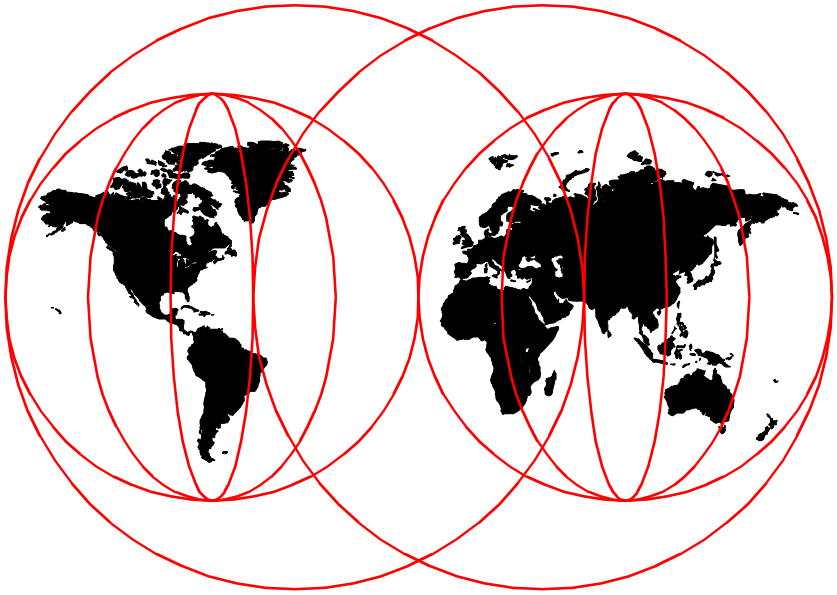
Building VLDB for BI Applications on OS/390: Case Study Experiences

Seungrahn Hahn, Nancy S. Ferguson, Markku Koskiniemi, Tarkeshwar Shrimali



International Technical Support Organization

www.redbooks.ibm.com

SG24-5609-00

SG24-5609-00

International Technical Support Organization

Building VLDB for BI Applications on OS/390: Case Study Experiences

January 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix F, “Special Notices” on page 259.

First Edition (January 2000)

This edition applies to DB2 for OS/390 Version 5, Program Number 5655-DB2, SmartBatch for OS/390 V1R2, OS/390 eNetwork Communications Server V2R6, DominoGo Webserver V5R0M0 for use with the OS/390 Version 2 Release 6.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook documents our experiences in building very large databases (VLDBs) for Business Intelligence (BI) applications through case studies with customers' real-life data, as well as from our work in the laboratory and the Teraplex Center. This book replaces *DB2 for OS/390 V5 Terabyte Database: Design and Build Experience*, SG24-2072.

This book describes the general considerations for each area of data warehousing and the approaches for addressing these considerations. Based on the results of case studies, it explains the various methods used and summarizes the pros and cons of each.

This redbook is written for anyone who is working on very large databases on OS/390 for Business Intelligence applications, including data warehouse architects, database administrators, performance administrators for data warehouses, storage administrators and system programmers who support data warehouses, and DB2 support teams.

Some knowledge of data warehouse concepts, DB2 for OS/390 V5 or later, and OS/390 V2.5 or later is assumed.

How to use this book

Designing databases, populating data, and managing workloads and end-user access to a VLDB warehouse are not trivial tasks. Such tasks require a variety of specific skills:

- Knowledge of the BI applications and how they manipulate the data
- Database design
- DB2 for OS/390 skills
- OS/390 skills such as SmartBatch and Workload Manager
- Network skills

Before you jump into a VLDB data warehouse project, we recommend you refer the appropriate systems staff to the following chapters.

Managers and project managers can get valuable information from Chapter 1, "Introduction to VLDB" on page 1, Chapter 8, "Scalability" on page 199, and Chapter 9, "S/390 strengths for BI applications" on page 221.

Those who are in charge of designing and building a data warehouse should read Chapter 2, “Database design for VLDB” on page 17, Chapter 3, “Data warehouse population” on page 57, Chapter 5, “Data mart on S/390” on page 99 and Chapter 6, “End-user access methodologies” on page 133.

Those who maintain the data warehouse should read Chapter 4, “VLDB utility maintenance” on page 91.

Those who have performance responsibilities should read Chapter 7, “Workload management in data warehouse environment” on page 163.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Seungrahn Hahn is a Business Intelligence specialist at the ITSO. Before joining the ITSO, she worked as an IT specialist in physical database design, performance tuning and data sharing implementation with DB2 for OS/390, and BI application solutions in IBM Korea.

Nancy S. Ferguson is an IT specialist working for IBM Global Services in support of various US customer accounts. She has 11 years of DB2 experience, five years with IBM. Nancy holds a M.S. in Computer Science and Engineering from the University of Connecticut. Her areas of expertise include DB2 systems programming, DB2 performance monitoring and tuning, database administration, disaster recovery, and project management.

Markku Koskiniemi is an IT architect specializing in data warehousing in IBM Sweden. He has 20 years of experience in databases and four years in data warehousing. He holds a degree in mathematics from the University of Umea, Sweden. He has worked at IBM for three years. His areas of expertise include Business Intelligence, project management and architectural solutions.

Tarkeshwar Shrimali is a Business Intelligence specialist at the BI Solutions Center in Dallas, TX. He has 19 years of experience in the computer industry, the last 10 with IBM. He provides technical support for DB2/MVS and is engaged in specialized consulting services emphasizing data warehouse design and performance tuning. He has done graduate research work on relational databases in North America and Australia. He holds a M.S. in Computer Applications from the Asian Institute of Technology, Thailand.

Thanks to the following people for their invaluable contributions to this project:

Viviane Anavi-Chaput
International Technical Support Organization, Poughkeepsie Center

Merrilee Osterhoudt
IBM S/390 Business Intelligence Technical Support Leader

M.H. Ann Jackson
Caryn Meyers
IBM S/390 World Wide Business Intelligence Technical and Marketing Support

Alice Leung
IBM S/390 Business Intelligence Initiative Management Leader

Chris Panetta
IBM S/390 Performance Analysis & Design

Ralph McCliment
IBM GBIS, Business Intelligence and VLDB Consultant

Dino Tonelli
M. Leticia Cruz
Andrea Harris
Nin Lei
Luanne McDonough
Darren Swank
Michael Tebolt
IBM S/390 Teraplex Center, Poughkeepsie

Stuart Pyle
Industry Consulting & Services - Americas

Bryan Smith
IBM Design Center for e-transaction processing

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks evaluation” on page 275 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com`
- Send your comments in an internet note to `redbook@us.ibm.com`

Chapter 1. Introduction to VLDB

What does the term “very large database” (VLDB) mean? There is no widely accepted definition for VLDB. As recently as the mid-1990s, a decision support database of 300 to 500 gigabytes (GB) was considered a very large database. By the late 1990s, databases of over a terabyte (TB) were not uncommon and with the growth of e-business and Web-driven commerce, data warehouses are growing even more rapidly. For the purpose of discussion in this book, we define a VLDB as a database that equals or exceeds one terabyte in total size.

This chapter provides an overview of the VLDB data warehouse environment and a summary of several case studies conducted by the S/390 Teraplex Integration Center (Teraplex) with IBM customers. We review the key issues many VLDB implementors encounter, and discuss how we addressed some of those issues in the context of the case studies. This chapter also reviews the design methods, strategies, and tools available to build a large data warehouse on S/390.

1.1 The demand for VLDB data warehouses

Studies by independent consultants have indicated that there is a rapid increase in implementations of data warehouses to support the growing number of business intelligence applications. Businesses of all sizes are proceeding at full speed to fund and deploy data warehouse applications. Many installations today have already implemented and are supporting data warehouses with multiple terabytes of raw data. One of the case studies discussed in this book was conducted with a financial institution that was building a warehouse based on 9 terabytes of raw data, with an anticipated growth rate of 500 gigabytes per month.

The number of users and query workloads is increasing in most organizations. Access to data warehouse systems by business analysts, marketers, salespeople, financial analysts and customer service managers is expanding rapidly, with increasing demands for faster access. The opening of data warehouses to customers, suppliers and business partners, via internet and intranet, is expected to accelerate this growth dramatically. Once a business has offered Web access to data, it is very common for the number of users of the data warehouse to increase much more quickly than originally projected.

Queries are also becoming increasingly sophisticated, and the growth of data mining, online analytical processing (OLAP), and other analytical tools

requires more powerful engines and sophisticated parallelism, both in hardware and software. Even though complex analytical queries typically make up only 10 percent of the total number of query workloads, they can consume more than 80 percent of system capacity.

Figure 1 shows the overall components of a BI system environment.

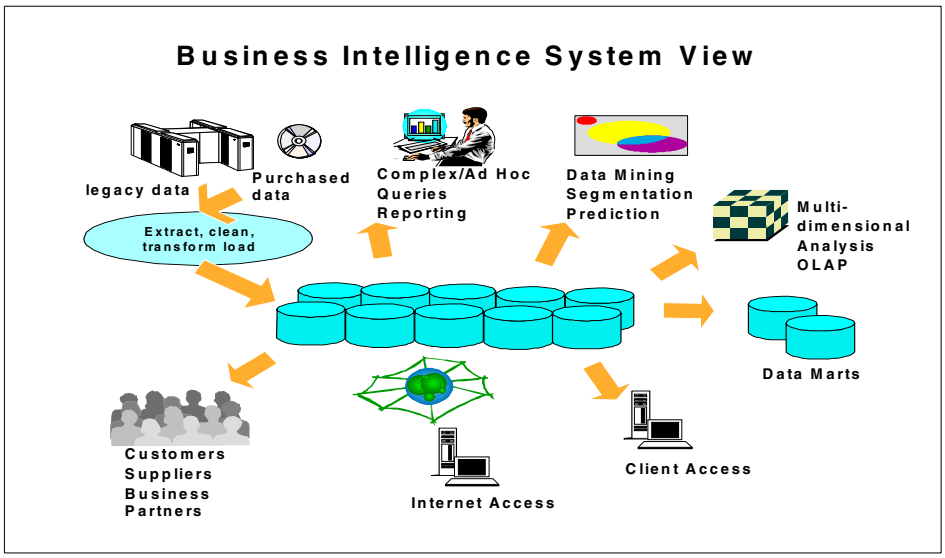


Figure 1. Infrastructure of a BI environment

Data warehouse solutions combine server and storage hardware, database software, and analysis tools to function as central collection points for information on a company's customers, products and business transactions.

Demand for data warehouses is being driven by marketers, salespeople, financial analysts and customer service managers across all industries, as they recognize the potential to improve the performance of their organizations and corporations.

1.2 VLDB issues and considerations

In this section we introduce the key issues and considerations that must be addressed when implementing VLDB data warehouses, across all hardware and software technology platforms. While most of the issues we will discuss are relevant to all data warehouse implementations, they can be particularly challenging in the VLDB environment. It is important to note that each data

warehouse implementation will have its own unique characteristics and the impact of individual issues will vary from implementation to implementation.

Key VLDB considerations that we discuss in the following chapters are:

1. Initial sizing and capacity planning for growth
   - Capacity planning for hardware, software and networks.
   - Implementation planning for building and maintaining data warehouses.
2. Data movement

   As the value of the business data warehouse becomes more evident, many organizations are building warehouses that are initially loaded with multiple terabytes. Businesses that are intent on leading market trends rely increasingly on business intelligence systems that reflect *today's* environment rather than last month's. The following must be considered:
   - Initial load time for very large volumes of data.
   - Batch window requirements for the data refresh or for periodic data update into a production data warehouse.
   - Fast purge of obsolete data.
   - Data currency requirements and real time updating.
3. Availability

   As business intelligence systems become more critical to the competitiveness of businesses and as businesses expand into worldwide enterprises, the requirements grow for continuous access to the data warehouse across all time zones. Batch windows decrease dramatically and high availability technology becomes critical. The following must be considered:
   - Continuous computing: 24 x 365 operation for highly distributed and globalized users.
   - Open data access to the data warehouse through the Internet, intranet and client/server environments.
   - Data currency: How current must the data be for analytical processing?
4. Scalability

   As a data warehouse becomes accepted in an organization, demand for its services grows. The need for new reports and aggregate tables increases, and the data warehouse explodes to several times its original size. More often than not, explosive growth occurs in the midst of a budget cycle and the supporting system must be able to respond acceptably to

the increase in data volumes, additional users and increased concurrency without an increase in capacity.

In a business environment of mergers and acquisitions, data warehouses are increasingly experiencing unplanned, spontaneous growth in data size. The data warehouse system infrastructure must be able to respond predictably to unpredictable growth in data and in users, so that batch windows are not compromised and end-user expectations can be appropriately set. The system must also perform predictably as capacity is added incrementally to CPU and I/O subsystems. The following must be considered:

- Granularity of upgrades of the chosen platform is needed.
- Will the expected improvement be achieved if processors and/or I/O bandwidth are increased?
- Data growth: After initial load, data maintenance should be completed within given batch windows, and query response time increase should be appropriate.
- End-user growth: How much impact will there be on query throughput and response time if the number of queries increases substantially?

5. Systems management

   As decision support systems evolve into mission-critical business intelligence systems, it has become increasingly evident that these data warehouse infrastructures require the same level of systems management focus as day-to-day operational systems. Ideally, these data warehouses and marts should be easily integrated into the enterprise's systems management infrastructure. The following must be considered:

   - Effective system resource management of CPU, DASD, channels, network bandwidth, especially I/O subsystem management.
   - Security and auditing for business-sensitive data.
   - Change management.
   - Automation of extraction, transformation, load and update processes.

6. Parallelism

   - Parallel data loading into data warehouse, data marts.
   - Parallel reorganization/recovery processing.
   - Parallel query processing.

7. Query performance and throughput

   Businesses are increasingly relying on BI applications to support the growing number of changes in their processes. As a result, many more queries are being sent, and those queries are scanning larger amounts of data, as well as becoming more complex and sophisticated. This means the following concerns assume even greater importance:

   - Consistent response times for short and medium queries.
   - Prioritization of critical queries according to business priorities.
   - Prevention of large queries from monopolizing system resources without depriving them of the ability to complete.
   - Managing throughput in a mixed query environment.

8. Data warehouse skills

   The fundamental database support skills are applicable across OLTP and data warehouse databases for a given database system (such as DB2 on OS/390), and IT shops can leverage their OLTP skills and people resources to start up and maintain their data warehouse environments.

   However, as the data warehouse grows in size and sophistication, additional skills must be brought into play in such areas as database architecture and design, tool selection and administration, performance, tuning, capacity planning, workload management and parallelization. The following must be considered:

   - What kinds of new skills are needed? Are data warehouse architects and/or business subject analysts needed?
   - Who in the organization will become more important in the data warehouse environment? Will multiple staff members become equally necessary, such as the capacity planner, network specialist, storage specialist, system programmer, data administrator or data warehouse coordinator?
   - If these skills are not available in the existing environment, how can they be obtained? What training or consulting services are available?

1.3 The case studies

Each case study is unique and is focused on S/390 capabilities such as parallel processing in data transformation, running queries in parallel, workload management, ease of connectivity, and scalability. For this redbook, we have selected representative case studies to illustrate suggested alternatives for addressing some key VLDB issues and considerations. The

results of these studies and the associated information are meaningful to anyone who is considering data warehouse design on a S/390.

The discussion in this book revolves primarily around case studies conducted by the Teraplex with two large US companies, one from the Banking industry (Customer A) and the other from the Health Insurance industry (Customer B), and the IBM Teraplex Center itself. We also make some references to case studies performed with companies in other industries such as Retail and Finance.

We cover only a subset of the tests and test results in this book. The names of the participating companies have been withheld due to confidentiality and the fact that BI applications can provide competitive advantages.

1.3.1 IBM S/390 Teraplex Integration Center mission

The S/390 Teraplex Integration Center (Teraplex) is missioned to support S/390 objectives to achieve growth and a leadership position in the very large database (VLDB) Business Intelligence (BI) arena in the following areas:

- Demonstrate S/390 hardware and software capabilities in the VLDB (1TB+) arena:
  - Provide supporting proof points
  - Publish VLDB information to customers and IBM sales force
- Optimize and stress test IBM BI software against TB size databases:
  - Reduce customer discovery of VLDB data warehouse-related problems
  - Provide optimization and tuning recommendations to customers for software
- Provide a facility for:
  - IBM customers and prospective customers to conduct proof-of-concept testing and case studies using VLDBs
  - IBM software and services partners (at our invitation) to stress and optimize their software against BI VLDBs
  - Expansion of technical expertise both within IBM and by our business partners in VLDB use and management

For details about the Teraplex Center, refer to *Data Warehousing with DB2 for OS/390*, SG24-2249.

1.3.2 Why choose S/390 for BI applications

Implementing a very large enterprise data warehouse requires a lot of work in organizing the data infrastructure environments, as well as in choosing and optimizing the right front-end tools.

When most of the source data originates on S/390-based operating systems, the S/390 provides a compelling platform choice for creating the data warehouse infrastructure. DB2 for OS/390's traditional strengths in data management and data movement, as well as OS/390's exclusive workload management technology and the cost advantages of being able to integrate the data warehouse into the existing enterprise systems management infrastructure, make the choice even more evident.

The customers who participated in the case studies wanted to exploit S/390 capabilities to reduce the learning curve to deal with VLDBs. IBM technical experts helped these customers understand and realize the benefits of handling a VLDB data warehouse with S/390, OS/390 and DB2 for OS/390, and to understand the requirements for the future production systems.

1.4 Banking - case study A

In case study A, the customer (known as Customer A) was in the early planning stages of a data warehouse that would exceed 9 terabytes of raw data in its initial implementation and would grow at a rate of approximately 500 gigabytes per month thereafter. There was little experience in the marketplace with such a large enterprise warehouse and Customer A sought Teraplex assistance in understanding the behaviors of this VLDB environment to gain insights into the following:

- How to do capacity planning for the data warehouse
- How to minimize the time for the monthly extract/transform/load (ETL) update cycle
- How to add/maintain data to existing tables in the warehouse while allowing access to *all* previously loaded data
- How to manage data accesses from the Internet
- How to handle changing data needs from the business
- How to minimize batch windows for backup/recovery
- How to ensure performance scaling with an increase in concurrent queries and huge increases in data (about 0.5 TB of data per month)

- How to provide consistent response time for small queries over large queries, and critical queries over trivial queries
- The impact on key DB2 resources of increasing concurrent queries
- How to use the total system resources efficiently by managing query parallelism
- The management of a VLDB

1.4.1 Objectives of the tests

In order to meet the customers' requirements, our tests were designed to:

- Validate S/390's ability to support future BI application growth for managing multiple terabytes of raw data (and DASD) and hundreds of concurrent queries.
- Demonstrate Workload Manager capabilities to dynamically manage system resources in a mixed query workload
- Demonstrate S/390's management of concurrent execution of online updates, database reorganization and queries that are required to achieve 24 X 365 continuous computing
- Demonstrate S/390's ability to scale:
  - Growth of users with no growth in CPU and I/O capacity
  - Growth of data with no growth in CPU and I/O capacity
  - Growth of CPU and I/O capacity
- Demonstrate DB2 for OS/390 functionality to carry out query parallelism with an appropriate partitioning strategy, indexing strategy, and rich DB2 statistics
- Demonstrate the effective coexistence of a data warehouse and data marts on a single S/390 system
- Demonstrate the ability to balance external sources of work using the Web enablement
- Facilitate knowledge transfer to the customer

1.4.2 BI applications in the tests

This case study focused on the analysis of account types like household, accounts and ATM transactions. Typical BI applications in the Banking industry are related to marketing and customer relationship management. Fast delivery of useful decision support information through the data warehouse and/or data mart to personnel at branch offices improves

marketing and increases competitiveness. Typical BI applications in Banking are:

- Marketing analysis
- Predictive marketing
- Marketing campaign planning and management

Because account analysis is the basis of marketing analysis, marketing campaign planning and management, the customer chose it as a case study test application.

1.4.3 Test system configuration

Two 9672 G5 Enterprise Servers coupled in a Parallel Sysplex were used for the test. Each server contained 10 CPUs (10-way) which made 20 individual CPUs available for processing. In some test case variants only 10 of the 20 CPUs were used. Figure 2 on page 10 shows the hardware configuration for the Banking customer.

There are:

- Two G5 RX6 servers
- Two G4 C05 Coupling Facilities
- One Sysplex Timer
- One N-ways Multiaccess Connector (Router)
- 10 TB RVA DASD with 20 9393-T82 disk controllers
- Eight 9032 ESCON Directors

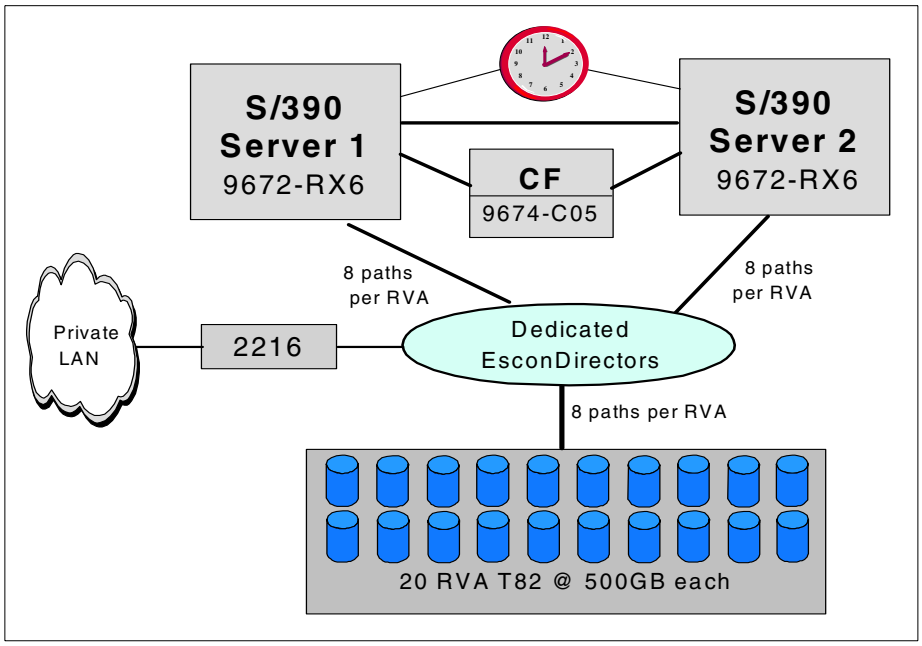


Figure 2. Overall hardware configuration of case study A

We had more than 5 terabytes of raw data which compressed into 1.3 terabytes of DASD using S/390 hardware compression. During the test, 20 RAMAC Virtual Arrays (RVAs) were used to contain user data, indexes, DB2 work files and temporary space for sorting.

We used an SMS storage group to automate the management and use of DASD. See 2.10.4, ”Data set placement” on page 47 for details.

1.5 Health Insurance - case study B

In case study B, the customer (known as Customer B) was in the early stages of designing a data warehouse implementation to replace their tape-driven reporting system. This customer participated in this study to understand the following issues and considerations prior to implementation:

- How to deploy a data warehouse solution quickly when most of the source data is on tape
- How to keep the data warehouse current with large volumes of updates, within the limits of a tight batch update window

- How S/390 scales to support a rapid growth in active users and increased query concurrency
- How to size your initial implementation
- How to effectively plan for data warehouse growth
- The characteristics of the Extract/Transform/Load layer and how to select the appropriate tools to meet the requirements of your data warehouse
- How OLAP tools can be used to improve the effectiveness of the decision support reporting system
- The VLDB considerations that must be incorporated into OLAP tool selection and deployment

1.5.1 Objectives of the tests

In order to meet the customer's requirements, our tests were designed to:

- Demonstrate the entire data warehouse process from the extraction/transformation of source data from large amounts of DASD and tapes to the population of the data warehouse within a given window and to query process by the business community
- Identify initial system resource requirements
- Demonstrate S/390 platform scalability in throughput with user growth while resources remained constant, with data growth while resources remained constant, and show the effects of adding system resources
- Demonstrate the ability of DB2 for OS/390 to handle 1+ TB of data in terms of better query performance and maximum exploitation of system resources
- Demonstrate Workload Manager capabilities to provide consistent response time of concurrent queries with different levels of priority
- Protect smaller queries from large (killer) queries, while protecting critical queries explicitly, and refresh data warehouse concurrently with query activity
- Demonstrate the effective coexistence of a data warehouse and data mart on a S/390 system
- Demonstrate the ability to analyze the data warehouse data using OLAP tools

1.5.2 BI applications in the test

We chose claim analysis as a case study application, which becomes the basis for further analysis.

This application will enable this customer to access health insurance claim data in a timely manner to answer questions from researchers, actuaries, statisticians and fraud and abuse management. Some of the questions currently require six weeks to extract and 36 hours to build a summary table. The DB2 data warehouse will enable them to access the same data and answer the same question in less than two hours.

The analysis performed can be considered as multidimensional analysis. Users want the ability to analyze the health insurance claim data from multiple dimensions such as year, quarter, month, claim parts, claim types, diagnosis, discharge status, surgical procedures, age, sex, race, and so on.

1.5.3 Test system configuration

The test system environment for case study B was basically the same as that for case study A, but based on each customer's specific requirements and current system environments, we added or modified our system configuration. We had:

- Two G5 RY6 servers
- Two G4 C05 Coupling Facilities
- One Sysplex Timer
- One N-ways Multiaccess Connector (Router)
- Four types of DASD with disk controllers
- Five 9032 ESCON Directors

We used several kinds of DASD, such as RVA, RAMAC I, RAMACII and 3390 to house user data, indexes, DB2 work files and temporary space for sorting. We also used SMS storage groups for the simple and effective use of DASD space.

Figure 3 on page 13 shows the system configuration with the DASD allocations used for the Health Insurance customer.

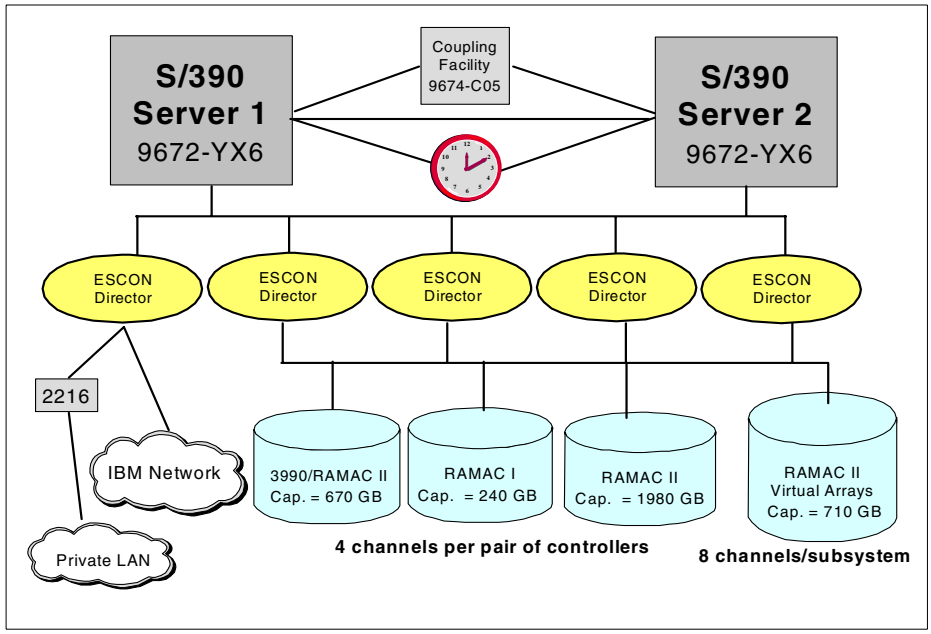


Figure 3. Hardware configuration of case study test B

1.6 IBM - case study C

This case study summarizes the 1 terabyte (TB) database build experience of the S/390 Teraplex team with DB2 Version 6.

1.6.1 Objectives of the test

The objective of this case study is to build one big table using DB2 Version 6 and its new features. With inline statistics option of DB2 Version 6, some DB2 utilities (LOAD, REORG, REBUILD INDEX) can be run with statistical information being gathered during their execution, which eliminates the separate execution of RUNSTATS utility following. Our focused areas were:

- Execute the DB2 LOAD and REBUILD INDEX utility with the inline STATISTICS option
- Compare the elapsed time of executing the LOAD and RUNSTATS utilities separately to executing the LOAD utility with the inline STATISTICS option

1.6.2 Test system configuration

Hardware configuration for this case study consisted of S/390 Enterprise Server, a mix of RAMAC DASD types and models, and the requisite connectivity between servers and DASD. Figure 4 shows the hardware configuration used.

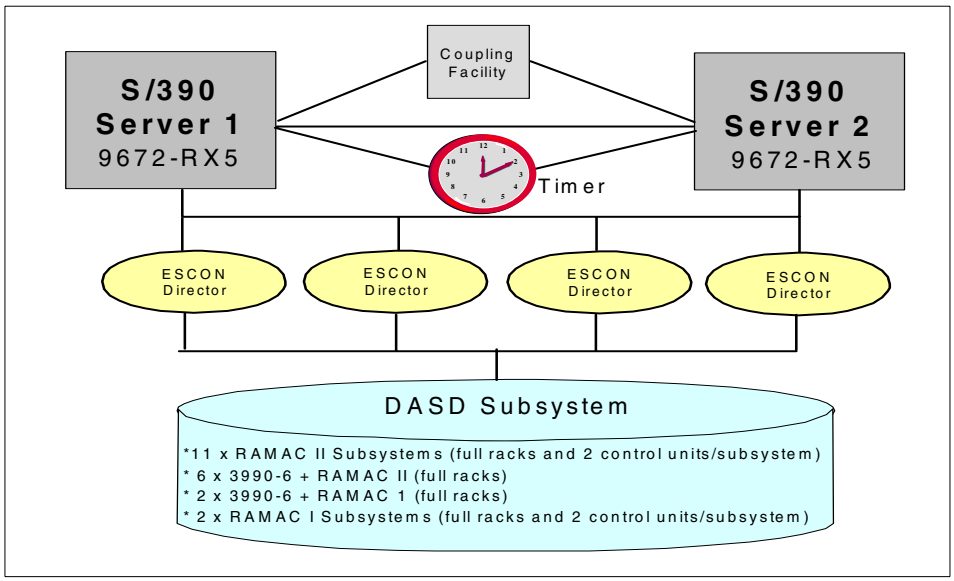


Figure 4. Schematic showing hardware configuration for case study C

The hardware configuration consisted of:

- Two S/390 G4 RX5Enterprise Servers
- One S/390 Coupling Facility Model C04
- Four ESCON Directors (9032-003)
- Eleven RAMAC II Array Subsystems (2 CUs/rack)
- Six 3990 CUs backed by RAMAC II Array DASD
- Two 3990 CUs backed by RAMAC I Array DASD
- Two RAMAC I Array Subsystems (2 CUs/rack)

Prior test had shown that for very I/O-intensive workloads, one G4 RX5 (10CPUs) server was capable of handling the bandwidth supplied by 20 RAMAC II control units (CUs). In other words, each single CPU could handle 2 RAMAC II CUs. Most workloads, ours included, are neither totally I/O-intensive nor CPU-intensive. Therefore, we selected 34 CUs (various

types/models) in combination with two G4 RX5 (20 total CPUs) servers as a reasonably balanced system configuration for our testing purposes.

1.7 Software used in the case studies

The key software we used for case studies were almost the same. Some vendor's software were used because the customers were using it in their current production systems, or wanted to verify the functionality of the products with their real data in a VLDB environment.

We used a variety of software products, but the key products were the following:

- OS/390 Version 2 Release 5
- DB2 for OS/390 Version 5 (for case study A and case study B)
- DB2 UDB for OS/390 Version 6 (for case study C)
- DB2PM for OS/390 V5R1
- QMF V3R3 for dynamic query
- Net.Data V2R1
- SmartBatch for OS/390 V1R2
- TME10 NetView for OS/390 V1 R1
- Domino Go Webserver for OS/390 V5
- Network Dispatcher
- Loadrunners
- Additional solution developers' products: BMC utilities, OMEGAMON for DB2, OMEGAMON for MVS, Platinum Technologies and so forth

1.7.1 Client tools available to use the data warehouse

We also used some end-user tools to generate queries and reports, and analyze the data in dimensional forms for the business end users.

If you already have these tools, you can use them in S/390-based BI system environments, too. They are:

- QMF for Windows
- DB2 OLAP Server
- MicroStrategy DSS family
- QueryObject

- Business Objects
- Visual Basic-based
- Brio Technology
- Cognos
- Hummingbird GQL
- PC SAS
- Focus on Windows
- PowerBuilder
- CA Easytrieve

If you use the double-byte character set (DBCS), contact the local software provider to verify that these products can handle DBCS data.

1.8 Naming conventions of this redbook

Each test uses its own unique names in this redbook. They consist of three fields, xxx-y-z, where xxx means test group name, y is A, B or C which stands for each case study, and z is the serial number of the test.

Test group names are as follows:

- DMT: Tests for data marts
- WEB: Tests for Web connections
- WLM: Tests for Workload Manager
- SCA: Tests for scalability

Chapter 2. Database design for VLDB

In this chapter, we document the design aspects of the very large databases (VLDBs) in the two customer case studies performed at the Teraplex Center. Business intelligence (BI) applications typically process a large amount of data to answer user queries. At very large sizes, typically terabyte+, the data warehouse design and the underlying database design play an increasingly major role in the success of the project. The small window of time (usually a weekend when the data warehouse is refreshed with current data from operational databases) and amount of hardware resources available, govern the physical design of the databases.

The BI use of the database (where large sweeps of databases are the norm) versus the online transaction processing (OLTP) use (where small amounts of data are updated) are also considered in the physical design.

We also describe the logical design, physical implementation, design goals, partitioning, compression issues and the decisions we took for the Banking and Health Insurance case studies. Most of the complexity of design and implementation is due to the nature of VLDB, and will have similar or more complexity in other database platforms. The S/390 and DB2 for OS/390 offers excellent design flexibility by way of design parameters, such as partitioning, hiperpools, SmartBatch, compression, WLM and so forth, to make it the most efficient implementation vehicle among IBM and non-IBM databases. We note decisions we took for DB2 for OS/390 to implement the multi-terabyte VLDBs at the Teraplex Center.

2.1 Logical DB design

A *data warehouse* or a *data mart* is home to interrelated business data. The logical database design helps us understand the enterprise's data and relationships between the data.

Many times this design is expressed pictorially in a data model. Data models are created in several forms. They are *entity-relationship* (E-R) diagrams, *snowflake* or *star schema* models. The modeling paradigm is influenced by the architecture of the underlying database platform, that is relational, or multi-dimensional (OLAP, MOLAP or ROLAP). However, it is possible to model a data warehouse with a relational metaphor, and proceed with the implementation using OLAP.

The data architecture of the case studies described in this book is relational. An end-user tool can access data from the data mart and provide

multidimensional processing to the user. The logical database design is the view of the data that an end user sees and accesses. This view is unencumbered by the implementation details brought on by the physical design to meet performance and other end-user requirements.

The business purpose of the data warehouse is expressed in the logical design. This design's physical implementation, as expressed in the execution of the logical design, helps achieve the business purpose. Having the correct and complete logical design is necessary to any warehouse implementation. Simply put, if you cannot answer a business question with the logical design, it cannot be answered by any physical implementation of the design. The logical design highlights only the data architecture and omits the process of populating, updating and querying information. The physical design is cognizant of the process issues, requirements, and constraints. In contrast, the logical design usually is vendor- or product-neutral; that is, the same design can be implemented on a multitude of vendor products and is not tied to a particular commercial brand.

The logical database design is best done by someone who is an industry expert. The expert knows the business objects, interrelationships and processing requirements well; however, that person may not be expert on the software and hardware platforms on which the design will be implemented.

At the Teraplex Center, IBM engages product experts to ascertain that the physical implementation of the logical design exploits the underlying hardware and software to ensure the efficient utilization of physical resources.The logical database design is provided by the customer. Figure 5 on page 19 and Figure 6 on page 20 are the subsets of the logical database designs of the Banking and Health Insurance VLDB implementations, tested at the Teraplex center. The Health Insurance logical model was developed by the customer with help of industry experts from the IBM Consulting Group.

Note that although the models depicted here seem like star schema models, they are E-R models. Star schema models are better suited and frequently used in conjunction with decision support processing like OLAP processing, using a multi-dimensional design paradigm.

The Banking enterprise data warehouse (EDW) is best viewed as a warehouse of household financial data. There is a Household entity which has one row for each household, and an Account entity with one row for each account.

A household can have many accounts. Each account is further described in detail in a CD, IRA, Checking or Savings entity. ATM and Teller entities have one record per month for each account holder keeping the summary data (Figure 5 shows a one-month volume of data).

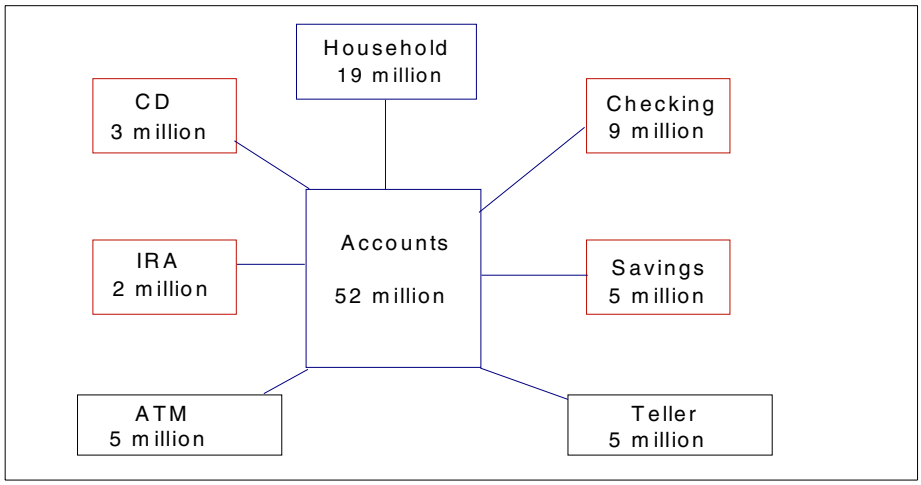


Figure 5. Subset of logical database model for case study A - Banking

The Health Insurance EDW keeps health insurance claim data. The Common Claim entity keeps data elements common on all claim types. Claims can be from providers. Claims can also be filed by physicians, institutions and non-institutions.

There is an entity for each type of claim and an entity for the Individual who has filed a claim. There is also a history entity; see Figure 6 on page 20.

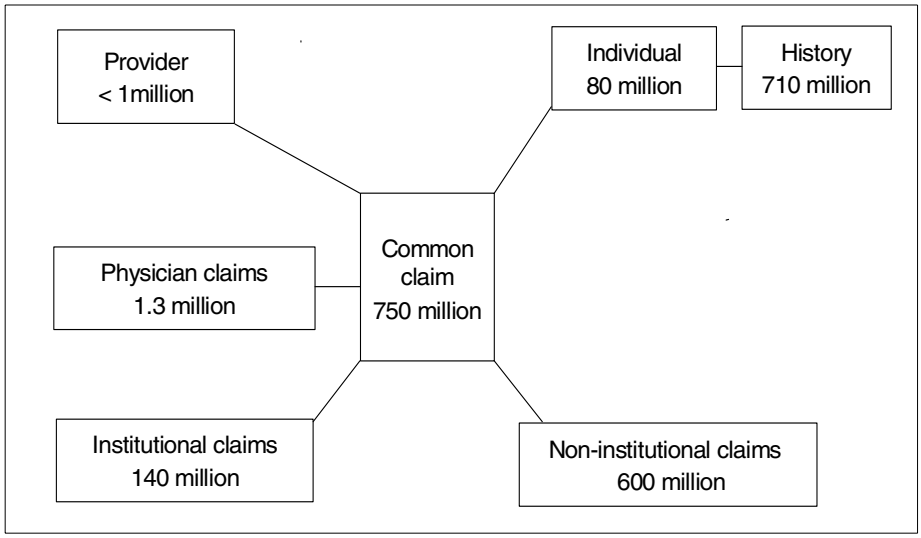


Figure 6. Subset of logical database model for case study B - Health Insurance

2.2 Physical DB design

The logical design is implemented using selected hardware and software. The physical design has to meet the operational requirements for the warehouse. We list here a few of the general considerations and some DB2-specific issues you have to analyze before you commit to a particular physical design.

Initial load and maintenance requirements

- **Operational constraints:** Are any time limits imposed in terms of loading and periodically refreshing the database? What is the batch window we need to conform to? How are we going to load the data? What is the size of the warehouse? How many tables and rows? Answers to these simple questions are of enormous importance and result in complex design decisions when the database size is in terabytes.

  How are we going to update the tables within the data warehouse--will we append data at the end of the data set, or in a new partition, or insert/update anywhere in the tables? For very large databases, appending data at the end reduces the time needed to refresh and maximizes the database availability during updates. Usually we will use a database utility to load a new partition, thus only the new partition being filled is accessible by queries.

For case study A, the customer needed three years of history, amounting to approximately 5.3TB of raw data. The individual table size ranged from 20 KB to 1 TB. The largest tables had 600+ million rows. The total build time for this effort including load, building multiple indexes per table and doing RUNSTATS, was 48 hours. This total timing was for three years' worth of data.

- **Availability requirements:** Are there batch window issues? How much down time is being planned for the operational databases to unload the data and for the BI data warehouse to refresh? For case study A, one of the main goals was continuous computing, that is, to have EDW available around the clock. We will see, in the physical design, that with partitioning we achieved this goal; at no time is the whole EDW unavailable. We apply maintenance to one partition at a time, thus we take only one partition at a time away from users. With this method, queries that do not read data from the partition under maintenance are not impacted.
- **Load times:** Loading the data warehouse entails extraction from legacy databases and transformation and cleansing of data. The DB2 LOAD utility can then be used to populate the warehouse. How many of these processes can be parallelized, that is, broken into multiple jobstreams to be run concurrently? Two methods (pipes or DASD striping) for optimizing load processes, and thus reducing the load times, can be the keys to achieving an efficient load.

  In one case study, MVS SmartBatch (with pipes) was used extensively to reduce the elapsed time of any job requiring intermediate files.
- **Utilities:** What concurrencies or efficiencies in utilities need to be implemented (for example, collect RUNSTATS while doing a data load)? Are there any OEM vendor tools available to us which will accelerate execution of any of the tasks in VLDB environment (for example REORG, Image copy or LOAD utility)?
- **Backup/recovery:** How are we going to recover in case of failure? How much time is allowed for reloading a table, the entire EDW? Shall we reload the data from operational data sources? Is this choice available or do we need to restore from image copies?

  All of this needs to be decided prior to starting the physical design. The amount of recovery time available and the data refresh or incremental update data size which needs to be maintained will have an impact on partitioning strategy and recovery utility processing.
- **Level of normalization:** Third normal form (3NF) is an ideal goal: it simplifies data maintenance. Higher order normalization in design is usually a trade-off with performance, but there are good reasons to have a

3NF-based database design. A big advantage of 3NF is reduced data redundancy, and thus reduced possibilities of data elements across different tables having inconsistent values.

- **Referential Integrity:** Use of Referential Integrity (RI) is popular in DB2 installations. If RI is implemented in operational data stores, should this be continued in EDW? When we implement RI in a VLDB environment it has load time implications, as referential constraints need to be verified for rows being loaded. Also, if data has come from a database where RI is implemented, the need to verify RI again is diminished greatly. If the original data source does not have RI, one needs to have a strategy and action plan in place to deal with the rows violating RI constraints.
- **Summary tables:** Summary tables can save a lot of processing time for queries which are very frequently executed, since results are pre-tabulated and queries do not have to plow through millions of rows each time. We preserve system resources by processing the data once and save the results for later use. This is a paradigm used by a few tools in the OLAP space. If we know the frequently asked questions or reports, it may be cost effective to pre-tabulate the results and save them in designated summary tables.

  If the EDW is being continually fed from operational sources, however, continually updating the summary tables may not be cost effective. In addition, business role changes in dimension tables (for example, product or sales geography realignments) may require the summary tables to be rebuilt. Sufficient system resources and batch window must be allocated for this.
- **Storage management:** DB2 allows you to manage the storage by yourself, that is, you manually place DB2 data sets on a set of volumes. Alternatively, SMS storage groups can be defined and placement of data sets can be automated within the SMS storage groups by pre-defined ACS routines.

  In a VLDB environment, thousands of data sets must be managed. One has to weigh the pro and cons of automating this task by using SMS Storage Groups, or letting DBAs manage storage manually. For the case studies, we used SMS and automated the task of data set placement and management. We used high level qualifiers (HLQs) within the ACS routines to spread the data sets on volumes. We did not notice any contention or hotspots on any of the SMS-managed volumes.
- **Partitioning:** DB2 allows a table to be split up to 254 parts. The number of parts may increase in future releases. The ability to split a table among multiple partitions presents many operational advantages. Each partition has a separate data set. Having partitioned tablespaces allows for higher

query parallelism and parallel use of utilities. In some ways, partitions can act as independent tables at a physical level transparent to the programmer and query user. We discuss partitioning in greater detail in 2.3, “Partitioning strategy” on page 23.

- **Use of hiperpools:** Hiperpools are cache for DB2 virtual bufferpools. They back up bufferpools and help improve the buffer pool performance by increasing the synchronous page hit ratio. This is a very effective technique to reduce I/O and pin tables in memory. We used this to pin large gigabyte-size lookup tables in memory and thus reduced the elapsed time for the initial load, data transformation and monthly update process in several case studies.
- **Load using SORTKEYS:** Using a SORTKEYS clause in the load statement causes keys extracted during the reload phase to be passed in memory. When sorting completes, the sorted keys needed for building indexes are passed in memory to the build phase. This allows for shorter elapsed time for load utility processing. The disadvantage is that the LOAD utility is not restartable following a failure during reload, sort, or build. Based on the restart requirements and load times available during the batch window, you need to decide whether to use a SORTKEYS clause.

  We used a SORTKEYS clause for the case studies to reduce load times.

2.3 Partitioning strategy

One of the benefits of DB2 for OS/390 is its partitioned tablespaces (TS), which allows physical subsetting of the tables. DB2 allows for three types of tablespaces: simple, segmented and partitioned.

Simple: A tablespace that can contain more than one table. The space is composed of pages, and each page can contain rows from many tables.

Segmented: A tablespace that can contain more than one table. The space is composed of groups of pages called segments. Each segment is dedicated to holding rows from a single table.

Partitioned: A tablespace that can contain only rows from a single table. The space is subdivided into partitions based on the key range of a nominated partitioned index. Each such partition can be processed concurrently by utilities and SQL.

Our experience shows it is best to use a partitioning scheme that considers query performance and overall system management requirements including

utilities and ongoing maintenance. Initial load should not be the primary consideration. We advise that all large tables be partitioned. For details on the use of tablespaces, see *DB2 for OS/390 Administration Guide,* SC26-8957*.*

A tablespace can be partitioned in 254 ways. Each partition is a separate data set. At the operating system level, all 254 partitions can have separate activities run against them. Thus a query or utility could run up to 254 parallel tasks on one (logical) table. In all large data warehouses, you exploit this feature of DB2 to reduce the elapsed time of tasks.

You have to decide on a partitioning key, a requirement for partitioned tablespaces. DB2 uses the key ranges of this key to put a record in the appropriate partition. Thus, selection of the partitioning key is one of the most significant decisions one makes in designing the BI data warehouse. The key also becomes the clustering index for the table. The partitioning key is used to spread the data within different “partitions” of a TS.

When determining the partitioning key, the key should be chosen such that it facilitates the design goals for the data warehouse, which may be:

- Maximizing availability or minimizing run time for specific queries
- Allowing load and refresh activities, including extraction, cleansing and transformation of data, in a fixed operational window
- Reducing hotspots
- Increasing parallelism for queries and utilities
- Accommodating the growth of data

You have to define the prioritized operational requirements first and design accordingly. Many of the goals mentioned here are not compatible with each other, that is, they are trade-offs and cannot be achieved together.

2.3.1 Partitioning methods

Partitioning is based on a key or a set of key values. One designates the ranges for the key columns for each partition in the Data Definition Language (DDL) statement.

Partitioning is done by a Key range, for example names starting with A to C in partition 1, Names starting with D to F in partition 2 and so forth. Thus a major decision is to choose the use of a partitioning key. Key range partitioning can be based on chronological order or any other key ranges (for example, Account Number, Social Security Number or Claim ID).

Whenever you use partitioning, prior analysis is always required. You need to consider the length of time for the data in the table to be kept, and amount of data. DB2 Version 5 supports a DB2 table up to 1 TB and allows partitioning of any table up to 254 ways. Thus, you have to consider data warehouse growth beforehand and have a strategy should you hit this limit. In DB2 for OS/390 V6, there is relief by being able to dynamically change key ranges. A single table also can be up to 16 TB in DB2 V6, which is also relief of DW growth.

In the following section, we describe some design options for partitioning.

2.3.1.1 Key range partitioning by time

Many times the data is generated or arrives in chronological order and the time dimension presents itself as a natural choice for the partitioning key. Usually this data is also indexed in operational databases in chronological order.

For this kind of data, use a key that is time-based, that is Time, Month, Day or so forth. Thus data from a contiguous period can be together in a partition; see Figure 7.

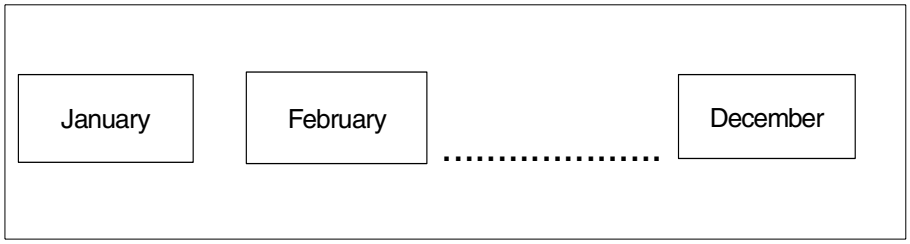


Figure 7. Partitioning by time

The resulting considerations of partitioning by time scheme are the following:

- The new data, for example, for a month or a week, can be loaded at the end of a tablespace without bringing the whole tablespace down.

  This allows for continuous availability, because you are building a new partition or inserting into a new partition while the rest of the partitions are available for query activity.
- There may be contention (hotspots) in the last partition. Usually the end users, the EDW customers, are interested in the recently added data. Thus a disproportionately large number of queries could refer to the data in the last partition, impacting the response time for these users.

One design alternative is to use a separate table for the current period. This new table is partitioned allowing parallelism. This segregates all of the query activity for the current period into a new table and will increase parallelism as the target data is in many partitions. DB2 V6 allows for dynamic changing of the key ranges for the partitions, reducing the need for separate tables.

- If there are Multiple Years of Data in one table, then you have a small number of partitions per unit of time. If we allow only one year of data in one table, DB2 can produce large numbers of partitions per unit of time. However, one year of data per table requires an SQL UNION when the result set must span more than one year.
- Older tables can be dropped and older partitions can be emptied and space released when data is no longer needed in a matter of seconds. In addition, as stated earlier, with the DB2 V6 dynamic key range update, the dropping and recreation of tables may not be required.
- A higher degree of parallelism for non-time-oriented queries is enabled, as now the query can be split and simultaneously access data in all partitions. Even for time-oriented queries, parallelism is enabled for those queries spanning multiple partition time periods.

2.3.1.2 Non-time based Key range partitioning

Here the key values from different ranges (for example, state names or zip codes) are used to select in which partition the data will be placed. In Figure 8 for example, we use state name as the partitioning key, resulting in the following partitions.

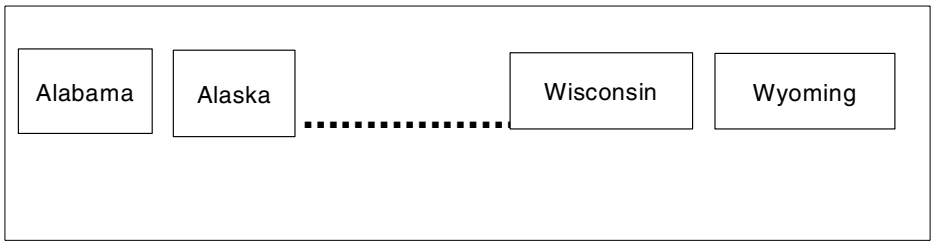


Figure 8. Partitioning by key ranges

Typically, an analyst looks at one region's data at a time. When this happens we can observe the following:

- This strategy allows for high levels of concurrency and thus produces less contention. Since analysts look for data from one state at one time, their individual queries are confined to one partition at any one time.

- Data is collected from one partition only. This strategy reduces parallelism and is thus potentially slower for individual queries, because only one task serially collects all the data and processes it from one partition. However, this approach results in higher collective throughput as queries can now “read” data from one place and do not have to “collect” data from multiple partitions, so they thus consume less resource. This is illustrated by Figure 9 as only one task is executed to retrieve the required state’s data such as Florida data, because Florida data is now only in one partition.

Resource consumption is reduced per query because you do not scan all partitions for the same amount of data, eliminating the overhead related to splitting the query and collecting all the data from parallel tasks.

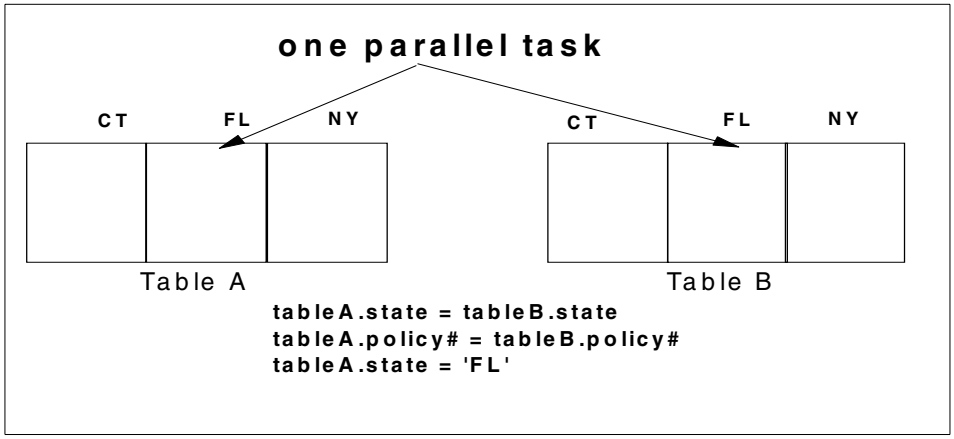


Figure 9. Key range partitioning - one parallel task

2.3.1.3 Random key partitioning

Sometimes it is necessary to randomize the distribution of rows within multiple partitions. This is particularly useful when the partitioning key has very skewed data distribution. We use a column or existing key to get the “random key”. Note that we keep the original key, along with the randomized key, in the table. Following are two strategies used to simulate “Random Key Partitioning” and distribute of data among partitions.

Use of a FIELDPROC

DB2 allows for a FIELDPROC user exit when inserting, updating or loading the tables. This exit allows data transformation when stored and retrieved. Thus the data presented to an application or retrieved by a query is different than the data that is stored in the tablespace. The stored data can be

transformed any way, for example compressed or bits transposed. This mechanism can be used to ensure uniform distribution of the data rows over partitions to avoid skews in the data distribution.

For detailed information about use of FIELDPROC, refer to *Data Warehousing with DB2 for OS/390,* SG24-2249.

Use of hash key

You can use a hash key to map data in order to randomize and distribute data among partitions. These methods are implemented programmatically. For example, you can use the Transpose function to map data from “Branch Number.Account Number” to “Account Number.Branch Number” by using a twos complement and so forth to get a new key.

You should realize that not all hash functions will distribute data evenly. Thus, if one hash function does not work with your data, you should try alternate functions.

It is customary to store the old key along with the new key in the database so that end users do not have to decipher the *newkey* to *oldkey* as shown in Figure 10.

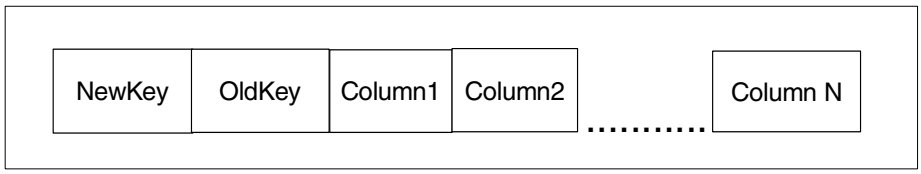


Figure 10. Hash key partitioning

NewKey is derived by a transform on the OldKey, for example:

Partition# = Remainder(OldKey/n)+1

NewKey=Partition #.Oldkey

When we randomize the data across partitions, the following can be observed:

- There is better distribution of the data because hash redistributed the data across the partitions. If the data distribution is not even enough, try another hash function, until the distribution achieves your goal.

- There are more parallel tasks as data needs to be retrieved from multiple partitions; see Figure 11. Note that in this example, state name is not the key used for hashing.

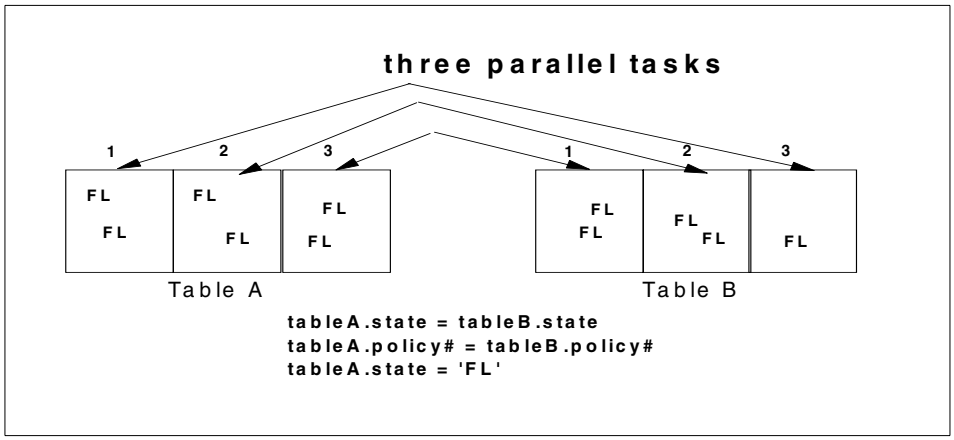


Figure 11. Hash key partitioning - three parallel tasks

This method is further illustrated in 3.3.3 *Partitioning Index* in *Data Warehousing with DB2 for OS/390*, SG24-2249.

By using a hash key, you distribute data across more partitions, which makes it more likely to generate more parallel tasks for queries and thus reduce the elapsed time of the queries. However, when you decide to use a hash key, consider the following factors:

- The joining tables should use the same column sequence in the partitioning key. If not, when join occurs, DB2 will not be able to capitalize on the similarity in clustering sequence and may choose a different access path, needing page scans or sort.
- When the qualified rows are large, having more parallel tasks run against more partitions is beneficial with a hashing key. If not, using key range partitioning is better than using hash key partitioning, because the latter often need more page scans.
- Queries running in a hash key partitioning environment tend to scan more partitions. This leads to higher consumption of system resources, which might have an impact in systems of high concurrency.

Partitioning affects the whole query load, and what works better for one query may not help other queries. The partitioning decision, the selection of the keys, hashing and so on are some of the most important decisions in a data

warehousing environment. You need to look at overall query effects for any decision taken and use an approach which will satisfy the business needs.

The use of ROWID for hash partitioning
This is a DB2 Version 6 feature and we did not use this for our case study tests.

The generated value of a ROWID column acts like a random value in the range of X'00000...' through X'FFFFF...'. Because of this semi-random nature, a ROWID column used as a partitioning key distributes the rows of a partitioned table across the partitions in a random way. In addition, you do not even have to specify the column. DB2 automatically generates it for you.

Therefore, if you would like to partition your table, and it has no other column that would serve as a good partitioning key, you can include a ROWID column in your table and use it as your partitioning key.

Example: Assume you want to create four partitions for an employee table, but because my employee numbers are assigned in ascending order, the employee number is not a good partitioning key. You can include a ROWID column in your table definition and use that as the partitioning key as shown in Figure 12:

```
CREATE TABLESPACE EMPSPACE NUMPARTS 4;
CREATE TABLE EMP
  (Id SMALLINT, Name CHARACTER(30),
   Salary DECIMAL(7,2), Deptnum SMALLINT,
    Emp_Rowid ROWID GENERATED ALWAYS,
    Picture BLOB(300K),
    Resume CLOB(100K))
    IN EMPSPACE ;
CREATE INDEX EMPARTIX ON EMP(Emp_Rowid)
CLUSTER (PART 1 VALUES(X'3FFF'),
                 PART 2 VALUES(X'7FFF'),
                 PART 3 VALUES(X'BFFF'),
                 PART 4 VALUES(X'FFFF')) ;
```

Figure 12. Use of ROWID as a partitioning key

Because DB2 pads the ascending PART values, the CREATE INDEX statement does not need to specify a complete value of the entire length of a row ID. (In a WHERE clause, though, the value compared to a row ID is not

padded, so complete values must be specified to confine an SQL statement to a particular partition.)

2.3.1.4 Number of partitions

In this section we discuss many factors which must be considered before deciding the number of ways a table may be partitioned.

Business considerations: There may be a good business reason to keep some data together, for example weekly or monthly data within a partition, so that when it ages it can be deleted altogether. What are the business growth plans? How many months', years', regions' worth of data will be housed in the EDW? You should avoid filling all the partitions in order to allow for growth. In DB2 V6, dynamic key range partitioning is available; that is, DB2 can move data to new partitions or rebalance partitions per new key ranges. After re-balancing, the affected partitions are put in reorg pending state.

(Prior to V6, if partition boundaries are to be changed, you must unload the data, drop the table, recreate the table, reload the data, create PI and NPIs, do new grants, and bind any application programs.)

Utility processing: Different utilities can be run on different partitions simultaneously.

Size of the data: Data of a large size can be better handled if it is partitioned; this will also reduce elapsed times for index creation, loads, reorgs and so on.

Batch window availability: This is a major design parameter. The amount of batch window available and the amount of data that needs to be added or maintained determines the level of parallelism needed to achieve the goal. The partitioning level implies an upper limit on how much query parallelism, batch processing or utility processing elapsed times can be reduced.

CPU and I/O configuration: Consider having the number of partitions that the I/O subsystem can deliver and CPU can consume; that is, you should try to balance the data production and consumption. Note, however, that this is only one of many factors in determining the number of partitions and not the only consideration. In most VLDB environments, the hardware configurations frequently change in order to support needed capacity increases, to support new data requirements, and to keep current with technological advancements. You should ensure that the physical design of the data warehouse exploits the hardware; however, you should not tie a design to the current I/O and CPU configurations very tightly.

Query parallelism: The most day-to-day benefit of partitioned tablespaces to end users is reduced elapsed time due to query parallelism. You should examine the key queries, their access paths and elapsed times to partition the data in order to get the desired elapsed times.

2.4 Index considerations

Non-partitioned indexes can be built during LOAD or can be deferred to later. We opted to defer to reduce LOAD process elapsed time.

The three methods of building non-partitioned indexes are:

1. Build all indexes while loading the table.
2. Build the partitioning index while loading the table. The RECOVER INDEX utility is then used to build the non-partitioned index. In DB2 version 6, you will use REBUILD INDEX instead of RECOVER INDEX.
3. Build the partitioning index while loading the table. Then, build the non-partitioned index using a process we call the parallel recover index method, which can reduce elapsed time significantly by parallelizing certain steps of RECOVER INDEX utility. In DB2 Version 6, the parallel index method is incorporated into the REBUILD INDEX utility for the unload and sort phases if the SORTKEYS parameter is specified. Refer to *DB2 UDB for OS/390 Utility Guide and Reference*, SC26-9015.

2.4.1 Design non-partitioned index for query parallelism

Physical contention can be reduced on non-partitioned indexes by breaking them into multiple data sets (pieces). These pieces can be spread across the I/O subsystem for maximum performance. In this manner, multiple accesses to different index pieces can take place simultaneously.

For non-partitioned indexes, use the PIECESIZE parameter on the CREATE statement to indicate the size of each index data set. Smaller values result in more data sets defined. The default is 4 GB for table spaces defined as LARGE and 2 GB for all others. Up to 128 data sets (pieces) per index are allowed (in V6, up to 254 data sets per index are allowed).

To choose a PIECESIZE value, divide the non-partitioned index size by the number of pieces desired and round up to closest valid PIECESIZE value. (In some cases it may be necessary to round down so the PIECESIZE does not have a lot of wasted space. In this event, there will be more pieces of smaller size.)

The more pieces spread across the I/O configuration, the better. If the non-partitioned index is likely to grow, a larger PIECESIZE value should be considered. Keep the PIECESIZE value in mind when choosing values for primary and secondary quantities for the VSAM data sets. Ideally, the value of the primary quantity plus the value of the secondary quantity should be evenly divisible into PIECESIZE.

2.5 Compression

In a typical database, there are strings of data which are repeated in part or in full. Here we document the reasons why you might use compression, and the potential cost and savings. Table 1 shows sample compression benefits documented by *International Technology Group, "Enterprise Solutions for Business Intelligence"*, Spring 1999.

Table 1. Effect of S/390 data compression

Industry	Database Size	Compression Level
Finance	4 TB	56%
Banking	1-2 TB+	50%
Insurance	1 TB+	45%
Government	100-300 GB	45-50%
Transportation	2 TB	45%
Utility	2 TB	40%
Retail	1.3 TB	40%
Manufacturing	300-450 GB	40-45%

Possible ways to implement compression are as follows:

Software compression: The data is encoded and decoded via software implementation of Ziv-Lemphal Algorithms. Typically, this is the slowest type of compression. DB2 compresses and decompresses the tablespaces only. Indexspaces are not compressed.

The software compression path length has been documented to be 4 to 6 times longer than hardware-assisted compression. Nowadays it is more practical to use hardware-assisted compression.

Hardware compression: Data is encoded and decoded by processor within the Central Processing Unit (CPU) via special hardware instructions

introduced in OS/390 in DB2 V3. DB2 only compresses the data in tablespaces. Indexspaces are not compressed.

RVA compression: RAMAC Virtual Array (RVA) is a high performance RAID-6 storage subsystem. The RVA uses fixed block architecture (FBA). The traditional storage subsystems use count key data (CKD) architecture. The RVA RAID implementation has very good data protection benefits. It also has unique compression benefits. RVA architecture is detailed in *Using RVA and SnapShot for Business Intelligence Applications with OS/390 and DB2,* SG24-5333.

RVA compresses all DB2 data including index, logs and tablespaces, sort work areas and so forth, while DB2 compression only performs compression on data in tablespaces. In warehouse environments, there is a large amount of space consumed by indexes (partitioned and non-partitioned), sometimes more than raw data space. RVA compresses everything. Additionally, RVA architecture eliminates the gaps between the records as they appear, before data is written to disk. Experience shows that with RVA, customers can achieve an overall compression and compaction ratio of 3.6:1. In contrast, typically DB2 compression yields 46 percent to 61 percent savings. Note that you can reach a higher compression ratio if your data has a larger amount of nulls, repetitive data and so on.

When DB2 compression is implemented, the additional RVA compression effects are minimal and not as pronounced. However, you still get the large RVA benefits on index spaces. In data warehouse environments, the partitioned index (PI) and non-partitioned indexes (NPIs) are also big and can rival raw data in size.

Compression benefits

DASD Saving: As shown in Figure 13 on page 35, compression typically yields a saving of 50 percent in DASD. (In the Banking study we measured the compression ratio to be 75 percent. This is extreme and was achieved mostly due to having an unusual amount of null values in columns.)

Even with careful planning you can end up with a large number of nulls in the warehouse because a data warehouse design is often preceded by an enterprise-wide data modeling effort. Data modelers often define new data elements, incorporating data from multiple sets of operational databases. Applications must be built or enhanced to incorporate these new data elements on data entry forms.

However, without proper discipline and controls, values for these new elements may still not be supplied and therefore data capture suffers. The users inadvertently fill these fields with nulls, spaces or zeros; or the application and/or DB2 software assigns these fields a default or null value. This leads to a disproportionate amount of nulls and such tablespaces benefit from compression.

Nulls can also be introduced in a warehouse as it is being populated from (multiple) legacy systems. As warehouse tables are built by joining data sources in these systems, null value could be assigned to the columns for missing key values in either system. DB2 utility DSN1COMP can be run on tablespaces or image data sets to give an accurate analysis for the DASD savings in case of compression implementation. We do not recommend implementing compression on tablespaces which do not yield at least 25 percent in space savings.

Figure 13 shows typical compression ratios from different types of tables.

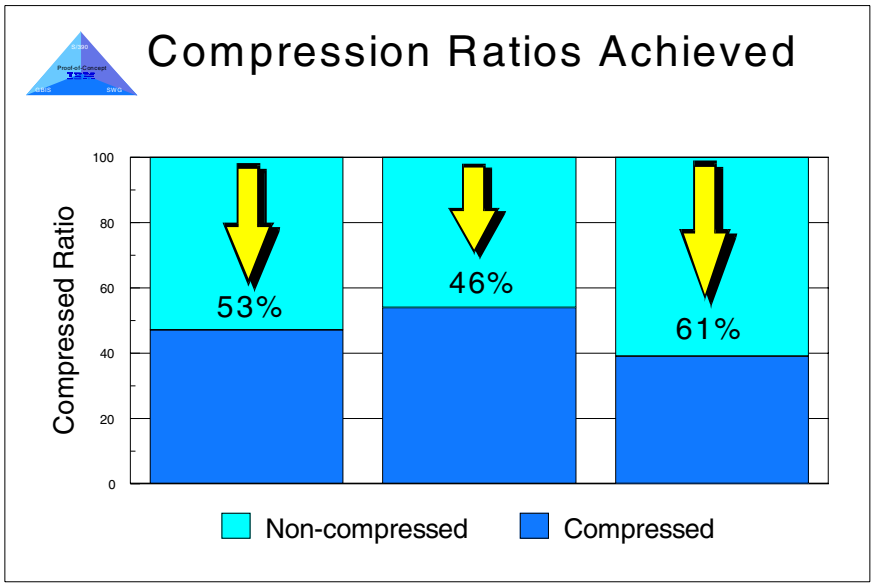


Figure 13. Typical compression savings

Figure 14 shows DASD savings in case study A.

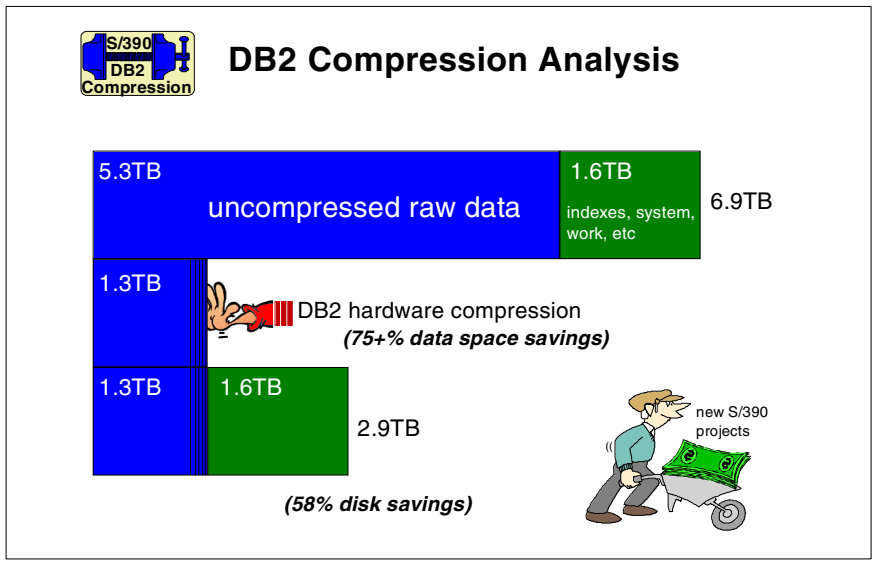


Figure 14. Case study A: compression savings

Improve buffer pool hit ratio: since DB2 reads data from buffer pools, with compressed data there are more data rows in storage and thus a greater probability of having a row in memory, increasing the hit ratio for the buffer pool. This results in reduced elapsed times for queries.

Compression effects on CPU and elapsed time: when data is compressed, data decoding uses additional CPU cycles for the queries. Thus, compression may increase the CPU cost of the workload. It impacts different types of queries differently.

The total CPU and elapsed time (ET) effects of compression vary based on the workload. Elapsed time and CPU time may decrease due to reduced pages to be brought into buffer pools, while CPU costs may go up due to the decoding of the compressed data.

Results will also vary depending on the types of accesses to the data. Index-only accesses incur no overhead, while indexed-data accesses might incur slight overheads. I/O-intensive table scans incur the most CPU overhead, yet they may also benefit dramatically in elapsed time reduction from increased efficiency of data accesses.

Compute-intensive queries with arithmetic functions incur relatively less overhead because the *compute* part of the query forms a large portion of the overall path length. Compression impacts mostly the I/O portion of the path length, which is minor in CPU-intensive queries by definition. Compression may also provide for more in-storage only accesses due to smaller data structures. With compression savings up to 50 percent, you can expect twice as many rows to fit into the buffer pool! You need to consider the overall effects on the workload versus potential DASD savings when evaluating the use of compression for the query environment. Refer to Figure 15 which shows the impact of compression on variety of workloads.

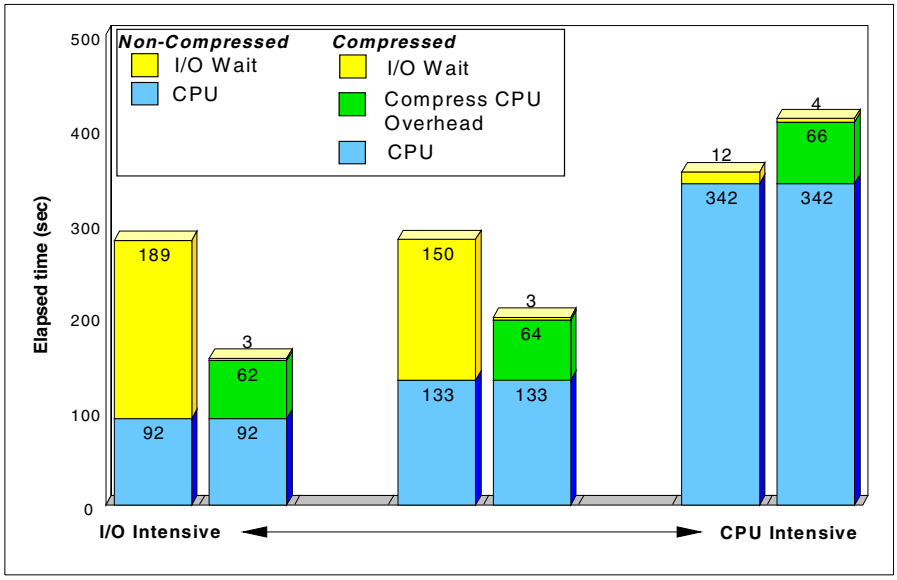


Figure 15. Effects of compression on elapsed time

RVA compresses all data, including indexes. DB2 compresses only data and not indexes. RVA implements compression within the RVA controller; data is compressed within RVA before being written to the DASD, and decompressed within the RVA controller after being read from DASD and passed to DB2. With RVA-only compression, the channel and buffer pool utilizations remain the same as within buffer pools where only uncompressed data resides.

Contrast this with DB2-implemented compression, where compressed data resides in the buffer pool, thus more data rows fit in a buffer pool; we can see an increase in the hit ratio for the buffer pool and increased I/O throughput as compressed data requires less I/O bandwidth.

To obtain the maximum benefits you should use DB2 compression with RVA implementation.

RVA architecture brings a whole new dimension to DASD planning due to its unique way of scattering data across disks and then assembling correct data when needed. For details on RVA compression, see *DB2 for OS/390 and Data Compression,* SG24-5261.

If you have the compression hardware, implementing data compression is by and large a good thing and may result in elapsed time savings in addition to DASD savings. The DSN1COMP utility can give you an estimate of savings. We decided to implement compression in our studies.

2.6 DSTATS statistics

The DB2 optimizer uses various kinds of statistics to create an efficient Query Execution Plan (QEP). It reads the statistics from many catalog tables. Statistics in these catalog tables are populated by the RUNSTATS utility.

By default the RUNSTATS utility will collect statistics like cardinality and the top 10 values by frequency, for columns which form the leading columns of the index. From DB2 Version 5 onwards, RUNSTATS can optionally collect both frequency and cardinality statistics for skewed combinations of columns, or correlated keys, and these are also held in SYSCOLDIST.

However, RUNSTATS collects statistics for only the leading columns of an index. The DSTATS utility is a supported program from IBM which will populate the catalog with statistics from any specified columns. Tests on a number of different customer databases suggest that this additional collection of statistics does in fact improve elapsed time for a significant proportion of queries, with response time improvements ranging from a few percentage points to factors of five or even ten in some cases; generally, these additional statistics seem to benefit the more complex queries.

DSTATS requires DB2 V5 or higher level. Currently this program is not part of the DB2 code. You can get DSTATS code from:

`ftp://www.redbooks.ibm.com/redbooks/dstats/statistics.zip`

or

`http://www.redbooks.ibm.com`

- Click **Additional Materials**
  - Click **dstats**

Then you can see the statistics.zip file, that is, DSTATS code.

The statistics.zip file will have the following four files:

- dstats.udb
- dstats.uld
- license.doc
- readme.txt

Non-uniform data value distributions in columns could cause less than optimal access path selections. The DB2 optimizer assumes a filter factor of (1/cardinality) for any particular value unless we have given the exception in the SYSCOLDIST catalog table. This does not bode well for columns with non-uniform or skewed distributions. Let us say a column called *Account Type* exists in an index and we have a cardinality of 50 for this column (bank having 50 products). If a query is looking to report on all accounts with Account Type = *Checking*, then DB2 chooses an indexed access based on its assumption of 100*(1/50) = 2 percent of the filter factor. However, in reality the Checking type accounts may be closer to 65 percent, for which a tablespace scan is a more efficient method. The DSTATS utility is a program which will collect statistics on any column the user specifies and populates the SYSCOLDIST catalog table. In the Banking and Health Insurance case studies, we ran this utility to collect statistics on various columns so the optimizer has correct distribution statistics to select an efficient access path. Figure 16 shows the improvements with the use of this DSTAT code.

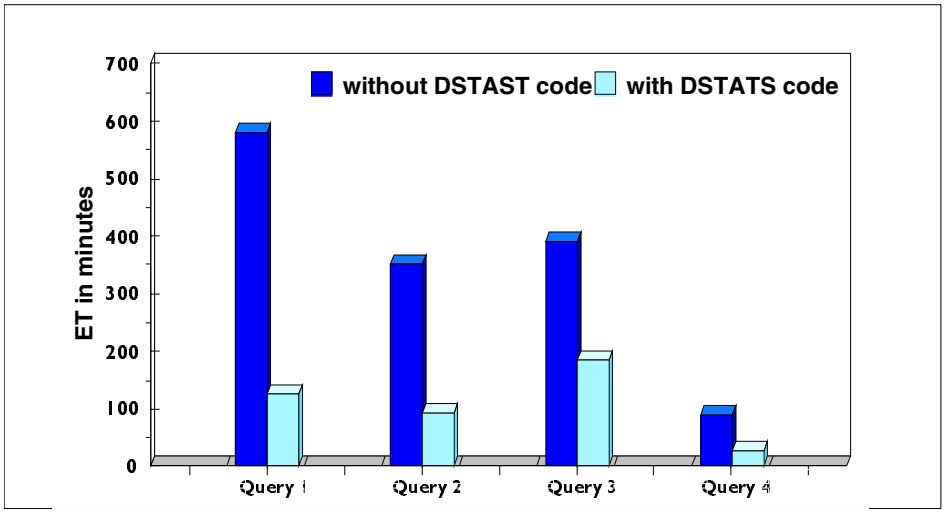


Figure 16. DSTATS measurements

When we have two columns involved in a search criteria, DB2 assumes both are independent and calculates the combined filter factor as (1/cardinality of column 1) * (1/cardinality of column 2). However, if the two columns are not independent, we are much better populating catalog tables with correlated distribution statistics. DSTATS also does this collection and population. RUNSTATS does the same, but only for leading columns of an index.

2.7 DB2 configuration

Following are recommendations for the DB2 configuration:

- Reserve some buffer pool space for synchronous reads, perhaps VPSEQT = 80. In a BI environment, a large amount of read is done by sequential prefetch. Thus, we allow the prefetched data much more buffer pool than in an OLTP environment.
- Allow 100 percent of the sequential prefetch buffer pool space for parallel tasks; set VPPSEQT = 100. Again, in a BI environment, we expect a lot of parallelism to process large volumes so we do not constrain parallel tasks from any memory.
- Buffer pool allocations: each parallel task needs 64 pages in the buffer pool. The suggested number of buffers are:

  ```
  64 * No. of parallel tasks * No. of concurrent tables being
  accessed within a query * No. of concurrent queries
  ```

  When buffer pool allocation is not adequate, DB2 reduces the degree of parallelism until all the parallel task data structures fit in the allocated pools.
- Assign hiperpools if expanded storage is available. Hiperpools are most useful for tables which are accessed using synchronous reads. Generally, they are less effective for the sort work buffer pool. However, as pages written to sort work migrate to DASD, they then become eligible for hiperpool. During the merge phase, these pages can then be read from a hiperpool rather than DASD saving the read from DASD.
- Aim for a hiperpool read-to-write ratio of better than 1 to 10.

The following pool assignments are suggested to facilitate performance and analysis. While multiple bufferpools are preferred, the size of each pool must be sufficiently large to support the level of concurrency and parallelism desired. The combined total of all the buffer pool is backed by sufficient virtual and real memory.

- BP0 - DB2 system catalog and directory
- BP1 - Large tables
- BP2 - Large indexes
- BP3 - Other non-pinnable tables
- BP4 - Other non-pinnable indexes
- BP5 - Pinnable tables
- BP6 - Pinnable indexes
- BP7 - Sort work

Be aware of database management system boundaries and design your configuration accordingly. For example, in DB2 V5:

- The maximum single table size is 1 TB.
- The maximum number of partitions is 254.
- The maximum number of pieces for a non-partitioned index is 128.
- The size of each non-partitioned index piece ranges from 256 KB to 4 GB. See *DB2 for OS/390 Version 5 SQL Reference*, SC26-8966, for a detailed explanation.

In DB2 Version 6:

- The maximum single table size is 16 TB.
- The maximum number of pieces for a non-partitioned index is 254.
- Size of each non-partitioned index piece ranges from 254 KB to 64 GB. See *DB2 UDB for OS/390 Version 6 SQL Reference*, SC26-9014, for a detailed explanation.

2.8 DASD storage management using DFSMS and RVA

There are two ways to manage storage: user-managed and system-managed. In user-managed storage, a user has to decide about placement of each data set. With partitioning, DB2 uses a separate VSAM data set to store each partition and index. SMS also implements the concept of Storage Group somewhat similar to DB2 STOGROUPS. Within SMS, a group of volumes can be assigned to a storage group. Subsequently, allocation of data sets on these volumes can be done dynamically based on various predefined criteria, such as high level qualifier (HLQ), or performance needs. All of the references to storage groups pertain to SMS groups within this section. We highlight here the SMS advantages within the VLDB environment.

It is clear that with large databases, you can easily get into thousands of data sets. Manually placing these thousands of data sets on each volume and also

ensuring the index and the data part utilize separate paths but not end up behind the same physical volume can be quite a cumbersome exercise. SMS allows easy and automatic placement of these data sets once the storage classes are defined. SMS does a good job of spreading data across multiple controllers and volumes.

SMS can allocate data sets to a volume based on pre-established policies. We use high level qualifiers of data set names as the basis for allocation within SMS groups. Thus, you will notice there is an SMS group for each category of DB2 objects. Each SMS group has granularity of space allocation. Table 2 shows the SMS allocation for case study A.

Table 2. No. of volumes allocated in case study A

SMS Group	Description	# of Addresses
A_RVA	DB2 tables and primary index	3,600
A_NPI	DB2 NPIs	320
A_WRK	DB2 workfiles	480
A_TEMP	Temporary data sets (DFSORT.)	360
A_RES	Reserved	360
Total		5,120

The Reserve Storage Group can be created to keep a set of logical volumes completely clean. If any of the other storage groups grow unexpectedly, we could easily move clean logical volumes into it. Additionally, due to the high address ability of the RVA, reserving 360 logical volumes does *not* signify you have to reserve 360*2.8 GB (maximum space on a 3390-3 volume). It is only the logical addresses that are being reserved; physical space is not reserved until required.

Following are the DASD implementation goals for a data warehouse; using SMS provides a simple mechanism to:

- Spread partitions for given table/PI
- Spread partitions of tables/indexes likely to be joined
- Spread PIECES of NPIs
- Spread DB2 work files
- Spread temporary data sets likely to be accessed in parallel
- Avoid logical address (UCB) contention

The following RVA characteristics facilitate a simple SMS implementation:

- Physical striping of data leads to less contention at the physical device.
- 256 logical addresses per subsystem leads to fewer data sets per logical volume (address) and thus, less UCB contention.
- Only what is used is needed; there is no overallocation wasting space.

We use the following guidelines when working with controllers having cache memory (RVA has cache in the controller):

- SEQCACHE=SEQ in DSNZPARM: this determines that DB2 will use cache in the DASD controller. Its default is set to bypass, to prevent the small cache in the non-RVA (traditional) controller from being overrun by DB2 prefetch and large scan activity. However, in RVA this cache tends to be of a large size; thus, BI workloads can exploit and achieve high cache hit ratios.
- DB2 Vertical Deferred Write Threshold (VDWQT): this is the percentage of virtual pool buffers that can be used for update activity by a single data set. The default is 10 percent. This may produce high bursts of DB2 write activity. This must be reduced to a small number, say one percent, to reduce contention at the RVA. This distributes the write activity uniformly and does not swamp the RVA cache, thus improving its read hit cache ratio.
- Caching: make sure that caching is enabled for all DB2 data sets at the controller. This can be done at DFSMS, DFP, data set, or volume levels.

See *Using RVA and SnapShot for Business Intelligence Applications with OS/390 and DB2*, SG24-5333, for an understanding of the workings of RVA DASD.

The direction of storage technology is towards reducing logical addressing contention and implementing physical striping of the data across large numbers of controllers. SMS storage groups facilitate the management of a large number of volumes. In the future there will be no IOS queuing, thus reducing the wait for an I/O request. The RVA technology allows us effective physical striping of data along with good compaction and compression. RVA also allows for fewer data sets per logical volume address, effectively reducing the UCB addressing contention.

2.8.1 Work file data sets

The number and size of work file data sets is workload-dependent. The number can be roughly estimated by taking the number of expected concurrent parallel tasks in the system at any one given time and dividing by

five. For example, for two large concurrent sorts with a database partitioned 250 ways, start with 100 work file data sets:

250 parts * 2 sorts / 5 = 100

The larger the work file data sets (up to a full volume), the better. This was only a starting point which we need to be prepared to vary as the workload increased.

2.9 Case study tests

So far we have described general technical issues for database design in VLDB environments for S/390 BI applications. In the following sections we describe some pertinent decisions in building BI data warehouses for the Banking and Health Insurance industry at the Teraplex Center.

2.10 Case study A: physical database design

The design considerations we document here are what we focused on in case study A, based on the customer requirements. These are not unique to the Banking industry and are widely applicable.

One objective of case study A is to demonstrate that with DB2, one can build a terabyte size-database and still be in third normal form (3NF). In the industry, sometimes a portion of EDW may be denormalized to meet certain query objectives. The database design for case study A is done in third normal form, which demonstrates that one can easily build large terabyte-sized data warehouses in 3NF and meet user expectations.

The main objective of converting a logical design to a physical design is to optimize the usage of the underlying hardware and database engine features. A logical model is a good documentation of the user view of data and its relationships. The next step is the physical design of the database. We determined the following:

- Any deviations from the logical model
- Which tables should be partitioned
- The number of partitions of each table
- The physical partitioning keys
- If tables should be denormalized to optimize query performance
- Placement of tables and indexes on DASD
- Indexing strategy

In the following section, we describe the abstract physical model of the database as well as the thought process behind the design decisions.

2.10.1 Deviations from the logical model

Each table in the logical model represents multiple years of data. With the large size of some tables in time-based partitioning, we might have run out of space. In order to add new data, we may require partitioning boundary changes. This necessitated having to drop and recreate the tables (because we used DB2 V5 for the Banking case study) which opened up some operational issues:

- Rebuilding and loading a large table could take a significant amount of time, especially for a table with many secondary indexes. During the rebuild, the table would not be available to the users.
- All the programs using this table need to be rebound. Although this restriction applies to static SQLs only, not all the data warehouse queries use dynamic SQL.
- Access control information (GRANTs) to this table is lost. DBAs have to rebuild this information with the new table.

We decided to use date-range partitioning for operational ease. The physical model specifies each table as storing one year of data. For example, there is a 1998 Household table, a 1999 Household table, and so on. Since the customer was interested in storing three years of history only, they could drop the 1998 tables in the year 2002 and reclaim the space for the year 2002 data.

A query accessing data crossing year boundaries (for example, between July 1998 and March 1999) would require writing an SQL UNION query accessing multiple tables. This observation does not present a problem to the customer since most of their queries access the current month of data. As a result, very few of their queries in their production system were modified.

2.10.2 Partitioning considerations

In this case, how to partition is an easy decision. All tables are partitioned to get the maximum benefit of parallelism. It is a requirement of DB2 V5 that the outer table of a join must be partitioned in order for a query to run in parallel. Partitioning should be performed even for small tables. This restriction is removed in DB2 V6, but the Banking study was performed using DB2 V5.

2.10.2.1 Number of partitions

In order to achieve the highest degree of parallelism for queries, each table is divided into 240 partitions. This yields 20 partitions for each month of data. Although it is possible to use 21 partitions instead, the decision was made to use a round number of 20 partitions for each month of data. This enables up to 20 degrees of parallelism for queries which access one month of data.

2.10.2.2 Partitioning keys

The partitioning keys are basically identical to what was depicted in the logical model. The tables are partitioned by one of two sets of keys. One set of keys has the following structure:

- extract_date
- account product category code
- company number
- account number
- system ID

The second set of keys uses the following columns:

- extract date
- state code
- ID - different IDs for different tables

It is important to have a predicate of the form tableA.extract_date= tableB.extract_date in every query that performs join processing. A query would probably produce incorrect business results if this predicate is missing. For example, it does not make business sense to join the May ATM activities with April accounts.

There is a concern that most queries would not experience high degrees of parallelism with this set of partitioning keys. These sets of keys were built based on key range partitioning, and data was clustered according to their contents. Florida data, for example, would be stored in a few partitions only, as opposed to being spread across all 20 partitions (remember the second column of some partitioning keys is postal_state_code). A query requesting Florida data would not exhibit a high degree of parallelism.

A common technique is to use random keys, similar to the hash functions employed in other database engines. This would distribute Florida data across all 20 partitions, which would make it more likely to generate up to 20 parallel tasks for queries.

It is not difficult to come up with a function to randomize the data. For the first set of partitioning keys, the function would make the account number the second column of the key:

- extract_date
- account number
- account product category code
- company number
- system ID

Account numbers are assigned randomly to the bank's customers. This function would randomize the distribution of data.

For the second set of keys, moving the ID column to the second position also distributes the data:

- extract_date
- id
- state_code

Again, the IDs are assigned randomly by households, accounts, and so forth.

Since the case study was not focused on single query measurements, but rather on high levels of concurrency, using random keys to improve single query measurement results was not a strong motivation factor.

2.10.3 Denormalization

In certain circumstances query performance could be improved when certain tables are denormalized. In particular, the Household and the Account tables are joined frequently. Denormalizing these two tables into a single table would speed up certain queries. The customer considered using denormalization in these tables in their production system; however, such a move was not adopted at the time of proof of concept because the change would make the physical design substantially different from what is used for the customer's production system.

2.10.4 Data set placement

We decided to use DFSMS for data set placement as there were over 5000 data sets to place. Refer to 2.8, “DASD storage management using DFSMS and RVA” on page 41, for data set placement strategy. Although we used RVA DASD for this, DFSMS can manage RVA and other DASD technologies (refer to the Health Insurance case for more information).

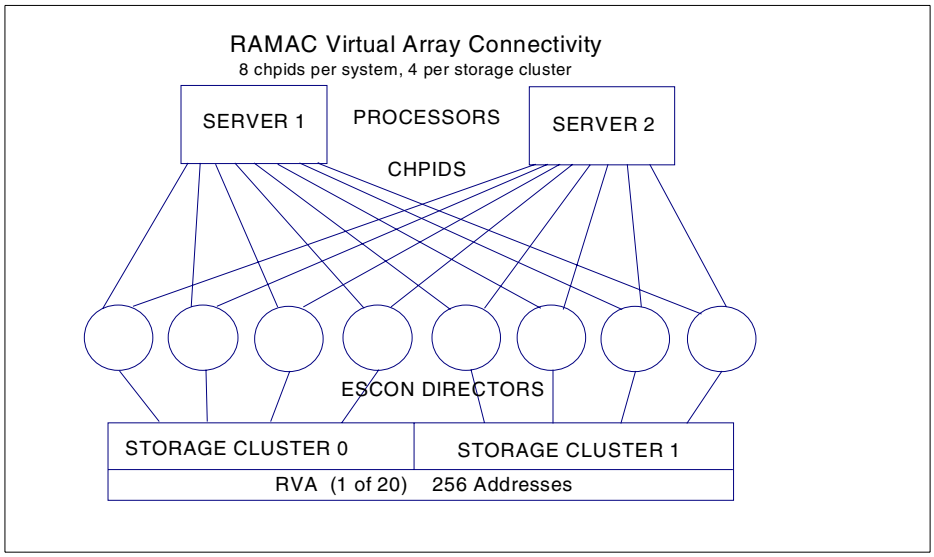


Figure 17. DASD configuration of case study A- Banking

The following SMS implementation was also defined (see Figure 18 on page 49):

- Pool1 for partitioned index spaces and data tablespaces: 3600 logical addresses.
- Pool2 for secondary indexes: 320 logical addresses. Pool3 for DB2 work areas DSNDB07: 480 logical addresses.
- Pool4 for other temporary work areas (for example, DFSORT): 360 logical addresses.
- Pool5 for reserve volumes in case of under allocation for pool: 360 logical addresses.
- All logical volumes are striped across 20 RVA subsystems.
- Simple ACS routines for data set placement:
  - HLQ are used to route to specific SMS groups.
  - SRM randomly selects from all volumes in the group.
- Total NPI space required approx. 100 GB with 750 PIECES.
- Total workspace required is 300 GB (300 1 GB workfiles).
- For temporary space we used a rough ROT of 10 percent of tables.
- RESERVE keeps a subset of volsers clean, which are movable to another SMS group. When a new volume needs to be put in a group, for example

to load a new partition, it can be easily done. If no clean volumes are available, we will have to first move data from an existing volume to another volume before adding it to an SMS group.

The SMS Storage Pool allocation worked well for case study A.

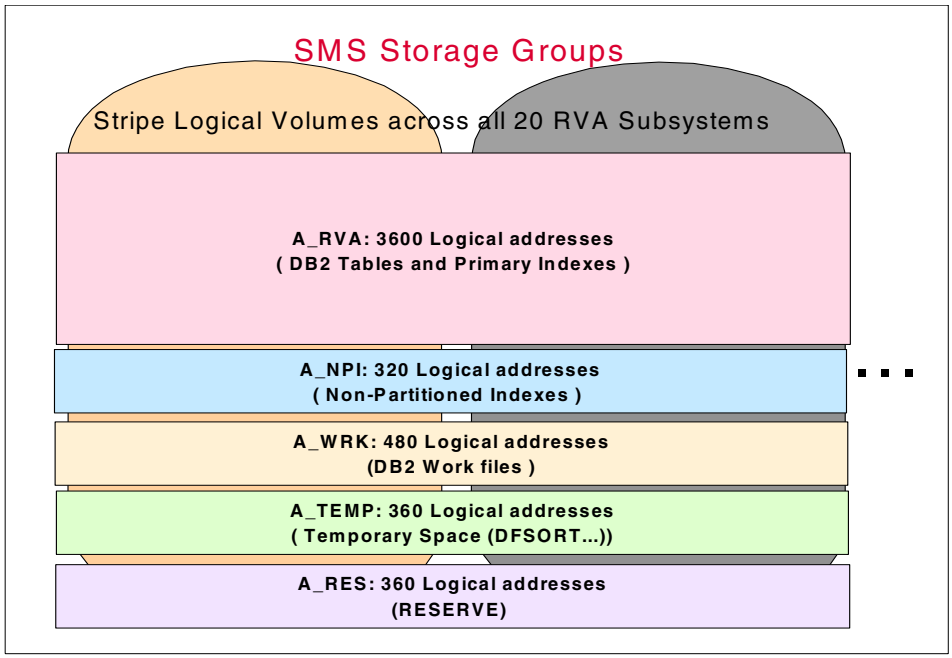


Figure 18. SMS definitions of case study A - Banking

2.10.5 Index strategy

The customer built 28 indexes in their production system for the 11 tables used in the case study. We built only the indexes used by the queries, after analysis of the key queries.

During the mini-database evaluation, the access path of each query was analyzed thoroughly. In addition to the secondary indexes defined by the customer, additional indexes were examined for their effects on query performance. At the end of the evaluation, it was believed the queries were using optimal access paths, and only secondary indexes used by these access paths would be built for the case study test system. The number of required secondary indexes therefore dropped from 28 to 8, per year of data. Thus, we built 24 NPIs for three years of data. The load time of 48 hours reported earlier includes building these 24 secondary indexes (NPIs).

2.10.6 Non-partitioned indexes

Secondary indexes are to be created as necessary. They can be built during load or can be deferred to reduce the data load elapsed time. Based upon past experience, we opted to build indexes after the initial data load. For detailed information, refer to 2.4, “Index considerations” on page 32.

2.10.7 Buffer pool allocations

The following table shows the bufferpools we allocated for DB2.

Table 3. Buffer pool allocations

BPOOL	Description	VPSIZE	HPSIZE
BP0	System	1,000	0
BP1	Large tablespace	60,000	120,000
BP2	Large index	60,000	120,000
BP3	Other non-pinnable tablespace	40,000	80,000
BP4	Other non-pinnable index	40,000	80,000
BP5	Pinnable tablespace	3,000	10,000
BP6	Pinnable index	3,000	10,000
BP7	Sort	100,000	200,000

2.11 Case study B: physical DB design

The design considerations we document here are what we focused on, based on the customer requirements. These are not germane to the Health Insurance industry in particular, and thereby have wide validity to other industries.

The development of the Health Insurance database considers both the data and technical constraints, since it involves a large volume of data. The data for this proof of concept was 500 gigabytes in size, and came in 400+ tape volumes. The raw data size of the data warehouse at the customer site was 1.5+ TB. The customer’s primary objective was to structure the Health Insurance database in the most efficient manner for data access, update and maintenance, and for backup and recovery. DB2 Version 5 was used in the Health Insurance case study.

The database design objectives for this study were:

- Establishment of an E-R data model for the Health Insurance study

- Efficient data access from both online and batch
- Data retrieval in a timely fashion
- Multitasking data access via query parallelism
- Ease of database maintenance
- Ease of backup and recovery
- Flexibility and extensibility in meeting changing business needs
- Being able to do a monthly refresh of EDW in a weekend window

2.11.1 Database design considerations

In order to assist in the development of an efficient database design, the team studied a representative set of the end-user business questions, the atomic-level data elements that are needed to answer the business questions, the business rules applied to each data element, the relative frequency of each business question, and the user interface requirements. Once this information was gathered, the logical data model was translated to a physical database design optimized for a DB2 relational database environment. Some of the tasks performed in transforming the logical data model to a physical database design optimized for DB2 were the following:

- Map business questions to data element mapping and rules.
- Source data volume and size database.
- Group logical and physical data elements.
- Normalize/denormalize the data.
- Obtain data access requirements.
- Identify unique record identifiers for each claim to avoid large composite keys.
- Minimize the need for non-partitioned indexes.
- When a non-partitioned index is necessary, use PIECESIZE to avoid I/O contention.
- Use data compression to reduce DASD usage.
- Build reference tables to support specific data access requirements.
- Partition data for manageability and performance.
- Assign specific DB2 buffer pools to improve access performance.
- Implement summary tables to reduce detail claim process.
- Implement large work files (DSNDB07).

2.11.2 Physical database design considerations

The Health Insurance logical model was translated into a physical design to support the selected business questions, optimized for DB2 on OS/390. The physical database design consists of a data model diagram showing tables, columns, data types, lengths, and join paths. Initially, the primary keys were identified.

The logical model was denormalized such that entities like Claim_Common and Claim_Institutional were combined with each institutional claim, for example, Claim_Inpatient_SNF, to form one large table. Similarly, Claim_Common and Claim_Non_Institutional are combined with each non-institutional claim, for example, Claim_Physician_Supplier. Therefore, each of the six claim types is represented by a single table consisting of common data, institutional or noninstitutional data, and data specific to each claim type. Due to this denormalization process, the Health Insurance physical database design model is comprised of only six major claim tables, which are as follows:

- CLAIM_INPATIENT_SNF
- CLAIM_OUTPATIENT
- CLAIM_HHA
- CLAIM_HOSPICE
- CLAIM_PHYSICIAN_SUPPLIER
- CLAIM_DMERC

In addition to the six major tables (one for each claim type), 14 minor tables have been created. Each minor table provides details for the major claim tables.

Figure 19 on page 53 shows the subset of the physical design for case study B.

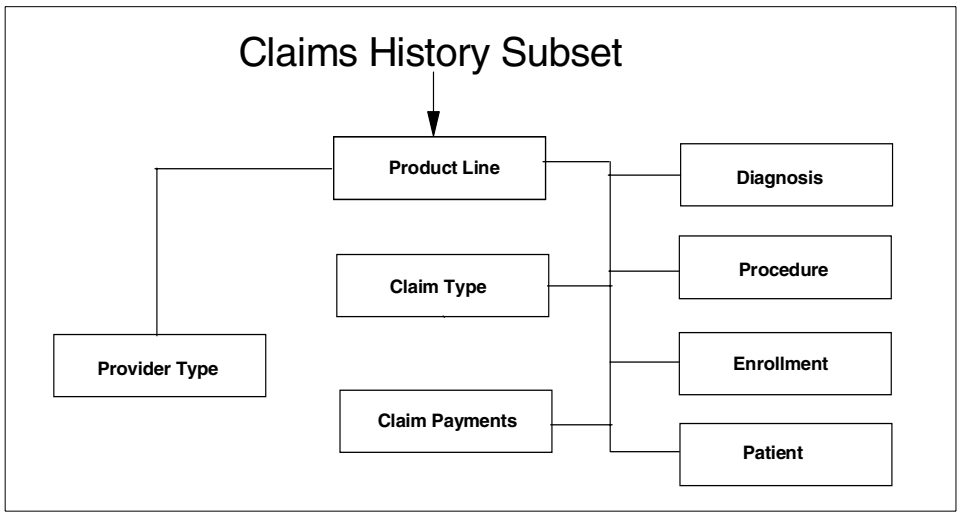


Figure 19. Subset of the physical DB design for case study B - Health Industry

2.11.3 Partitioning

In the case study, each medical claim record is assigned a unique record identifier. This unique claim record identifier (Claim_UID) is used for partitioning and for joining related tables in the Health Insurance database. Claim IDs are grouped into six claim types. Data is further grouped into 10 groups, to create 10 process streams to cleanse and do transformation in parallel. Claim_UID are closely correlated to chronological sequence. We partitioned this data 65 ways to allow for 10 years of history.

Specifically, the unique Claim_UID is a sequential number assigned to each claim. This Claim_UID will be used as the partition key. The advantages of partitioning tables by Claim_UID are:

- Each partition can be processed independently thereby increasing the depth of parallelism in data access and other activities.
- A simple partitioned key will be used to join all related claim tables instead of using large composite keys.
- There is a simplified archival procedure by Claim_UID using the Unload utility. We use customer selected utilities to mimic their environment.
- There is low table maintenance.
- Table backup and recovery is easier.

2.11.4 Physical DB2 objects placement

To evenly distribute the tables and indexes across the I/O subsystem, a mix of DB2 STOGROUPS or SMS-managed data sets is used. We used DB2 storage groups to avoid the need of creating VSAM data sets explicitly. We then used SMS to manage the DB2 storage groups. We used a mix of RAMAC and RVA DASD for the Health Insurance test that was similar to what we used for the Banking test.

We used multiple Storage Groups, backed by horizontal striping of volumes across all controllers. Simple SMS routines utilizing data set HLQs were defined. Refer to 2.8, “DASD storage management using DFSMS and RVA” on page 41, for data set placement strategy.

Table 4. No. of volumes allocated in Health Insurance case study

SMS Group	Description	No. of Addresses
M_ST1	Striped SMS Group for striped data set performance	57
M_ST2	Striped SMS Group for striped data set performance	49
M_DB2	DB2 tables, PIs, NPIs	464
M_USER	Intermediate DB2 tables, query answer sets, end-user tables	174
M_WORK	DB2 workfiles	72
M_FLT	Flat files for refresh test	60
M_TEMP	Temporary data sets for DFSORT	134
A_RES	Reserved	20
Total		1,030

2.11.5 Buffer pool allocation

During the query performance tests, we aimed for a very high level of concurrency. Our buffer pool allocations are shown in Table 5 on page 55.

Table 5. Buffer pool allocations

BPOOL	Description	VPSIZE	HPSIZE
BP0	System	1,000	0
BP1	Large tablespace	30,000	150,000
BP2	Large index	30,000	150,000
BP3	Other non-pinnable tablespace	20,000	100,000
BP4	Other non-pinnable index	20,000	100,000
BP5	Pinnable tablespace	3,000	10,000
BP6	Pinnable index	3,000	10,000
BP7	Sort	50,000	250,000

2.12 Summary

The following conclusions from the two tests at the Teraplex Center can be drawn in regard to physical design for the EDW:

- Partitioning is very important in achieving operational goals in VLDB environments. With effective partitioning, we can enable the EDW to be available for 24 x 365.
- Parallelizing in all aspects of EDW must be exploited, from loading the EDW to building index jobs.
- SMS Storage Groups reduce the complexity of placement of data dramatically. These are highly recommended in any sizable environment.
- A third normal form (3NF)-based design goal is very easily achievable.
- Summary Tables can save a lot of processing time if you know the nature of queries beforehand. Thus, end users should be solicited to help explore possible summary tables to be built and maintained.
- Hiperpools are very effective in reducing synchronous I/O.
- We should use the SORTKEYS clause in the LOAD utility to pass sorted keys in memory. This clause is especially useful when data is not in sorted order and we have more than one index in the table.
- A balanced system with matching I/O and CPU capacities is desirable; that is, the I/O subsystem generates just enough data to saturate the processor, thus there is no underutilization of any hardware capacity.

- You should seriously consider implementing compression within DB2. Use RVA if possible.
- The DSTATS utility should be used to collect statistics for columns with possibly skewed data distribution.
- Allocating most of the buffer pool for parallel tasks benefits BI queries. Assign hiperpools if expanded storage is available.

Chapter 3. Data warehouse population

One of the biggest challenges in building an enterprise data warehouse (EDW) is data population. When performing the initial database load with very large volumes of dirty data, you may face more problems than you expected. Data population is the most complex data warehouse building block. This building block includes extracting data from an operational system, mapping, cleansing, and transforming it into a form suitable for a new Business Intelligence (BI) environment and loading it into a data warehouse.

To successfully populate a data warehouse, the architectural solution is important. It is necessary to be familiar with Online Transaction Processing systems (OLTP), data warehouse, and business needs in order to understand the degree of complexity of transformation and the potential impacts of the volume. This is particularly important when the data warehouse is classified as a Very Large Database (VLDB).

The main design considerations for data warehouse population are:

- Identifying critical success factors
- Managing risk
- Driving application parallelism

These considerations apply to both the initial load and regularly scheduled data refresh. Even though data capture, cleansing, and transformation are company-specific, depending on the tools used and the business rules, we use the term *transformation* to include the processes of cleansing, mapping, and transforming.

3.1 Architectural considerations

This section discusses the architectural design considerations for populating the data warehouse.

3.1.1 Initial load

To define the design on the architectural level we examine a sequential solution and a parallel solution, using a Health Insurance company as an example.

3.1.1.1 Sequential solution

Figure 20 shows the sequential design of the end-to-end data flow up to the load processing, which is parallel.

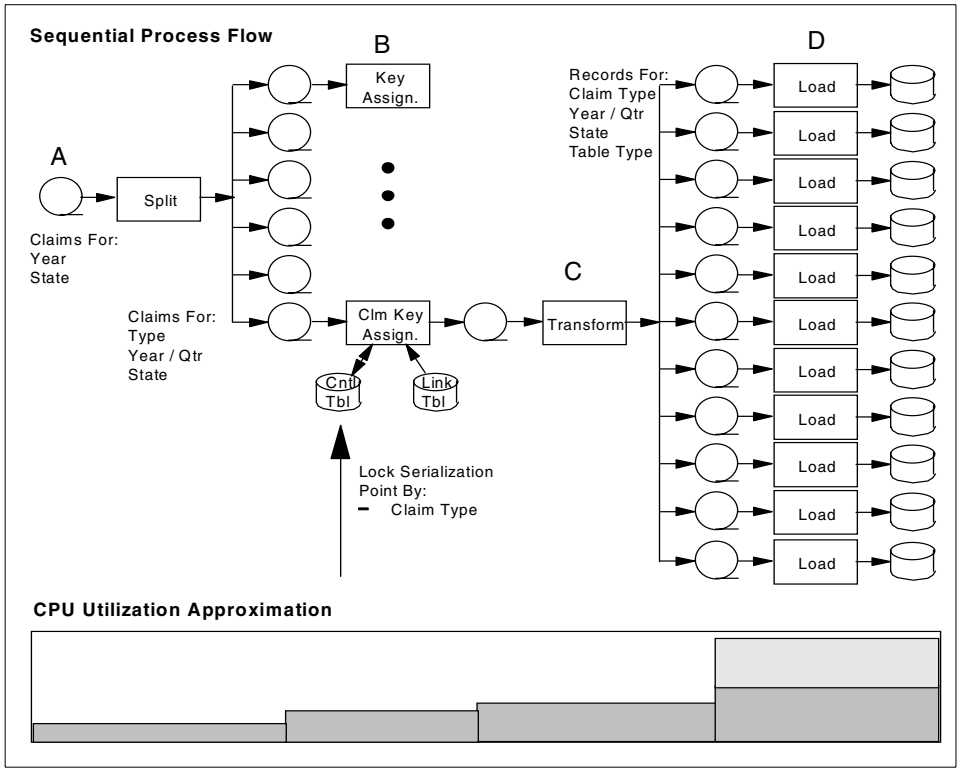


Figure 20. The sequential process flow

The input file (**A**) is split by claim type, which is then updated to reflect the generated identifier (**B**), such as claim ID or individual ID. This data is sent to

the transformation process (**C**), which splits the data for the Load utilities (**D**). Partition boundaries are managed by judicious use of Load Replace Partition Number and Load Resume.

This process must be repeated for every state, each year, for a total of several hundred times. Given the tape bandwidth constraints, and the limited DASD beyond the space allocated for DB2 objects, this results in a very long process.

There are some issues for completing the transformation and load in a timely manner:

- With the high degree of variability in input file volumes, predicting when certain processes will complete is very difficult.
- The lock serialization will prevent two streams from being in the key assignment process concurrently; see the large pointer in Figure 20 on page 58.
- Managing the Load Resume versus Load Replace specification is a manual process.
- Some installations may have limited tape throughput capability.
- If the installation is stressed with DASD capacity, much of the intermediate data will need to be put on tape cartridges, and throughput limitations are magnified by the number of intermediate files.

We found that this model was not suitable for transforming very large data volumes because:

- The model does not support efficient usage of CPU and DASD, that is, parallelism is not supported.
- The elapsed execution time is probably too long.

Some comments regarding this design:

- Media failures - Based on the number of input cartridges that must be read, some media failures can be expected if the volume is large enough. From our experience, between 0.1 and 0.2 percent of inputted cartridges fail. So if the inputted cartridge volume is lower than 500, the possibility for media failure is zero. But with a volume of 4000 cartridges, we might have two to four failures. This design minimizes the risk associated with media failure because nothing that has not been created locally is loaded to tables.

- Every source record is read or written many times before being stored in a table. If the source data is in the terabyte range, this process will require several days to complete through a single controller.
- A large number of scratch tapes is required to complete processing for a single input file.

When the source volume quantity passes multi-terabyte limits, the need to parallelize all processing and minimize I/O handling is acute.

3.1.1.2 Parallel solution

There are several opportunities to reduce the I/O impact and streamline the process, and to simplify the transformation and load process, by using SmartBatch to achieve a parallel solution.

The *high level* design shown in Figure 21 shows how the data flows from the initial read through the load. The objective of this design is to minimize the amount of data that must be externalized through I/O controllers.

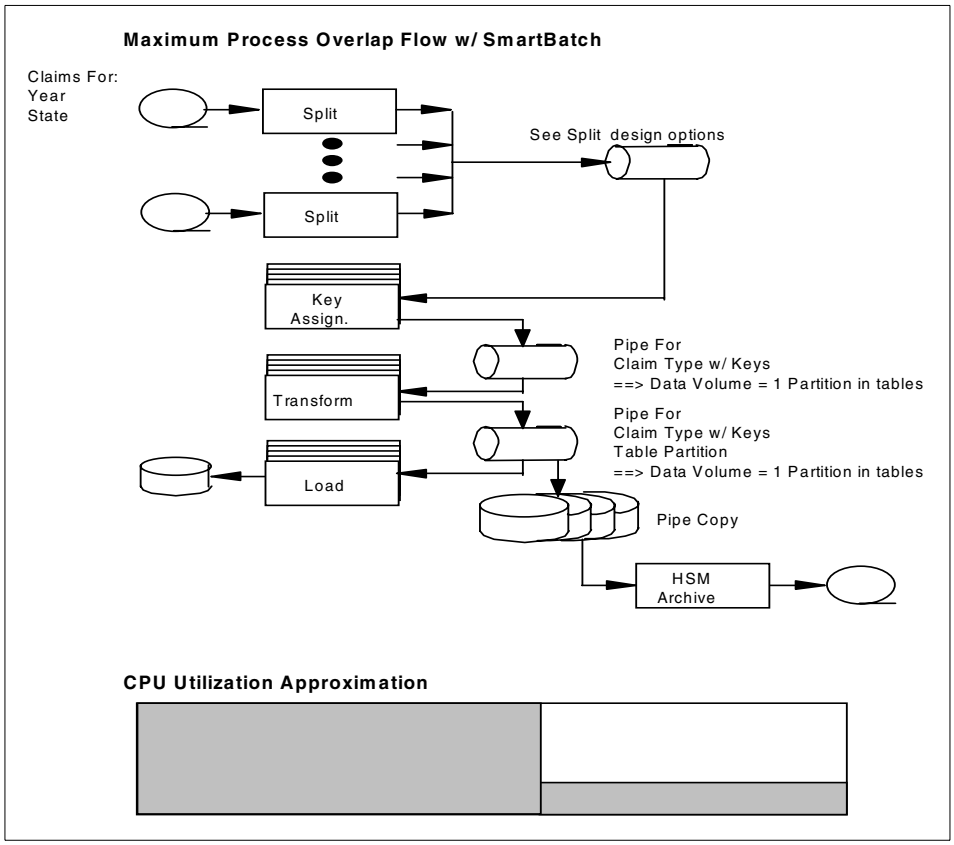


Figure 21. Parallel solution to load a data warehouse

The basic idea behind this design is that enough input files are read in parallel so that the data defines a partition in the resulting tables. Additionally, enough processing is occurring concurrently to consume an entire server. This has the effect of gating the throughput based upon processor capacity, thereby relieving the tape control unit constraints in the process. This type of solution is called the piped solution (see Appendix D on page 251).

3.1.1.3 Split design options

There are a number of ways to design the initial split process. Using sort is one method, and using a piped solution is another. The two options depend upon the nature of the records.

- Single record: the repeating groups are part of a single variable-length record. This is typically thought of in COBOL as OCCURS DEPENDING ON group items. This format allows record level processing to be used throughout the process, and opens more options.
- Multiple records: the data for a single logical record (for example claim data) consumes multiple records, using record types to delineate core logical data from repeating group data.

Figure 22 shows that the format of the input file will determine which of the following are viable options.

If claims are represented as a single record, then a single pipe is required. Multiple readers are started for high volume claim types (for example carrier), with the number of readers set to the number of high volume claim type partitions for each partition for other claim types.

If multiple records contain the data for a single claim, then care must be taken to ensure that they reside on the same pipe and are not interleaved with other data. The merge function is used to accomplish the latter.

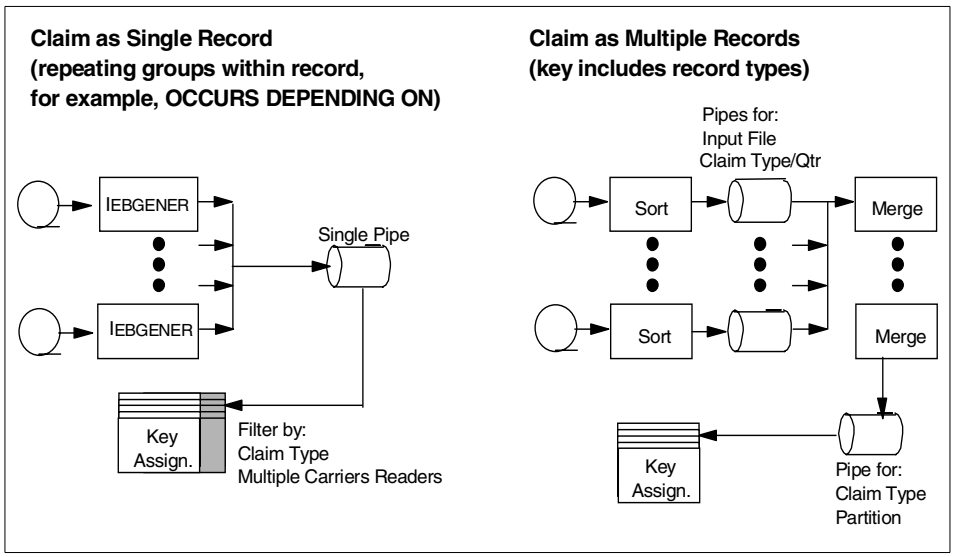


Figure 22. Pipes for multiple records design

Figure 22 on page 62 simplifies (for readability) the number of pipes required in the Multiple Record design. In actuality, there is a pipe for each input file by claim type feeding a merge process. The merge creates one pipe for each resulting claim type/partition.

How will the transformation process, which takes most of the end-to-end time, fit into the batch window? The batch window is always limited. The transformation process is part of the initial load and refresh.

3.1.1.4 Transformation

The transformation process shown in Figure 22 on page 62 can be a hand-crafted COBOL program or it can be code generated by a population tool like Visual Warehouse, ETI, Vality, or DataStage Suite.

The transformation contains the following types of processes:

- Data model mapping between legacy OLTP data models and BI data models
- Decoding encoded values
- Data consistency checking
- Identifying additional data relationships
- Representing missing data in the data warehouse

The BI data model and quality of the source data decide the scope and use of the transformation processes.

3.1.2 Refresh of data warehouse

In a data warehouse environment, refreshes and maintenance must occur frequently, but the availability of the data must remain as high as possible. For this reason, it is very important to have a solution where data refreshes can occur concurrently with queries, minimizing the need to shut down the data warehouse environment for end users.

Data currency is important to the business and the data refresh is the tool to realize it. Mission-critical queries often require current and frequently refreshed data. However, the business needs determine when and how often the data warehouse should be refreshed to accomplish the company's business. In some cases, high availability is the primary goal, while throughput might be a priority in other scenarios.

On the architectural level data refresh is similar to the initial load, especially when the volume of refresh is very large. Differences are on the technical

level, that is, how to partition the data, what type of parallelism is used, and the size of the batch window. The initial load as shown in Figure 21 on page 61 is usually executed once, but the data refresh executes frequently and is dependent on available window.

The objectives for designing a solution for a data refresh are:

- The design must be scalable. The database downtime for updating a data warehouse is often limited to short windows, often weekends.
- The solution must maximize the use of parallelism.
- The number of jobs and files for the data transformation data flow must be minimized.

Figure 23 on page 65 shows the data flow for refresh for our health insurance example. The input (**A**) file is split by claim type, which is then sorted (**B**). The data is processed for final Action (**C**) and then updated to reflect the generated identification (**D**). This data is sent to the transformation process (**E**), which creates the data for the Load utilities (**F**). Partition boundaries are managed by judicious use of Load Replace Partition Number and Load Resume.

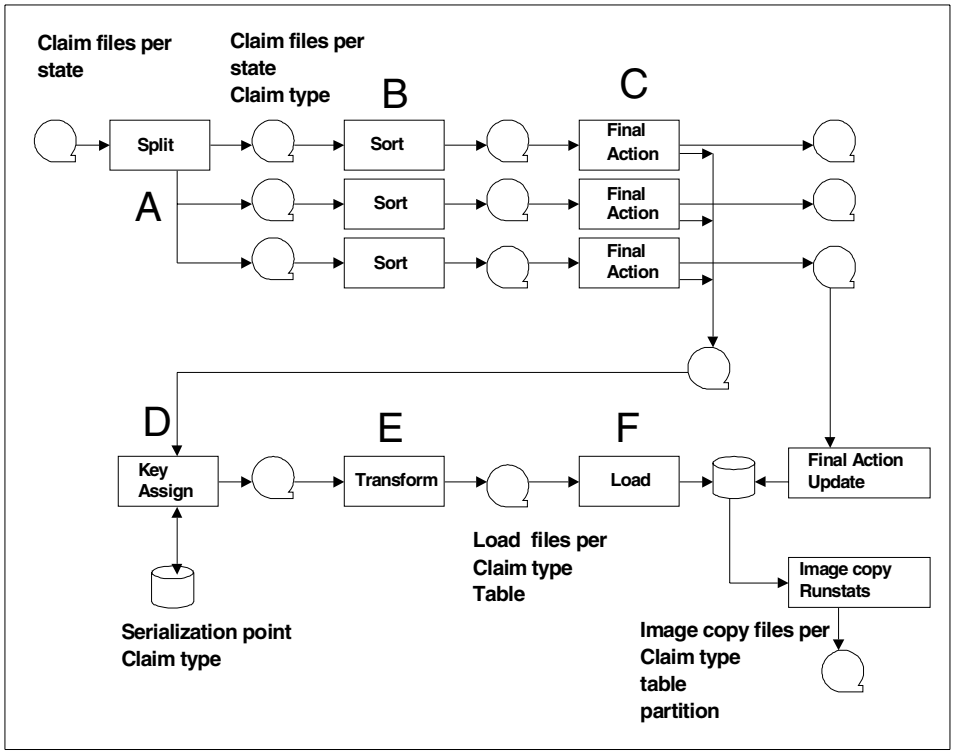


Figure 23. Refresh of data warehouse

In this solution, CPU parallelism can be achieved by running Sort and Final Action in parallel. The parallelism will be limited for tape read processes by the number of tape units. The split must be executed for each source file.

A load balance analysis is very important before final design is done. In our health insurance example, several sorts were used for the high volume claim type, and the results were then merged. This enabled all sorts to complete in a similar time frame.

3.1.3 Critical success factors (CSF)

To succeed with data population, CPU and I/O parallelism are critical. Without parallelism, the probability of completing the population in a reasonable time is small. The fragmented input data comes from many internal and external sources and varies in format and quality. Additionally, when the data source contains large volumes of data, it is often fragmented to optimize maintenance processing. The partitioning strategy for the data warehouse is

optimized for retrieval processing, usually resulting in a requirement to realign the data across different fragmentations to support load.

In many cases the batch window is narrow and the selected design may contain risks.

The critical success factors for data population are:

- I/O optimization
- Fragment realignment
- Risk management
- Batch window (refresh)

In this chapter, we will confine the discussion to risk management.

3.1.4 Risk management

Each design alternative should be evaluated from a risk perspective, and no solution is without some risks.

From a risk management point of view, all solutions have some risks that must be highlighted. Risks can also be viewed as an opportunity or benefit.

If a risk is taken and causes rerun of all work, the total time for throughput is shorter than the time with the traditional sequential way.

Let's look at the risks associated with the Health Insurance example.

- Partition sizes may vary. However, we should be able to come within 10 to 20 percent of a single table. This depends in part upon the business volumetric by state and year. Our approach selects different states to minimize regional volumetric differences (local economies and state regulations), thus mitigating this risk.
- Cartridge media failure requires full rerun of all work impacted. That is, if one task abends, SmartBatch will automatically fail upstream and downstream tasks. When a cartridge media failure occurs, the backup copy of the file is used, and the entire process is rerun for the set of input files. Media failure when using cartridges is 0.1 to 0.2 percent.
- Return code level failures - SmartBatch does *not* presume that a non-zero return code constitutes a failure. Consequently, failures that are reported as return code values require manual detection and intervention.

- Multiple jobs to track. SmartBatch requires that overlapping processes (writers to, and readers from, the pipe) be in separate jobs. This allows JES to schedule the processes concurrently.

  Suggestion: Follow all programs with steps that abend if executed. This will allow you to detect errors more effectively.

3.1.5 Application parallelism

How will the transformation and other processes fit into the limited batch window? Let us discuss how we can manage different scenarios, assuming that the source and the target are two different systems.

There are two forms of parallelism:

- *Process overlap* is implemented with piping or messaging technology, allowing dependent processes to run concurrently.
- *Symmetric processing* is implemented by running multiple occurrences of a process, processing different fragments of the data.

The relationship between the amount of data to be moved and the size of the batch window determines the appropriate solution. In the following recommendations, the term “batch window” is broken down into its constituent parts:

- *Transfer window* is the period of time when there is no, or negligible, database updating occurring on the source data, and the target data is available for update.
- *Reserve capacity* is the amount of computing resource available for moving the data.

The relationship between data volume and the size of the transfer window is the starting point for a transfer design. Figure 24 on page 68 shows when each solution is appropriate. Case 1 is the simple case where there are no batch window constraints for the data to be moved, and Case 5 is nearly “mission impossible”. The reserve capacity is then analyzed to determine if sufficient compute power and DASD exists to support the transfer.

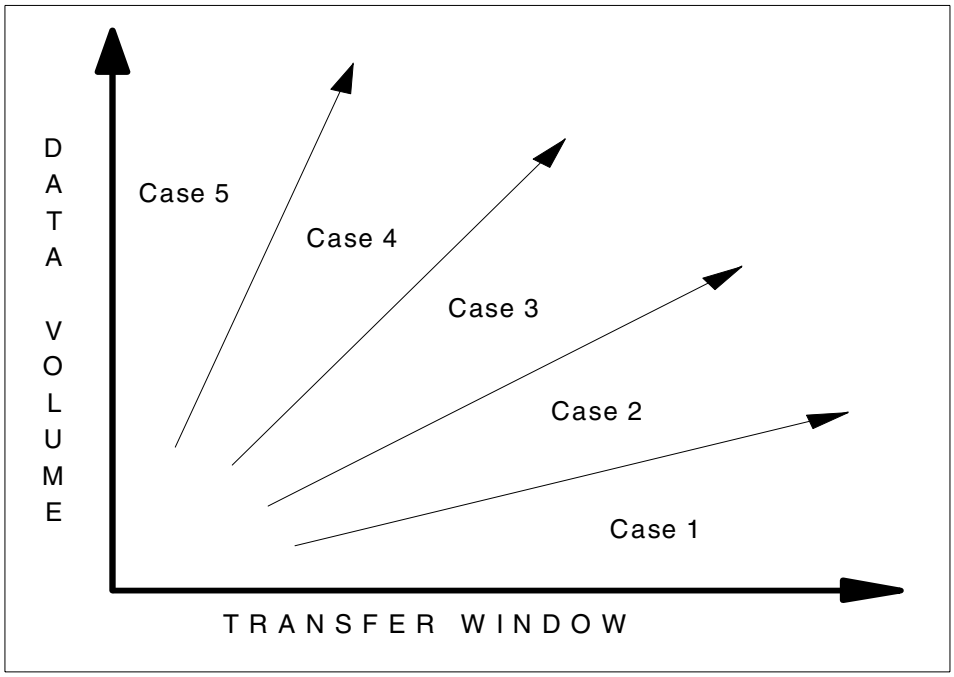


Figure 24. Data transfer design cases

Case 1:

When sufficient capacity exists (CPU and disk) within the transfer window on both the source and target systems, simplicity rules. Use the method that is most familiar to your company. A typical packet solution could be to use different vendor query, FTP, and population tools.

Case 2:

In this case, the transfer window is shrinking relative to the data volumes. The Process Overlap technique works well in this situation. It does not incur any significant additional resource cost (compared to Case 1), and DASD is eliminated from the process. Hardened data backup on disk can be achieved on OS/390 by piping a second copy of the data to DASD. To minimize the elapsed time impact of hardening the data, consider using data striping.

Note that the resource consumption profile is changed from Case 1. That is, the query and populate processing are occurring concurrently instead of

being offset. Depending on the source and target systems' workload profiles, implementing Process Overlap may require shifting some other work.

Case 3:

In this case, the transfer window is becoming shorter compared to the data volume. If there is sufficient capacity on both systems during the window, the symmetric processing technique works well. Again, a hardened copy of the data can be made for recovery/rerun purposes.

Case 4:

In this case, the transfer window is very short compared to the volume of data to be moved. A strategy must be employed to capture changes. Two alternatives exist: either replicate the update transactions and reapply them to the target system, or use a change data capture technique.

This replicate and reapply alternative is only viable when a very small set of simple transactions drives the data changes. With this alternative, update costs are doubled.

The change data capture technique accumulates data changes (additions, updates, and deletions) into a change table (essentially a queue). When the source and target data reside on different systems, the change table should reside on the source system to minimize the impact of capturing changes. The data is extracted, moved, and applied during the very short transfer window. In this situation, an extract query program must be tailored to delete data from the change table. The results of the query are provided to the populate program, which can either LOAD RESUME (append) or do updates, depending on the table design. Either the Process Overlap or the Symmetric Processing techniques can be used for the data transfer. This approach has both disadvantages and advantages.

The disadvantages are:

- There is more complexity due to maintaining a second copy.
- More computing resources are consumed than Case 1, Case 2, or Case 3 for the number of records actually moved.

The advantages are:

- The change data capture and query do not access the same data being accessed by existing applications, thereby avoiding DASD contention with existing workloads.
- Only changed data is transferred between the systems, as opposed to full capture with LOAD REPLACE processing on the target system.
- The change table can be shared among OS/390 servers. This enables the query to run on the server within a sysplex that has the most reserve capacity.

Case 5:

In this case, the transfer window is essentially nonexistent. Data transfer under these conditions cannot capture the source data at a point in time, unless RVA SnapShot functionality (hardware capture) can be used.

In a situation of frequent updates, a continuous data transfer is required. With this technique, the data transfer is initiated multiple times a day to keep the data volume in the change table and target system updates manageable. Again, the query program must be changed to delete extracted data. Care must be taken to avoid deadlocks between the change data capture process and query. This can be accomplished by using either a high commit frequency or row level locking. Your DBA will need to take into account the particular situation to meet your business needs.

With many queries per day, end-to-end process controls are essential to ensure that the latest updates are applied in the correct sequence on the target system.

This approach has both disadvantages and advantages in addition to those associated with Change Data Capture.

The disadvantages are:

- Queries compete with other jobs for resources. (However, Workload Manager can be used to minimize the impact of this contention, and extraction can run as a background process.)
- The extracted data does not reflect a single point in time.

The advantages are:

- CPU capacity is consumed on an “as available” basis. This avoids the need to commit a significant block of computing resources to a particular window.

3.2 Implementing data population

To populate data into a data warehouse, we can use either DB2 utilities or other vendors' utilities. In this section we describe the characteristics and features of DB2 utilities, and how we can use them effectively in the initial load phase.

3.2.1 DB2 Utility

The transformation program creates a load file for each table in the database. These load files can be pre-sorted into the partition key sequence and split into multiple load files, depending upon the partitioned key range. Multiple DB2 Load utilities can run in parallel to fully exploit the MVS servers and I/O subsystem bandwidth. The image copy of the table could be simultaneously created at load time with DB2 V5. The new DB2 V6 LOAD utility enhances parallelism in the load process by allowing the collection of statistics during the load phase, eliminating the need for running RUNSTATS afterwards.

3.2.1.1 SORTKEYS clause

When running the LOAD utility, the SORTKEYS clause in the load statement causes keys extracted during the reload phase to be passed in memory for sorting unless the estimate of the number of keys to be sorted is omitted or specified as 0. When sorting completes, the sorted keys needed for building indexes are passed in memory to the build phase regardless of whether or not an estimate is specified. The SYSUT1 (assuming an estimate was specified) and SORTOUT files are not used for these keys. However, any sorted foreign keys are still written to SORTOUT for later processing in the enforce phase as usual. The disadvantage is that the LOAD utility is not restartable following a failure during reload, sort, or build.

In general, use of the SORTKEYS clause in LOAD allows for shorter elapsed time. It improves load performance whenever there is more than one index. If data is already sorted, you will not see much benefit from it.

3.2.1.2 Inline statistics

In DB2 version 5 or prior, it is necessary to run the RUNSTATS utility to obtain new statistics following a LOAD or REORG. DB2 V6 allows the statistics to be gathered during the execution of some utilities, such as LOAD, REORG, and REBUILD INDEX.

Using inline STATISTICS will eliminate the need for separate RUNSTATS after these utilities and will, therefore, result in reduced elapsed time.

No more additional scanning of the underlying data for gathering the statistics is needed. By having the statistics gathering function within the utility job, administrative as well as scheduling procedures can be simplified. You collect inline statistics by specifying the new STATISTICS keyword. All RUNSTATS options are supported.

3.2.2 LOAD and RUNSTATS parallelism

To achieve parallelism during raw data load, a LOAD for each partition is required. This is not new; DB2 has used partition independence for years to let the DBA work with individual partitions while not affecting the rest of the table or database. We discussed spreading the data uniformly across the I/O subsystem. If each table is spread in this manner, it is easy to achieve optimal parallelism.

For each table, each job could be basically the same; only the job name, partition number, and utility-id (UID) require changing, and possibly SYSREC DD card requires changing if you split the raw data.

3.3 Case study tests

This section describes three case study tests:

- Initial load
- Refresh
- Concurrency

3.3.1 Objectives

We studied the initial load and data refresh processes for the Health Insurance and Banking customers, and have identified the following objectives:

- **Throughput:** Manage to load a data warehouse the quickest possible way and without impact to the workload.
- **I/O handling:** Minimize the amount of data that must go through the tape controllers.
- **Capacity:** Maximize the usage of CPU.
- **Systems Management**: Use automation.
- **Availability**: Manage data warehouse refresh concurrently with query activities.

The Banking customer (case study A) selected high availability as the key goal for data refresh while the Health Insurance customer (case study B) chose very high throughput as its goal.

We used DB2 utilities for case study A (DB2 Version 5) and case study C (DB2 Version 6). For case study B, other vendor's utilities were used; DB2 RUNSTATS were used for gathering statistics.

3.4 Initial load

In this section we discuss our implementation experience for the initial load of case study B. Similar experiences were obtained for a case study A but are not presented here.

Populating the data warehouse with the initial load is a significant effort. The initial load for VLDBs needs to deal with huge volumes of data. To keep the loading time short and the process efficient, some basic choices have to be made such as:

- Load methodology
- Type of parallelism

We made the following choices for our case study tests:

- Load method - piping
- Parallelism - sysplex

3.4.1 Objectives

The objective of the initial load test was to load the database as quickly as possible with the configuration that was available. Our design had to take into account that all the raw data was behind one tape controller.

3.4.2 Methodology

Piping technology (SmartBatch) was used to minimize the single tape controller bottleneck. Piping allowed us to overlap processes and keep the tape controller 100 percent busy. In this case, running 60 concurrent load jobs saturated the bottleneck, however, it did not drive the CPU to 100 percent utilization. If we had additional tape controllers available, it would have been possible to double the level of concurrency to achieve 100 percent CPU utilization and reduce the elapsed time of the initial load even further.

Input sets

The input arrives fragmented by geographical area and time period, states and years. The volumes associated with the various states vary widely:

- Total claims = 800 million
- Number of states = 37
- Number of years = 3

For this case study, we selected which files to process as a single entity to generate an equitable number of claims to process, automatically selecting enough records to fill a partition. This assumes that the claims have similar profiles across regions and through time.

Control Records

Control records are used in the health insurance design to generate identifiers. To support the Input sets, multiple control records are necessary. Each control record supports a partition or set of partitions. This make it possible to run multiple Key Assignment processes in parallel.

Piping technology and SmartBatch

An approach to achieving efficient execution of the initial load process is to minimize I/O resources and automatically manage the execution of job steps by assigning them to available CPU resources within the sysplex. We chose to use SmartBatch for OS/390, which is a piping technology that allows dependent processes to run in parallel (see Appendix D on page 251). The data created by one process is fed directly into the receiving process as it is generated. SmartBatch has several advantages over a DOS-like piping mechanism:

- Multiple writers - several programs running in parallel can place records onto a common pipe.
- Multiple readers - several programs running in parallel can retrieve data from a common pipe. Presuming that they each obtain resources with the same priority, each will read a similar-sized portion of the data. Combining multiple readers and multiple writers allows optimizing the data flow between parallel tasks.
- Full data copy - a process can read all data that flows onto a pipe. This makes it possible to harden a copy of the data while the other readers are consuming the data. Hardening is desirable for rerunning a piped process at a later point in time. See also "SMS data striping" on page 75.

- Filters - SmartBatch offers filters, which can be associated with a particular reader. This is an alternative to Sort for splitting the data.
- Pacing - SmartBatch makes writers wait when the reader cannot keep up, based upon pipe depth specifications (number of buffers). Similarly, readers who do not have data to process are placed in a wait state until the creation process can catch up.

SmartBatch for OS/390 can dramatically shorten the business-critical batch processes in your system, as well as reduce the amount of intermediate disk space needed.

By running jobs and job steps in parallel, SmartBatch for OS/390 can significantly reduce the elapsed time of batch jobs or job streams. Further elapsed time reductions result when SmartBatch for OS/390 is used to minimize I/O resource usage and automatically route job steps to available processors in the Parallel Sysplex.

SmartBatch for OS/390 does the following:

- It reduces your batch window. It improves batch performance in single image and Parallel Sysplex (TM) environments.
- It runs your batch applications in parallel. It supports BatchPipes across the systems in a Parallel Sysplex.
- It enables you to better manage your workload. It provides automation capability for tailoring your batch processing.
- It improves I/O efficiency through advanced techniques. It reduces VSAM I/O counts up to 80 percent and non-VSAM I/O counts up to 90 percent.

SMS data striping

SMS offers the opportunity to stripe data across multiple DASD volumes, increasing I/O bandwidth. This is what was used together with SmartBatch for hardening the data to avoid introducing I/O delays in the overlap processing. The volumes containing the data striping reside on different controllers.

HSM Migration

HSM uses software compression before archiving data. We used HSM to compress the data before moving it to tape cartridges, minimizing the number of bytes that must be moved through the tape controller.

The minimum amount of SMS Data Striping storage required to support a single load is equivalent to a single copy of the set of input files processed.

DB2 Hiperpool

The DB2 Hiperpool is used to reduce or optimize I/O for high data access like transformation tables or whatever high accessed data. We used Hiperpool to store a critical NPI for the duration of the Key assignment, and to store large gigabyte-size look-up tables in memory in order to reduce the elapsed time for the initial load and monthly update process. Using Hiperpools to reduce or eliminate I/O out to DASD can significantly reduce the elapsed time of an application.

Figure 25 shows the data flow from the operational data to the data warehouse using pipes for the Health Insurance customer case. For detailed information about each process, refer to 3.1.1, “Initial load” on page 58.

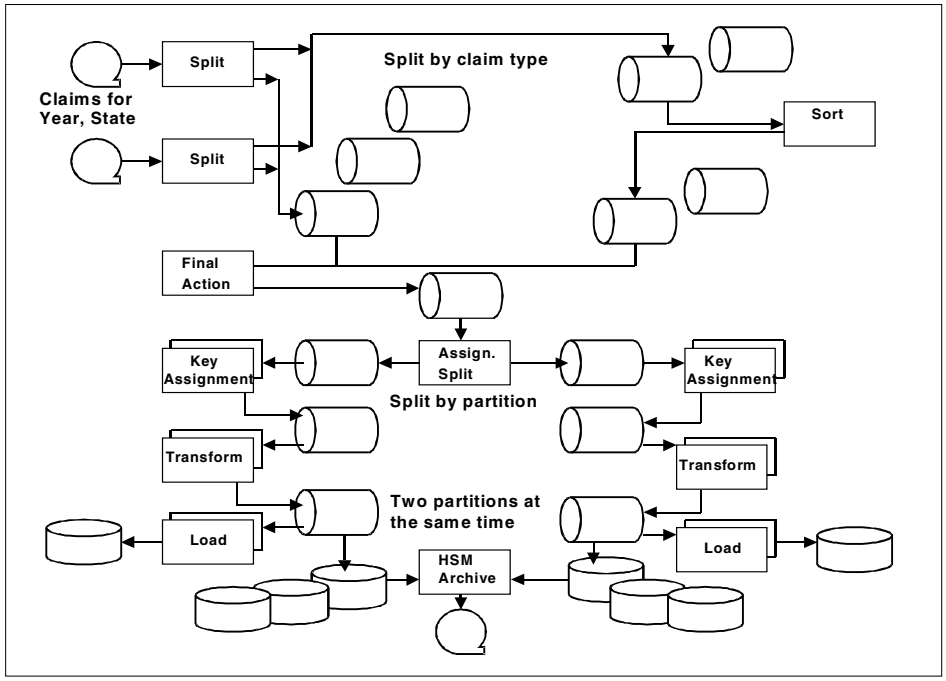


Figure 25. Detail design of initial load using pipes

Benefits for this specific method are:

- All processes, including load, are going on concurrently.
- The transformation and the load complete in the minimum possible elapsed time, based upon processor capacity.

- HSM Archive captures data on partition boundaries, simplifying recovery if a disk media failure occurs during the downstream processing.
- There is no tape storage for intermediate results. Only data which may be required to reload partitions at a later date is archived to tape. That data contains all generated identifiers, minimizing the reload complexity.
- This reduces the tape controller bandwidth constraint. With more tape controller bandwidth, you can drive more parallel processes at higher CPU utilization.
- Only 3 physical read/write transactions occur. Compare this with 3.1.1.1, “Sequential solution” on page 58.
- The largest claim type (70 percent of the volume) used 4 sorts reading from a common pipe to balance sort throughput with other claim types.
- The amount of intermediate disk space needed during the processes was reduced.

3.4.3 The case study test results

We used piping technology (SmartBatch) to move data through this process. The piping made it possible to create several layers of concurrency. The piping methodology that we used greatly increased the throughput, thereby allowing the database to be loaded in a reasonable amount of time; see Table 6. This piping methodology, even with the single tape controller constraint at the Teraplex center, enabled us to process 800 million claims in 64 hours. That is 12.5 million claims per hour (see Table 6).

Processing of the claims includes split, sort, transformation, key assignment, and the load and statistic collection of six billion rows across 63 tables. The largest table was one billion rows.

Table 6. Processing throughput

Number of claims	Total number of rows	Elapsed time	Throughput/hour
800 million	6 billion	64 hours	12.5 million claims

Figure 26 on page 78 shows the CPU consumption in each process of initial load. The majority of the elapsed time occurred during the split process, feeding the final action sorts. The average elapsed time for a state set was 2 hours and 8 minutes. This was also the least CPU-intensive process. With additional controllers, multiple state set streams could be driven concurrently. Theoretically enough concurrent state set streams feeding the load process could drive the load process CPU utilization to 100 percent. This would substantially reduce the elapsed time of the initial transformation and load.

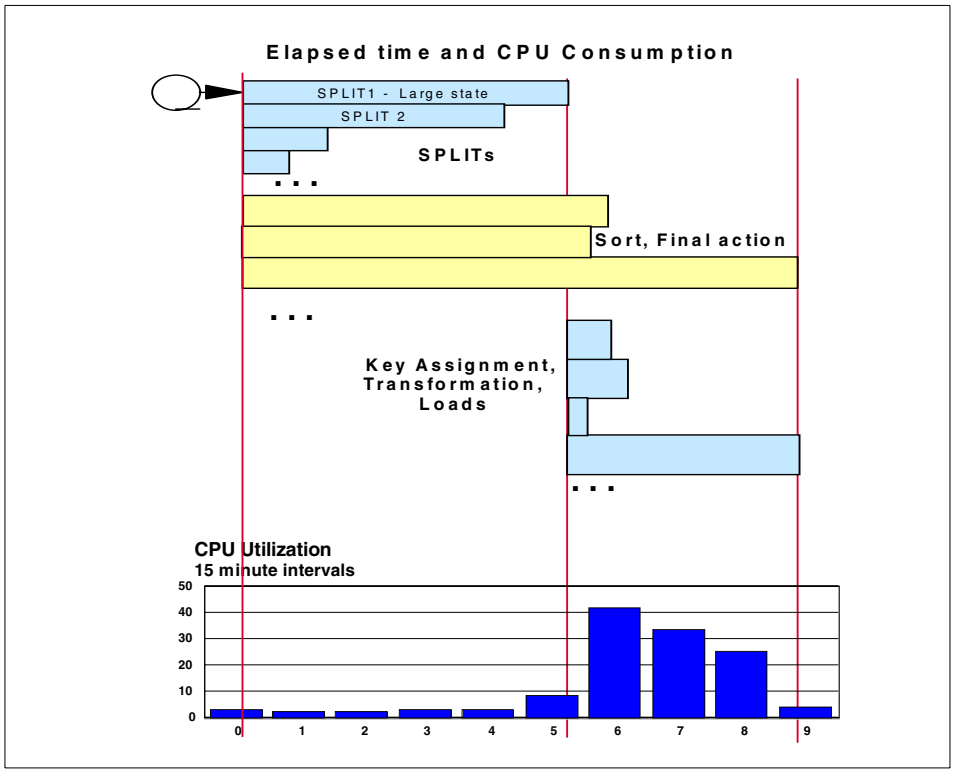


Figure 26. Typical CPU consumption

The processor never exceeded 50 percent utilization and averaged 12 percent utilization. Figure 26 highlights the impact of I/O on elapsed time. The split and sort processes were very I/O-intensive, therefore, CPU consumption was very low. In contrast, transformation, key assignment, or load consumes many more CPU cycles. The split was gated by the throughput of a single tape controller.

Note

The actual sort of the data must complete prior to the downstream processes becoming active. When the sort program writes out the first sorted output record, the downstream processes will then be able to run in parallel.

The case study test showed that:

- All processes, including load, are going on concurrently.

- The transformation and load completes in the minimum possible elapsed time, based upon processor capacity.
- The HSM Archive captures data on partition boundaries.
- There is no tape storage for intermediate results.

The test was executed based on 3.4, “Initial load” on page 73 and all results are based on the configuration and other metrics in 1.5, “Health Insurance - case study B” on page 10.

3.5 Monthly refresh of data warehouse

This test was done with case study B.

3.5.1 Objectives

The objective of the monthly refresh test is to append one month of new claims data from DASD, and contain the database downtime within a weekend window of no longer than 60 hours.

3.5.2 Methodology

The monthly refresh was divided into three categories: pre-processing, database maintenance window, and post-processing (see Figure 27 on page 80).

During pre-processing and post-processing the database is available for query processing. During the maintenance window, the database is taken offline for query processing. The maintenance window is the only category that requires the database to be taken offline, therefore, it is the only category of processing that was required to fit within the weekend window.

Pre-processing consists of splitting, sorting, final action processing, key assignment, and transformation of claim records into load files. During this time, the database is available for query processing. The database is then taken offline to open up the maintenance window.

Database maintenance window processing consists of loading the transformation files and processing the final action updates against the database. Prior to the start of post-processing, the database is brought back online for concurrent query processing.

During the post-processing, the database is brought back online for concurrent query processing. A backup copy and statistic collection is taken for new partitions and any partitions that were changed during the final action

update. The large NPI that is used during pre-processing is rebuilt at this time, so that it is ready for the next month's update.

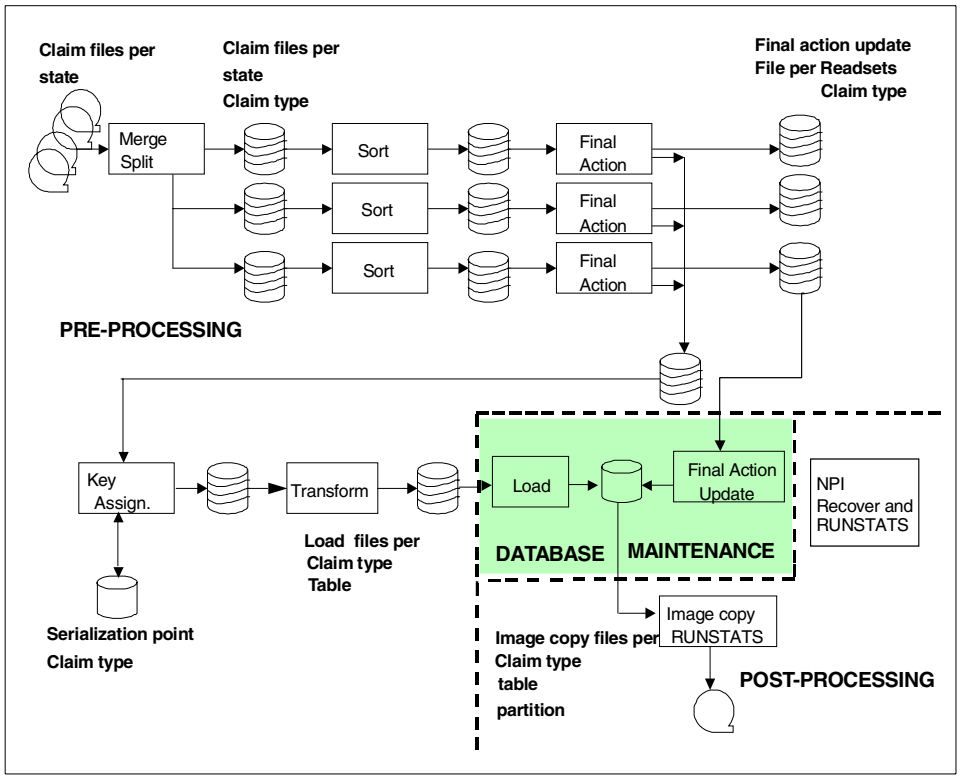


Figure 27. Data refresh processing

1. Pre-processing

   During preprocessing, the data was extracted from the operational system and transformed into load files. We ran the split, sort, final action and key assignment processes while the data base was online for end-user query activity. This effectively removed those processes from the critical batch window. An advantage was that they could run anytime when traditionally less tape activity is occurring.

2. Database maintenance

   During the database maintenance window the data warehouse was offline for query access. This was when the data is loaded into the data warehouse and previous claims were updated reflecting they were no longer the current claims. For the case study test, the backup and statistic collection (backup phase) was run during the database maintenance

window. Depending on the production implementation and environment, this backup phase task could be deferred to the post-process phase when the database is back online for queries.

If queries are expected immediately against new partitions, RUNSTATS must be in the maintenance windows. Consider using RUNSTATS sampling. It can significantly reduce elapsed time, and only 20 to 25 percent of sampling is enough to get the required statistics.

3. Post-processing

   We could run the image COPY and RUNSTATS utilities while the database is available for query activity, too. The copy pending state does not preclude reading the database. So we can move the utilities from the critical batch window. For additional information, refer to Figure 31 on page 92.

   During this phase, the data warehouse was back online for queries. This was when the nettting key index was recovered. This was a very large index, and its existence caused the database maintenance phase to run a long time. Since this large index was only used during the pre-processing phase, we dropped this NPI before the database maintenance phase, and then recreated it during the post-processing phase. In reality, the NPI could be recreated any time before it is needed again for the next monthly pre-processing phase.

We used WLM to ensure that some of the processes with larger volumes get a significantly larger share of the cycles available, to minimize the critical path.

Input sets

The input arrives fragmented by geographical area, that is, by state. The read sets contain up to eight input files in parallel. Each input set can then consolidate its data for later processing. Several input sets can be processed in parallel. The current configuration and load balance analysis determined the number of read sets processed in parallel.

Control records

Control records were used to generate unique identifiers during the key assignment process. To support the read sets, we need to have multiple control records. Each control record supports a partition or set of partitions. This make it possible to run multiple key assignment processes in parallel.

OPC-TME

OPC was used to control the job flow. OPC is a batch job scheduler that controls the job flow and dependencies one job may have over another. OPC made it very easy to manage the monthly refresh process.

SMS data striping

Using SMS managed data sets, the data can be spread across multiple DASD volumes across multiple controllers. This is called *data striping* and can yield significant increases in I/O bandwidth.

The benefits of this methodology are:

- Only DB maintenance portion is executed during the critical batch window.
- The pre-processing such as transformation, and post-processing such as backup or RUNSTATS can be executed outside of the DB maintenance windows concurrently with query processing.
- There is no tape storage for intermediate results. Only data that may be required to reload partitions at a later date is archived to tape.
- Using split and merge technique improves the performance by accessing the source data in parallel.
- SMS data striping increased the I/O bandwidth for all intervening files.

3.5.3 Case study test results

During this case study test, one month's worth of claims data (500 million rows) was added to the database. The actual measurement was taken on a 10-way G5 server, using data for 37 states.

The test result of monthly refresh is shown in Figure 28 on page 83. The total elapsed time for pre-processing, maintenance window, and post-processing was 30.5 hours for 37 states. The projected time for claims across the entire 50 states is 75.5 hours. The projected query downtime to execute the maintenance window for the entire 50 states is 27 hours. This far exceeds the requirement to keep the maintenance window within a weekend, no more than 60 hours, for the entire 50 states.

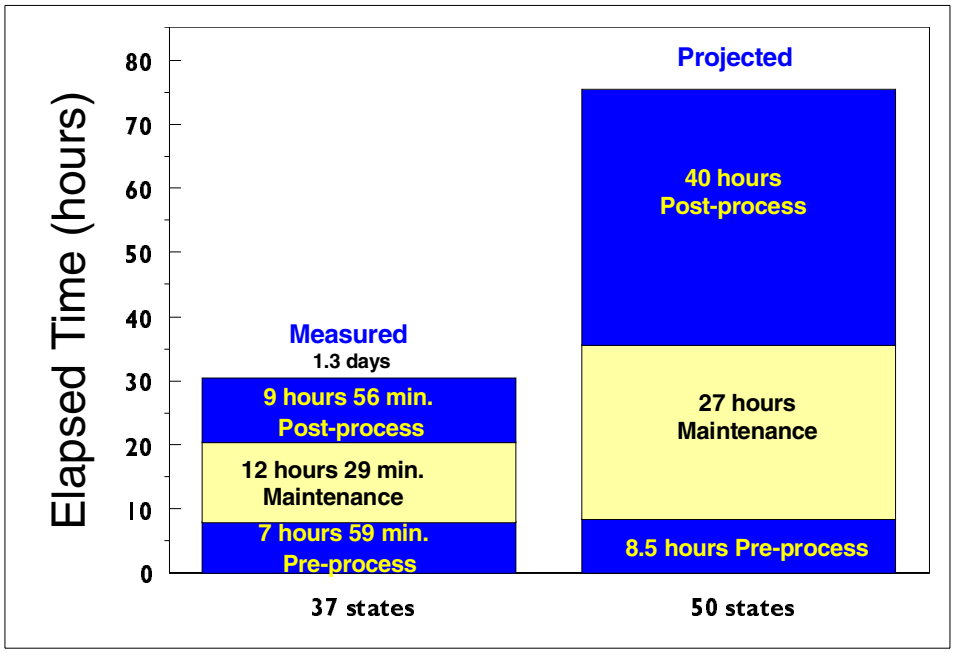


Figure 28. Monthly refresh test result

3.6 Concurrency

One customer requirement is concurrency of the data. To load the data (initial load or refresh) at the same time as the query activities are ongoing is a challenge. This test is done for case study A.

3.6.1 Objectives

The objective of the test was to show that data warehouse refreshes and queries can run concurrently with little impact to the utilities and query workload.

3.6.2 Methodology

The test methodology was based on four test cases as follows:

- **WLM-A-31:** Load one month of data into one table and collect DB2 catalog statistics for the table, table space, primary index, and non-partitioned indexes (NPIs).
- **WLM-A-32**: Run the ad hoc query workload (20 users).

- **WLM-A-33**: Combine WLM-A-31 and WLM-A-32. The WLM policy gave the ad hoc query workload priority over load and RUNSTATS.
- **WLM-A-34**: This was like WLM-A-33, but the WLM policy gave the load and RUNSTATS jobs priority over the query workload.

Table 7 shows the configuration of the test cases.

Table 7. Configuration for concurrency test cases

	WLM-A-31	WLM-A-32	WLM-A-33	WLM-A-34
No. of servers for queries	N/A	2	2	2
No. of trivial users	N/A	2	2	2
No. of small users	N/A	2	2	2
No. of medium users	N/A	2	2	2
No. of large users	N/A	14	14	14
No. of Servers for Loads/RUNSTATS	1	N/A	1	1
WLM Policy	N/A	N/A	Favor query	Favor load
Measurement interval	Start to finish	3 hours	Start to finish	Start to finish

3.6.3 Case study test results

The results of the refresh test are presented in Table 8 on page 85. The table shows the actual elapsed time (ET) and number of queries completed during these measurements.

Table 8. Result of the test

Test case	ET in minutes	Trivial users	Small users	Medium users	Large users	Total queries completed
1. Refresh	173	N/A	N/A	N/A	N/A	N/A
2. Queries	180	81	21	67	156	325
3.Refresh and queries (favor queries)	365	79	21	62	134	296
4.Refresh and queries (favor refresh)	186	64	17	47	105	233

Figure 29 shows data refresh and queries executing at the same time, but the queries were favored, that is, the queries have higher priority in the WLM policy. As you can see, since the refresh was lower priority, its elapsed time was almost doubled. But the queries, which are the most important piece of work, were hardly affected, as you can see by comparing the black query bars in Figure 29 or the numbers in Table 8.

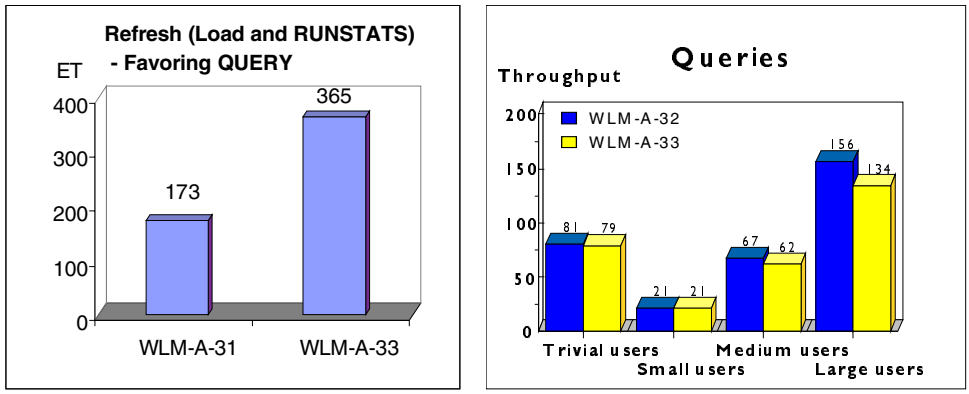


Figure 29. Refresh and queries (favor queries)

Figure 30 shows data refresh and queries executing at the same time, but in this case, the refresh was favored (perhaps because, for example, it is the end of the day and you want to give the refresh a boost in priority or maybe the most recent data is needed as soon as possible). This case shows that when the refresh was favored, the refresh elapsed time barely increased and

furthermore, the queries were not impacted that badly even though they were lower in priority. Again, compare the second and fourth row in Table 8 on page 85 or the numbers in Figure 30.

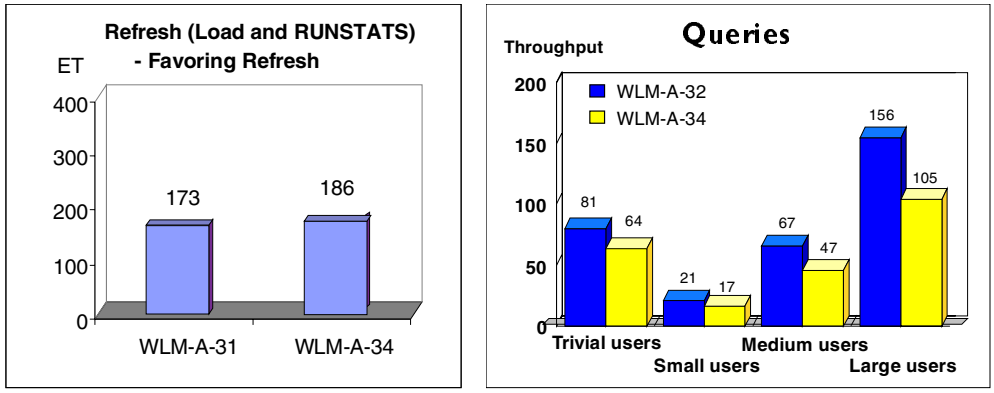


Figure 30. Refresh and queries (favor refresh)

The test results show that data warehouse refreshes and queries can run concurrently with little impact to the utilities and query workload. This can be achieved by having an appropriate WLM policy in place that prioritizes the queries and the refresh based on the business requirements. The prioritization can be changed dynamically as needed.

In addition, the database partitioning scheme must be considered if continuous computing is a business objective. The data used for this case study was partitioned by date/time. It was primarily this partitioning scheme that allowed refreshes and queries to run concurrently with little impact, because the new data was being loaded into data partitions and primary index partitions that the queries were not accessing. The accesses to the NPIs by the queries and loads also caused little impact. For more information about partitioning, refer to 2.3.1, “Partitioning methods” on page 24.

Analysis and conclusion of the test

The following observations were made during this test:

- Refreshes and queries can run concurrently with minimal impact.
- WLM should be used to ensure balance between load and queries.
- Type 2 indexes should be used.

3.7 1 TB database build

This test case was run with pre-GA DB2 Version 6 code, and DB2 Version 6 performance has been improved since our test.

During case study C, we focused on building one big table using DB2 Version 6 and its new features. The total elapsed time to build our 1 TB database included loading the data, building the primary index, building a non-partitioned index, and collecting statistics. The table was then ready for query execution.

With DB2 Version 6, LOAD can be run with the STATISTICS option, which enables gathering statistical information during the load of the table. Our test objectives were:

- Execute the DB2 LOAD and REBUILD INDEX utilities with the inline STATISTICS option.
- Compare the elapsed time of executing the LOAD and RUNSTATS utilities separately to executing the LOAD utility with the inline STATISTICS option.

Next, we proceeded to build a 340 GB non-partitioned index. We used the REBUILD INDEX utility, which was called RECOVER INDEX in DB2 Version 5. In DB2 Version 6, an index can be rebuilt in parallel if the new option SORTKEYS is specified.

The STATISTICS option of the REBUILD INDEX utility was also used to gather statistical information from the non-partitioned index.

General table information

The total size of the table, not including partitioning or non-partitioned indexes, work files and free space, was 782 GB.

- Number of rows: 6 billion
- Row size: 141 bytes
- Number of partitions: 209
- Number of pages: 216815300
- Average number of pages per partition: 1037400
- Tablespace is not compressed

Due to the 2.8 GB volume capacity, each tablespace partition spanned two volumes. Data was evenly distributed across partitions and across the 34 control units.

Load information of the partition

The information in Table 9 includes the elapsed time of the loading of the raw data into the table and primary index, and collecting statistical information. In DB2 Version 6, you can gather statistical information from the tables and their indexes by running the RUNSTATS utility separately or by using the inline STATISTICS option while running the LOAD utility. You can see the benefits of the new DB2 Version 6 LOAD with inline STATISTICS in Table 9.

Table 9. Load information of the tablespace partition

	Load (min.)		Runstats (min.)		Load with inline stats (min.)	
	Elapsed time	**CPU time**	**Elapsed time**	**CPU time**	**Elapsed time**	**CPU time**
one partition	52.0	47.70	44.80	14.84	54.2	58.74
20 partitions	107.7	1004.04	78.37	284.48	132.6	1213.35

In addition, we first loaded data into one partition only, and then loaded data into 20 partitions using one batch job for each partition, to show the benefits of the parallel loading. We used one 10-way G4 processor.

One LOAD parameter worth mentioning is SORTKEYS. We used it to reduce the total sort work space needed and to speed up LOAD. One advantage of using SORTKEYS is that the index keys are passed to sort in memory, rather than written to sort work files. This avoids the I/O to SYSUT1 and SORTOUT. There was also overlap between loading and sorting, since the sort was running as a separate task. Avoiding I/O to sort work files and the overlapping of tasks improved LOAD performance.

For RUNSTATS and inline STATISTICS, we used 25 percent sampling, which means 25 percent of rows were sampled for collecting non-indexed column statistics.

Load information of a large table

For the load of all 209 partitions of the large table (782 GB), we used two 10-way G4 processors in a sysplex environment. The information in Table 10 on page 89 includes the results of the load. The jobs were executed in a manner designed to keep processors 100 percent busy and to minimize I/O contention.

Table 10. Load information of a large table

	G4 Processor (Measured)		G5 Processor (Projected)	
	Elapsed time (min.)	**CPU time (min.)**	**Elapsed time (min.)**	**CPU time (min.)**
Load with inline statistics	716	12680	362	6357

You might notice in Table 10 that the elapsed time savings does not appear as great as in the one partition test and the 20 partitions test. The reason for this is that now the processors are saturated; therefore, the elapsed time savings at saturation is only directly proportional to the CPU time savings.

For the LOAD utility, SORTKEYS was calculated for each partition and since the data was distributed evenly, it was the same for each partition. Refer to *DB2 for OS/390 Version 6 Utility Guide and Reference,* SC26-9015 to calculate the SORTKEYS value.

Non-partitioned index build information

At most installations, non-partitioned indexes are developed to improve performance of business queries. For our large table, we defined one large 340 GB non-partitioned index.

In order to achieve parallelism when building a large non-partitioned index, we used the REBUILD INDEX utility with the SORTKEYS and STATISTICS options. With the SORTKEYS option, the data is unloaded and the index keys are sorted in parallel, which significantly reduces the total elapsed time of rebuilding large non-partitioned indexes from a partitioned table. The STATISTICS keyword allows additional subtasks to collect the sorted keys and update the catalog table in parallel, eliminating the need for a second scan of the index by a separate RUNSTATS job. For more information on the parallel rebuild index, refer to *DB2 UDB for OS/390 Utility Guide and Reference*, SC26-9015.

Table 11. Large NPI build information

	G4 Processor (Measured)		G5 Processor (Projected)	
	Elapsed time (min.)	**CPU time (min.)**	**Elapsed time (min.)**	**CPU time (min.)**
NPI build with inline statistics	3288	4075	1675	2038

The total elapsed time to build the 340 GB non-partitioned index using the REBUILD INDEX utility was 54.5 hours. Had we built this without the parallel rebuild, it would have taken about 130 hours.

Note: Non-partitioned indexes are not required to achieve query parallelism. The number and size of non-partitioned indexes varies with different workload and application requirements.

3.8 Summary

The design methodology for initial load and monthly refresh consists of:

- Examining the process flow for a serial process.
- Examining the process flow for input fragment-to-partition mapping opportunities.
- Identifying opportunities for improvements.
- Developing a design for throughput alternatives.
- Performing risk analysis for these steps.

The case study test shows that the objectives for the test were met as follows:

- **Throughput**: The CPU and I/O parallelism created the possibility to load the data warehouse the quickest possible way and without any impact to the workload.
- **I/O handling:** A parallel design reduced the amount of data that must go through the I/O controllers.
- **Capacity:** WLM maximized the CPU usage.
- **Systems Management**: SmartBatch managed the execution automatically.
- **Availability**: The data warehouse refresh can occur concurrently with query activities, or can be split into online and offline activities.

Chapter 4. VLDB utility maintenance

Maintenance in very large database (VLDBs) environments is complex because it involves updating a large amount of data in a short, well-defined timeframe. In addition, the data may be coming from a variety of sources, and usually is cleansed and massaged before it enters the BI data warehouse or data mart. Frequently we need to make use of advanced techniques, such as using BatchPipes, overlapped processing, parallelizing programs and others to exploit partitioning to meet the batch window. In this chapter, we discuss the issues related to utility processing. For advanced techniques see Chapters 3 and 5.

By their very nature, VLDBs require a relatively long build time and they also need to be refreshed with new data periodically. Data update can impact the availability of the data to end users, therefore, the available update window usually conforms to an existing batch window. Frequently, the design of the database is guided by the amount of data that needs to be updated and the architecture which will allow these many updates to be completed within the desired timeframe.

The design of a VLDB can alleviate the need for reorganization, if the updates do not change the clustering of the data significantly. Many designs will add new data at the end of lasting partitions or in separate partitions. With this approach, data reorganization is not necessary.

An integral aspect of maintenance is to be able to recover the database in case of an outage or disaster. How are you going to recreate the database? You must decide if re-building the VLDB is feasible and desirable, as opposed to taking image copy backups and restoring the database from image copies, which is a more traditional approach. The procedures for recovering the VLDB environment must be defined, documented and tested along with the disaster recovery plan.

With VLDBs, we try to exploit the parallelism of DB2 and a sysplex to reduce the elapsed time for all the maintenance jobs. As depicted in Figure 31 on page 92, we try to define critical processes which must finish within a certain window. Typically load and update are the only activities which must happen in this batch window. RUNSTATS for the *new* partitions or tables should also be done during this window, as they are critical for queries to complete in a optimal fashion. Data preparation including cleansing, transformation and staging can occur beforehand. Similarly, image copies can occur after this window.

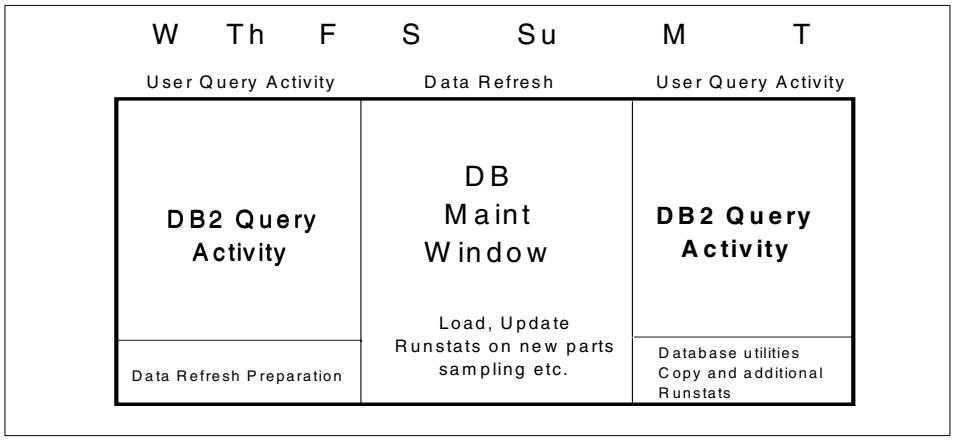


Figure 31. Batch window analysis

The key to increasing parallelism is to exploit partitioning in DB2 in support of your design goal. If availability of data is paramount, then it is useful to have a time-based partitioning key. This allows new data to be appended at the end of data sets and typically within new partitions. Thus all of the data in the VLDB is available when these new partitions are built or loaded. VLDB data remains available even when NPIs are being built, if that non-partitioned index (NPI) is not used to run your queries.

Especially with online reorganization, data remains available for almost the entire duration. During online reorganization, DB2 unloads the data into a new data set in correct sort order. Subsequently, DB2 switches the data set names for the reorganized tablespace with this newly created data set. For a very brief period when this switch takes place, the particular tablespace is unavailable. The NPIs also can be built in parallel with very little outage with the *parallel recover index* method. This functionality is now available in DB2 UDB V6.

We use sampling for RUNSTATS. Sampling 20 percent of the data reduced the elapsed time by 60 percent, with little or no impact on query performance. With larger databases the sampling becomes much more attractive and effective, with significant savings in the elapsed time without appreciable loss of data characteristics. For further information, see *DB2 for OS/390 V5 Performance Topics*, SG24-2213.

We used LOAD, Image copy, RUNSTATS and DSTATS in sequence for all tablespaces. With DB2 V5, Image copy and LOAD can be done together in one sweep of data and therefore we used this. DB2 V6 will allow LOAD,

Image copy and RUNSTATS in the same sweep. We recommend utilities that run against all the partitions be run in a staggered fashion so that all of the data is not unavailable in the same time period and the workload is distributed over time.

In summary, the critical success factors for VLDB maintenance are the same as those for data population:

- Parallelism exploitation
- Minimizing I/O
- Fragment realignment or partition balancing
- Risk management
- Batch window conformance

4.1 Reorganization

At the Teraplex Center, we designed the case study A (Banking) and case study B (Health Insurance) using time-based partitioning. Thus all new data is added in new partitions at the end of a tablespace, mitigating the need for reorganizing the data.

However, in the Health Insurance study, the data was organized by claim type which is heavily correlated to chronological order. Thus in this study, while almost all new data is added at the end or to new partitions, there can be some updates related to earlier claims updating existing partitions. We programatically maintain a list of partition numbers affected so that only these partitions are backed up.

Maintenance of a VLDB is very much dependent on the organization of the data. See Figure 32 on page 94 for partitioning Health Insurance case study data. With this method, we add new data in new partitions at the end of the tablespace. For this study we defined five partitions for testing the monthly reload process.

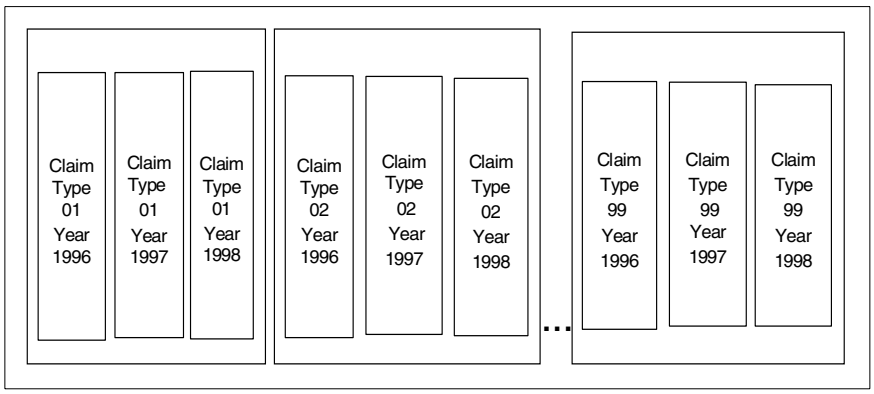


Figure 32. Partitioning by time

Since all the partitions are separate data sets, we recommend using parallel reorganization jobs when you do need to REORG tables due to having a significant number of updates.

To increase the availability of the data warehouse, the REORG utility can also be run in online mode. In this mode, tables are available during most of the utility run time. DB2 unloads and sorts while tables are available for query. Now DB2 builds the tables in shadow data sets and for a brief period freezes activity in tables. During this time, DB2 switches the tables, matches data set names with shadow data set names (reorganized data) and then makes tables available again for query activity. For a large table with 64 partitions, the outage time at Teraplex system was only a few minutes. Note that this process doubles the disk requirement for concurrent partitions being reorganized.

Figure 33 on page 95 illustrates the impact of the new feature of DB2 V6 inline utilities, showing the reduced total elapsed time for REORG+Image Copy.

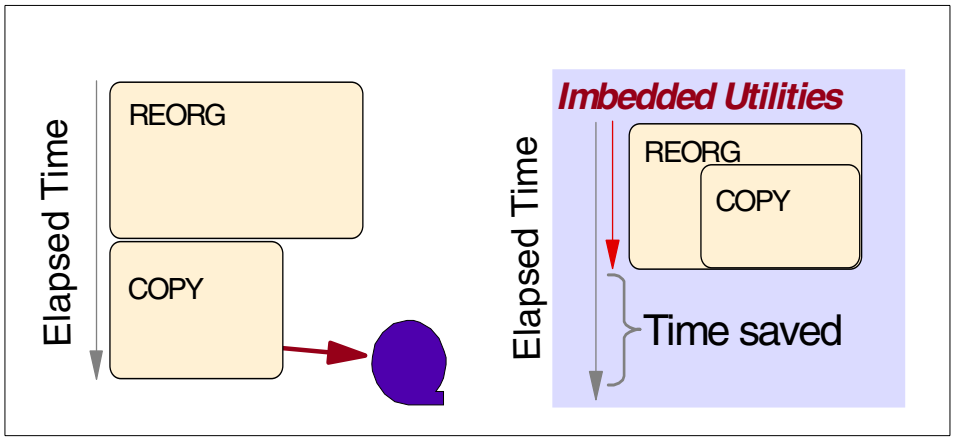


Figure 33. Imbedded utilities

4.2 Backup

Our emphasis in the case studies has been scalability, workload balancing, query performance in the VLDB query environment and data mart co-existence. Taking backups, which is a high priority in production environments, is not a priority in our test environment.

Backups can be run in parallel. However, you have to run multiple COPY jobs with separate DSNUM in separate jobs to parallelize the copy operation.

Image copies can be run online to gain improved availability. To run online, you can run the Copy utility with the SHRLEVEL REFERENCE parameter, or by using Concurrent Copy functions of DASD Facility Storage Management Subsystem (DFSMS). See *DB2 for OS/390 Version 5 Utility Guide and Reference*, SC26-8967 for details.

At the Teraplex Center, and more and more in large-size warehouse environments, RVA DASD is being used. RVA DASD has some operational advantages.

- Space savings

  The data is compressed and compacted by the RVA Controller. The RVA does not differentiate between data and index and compresses them both. RVA also compacts (that is, eliminates inter record gap) from the tablespaces, which is a unique benefit of using RVA DASD. See 2.5, “Compression” on page 33 for more details.

- SnapShot

  RVA DASD has the SnapShot feature, which allows an instantaneous backup. RVA produces a snapshot without moving or duplicating data. Within the RVA architecture, a FTD pointer is akin to a logical data set. The FTD pointers point to a physical data set on the RVA DASD. For backup, a copy of the FTD pointer is made. Thus, the original and the copy (also known as logical data sets) point to the same physical data set. SnapShot takes seconds per data set compared to minutes or hours with traditional DASD.

  Figure 34 shows measurements done at the Teraplex Center as part of one case study. SnapShot took only one second to create a virtual copy of a partition of a large table. This contrasts to the standard DB2 image copy which took over one minute. When the snapshot operation is complete, DFSMS writes the virtual copy to another data set for data hardening.

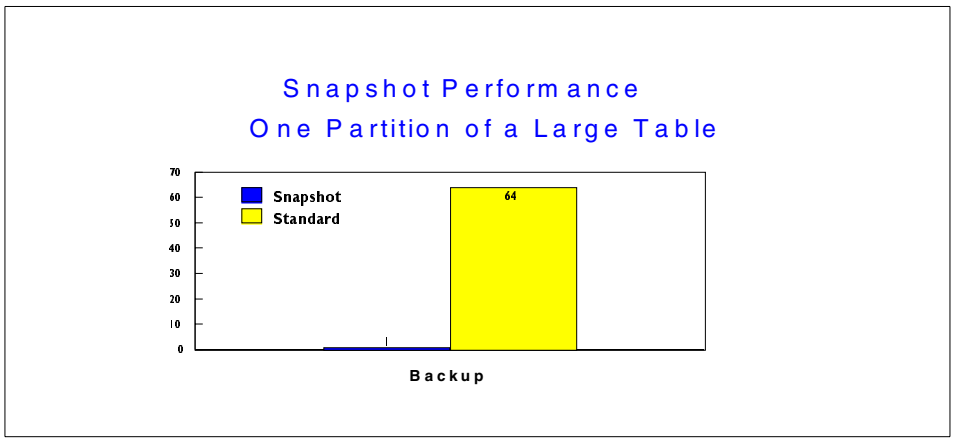


Figure 34. RVA SnapShot vs. DB2 backup times

The recovery process uses the hardened snapped data set. Refer to *Using RVA and SnapShot for Business Intelligence Applications with OS/390 and DB2*, SG24-5333 for more information about RVA DASD with DB2.

4.3 Recovery

VLDBs in the decision support environment are primarily read-only warehouses, thus recovery is not as critical as in OLTP environments. However, as business intelligence applications grow in importance to the performance of the business, availability of the warehouse and marts becomes more critical.

We recommend taking image copies after each refresh. This image copy is used to recover the database at a customer's primary or disaster recovery site.

Sometimes customers save the flat files which were used to load the data warehouse in order to rebuild the data warehouse in case of recovery. This approach requires a large amount of storage to save these flat files on DASD or tape. If you use an application to update databases, then this approach is not viable.

Image copies are used to recover the tablespace in case of any corruption of data or hardware failure, and therefore provide insurance against failure. You can take hundreds or thousands of image copies before one is needed for restore. However, image copies may make for faster recovery, especially in the case of compressed data. Image copies store compressed data and DB2 does not have to encode data while restoring. In contrast, if you load from a flat file, DB2 has to encode and compress data.

After restoring data from an image copy, you must rebuild indexes via the Recover index utility. With DB2 UDB V6, you have two options: the recover indexes with an image copy of the indexes, or rebuild the indexes from the table. Especially in a VLDB environment, the ability to recover an index from the image copy of the index is very useful and provides reduced elapsed time. This utility can also be run in parallel to cut the elapsed time even more.

Refer to *DB2 for OS/390 Version 5 Utility Guide and Reference,* SC26-8967 and *DB2 UDB for OS/390 Version 6 Utility Guide and Reference,* SC26-9015 for further details.

4.4 Summary

We make the following recommendations regarding to VLDB maintenance:

- Exploit partitioning to do parallel utility processing on separate partitions.
- Exploit synergy in utilities to do multiple tasks in one sweep of the data. For example you can do LOAD and Image Copy together. In DB2 V6, we will be able to do LOAD, Image Copy and RUNSTATS together.
- Run the DSTATS utility for collecting statistics for skew in the data distribution.
- Explore advanced techniques, such as sampling of data for RUNSTATS and piping for batch jobs, to reduce run time for maintenance jobs.
- Run utilities in online mode.

- When using RVA DASD, use the SnapShot feature to take backups.
- Take an image copy of the database after each refresh.

Chapter 5. Data mart on S/390

Frequently, end-user departments require access to only a subset of the detail information found in a data warehouse. Due to the amount of data and the database design for the warehouse, it may be appropriate to extract a subset of information and store it separately for access by the end-user tools. The detail data in this structure is often aggregated into summary tables to improve query execution in this departmental entity, referred to as a data mart.

A *data mart* is a collection of data for a particular focused subject area, such as customer and sales data for the marketing department, or data pertaining to a specific geographical region for a regional mart.

For example, the East region for a company may only care about data that pertains to them and would only need access to data for that geographic subset. A regional mart would be created at the regional office to meet their requirements, thereby avoiding network access to a centralized data warehouse containing data pertaining to all the regions in the company. Were they to access the warehouse directly, the queries may run longer, due to network connections, and the fact that all the stored data in the warehouse might have to be scanned to derive the same answer.

A mart is tailored for use by either a single department or a part of the organization. These marts are sometimes considered departmental data warehouses. When constructing a data mart, the business must decide the subject, the volume, the aggregation level and how the information is to be organized.

Determining the need for a data mart depends on the needs of the business organization, the architecture of the data warehouse environment, the data and performance requirements of the end users, and maintenance and data population issues to support the business intelligence (BI) databases. This chapter will discuss these issues to assist you in determining the need for a mart, as well as the options you have in implementing one. The key issues related to implementing a mart are:

- Architecture
- Performance
- Population
- Maintenance
- Resource management

5.1 Architectural consideration

From the architectural point of view, the data environment may be considered as a single virtual database (see Figure 35). The data within this database flows from one layer to the next through a structured process.

A three-layered architecture optimally involves taking data from the operational source (as well as external sources), extracting, cleansing, and transforming the data into a single data warehouse within the virtual database. The data is then filtered and transformed into more subject-specific data marts to support various departments or end-user requirements. By using a centralized warehouse repository to draw information for all departmental data marts, the company can have consistent data throughout the corporation.

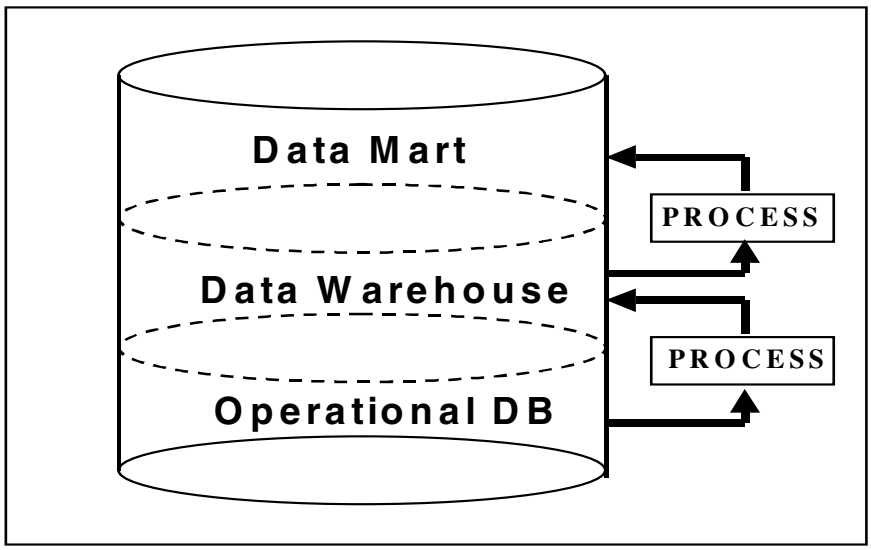


Figure 35. Logical data environment n-tier architecture

The implementation of the underlying physical systems to support this environment depends on the company's IT strategy and the current and planned computer capacity. All layers of the logical data environment shown here can be implemented on one or on several systems. Incorporating different platforms and software for each data tier results in a more complex environment to implement and maintain.

In this chapter, we will focus on two environments for implementation:

- Two physical systems
- Three or more (multi-)physical systems

5.1.1 Two physical systems

The two-system architecture, where the operational data stores and the data warehouse/data mart databases reside on different systems (see Figure 36), offers centralized data management operations and efficient resource management for the business intelligence environment. S/390 is a perfect system for this purpose.

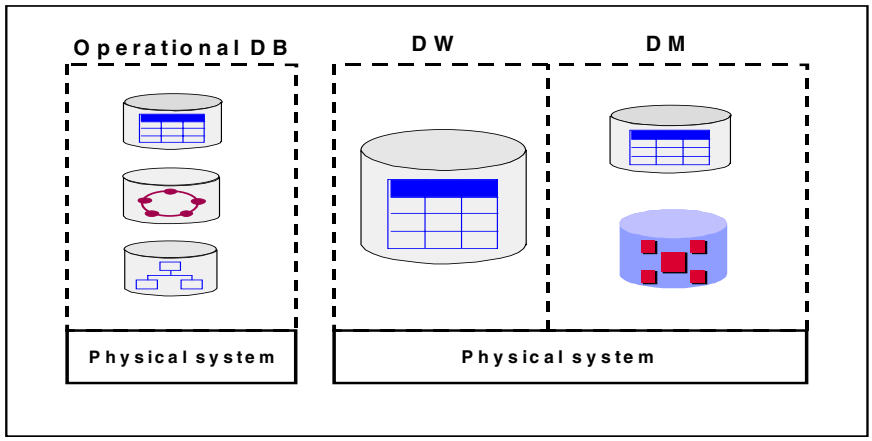


Figure 36. One physical system for data warehouse and data mart

Pros (+) and cons (-):

- Data management (+): By centralizing all BI databases on a single system, the data does not have to be transferred between systems. Utilizing a single software/hardware platform avoids lengthy translations between formats (such as EBCDIC to ASCII). Having the data on the same physical system eliminates lengthy transmission issues associated with distributed marts. While initial update processing volumes may allow for these lengthy transmissions, users invariably demand more timely data and more types of data, requiring more frequent updates, in perhaps, constrained batch windows. Meeting these additional requirements with increased transmissions can pose a significant hurdle later in the life of the BI environment.

- Skills management (+): creating a diverse environment of a number of systems and databases, requires the development of a wide spectrum of skills. This spreads support personnel very thin at a time when technical resources are difficult to locate and retain. By implementing the BI databases on a single hardware/software platform, you can better leverage existing skills and reduce the complexities within the environment.
- Operation management (+): with a single platform, hardware resources can be automatically shared between all BI processes, avoiding isolated islands of capacity that result when the warehouse and each mart are implemented on separate systems. Additionally, a single support staff can be leveraged to maintain the centralized system, insuring adequate support.
- System availability (+):Alternatively, a centralized system supporting all marts and warehouses, can cause some logistical issues for the system. Having all the users throughout the country (or world) accessing the same server, can restrict maintenance and update windows. Accessing remote central servers can also demand significantly larger communication lines to transmit large volumes of data that may be extracted during a query. System availability can be an issue, depending on the central platform selected. Choosing a server with superior availability, recovery, and accessibility characteristics, can avoid many of these issues.

In this document, the case studies were implemented on this architecture due to the overwhelming benefits of this environment.

5.1.2 Three or more (multi-)physical systems

Figure 37 shows another way to spread the logical data layers across physical systems. This architecture provides an independent system for each layer when issues like ownership and delineated query processing power are important in determining the final solution.

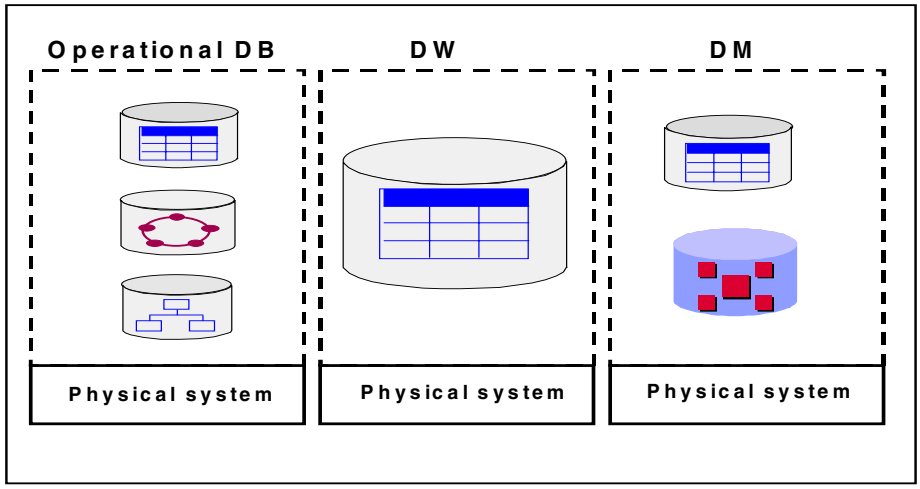


Figure 37. Having a separate system for each logical layer

The transfer of data from the warehouse to each mart is more demanding in resources, and network capacity than the two-system approach. Each system independently controls its own data, however, the data flow must be synchronized from the centralized data warehouse to each data mart.

The individual data marts can reside on separate systems from different vendors placed by geography or organization, or these logical marts can be combined into a single physical system, like S/390. In the decentralized solution, it is probable that the end users will influence the choice of the data mart system and choose a system with which they are already familiar.

Pros (+) and cons (-)

- Flexibility (+): Distributed data marts offer the highest degree of end user flexibility, by allowing users to access the system whenever it is needed without regard to other departments' uses, and permitting the user department to upgrade, and control the support of the system locally.
- End-user proximity (+): Locating the data mart close to the end users facilitates access to the data. Should queries pull large volumes of data from the mart, it does not have to be transmitted over communication lines. This can improve the overall performance of the system for the end users.
- Data movement (-): On the other hand, large volumes of data must be transmitted for regular updates and refreshes to the system. Occasionally, it is faster to completely drop a table and rebuild it, rather than update it.

When that is the case, there can be very large quantities of data sent across the transmission lines to these distributed marts. Hopefully, these transmissions can be scheduled for off-shift hours, avoiding congestion on the lines. This is not always the case, however.

- End-to-end synchronization (-): When marts are created, they result in multiple copies of the same data being distributed through the BI environment. It is critical as data is updated and changed within the BI environment, that these changes are replicated across all marts built from the data. This can result in complex data flow and management concerns, that must be considered when implementing a distributed approach. There may be a reliance on end user personnel to complete a mart update, and it may not always be handled in an appropriate or timely manner.
- Support costs (-): When design changes or system problems occur, end users are generally unable to handle these issues, and the technical support personnel must attempt to address the issues remotely, or fly to the remote locations for specific needs. Adding a support person to the end user department is a common solution that it is often expensive. Frequently, an end user is trained to handle the system, however they rarely have the technical background that allows them to keep the system in top running order.

5.1.3 Data warehouse and data mart coexistence considerations

Combining the data mart and data warehouse workloads on a single server can create some significant hurdles in many hardware environments. Only S/390 can support both a data warehouse and a data mart on the same system with guaranteed CPU resource allocation. The fact that the data warehouse and the data mart coexist on the same system offers many benefits, such as:

- All data needed for the data warehouse and the data mart is stored on shared DASD devices and can be easily accessed for both data warehouse and data mart activities, thereby avoiding data synchronization issues.
- Implementing on a single S/390 platform avoids the additional processing that is needed for data type conversion which is required when mixed systems such as mainframe, UNIX or OEM are used together.
- The same product suite can be used to manage and access the data warehouse and the data mart, such as the database system, operating system and database monitoring tools, database utilities (load/reorg/backup/recover), SmartBatch, and so on. Limiting software

products can significantly decrease the costs associated with implementing the environment.

- Maintenance procedures and skill requirements for the data warehouse and the data mart are the same.

When evaluating the overall architecture implementation for the BI environment, it is critical to balance the user requirements and the complexity and costs to the company. While end users may initially want a distributed mart solution, the overall cost to the corporation has to be weighed into the decision. Benefits like those listed above should be considered when making the decision.

5.2 Performance

A mart is constructed to address the specific requirements of a small group of users within a department or region. Generally these users have similar data needs, and ask similar questions. The performance requirements for a data mart are generally defined by the end users and are an important consideration for BI designers and architects who establish the data mart environment.

The architectural solution (2-tier or n-tier), together with the overall query and update workload, define performance for the end-user community. In the n-tier solution, where connections between environments are network-based, data movement is a performance issue, and should be considered early in the design phase.

The load method and the partitioning strategy for the marts will also greatly impact overall performance and should be considered carefully. Often, it is important to develop a profile of key queries the end users will be submitting to a system, to better understand the volume of data being accessed by the users. This will allow for the best overall design of the database. Additionally, this query profile will identify information that will be commonly accessed by the users. This frequently used information can be integrated into the data mart by data aggregations. Aggregation of the detail warehouse data to create the informational density of data within a mart can be key to performance. For example, including summary tables of sales by time period, such as daily, weekly, or monthly, as well as averages of productivity by organizational unit or product set, can avoid repeated processing of the same query over and over again within the system. Incorporating these precompiled results allows queries to fetch commonly requested information quickly, resulting in superior performance for the end users. Thus, identifying

those aggregations that would have the greatest impact is key in the life of the data mart.

5.3 Population considerations

Many corporations begin their BI activities by building a departmental data mart. Often these marts are populated directly from the operational environment, and as more marts are added to the company, the batch demands for extracting and populating marts can be excessive.

The process to populate the data mart is very similar to the process used for the data warehouse. The following steps can be identified:

- Data modeling
- Cleansing
- Transformation
- Aggregation
- Initial load
- Data maintenance

Every company must develop a strategy to address these steps. Some of the steps, like transformation, may or may not exist for a data mart, depending on whether the company has a warehouse in place. If they do, then it is easier to create a mart because most of the data is already transformed and cleansed during the initial load of the data warehouse. The aggregation step is optional, dependent upon business requirements for low response time or frequent repetitive summing of the same data by many users. The initial load is always concerned with the volume of data and the associated business rules. Some marts are only updated once, while others are updated frequently. Architecturally, data maintenance is similar to the initial load in these environments.

Depending on the size of the data mart and the aggregation level within the mart, the population strategy may vary. If the data requirements based on business needs are simple, then there is no need to make the population more complex than extracting the necessary data from the warehouse. However, the population process often requires additional aggregation of data, thereby lengthening the batch window, but creating a more effective data pool.

The flow from the data warehouse to the data mart can be parallelized if business needs demand it or if the batch window is insufficient. The techniques of parallelization are described in Chapter 3.

Figure 38 shows the sources for a data mart: a data warehouse, an operational database, or an external source. Generally, all data from operational and external sources should be funneled through the central warehouse to insure consistency of data between all the marts within the company. Directly adding information from any outside source that has not been standardized or cleansed, as it typically is when incorporated into the warehouse, should be done very cautiously to avoid contaminating the mart with bad data.

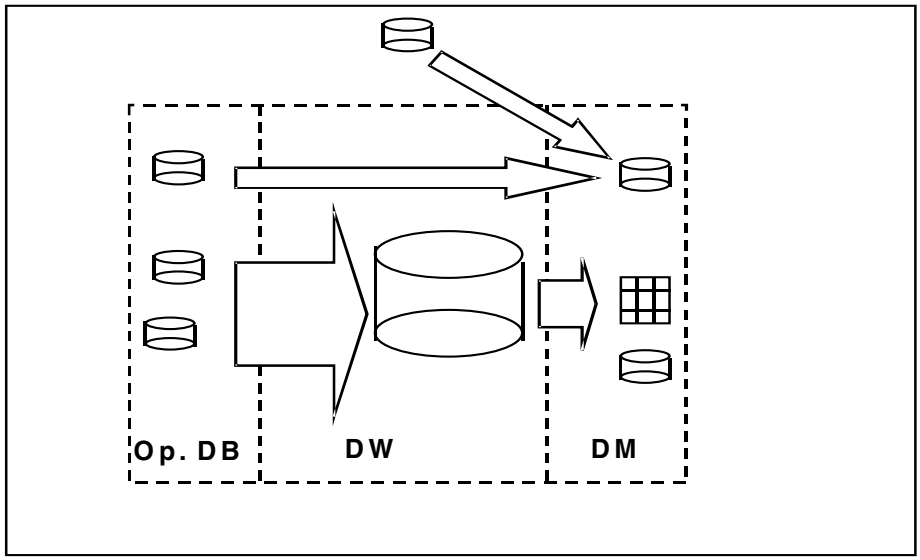


Figure 38. Data mart and sources

Data modeling

Data modeling for a BI implementation (either a data mart or warehouse) is a significant activity that facilitates the analysis and documentation of business needs against the data. Modeling techniques are divided into two classes:

- Entity relationship modeling
- Star schema modeling

The data mart can be built on either the entity-relationship model or star schema model. Either model is feasible and your company's situation and business needs will be the deciding factor.

An entity relationship model is developed around an entity or object (such as a customer, a product, and so on) in which the business is interested, and separates these entities into their own tables. The second part of this design consists of the relationships between entities. The purpose of this model is to remove any redundancy of data within the database, thereby greatly improving update processing by limiting changes to a single piece of information. This model has been most frequently used to design systems, and it can be very complex to understand and interpret by users. It functions best in operational environments where the data is accessed by programs and transactions.

A star schema model is a design strategy that communicates clear and simple relationships between all the pieces of information stored in the database. A simple star consists of a group of tables that describe the dimensions of the business arranged logically around a huge central table that contains all the accumulated facts and figures of the business.

For more information about data modeling, see *Data Warehouse with DB2 for OS/390*, SG24-2249. For information about logical DB design, see *Data Modeling Techniques for Data Warehousing,* SG24-2238.

Generally, the primary rule for designing a mart (or warehouse) is to keep the database design simple and easy for occasional users to understand and interpret. The database should be visualized as a schema that the business analysts can readily use, thereby allowing them to gain real value from the mart. A data mart may contain the following types of data:

- Dimensional data
- A subset of the data warehouse
- Aggregated data

The database for a mart is designed so that:

- The schema is simple
- Query processing is relatively fast
- Data load can be accomplished fairly quickly

In our case studies, the following modelling techniques were used:

- Banking customer - entity relationship

- Health insurance customer - star schema

Figure 39 and Figure 40 on page 110 describe a high-level data model of the data mart used in our case studies for the Banking and Health Insurance customers.

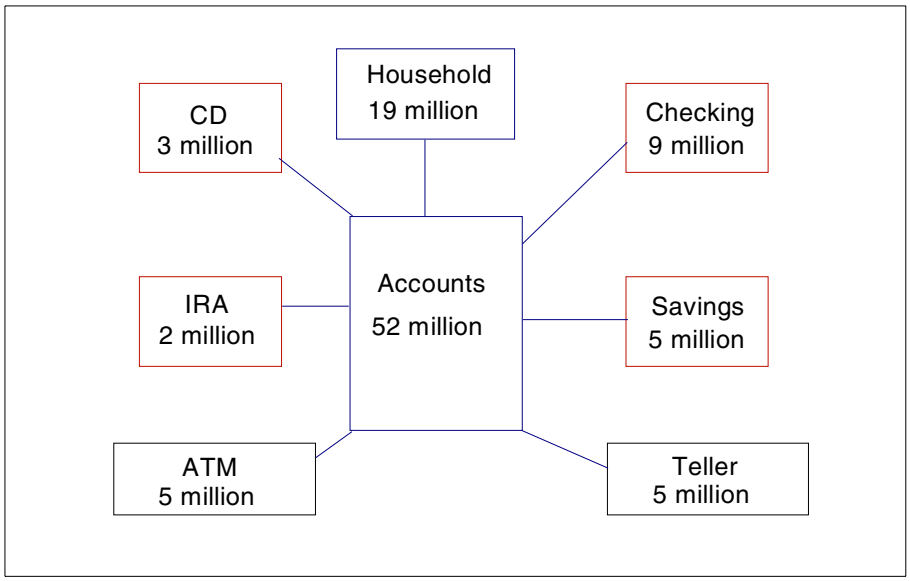


Figure 39. Data model of a data mart for Banking customer

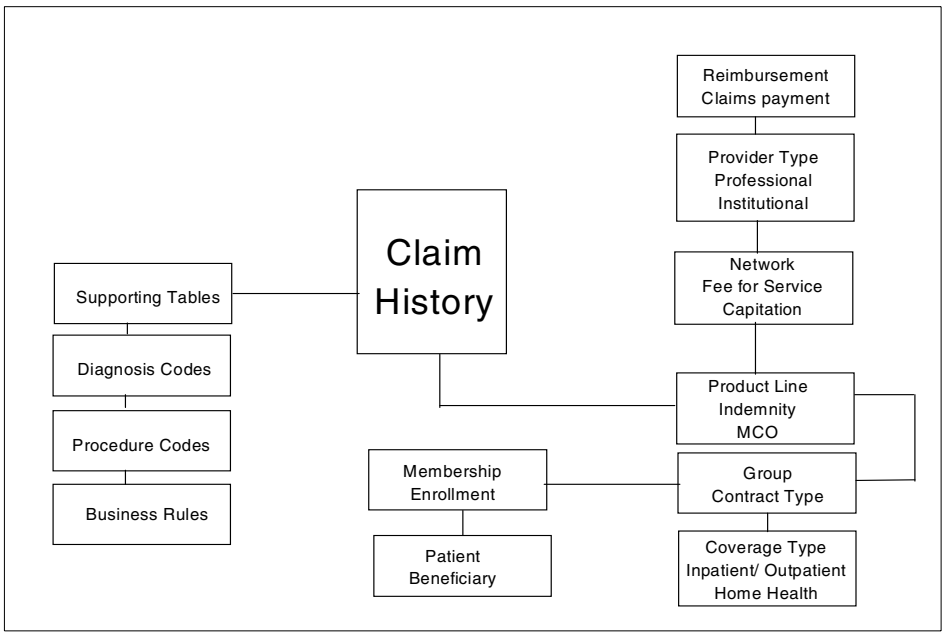


Figure 40. Data model of a data mart for Health Insurance customer

Data mart population process

Once the mart design is established, the mart is built by performing the following tasks:

- Create the database and its tables
- Populate rows into the table
- Build indexes
- Perform RUNSTATS

5.3.1 Create the database and its tables

To begin the process, it is necessary to generate the data definition language (DDL) to define the database structure for the data. This requires the translation of the logical database structure into the physical structure for the system, similar to building the database as discussed in Chapter 2. Refer to that section if you have not read it at this point. Additionally, refer to *Data Warehousing with DB2 for OS/390,* SG24-2249.

5.3.2 Initial load of data mart

Once the tables are created, it is necessary to extract the data from the warehouse and load it into the mart structure. There are two primary methods used to populate the data mart. Both were used during our case studies:

- Loading data by using the INSERT/Subselect SQL statement
- Loading data by using DB2 utilities with or without piping

Table 12 shows some differences between the SQL INSERT statement and the LOAD utility.

Table 12. Differences between SQL INSERT and LOAD utility

Description	SQL INSERT	DSNTIAUL/DB2-LOAD
Parallel extract process	Yes	Yes
Parallel load process	Yes*1	Yes
Partition is available for queries before load procedure has ended	Yes	No
Logging	Yes	User specified*2
Incremental image copy after the load	Yes	No
Execute the Copy utility concurrently with your queries	Yes	Yes
Usage of subselect clause	Yes	Yes
Volume to Load	Small	Large
Use of piping technology	No	Yes

*1 Using multiple INSERT statements by partition, parallel load is possible.

*2 After executing the load utility with LOG NO, an image copy is highly recommended.

Either method can be used, and each organization should consider what is appropriate for their company. When doing the initial load, consider the following items to determine which approach is best:

- Available work space
- Volume of data to be populated and the impact on DB2 logs
- Partitioning of source and target tables
- Sufficient hardware capacity for the duration of the population process

5.3.2.1 SQL method

For data marts, where the extraction rules are simple, it may be advantageous to use INSERT INTO ... SELECT FROM, especially when the data mart is small compared to the amount of the data in the data warehouse tables. With DB2 for OS/390, the SELECT may be parallelized automatically. The INSERT phase is serialized. Should full parallelism be needed, multiple INSERT statements should be used.

Figure 41 shows the data flow from a data warehouse to a data mart. A DB2 internal sort can occur when there is a GROUP BY or ORDER BY clause in the sub-select statement.

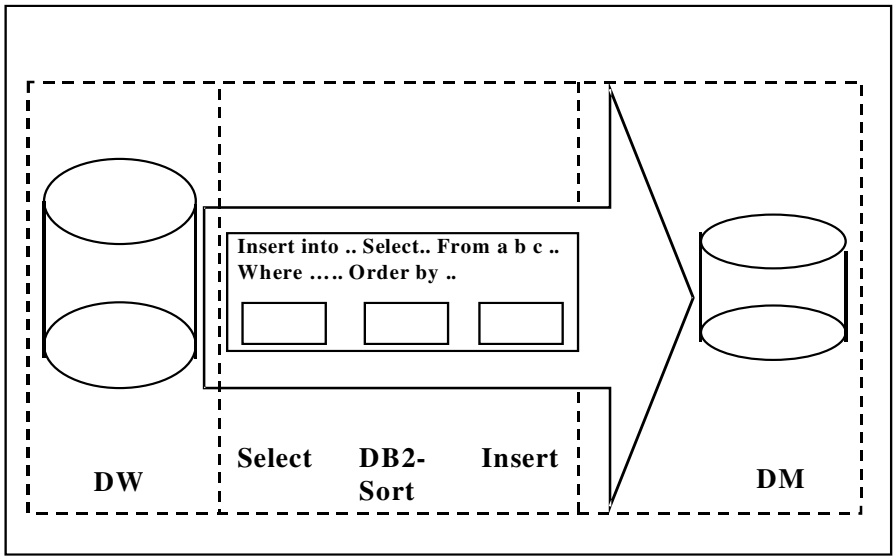


Figure 41. Load of the data mart with the SQL method

When using the SQL method, you should consider the followings:

Space

Significant sort work space may be needed when executing an ORDER BY or GROUP BY clause in the sub-select statement. The amount of space depends on how complex the statement is and the number of rows to be sorted and produced. By implication, extracting from VLDB tables can result in large mart table volumes as well. Should the mart table populations execute with end-user query activity, sufficient DB2 temporary work space (DSNDB07 if non-data sharing) and buffer pools must be allocated to support the concurrency level. Chapter 2 provides formulas for estimating this.

Logging

When using the SQL statement INSERT INTO .. SELECT ..FROM, the logging option is YES. Therefore, DB2 will record each INSERT into the DB2 active log. This can have an impact on elapsed time and require additional active log disk space.

Parallelism/partitioning

DB2 query parallelism will not occur during the INSERT phase of the INSERT/sub-SELECT statement. To achieve insert parallelism, multiple INSERT/sub-SELECT statements should be used with the WHERE clause including predicates that map to the partitioning keys of the target mart table as well as the very large source warehouse tables. Multiple INSERT statements which put rows into different partitions of the same table can be executed in parallel with minimal I/O and locking contention. See 2.3, “Partitioning strategy” on page 23, for detailed information about partitioning.

Resources

The INSERT statement will consume more CPU resources than the DB2 LOAD utility to place the data into a table. The INSERT/sub-SELECT is also one unit of work and may be long-running. Other query users on the system who have a need for the same resource, such as temporary work space, may be impacted because some DB2 resources for SQL statement execution are shared by all users.

5.3.2.2 DB2 utility method

All types of data marts can use the DB2 utility method, as shown in Figure 42.

When using this method, you can unload data from the data warehouse using DSNTIAUL (a DB2 sample program) or your own applications, sort the data with DFSORT if required, and then load the data into data mart tables using the DB2 load utility. In DB2 Version 6, the REORG utility has been enhanced to produce unload formats that can be used as input to the LOAD utility. It is significantly faster than DSNTIAUL.

Parallelizing the unload process is much easier with partitioned tables. Moreover, the LOAD utility can load data into partitions in parallel. The benefits obtained by utilities and parallelism can be incorporated into the entire data warehouse process from data capture to the load**.**

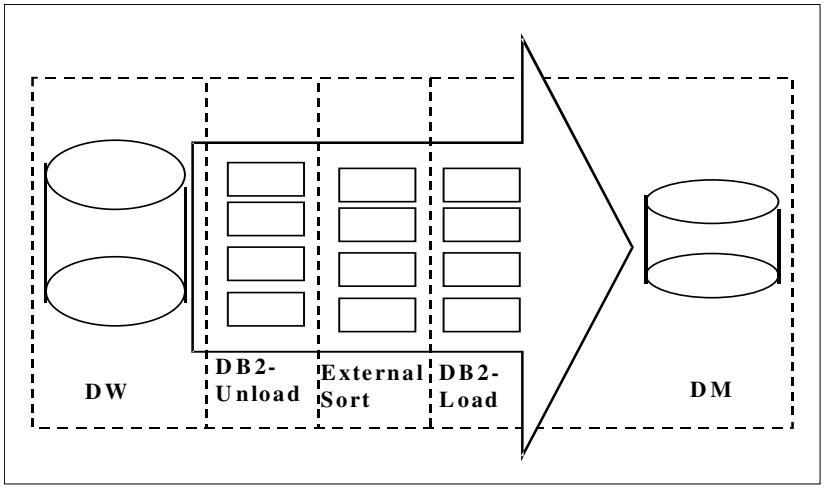


Figure 42. Load of the data mart with DB2 utility

Space

The unload activities need OS/390 work space for the unloaded data; the external sort (DFSORT) also needs sort work data sets. Ensure the shared OS/390 DASD work pool is sufficient to handle the volumes for the mart population process concurrent with other batch processing in the system.

Logging

By specifying LOG NO for LOAD utility, no records will be written to the DB2 active log. This is the preferred specification when loading more than a few thousand rows into a table. We recommend you take an image copy after each load and only before updates are allowed.

Parallelism/partitioning

Partitioning is strongly recommended for large tables, whether they are in a data warehouse or data mart. See 2.3, “Partitioning strategy” on page 23 for detailed information. Implementation of parallelism is simple. Once a partitioning strategy is developed and you can generate multiple unload data sets by executing multiple unload jobs by partition, sort the output data sets by the partition key of the target mart table if required, and then run multiple load utility jobs by partition (with LOAD .. INTO TABLE ... PART.. control statement) in parallel.

Resources

This method makes very efficient use of system resources.The degree of job parallelism is connected to the amount of CPU and DASD work space available as well as the desired elapsed time. Multiple independent units of work or jobs can be scheduled and managed independent of each other throughout the population process or can be simply run end-to-end with SmartBatch to minimize elapsed time. WLM can dynamically allocate resources to the population jobs when spare cycles are not needed by the query community or the population jobs can be favored over the query workload.

Combined parallelism and piping technology

BatchPipes facilitates the parallel execution of jobs that extract data from a warehouse, sort the data, and apply it to the mart. Without piping technology, the DB2 programs require DASD space to temporarily house the unloaded data. The data is then read back in and sorted into the partitioning key sequence of the mart tables, and loaded with the utilities. This process results in lots of I/O and increased elapsed time. With piping technology (described in Chapter 3 for data warehouse population), you need less work space, less I/O, and reduced elapsed time for the whole data population process for the marts.

Figure 43 on page 116 shows the possible parallel load options with DB2 utilities. You can choose one of them based on table structure, skills, and system resources.

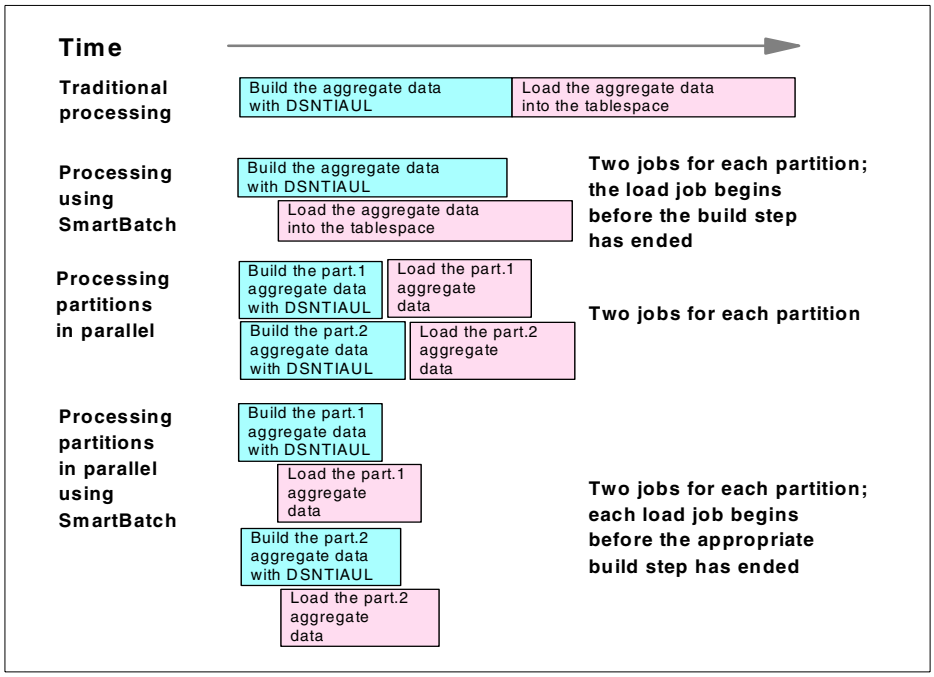


Figure 43. Elapsed time in data mart population using DB2 utilities

For more information, refer to *Data Warehousing with DB2 for OS/390*, SG24-2249.

5.4 Data mart maintenance

Data mart maintenance may be simple or complex, just like the data warehouse. End-user requirements for the currency of their data determines the maintenance strategy for a data mart.

Figure 44 shows how the data mart maintenace cycle is related to the life time of data mart. This data mart maintenance cycle is related to the data warehouse's maintenance periodicity; however, business needs should decide the final frequency at which marts are updated.

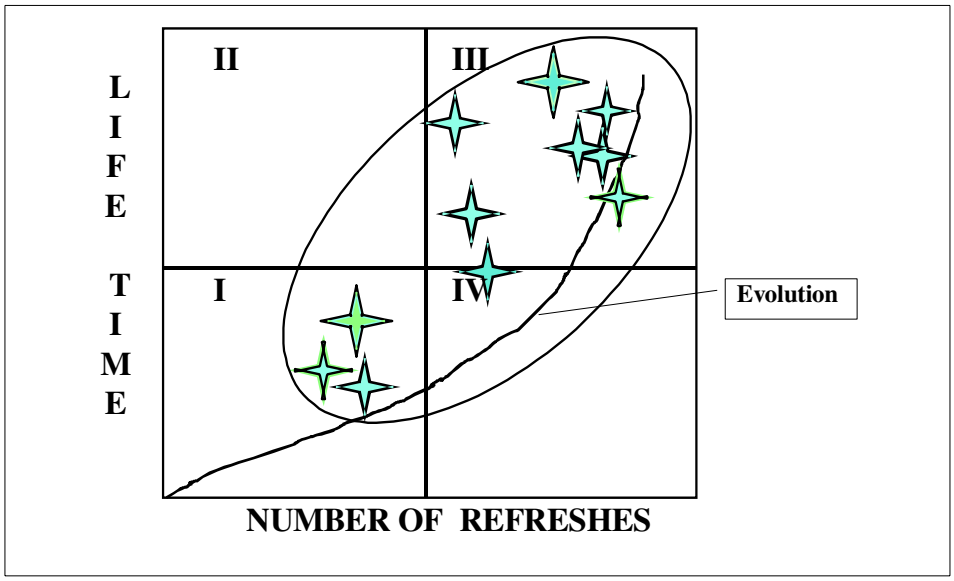


Figure 44. The data mart maintenance strategy

Other parameters which affect the strategy are:

- Volume
- Batch window

In the VLDB environment, the maintenance volumes and the batch window are the critical parameters. In Chapter 3 "Data warehouse population" on page 57 we describe a method to parallelize the initial load and data maintenance in order to use resources efficiently and to have the shortest possible execution time. The objectives of ensuring that end users have current data when needed, and of fitting the maintenance process into the batch window, are in direct conflict with each other.

The serial process of moving data from the data warehouse to the data mart is often too slow. For a very large data warehouse, it is necessary to think in parallel.

In general, to improve the elapsed time of the data mart maintenance process, a part of the data mart process may be implemented with the data warehouse maintenance process after the data has been cleansed and transformed. While the data warehouse is being maintained by using the LOAD utility, the same input file could be extracted for data mart maintenance; sometimes a simple sort INCLUDE will suffice. The extracted

data can then be stored on a different set of devices (DASD or tape), or can be piped directly into the data mart maintenance process.

Whether such an implementation is possible depends upon the business logic requirement at the mart as well as the original data warehouse maintenance strategy. For example, if the data warehouse maintenance strategy is implemented by using an SQL application with conditional business logic, the warehouse input file is likely not a candidate for a simple extract for use at the mart. However, if a warehouse maintenance program performed the conditional logic processing and produced a sequential image of maintained rows, that file may be usable as input to the data mart maintenance process. When there are multiple data marts in the BI environment, mission-critical marts should be maintained first.

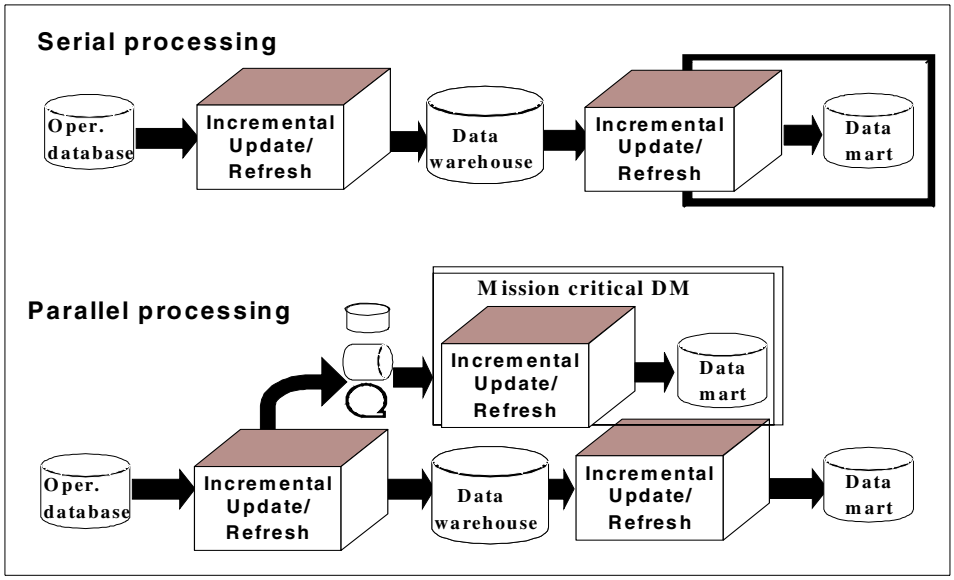


Figure 45. The data mart maintenance

The maintenance process for data marts can be executed serially or in parallel. In Figure 45, *parallel processing* means that several data marts can be loaded in parallel with each and/or in parallel with the data warehouse.

Pros (+) and cons (-):

+ Maintenance can be done in a shorter time than serial processing.

+ Maintenance is likely to fit into the batch window.

- The maintenance process for the data warehouse becomes complex. A redesign of the data mart can alter the design of the data warehouse maintenance process.

- More CPU, DASD and tape resources are needed during parallel execution of multiple job streams.

- A parallel maintenance process requires a solid understanding of the data warehouse architecture, business needs, and piping technology.

5.5 Resource management: intelligent resource sharing

While combining mart and warehouse workloads on a single server may seem optimal from a resource, support, and maintenance perspective, activities such as ad hoc queries and even some planned queries can absorb 100 percent of the processing capacity of an entire computer system.

To insure that all workloads are serviced, it is critical to have a sophisticated workload management capability in the system. This capability cannot be satisfied by the simplistic time slicing implementations currently being delivered on distributed platforms. The full breadth of this function is discussed in Chapter 7 of this book. Refer to that chapter for a clear understanding of the intelligent resource sharing that is unique to the OS/390 environment.

With the advanced resource sharing features of S/390, such as LPAR and the OS/390 Workload Manager resource groups, multiple/diverse workloads are able to coexist on the same system. These intelligent resource sharing features can provide significant advantages over implementations where each workload runs on separate dedicated servers or with time slicing. Additionally, these S/390 capabilities provide the same ability to dedicate capacity to each business group (workload) that dedicated servers provide without the complexity and cost associated with managing common or heterogeneous software components across dedicated servers.

Another significant advantage comes as a result of the *idle cycle phenomenon* that can exist where dedicated servers are deployed for each workload. Inevitably there will be periods in the day where the capacity of these dedicated servers is not fully utilized, hence there will be *idle cycles*. A dedicated server for each workload strategy provides no means for other active workloads to tap or *share* the unused resources. These unused cycles, in effect, are wasted and generally go unaccounted when factoring overall cost of ownership. Features such as LPAR and WLM resource groups can be used to virtually eliminate the idle cycle phenomenon by allowing the sharing

of these unused cycles across business units (marts), automatically, without the need to manually reconfigure hardware or system parameters. The business value derived from minimizing these idle cycles maximizes the return on your dollar investment for the solution.

A final advantage comes from the fact that not only can capacity be guaranteed to individual business units (marts/workloads); but through this intelligent resource sharing, business units can make use of additional cycles unused from other business units. The result is a guarantee to each business unit of X MIPs with a potential for X+ MIPs..., and who could refuse? This results in *bonus capacity*, that comes from the sophisticated resource management features available on S/390.

5.6 Case study tests

The test cases for data mart build and execution included:

- Extraction of data from cleansed, transformed VLDB data warehouse tables by using SQL statements submitted through DSNTEP2 or DSNTIAUL, IBM-supplied DB2 sample batch programs which can execute one or more SQL statements.
- Placement of data into the data mart tables by using SQL or the DB2 load utility.
- Validation that hardware resource allocation of combined data warehouse and data mart extract, load, and query activities on a single system can be simply prioritized and managed with the OS/390 Workload Manager (WLM).

Each case study test had unique objectives that focused on the functionality of S/390 capabilities:

- Parallel processing techniques for data mart population: could the data mart tables be simply and quickly populated given the very large source tables in the warehouse and potentially very large data mart tables?
- Workload management: would WLM allocate hardware resources fairly between the mart and the warehouse query workload according to installation-defined parameters?

For the business queries, both customers supplied typical business scenario questions which are frequently requested by the end-user community, but which previously required multiple weeks for answers or were unanswerable due to never-ending queries or lack of real-time, dynamic access to electronically captured historical data.

We did three types of case study tests:

- Data mart population by using the SQL statement method
- Data mart population using the DB2 LOAD utility
- Intelligent resource sharing using WLM

5.6.1 Data mart population

5.6.1.1 Data mart models

The data mart data models for both case studies are very similar to their respective data warehouse models. In the banking case study, the customer anticipated a significant number of future requests for smaller subset table extracts of the warehouse data. Understanding ease of use and the performance implications of the various techniques available to build the mart tables with the given product set overrode the data model concerns. The entity relationship (ER) model was retained for the data mart test cases. See Figure 39 on page 109.

The Health Insurance case study utilized a star schema implementation in its data mart rather than the data warehouse ER model to facilitate data access using an OLAP tool. See Figure 40 on page 110.

5.6.1.2 SQL method (DMT-A-1)

Case study A used a DB2-supplied sample batch program DSNTEP2 to submit the SQL statements for the tests because the transformation logic could be easily accomplished with SQL functions and the mart table build was estimated to require more than a few minutes.

Objectives

Show that summary and subset tables can be efficiently constructed from the data warehouse using SQL-statements only.

Metrics

Table 13. Metrics for building a subset table for case study A

Objects	Amount/Description
DW size in TB (raw)	5.0 TB (3 years)
No. of rows in DW	500 million
Unload/Sort/Load time	38 minutes
No. of rows in DM	5.5 million
DM size in GB (raw)	1GB
Number of tables (DM)	1
Number of partitions (DM)	-

For configuration of the test system, see Figure 2.

Method

Four steps were included in a job:

- Create the subset table.
- Insert/subselect the data warehouse source rows into the data mart subset table.
- Build an index.
- Execute the RUNSTATS utility.

Case study test result

Space

In some of the population tests, the sub-select query elapsed time increased noticeably. DB2PM reports indicated the elongation in elapsed time was due to I/O contention on the DB2 work file data sets. With the INSERT INTO...SELECT ... FROM-statement for an aggregate table build over hundreds of millions of rows, DB2 was performing using the GROUP BY and ORDER BY functions while other warehouse and mart queries were also executing multi-table joins, creating temp tables, and using other DB2 functions which required the use of the shared DB2 temporary work space. Work files data sets were increased to eliminate the I/O contention.

Note

Care must be exercised when determining the number and size of the work space and sort work space. Actually, there is no separate sort work space. When doing sorts for queries, DB2 uses its buffer pool together with the physical work file database (DSNDB07 if non-data sharing), as needed. 2.8.1, “Work file data sets” on page 43 provides a method for calculating the number of work file data sets. We only observed contention on the work file data sets associated to the database (DSNDB07 or equivalent), not the associated buffer pool and hiperpool.

Logging

DB2 logging is automatically done, no option, when you use the SQL statement INSERT INTO .. SELECT ..FROM method. Each INSERT is logged in the DB2 active log. The volume and length of rows to be logged during the INSERT phase exceeded the capacity of our active logs. To sustain maximum throughput during mart population, we enlarged our active logs to eliminate the archive offload.

Parallelism/Partitioning

The data mart population was parallelized by concurrently running several INSERT statements with the WHERE clause of the SELECT of the source data based on the partitioning keys of the source table.

Summary

Our recommendations from this case study are:

- Allocate enough work space when loading the data mart with SQL statements.
- Ensure that sufficient active logs are available to support the volume of inserted rows that may result from INSERT INTO .. SELECT ..FROM statement for VLDB tables.

5.6.1.3 DB2 utility method (DMT-B-1)

This test was done for case study B, the Health Insurance customer, and also used SQL statements to build the data mart rows. Due to the significant projected size of the individual mart tables, the faster-performing, less-resource-cost DB2 LOAD utility was chosen to populate the mart tables rather than insert statements.

Objective

Unload and load the data mart with standard DB2 utilities in parallel.

Metrics

Table 14. Metrics for the Initial load test for case study B

Objects	Amount/Description
DW size in TB (raw)	0.8 TB (37 states 3 years)
No. of rows in data warehouse	6.3 billion
Unload/sort/Load time	5.2 hours
No. of rows in data mart	560 million
DM size in GB (raw)	30
Number of tables	4 fact tables and 8 dimension tables
Number of partitions	60 per table

For the configuration of the test, see Figure 3 on page 13.

Method

We used DSNTIAUL, an IBM-supplied DB2 sample program, which executes a single SQL SELECT statement to produce an unload file and an associated DB2 load control card, DFSORT to sort/merge the unloaded records, and the DB2 LOAD utility to load the data mart.

Extracts from the health insurance data warehouse were built using the DB2 unload program DSNTIAUL. Multiple unload jobs ran concurrently. The unload statements for the fact tables included joins between three tables. Permanent DASD allocations were specified to ensure restartability after the extract step. We used DFSORT to sort the output files into target mart table clustering key order, and loaded the data with the DB2 load utility.

Case study test result

The unload, sort, and load were executed in parallel. Results of our test are:

Space

Multiple batch parallel jobs executing concurrently placed a high demand on OS/390 work space DASD to store the extracts and then sort them. Given the very large output files from the joined extracts as input into the

following DFSORT steps, trade-offs had to be made between the batch job concurrency level to meet the desired elapsed time for multiple mart table populations versus the available DASD to support the unloads and sorts.

Logging

Because of the high volume of rows to load, the DB2 LOAD utility log setting was set to NO to maximize throughput; no records are written to the DB2 active log. Image copies were made to enable recoverability.

Partitioning

The projected size of the mart tables dictated usage of partitioning. New partitioning keys had to be determined because the mart tables represented joins of multiple warehouse tables. We decided to change from the data warehouse geographic-oriented key to a claim-type partitioning to facilitate the departmental OLAP query access. This also encouraged the decision to use of the extract/load utility method to maximize mart population I/O efficiency in both the warehouse and mart.

Resources

This method makes very efficient use of resources; the DB2 LOAD utility is more efficient than INSERT for placing rows into a table. The degree of job parallelism to implement was defined by the amount of physical DASD and CPU resource available and the desired elapsed time

Summary

Our recommendations from this case study test are:

- Analyze and allocate sufficient DB2 work space for the extract if joining tables or sorting in the extract SQL statement and OS/390 DASD space to store the extracts, to support the sort of the extract data sets, and the sorts for indexes during the LOAD utility.
- Use SmartBatch or a similar product if piping is required. For very large table builds, when system load management becomes too complex, consider using SmartBatch to reduce the DASD requirements and increase throughput and WLM to manage the concurrency.

5.6.2 Intelligent resource sharing between DM and DW

This case study test demonstrates S/390 can support both a data mart (DM) and a data warehouse (DW) on the same system in a managed way using Workload Manager (WLM) to manage the concurrency and priority of both workloads. (Chapter 7 "Workload management in data warehouse environment" on page 163 contains additional information about WLM and other tests.)

Objectives

- Prove WLM provides the resource management capabilities necessary for both workloads to meet their specified goals.
- Prove the efficient use of total system resources by allowing the data mart to utilize any additional resource that the data warehouse is not using.
- Prove the ability to cap the amount of processing resources used by a given workload.
- Show the dynamics of WLM by changing the policies on the fly.

Method

WLM resource groups were designed to utilize/manage the amount of processing resources available for the data warehouse and data mart workloads.

We executed the general ad hoc query workload for the data warehouse at an active concurrency level of 150 users (well beyond saturation for its allocation of processor resources) and 50 active concurrent users for the data mart. Six batch data mart queries (three per server) were added to the data mart workload to saturate the system. The data mart exists on the same servers but has its own DASD pool and DB2 subsystems.

In both cases a maximum capacity of 100 percent was defined for the resource group to allow any unused capacity to be absorbed by either workload. That is to say if either workload was using less capacity than its minimum, then the other resource group (workload) could use any idle cycles.

Test configuration

Table 15. Configuration for the test

	DMT-A-21	DMT-A-22	DMT-A-23	DMT-A-24
Database	DW/MART	DW/MART	DW/MART	DW/MART
No. of servers	2	2	2	2
No. trivial users	15/0	15/0	2/0	2/0
No. small users	15/40	15/40	2/40	2/40
No. medium users	15/10	15/10	2/10	2/10
No. large users	105/0	105/0	4/0	4/0
No. total users	150/50	150/50	10/50	10/50

	DMT-A-21	DMT-A-22	DMT-A-23	DMT-A-24
WLM policy	policy 1	policy 2	policy 1	policy 1c
CPU allocation	75/25	50/50	75/25	75/25c
Measurement interval	1 hour	1 hour	1 hour	1 hour

Table 15 shows the query workload combination and the allowable processor consumption for the data warehouse and data mart workloads as defined by the WLM policy used in each case.

1. Test case: DMT-A-21

Each resource group was assigned specified minimum and maximum capacity allocations.

This measurement shows the WLM policy giving a minimum of 75 percent of the system resources to the data warehouse and 25 percent of the system resources to the data mart.

2. Test case: DMT-A-22

This measurement shows the WLM policy giving equal processing resources to both workloads: 50 percent of available processor went to the data warehouse and 50 percent went to the data mart.

3. Test case: DMT-A-23

This measurement shows both workloads still with 75 percent of the processing resources allocated to the data warehouse and 25 percent allocated to the data mart.

The difference is that the warehouse users have been reduced to only 10, and they are not able to consume their full 75 percent. WLM has the capability to give those idle cycles to another workload until they are needed. Here you see the data mart making use of the unused cycles.

4. Test case: DMT-A-24

This measurement shows both workloads still with 75 percent of the processing resources allocated to the data warehouse and 25 percent allocated to the data mart.

The difference is that a cap has been put on the resource consumption of the mart.

Case study test results

Figure 46 shows the test results which examined how WLM resource groups, one for the data mart and one for the data warehouse, can be used to enable the consolidation of a data mart and a data warehouse on the same system while guaranteeing dedicated capacity for each business unit (workload).

Results show capacity distribution very closely matches the desired distribution.

Test DMT-A-21 was the initial test to show that defined allocation and actual usage would correlate.

Test DMT-A-22 was done to show that capacity allocations could be dynamically altered without interference to workloads. A dynamic switch from policy 1 to policy 2 was done with the full workload already executing on the system. Such dynamic alterations of capacity allocations could be automated to reflect business preference of certain workloads at different periods of the day, week, month, or year.

Test DMT-A-23 demonstrated the ability of one workload to absorb unused capacity of another workload when workload activity subsides on the data warehouse. The concurrent warehouse query level was significantly lowered. The CPU usage for the data mart increased to 68 percent from 27 percent (which is from DMT-A-21) because the warehouse no longer was consuming its allocated 75 percent. Its idle cycles were absorbed by the data mart, accelerating the processing while maintaining constant use of all available resources.

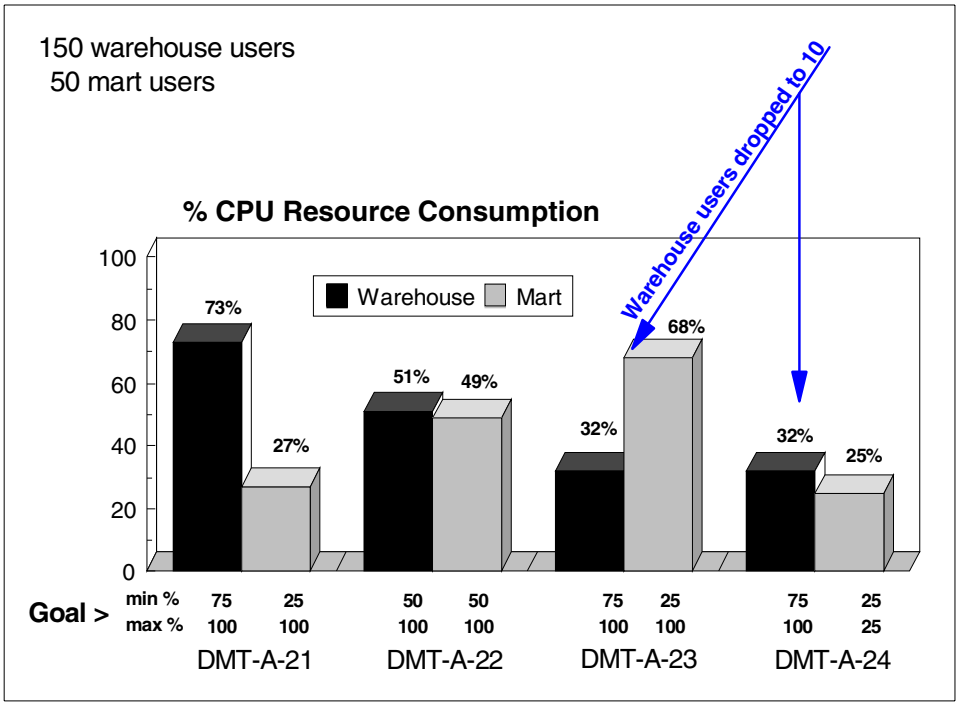


Figure 46. Intelligent resource sharing

The final test (DMT-A-24) demonstrated that even though there were idle resources from the 75 percent allocation to the warehouse and the mart could use more resources to satisfy the demands (as shown in DMT-A-23), WLM could be configured to not allow the mart to use more than 25 percent of processing resources.

Some comments

- Four measurements comprised the test case. To demonstrate the dynamics of WLM, all four measurements were run sequentially as one big measurement. The total test interval, including all four measurements (each measurement interval was one hour), as well as warm-up and cancelling of users, was more than seven hours.
- There was also a three-hour warm-up period for test case DMT-A-21 settings to get the system into a steady state.
- The WLM policy and the number of users were changed on the fly for each measurement.

- Between test case DMT-A-22 and DMT-A-23, 140 data warehouse users were cancelled. This was done over a 15-minute period to give the system time to adjust.

Summary

Our experience from the case study is:

WLM correctly allocates system resources to different workloads according to installation-specified targets in resource group definitions.

Data marts and the data warehouse can coexist on the same system with guaranteed allocation of hardware resources to each. In addition, cycles, if unused due to low activity in one workload, can be used by the other workload automatically. Thus maximum return from the hardware may be achieved.

Data marts and the data warehouse can coexist on the same system with either or both prevented from utilizing more hardware resources than desired, even if unused cycles are available.

5.7 Summary

S/390 is an optimum platform for building and housing data marts for some very key reasons:

- The ease of moving data from the warehouse to the mart, avoiding lengthy transmission and format changes to data, is an important consideration not only for the initial load, but for the entire life of the mart. The ease of sharing data within the same system allows for easier user access.
- Sharing data across systems is facilitated through data sharing, which also minimizes/eliminates the volume of data that must be moved from one system to another.
- Parallel processing of queries and DB2 utilities optimizes performance.
- Centralizing processor resources allows for more effective sharing of the capacity, without creating isolated islands of capacity.
- WLM functions permit integration of data warehouse and data mart queries within the same system, while delivering the needed capacity to each group of users.

As Business Intelligence systems become more critical to business operations, their users will demand significantly higher levels of security and availability, the hallmarks of the S/390 environment. When considering the

appropriate platform for these applications, it is important to consider not only the users' needs today, but 24 and 36 months from now.

Selecting the right platform today will avoid timely and costly conversions when the users' requirements exceed the capabilities of a platform selected for its initially low cost. Additional data and information cited by the consultant community on these items and others are listed in Appendix E, “Reasons for choosing S/390” on page 255.

Chapter 6. End-user access methodologies

A very important aspect in designing BI applications is determining what end-user access methodologies to use. The methods used will vary depending on the overall environment in which the data warehouse resides. You must consider the existing network structure, end-user specifications and the availability of specific tools, as well as IT system requirements. You must also decide if end users are to directly access the data warehouse, the data marts, or both. The decision is dependent on your business and end-user requirements, and should be determined early in the data warehouse design in order to incorporate potential co-requisite modifications into the architectural design.

In this chapter, we introduce various means of accessing DB2 data in a BI environment with the intention that it may serve as a reference for you when you analyze your own environment.

This chapter focuses on three topics:

- Tool selection considerations
- Options for end-user data access
- Case studies

6.1 Tool selection considerations

A business that seeks to make its data mart or data warehouse accessible by end users must define a comprehensive implementation strategy. You must consider many factors:

- The end-user functionality requirements, for example simple reporting, analytical processing, and/or sophisticated data mining
- The current network topology and additional bandwidth requirements of the BI application, for example volume of data to be processed by the tool and volume of data to be transmitted to the end-user workstation
- The network connectivity desired to support the current and future user population including intra-, inter- and extra-company access to the data warehouse or mart
  - ODBC protocol for data access only
  - HTTP for Web browser to Web application access
  - Mainframe dialup (ODBC, Web, direct host application)

- The security and audit policies governing the access of the data

Only then can you make wise decisions regarding end-user tools; tool selection is more than simply a matter of evaluating a tool for analytic and presentation functionality. You should focus on the following tasks:

1. Understand the general inter-dependencies between your current IT hardware and software infrastructure requirements, the BI application toolsets under consideration, and the organization's security and audit policies
2. Determine analytic objectives for your end-user solutions so the correct application tools are chosen
3. Investigate various end-user software products focusing on joint business and IT requirements and then choose the one that best fits your environment

To assist you in understanding the issues, a list of questions is provided. The answers to these questions should help you decide “What issues must the business consider to accommodate end-user data access?” This is not a complete list, but rather a guide to get you started.

- What type of network configuration currently exists?
- Can the existing network handle additional workload?
- Will changes have to be made to the network configuration to accommodate end-user access to the data warehouse?
- Will additional end user or system hardware be required? If so, how much? What kind?
- How will end users access data? From the Web? Using client-server tools? From TSO?
- If client/server access is desired, has DB2 been implemented with distributed work?
- Will there be “power” users with special privileges? If so, how many?
- What types of user access awareness to the data warehouse/data mart is required by the business? Does the tool interface to DB2 support or inhibit the ability to meet this requirement?
- What type of resource allocations will end users require versus production operational work?
- What type of data are end users interested in viewing? Is it primarily pre-generated reports or will they perform ad hoc queries?

- What tools are readily available inhouse to accommodate end-user access to the data? Will additional software need to be ordered or tested?

Hopefully, these questions raised other questions and helped you realize the importance of an organized approach. Be sure you address these questions before you proceed.

To obtain the right end-user solution for your company, you must first set objectives; what must be achieved with these products to fulfill business requirements, yet ensure a stable IT environment? A number of common objectives follow:

1. Good query performance as the user population increases
2. Proper workload balancing
3. Prioritization of incoming requests with existing queries
4. Data accessibility at multiple levels of detail
5. Good reporting capabilities

These objectives are discussed further in the following sections.

6.2 Options for end-user data access

Once you have addressed the issues related to end-user access and have established your goals and objectives, it is time to assess various tools and software products that will help you achieve your goals. In this chapter, we introduce a few products and tools that run with S/390. They are categorized by access method, that is, those that enable Web access and those that strictly use distributed access to DB2 using the DRDA protocol. All of the products and tools we mention were used in our case study tests at the Teraplex Center. We provide you with actual examples of how they work and the benefits that can be obtained from using them. All tests were done within the scope of customer objectives.

6.2.1 Web enablement

Businesses today are realizing that accessing data through the Web is an integral part of their business processes. Information is now readily available with a click of a mouse. In terms of data access, Web pages allow users to pose ad hoc queries, view reports and invoke predefined queries from Web menus. A data warehouse on S/390, therefore, becomes a prime repository for data accessible from the Web. But what software products are available to simplify Web access to the data warehouse and what performance impact will Web access have to the end user?

6.2.1.1 Net.Data

Net.Data is a no-charge feature of DB2 for OS/390. Net.Data enables both intranet and Internet access to relational data on various platforms, the more common databases being DB2, Oracle and Sybase. Net.Data runs as a common gateway interface (CGI) program or uses the Go Webserver Application Programming Interface (GWAPI). It is used to create dynamic Web pages with macros. These macros combine HTML with Net.Data's own macro language to invoke SQL, as well as provide language support for Java, REXX, Perl and C++. It enhances your ability to create your own Web page applications tailored to your end users.

To use Net.Data for OS/390, you only need a Web browser on the client. The Web browser interacts with the local Web server, Lotus Domino Go Webserver for OS/390, which runs the Net.Data Macro Processor (Figure 47 on page 136). This processor directly interacts with DB2 for OS/390 using either the Call Attach Facility, if Net.Data runs as a CGI program, or the RRS Attach Facility, if run as a GWAPI. The macro language is used to create the dynamic Web pages that appear on the client's machine.

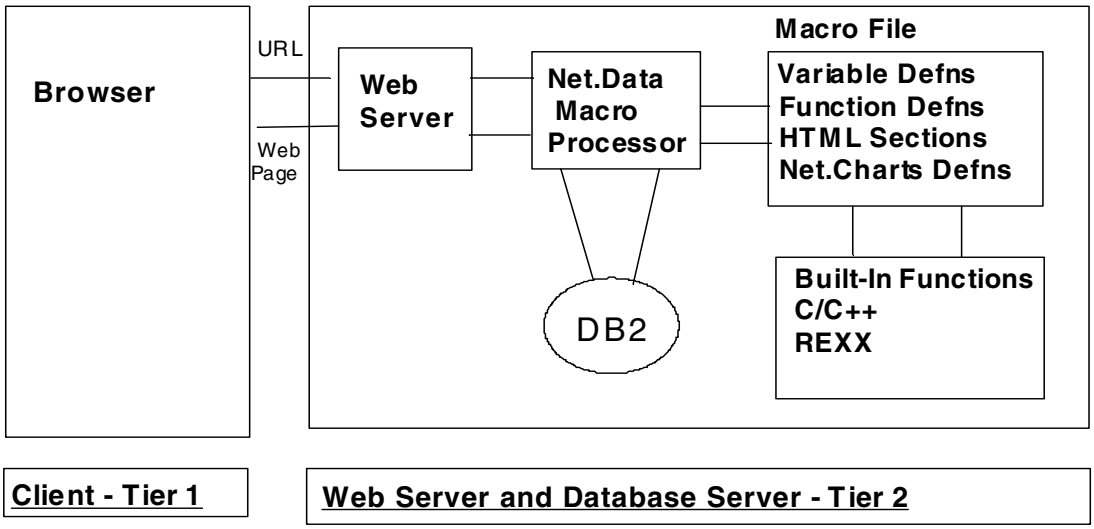


Figure 47. Net.Data connectivity

Using Net.Data offers the following advantages:

- It is useful for ad hoc queries because interactive Web pages can be developed to allow end users to specify their own search criteria (thereby creating ad hoc queries). The SQL always remains hidden to the user.

- It enables easy-to-create Web pages: a simple macro language mixed with HTML provides an easy way to develop your own Web pages.
- It integrates well with any Web browser to create end-user customized pages.
- It provides fast data retrieval: Net.Data with DB2 V5 boasts improved performance geared toward VLDBs.
- It uses the Call Attach Facility or the RRS Attach Facility to connect to DB2, thereby requiring no additional network software.
- It is a quick option for creating summary tables against large amounts of data, which is useful for end users who only need to see “the big picture”.

When Net.Data is defined to run as a CGI program, each time Web query is invoked, a new address space is created to service the request. Noticeable activity against the Web server *steplib* data sets may occur as each address space is started.

To rectify this, we recommend placing the Lotus Go Webserver load library (hlq.SIMWMOD1) and the Net.Data load library (hlq.SDTWLOAD) in LPALIB. Another option is to run Net.Data using GWAPI. This, however, requires the RRS address space.

Refer to `www.software.ibm.com/data/net.data/docs` for more information on Net.Data. Also refer to *Net.Data Administration and Programming Guide for OS/390.*

6.2.1.2 Lotus Domino Go Webserver for OS/390

Having a Web server reside on OS/390 is not a traditional approach, but there are advantages to doing this. As most businesses today store significant amounts of data in a S/390 environment, direct accessibility from the Internet to this data reduces the need to extract and transmit the data to alternative Web servers. The sophisticated systems management functions of OS/390 is another reason for a local Web server. OS/390 boasts strengths in capacity management, availability, security and data integrity; important features for Internet access as well as existing batch and transaction processing.

Lotus Domino Go Webserver for OS/390 supports Web-to-database application development with Net.Data. It is a powerful server that boasts a variety of server features like Java Web support, Web usage mining, client and server authentication, security features, data encryption, Simple Network Management Protocol (SNMP) support and expanded CGI support for Java, C, REXX and Pascal.

Lotus Domino Go Webserver can operate in three modes: standalone server mode, multiple server mode, and scalable server mode. In a *standalone* mode, you have a single Domino Go Webserver address space running in the system. It has a console interface so that the command can be issued at the operator console to communicate directly with the server address space (default name is IMWEBSRV). In the *multiple server* mode, it is possible to run multiple Domino Go Webservers on one OS/390 system. There can be multiple standalone servers, multiple scalable servers or a combination of both. In a *scalable* server mode, the Web server is configured to allow requests to be divided into application environments, each managed by WLM according to installation defined priorities and service goals.

Figure 48 gives an overall picture of how a Web request is processed by Lotus Domino Go Webserver in a *scalable* server mode. Incoming Web requests arrive at the Webserver Queue Manager. The URL is then used to determine the application environment in which to run the request. The request is passed to a queue server assigned to process requests for the application environment. If an appropriate queue server is not yet running, one is started. WLM determines how many queue servers to initiate based on the volume of requests for the application environment.

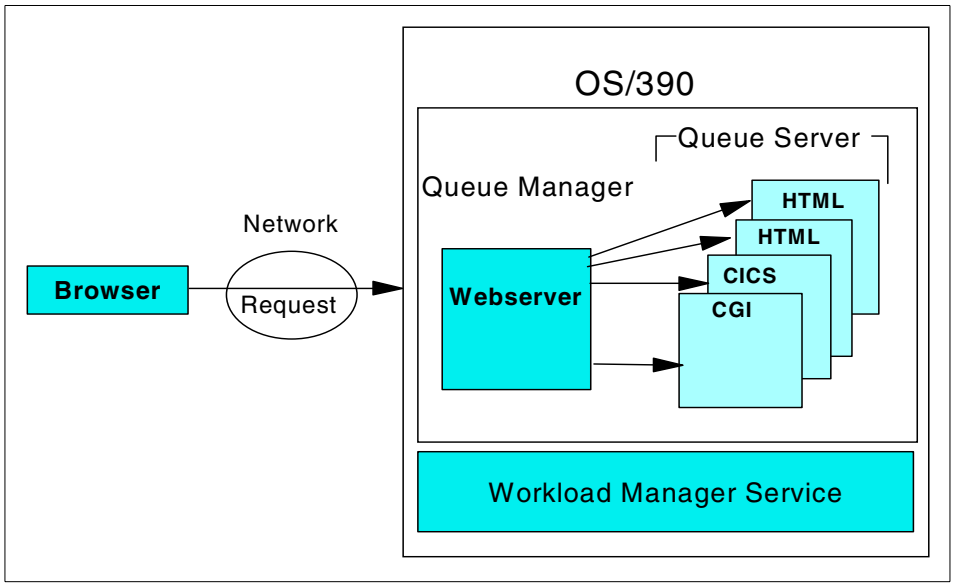


Figure 48. Lotus Domino Go Webserver environment in a scalable server mode

An example follows:

Transaction class for a Web request is an installation-chosen value, assigned according to the URL template specified in an application environment directive in the Web server configuration file. For example, the application environment directive:

APPLENV /web-cgi/db2www/hi* WEBHI WEBHI

means that all requests with URLs that begin with /web-cgi/db2www/hi* will run in application environment WEBHI, and be assigned WLM transaction class WEBHI. In WLM, IWEB is an OS/390 predefined subsystem used to classify work that enters the system via the Web server. A classification rule for IWEB such as:

Classification:

Default service class is WEB

Default report class is WEB

would then take that Web request which has been assigned transaction class WEBHI, and assign it to WLM service class WEBHI. Those URLs following WEBHI's specification will be serviced under WEBHI's WLM prioritization rules as opposed to the WEB rules. The WEBHI service class is given a higher priority than service class WEB, in which all other Web work ran.

Table 16. Example of WLM definition for a Web request

#	Qualifier Type	Qualifier Name	Starting Position	Service Class	Report Class
1	TC	WEBHI		WEBHI	WEBHI

Using the Domino Go Webserver offers the following advantages:

- It is located on the same operating system as the data warehouse, thereby reducing data access time.
- You are able to create a scalable network configuration to direct Web requests to specific application environments and prioritize the tasks.
- You are able to benefit from the strengths of the OS/390 operating environment.

Refer to *Domino Go Webserver Release 5.0 for OS/390 Web Master's Guide,* SC31-8691 and *OS/390 MVS Planning: Workload Management*, GC28-1761 for descriptions of setting up the scalable Web server. See Appendix A, "System settings" on page 225 for the Web server configuration file

statements and WLM panels showing the application environment and classification scheme used in our case study tests.

6.2.1.3 Network Dispatcher

Network Dispatcher, also known as SecureWay Network Dispatcher, runs on various routers like IBM's 2216 Nways Multi-protocol Router. It is a traffic management solution that can be configured to support multiple clusters of TCP/IP application servers. Network Dispatcher intercepts incoming requests and then decides where to route the request based on several factors:

- Number of current connections to each server
- Number of new connections routed to each server
- System capacity on each server
- Input from a protocol advisor

The system capacity value represents the available capacity on the system to do work, where a high number indicates the system has ample capacity for more work, and a low number indicates little capacity. Network Dispatcher periodically obtains from WLM. Thus a system that is busy running critical work is protected from being hit by new work, while allowing a system with minimal usage to receive additional work.

Network Dispatcher assigns each of the four factors a proportion per server. Depending on these values, Network Dispatcher will calculate the total weight to give each server within a cluster. Based on these weights, Network Dispatcher decides where to send new requests. For, example a server with a total weight of 80 will get 80 out of 100 new requests, while a server with a total weight of 20 will get 20 out of 100 new requests.

We used Network Dispatcher in our case study tests. It played a vital role in our scalable network configuration. Using Network Dispatcher offers the following advantages:

- It is transparent to the user.
- It recognizes when a server is down and redirects traffic.
- It does not require any software to be installed on the servers.
- It can balance the load across different servers of different sizes and different operating systems.
- It includes extended Java support.

Be aware of how Network Dispatcher determines system capacity when working with Workload Manager. As long as the importance of the work

running on a S/390 server is 2-7, Network Dispatcher will consider all of that server's capacity to be available, even if the server is 100 percent busy. The system capacity value only reflects capacity used by importance 1 work. An APAR is now available, OW39000, which modifies how system capacity is reported to Network Dispatcher. With the APAR, the system capacity value reported to Network Dispatcher will reflect capacity available at all importance levels.

For more information on configuring Network Dispatcher on a 2216 for your site, read Chapter 8 in *Nways MAS Using and Configuring Features Version 3.4,* SC30-3993. This book can be accessed on the Web at

`www.networking.ibm.com/216/216prod.html`

under **Documentation** and then **Technical Publications**. (The installation and configuration instructions in this manual are more detailed than those referenced in the eNetwork Dispatcher Web page.)

In this redbook, refer to Appendix A, "System settings" on page 225 to see how we configured the 2216 router for our case study tests. This includes the IOCP, HCD and TCP/IP statements.

For additional information on WWW access methods to DB2, refer to *WWW Access to DB2,* SG24-5273.

6.2.2 Distributed connections to DB2

A client-to-server distributed connection to DB2 involves a client residing on a workstation running OS/2 or Windows, for example, that uses DB2 Connect to establish a DRDA connection to DB2 for OS/390. The client typically runs tools that allow users to answer their business questions using a GUI interface, without the need to know SQL.

DB2 Connect allows clients to access host data and DB2 client application enabler (CAE) provides a run-time environment for database applications written in C, C++, Java, COBOL, FORTRAN, SmallTalk, REXX, and other programming languages. The latest development tools can be used to create database applications using Application Programming Interfaces (APIs) such as ODBC, JDBC, DB2 Embedded SQL and DB2 Call Level Interface. APIs allow you to use DB2 Connect with spreadsheets, decision support tools, database products and various development tools.

Two tools that use DB2 Connect as their vehicle to connect to DB2 are described in this section. Other products are listed in Chapter 1.

6.2.3 OLAP tools

As business sophistication grows, the need to view business-critical information in new formats is becoming key for decision makers. Users must have the ability to dissect the data and access it at various levels within dimensions to examine detail that before now was difficult to obtain. New technologies like online analytical processing engines (OLAP) have brought this information to our fingertips.

Why choose an OLAP tool?

- It allows you to view data in multiple dimensions.
- It allows you to drill down to various levels of the data to acquire specific information.
- It allows you to obtain detailed data fast.
- It allows you to obtain data that may have been previously unattainable because of the complexity involved in gathering it.

6.2.3.1 OLAP tool architectures

OLAP engines use either relational databases (ROLAP) or multidimensional databases (MOLAP) as their data repositories. See Figure 49.

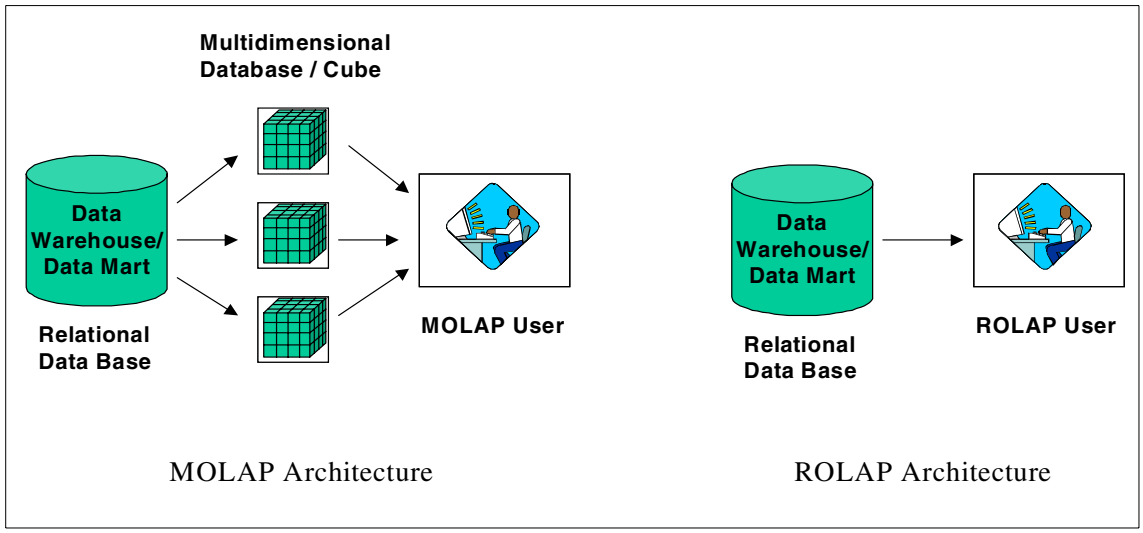


Figure 49. MOLAP and ROLAP architectures

MOLAP tools rely on multidimensional databases (MDDB) or cubes to establish the boundaries of the queries to be posed against its data. They are separate databases built from the data warehouse or data mart. A query

requiring data outside the dimensions of an MDDB cannot be posed until a new MDDB is defined. As a result, flexibility and scalability are limited. However, if the MDDBs are built such that they can answer a majority of the queries, then it can be extremely efficient. Because of the way the MDDB is designed, it tends to cater to pre-defined queries rather than to ad hoc queries.

ROLAP tools, on the other hand, do not require prebuilt dimension databases. Queries are posed directly to the relational database, giving ROLAP tools more flexibility and scalability. Data is separated into dimension and fact tables in a multidimensional data model. The dimension tables directly relate to data warehouse tables, therefore allowing cross-dimensional queries. The drawback is the possibility of having a lot of dimension tables with large amounts of data.

OLAP tool architectures can be multi-tiered. What does that mean? Basically, it means that additional layers may exist between the OLAP tool and the actual data. In a two-tier architecture, the OLAP tool directly accesses the data warehouse without any intermediary and controls the caching of online reports. See Figure 50.

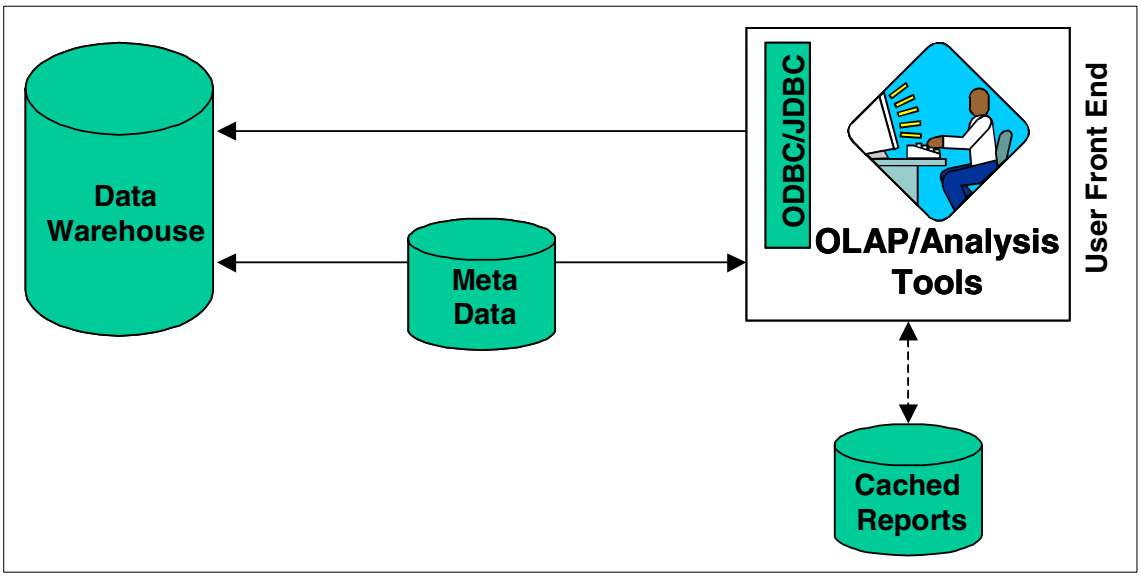


Figure 50. Two-tier architecture

A three-tier architecture, however, introduces a query server that acts as the primary application logic hub for query activity and further data manipulation.

Any data passed to the OLAP tool is done through a remote call. See Figure 51.

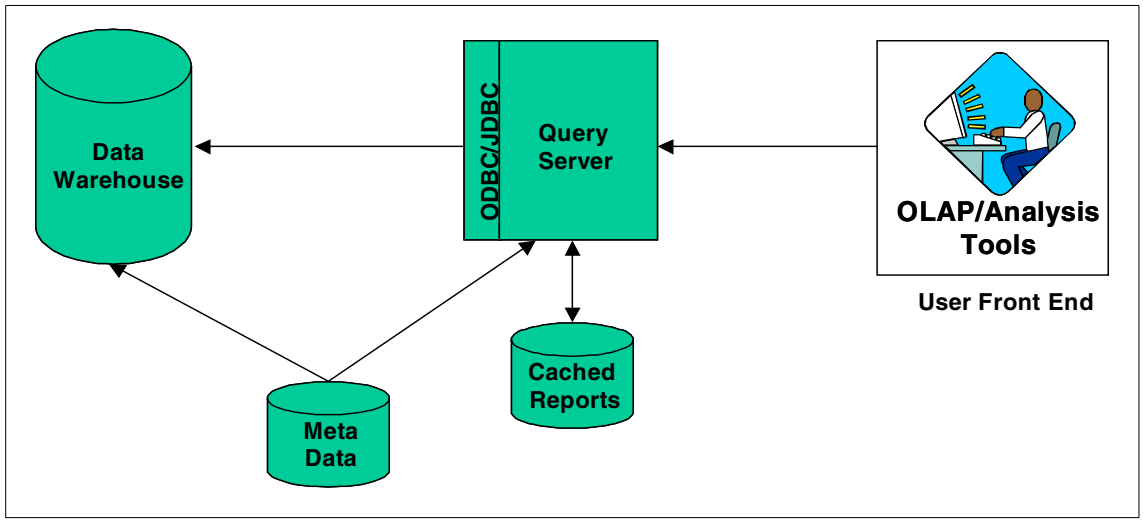


Figure 51. Three-tier architecture

In a four-tier architecture, we introduce a Web server that directly communicates with the OLAP tool. Again all application processing is done at the query server. See Figure 52.

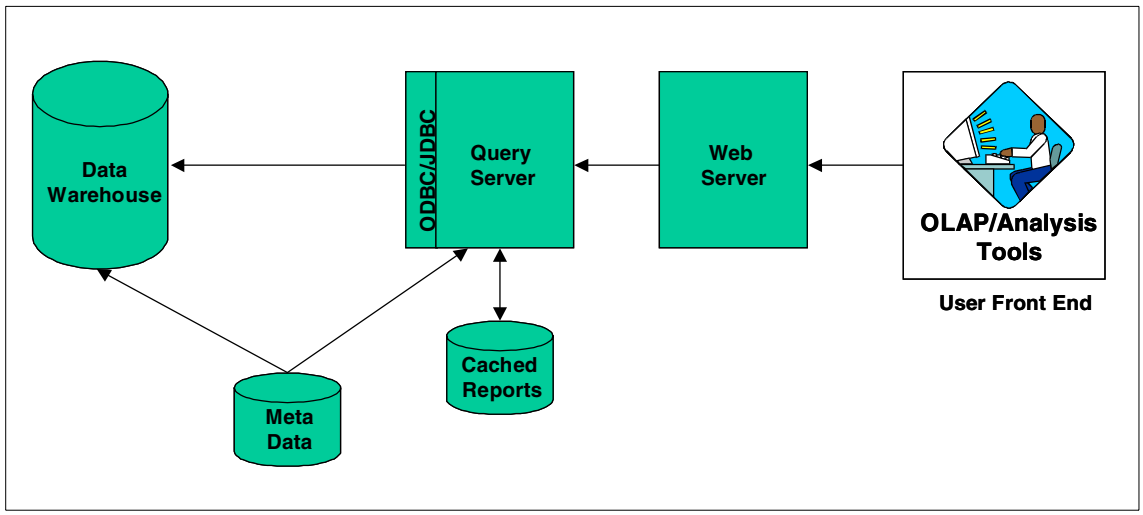


Figure 52. Four-tier architecture

6.2.3.2 Tool description

The ROLAP product suite we discuss in this section is from Microstrategy. It provides sophisticated decision support for very large relational databases. Much of its functionality is very similar to other ROLAP tools, so it acts as a good example.

Three of its components mentioned are here DSS Architect, DSS Agent, and DSS Exec (EIS). Figure 53 shows how they interact with each other and the data mart.

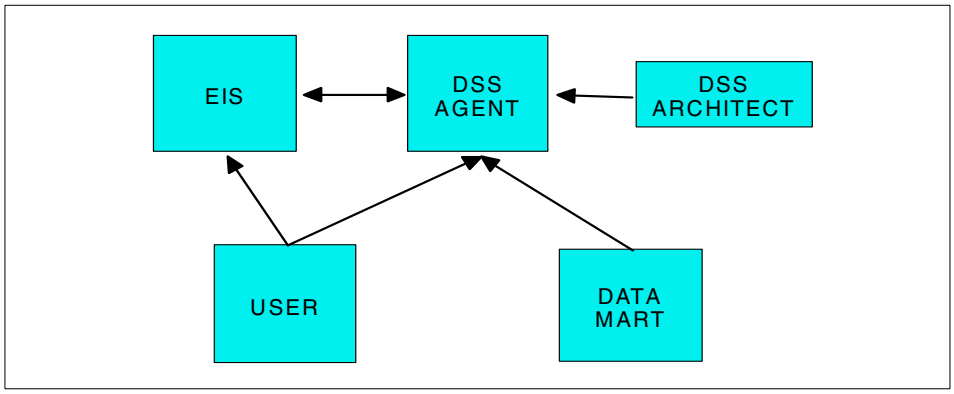


Figure 53. DSS product component association

DSS Architect is used by the administrator to define Projects (the logical setup of the data as it pertains to the physical design) as well as to build the metadata. Defining a project includes the following tasks:

- Setting up connections to the Meta Data and the data warehouse
- Selecting fact tables
- Defining dimensions and attributes and their relationships
- Specifying hierarchies within each dimension, if any

DSS Architect will then use this information to build the Meta Data. Once the Project and Meta Data have been created, someone familiar with the user requirements can customize the application further using DSS Agent. This component uses the Project information and the Meta Data and allows you to do the following:

- Create metrics using fact columns (specific column computations used in reports)
- Create templates (specific columns selected from the data mart)

- Create filters (defines search conditions)
- Create reports (combines variations of the previous three entities)

DSS Agent is also the primary user interface. It allows you to execute the reports and drill down to several levels of data, in other words, to do ad hoc queries. Some users, however, may not be interested in that level of detail. For those individuals, specially designed front-end panels can be created using DSSexec. This component gives the designer the ability to create customized panels by user type. Reports are run by simply selecting options off a panel.

6.2.4 QMF for Windows

The Query Management Facility (QMF) product, a feature component of DB2, has long been an ideal, simple DB2 for OS/390 report tool. QMF for Windows brings the same capabilities and features of the QMF product to the Windows environment. The primary difference is in the look and feel of the product; a rich GUI interface is in QMF for Windows. Some additional features make it an ideal BI tool:

- It automates DB2 queries from Windows applications.
- It provides transparent integration with many popular Windows applications.
- It includes an easy-to-use editor to change table rows from a query results table, a row at a time.
- It eliminates the use of database gateways, middleware, and ODBC drivers.
- It provides additional Web publishing capabilities with HTML.
- It works with Data Joiner to access non-relational databases.
- It is easy for both novices and expert-level users to learn.
- It tracks data accessed down to the table and column level.

The URL for the QMF home page is `http://www.software.ibm.com/data/qmf/`. To obtain a free trial sample, select **downloads**. For more information on QMF for Windows, refer to *Installing and Managing QMF for Windows,* GC26-9583.

6.3 Case study tests

The following case study tests relate to Web data access and to the Microstrategy ROLAP product suite. The Web-related tests use Net.Data,

Lotus Domino Go Webserver, Network Dispatcher, and Workload Manager to create a scalable Web server environment. The tests address query performance, query prioritization, workload balancing and server scalability. The tests implemented using the Microstrategy product suite show the usefulness of a ROLAP tool for EDW and/or data mart access to answer complex questions. We also demonstrate, at the same time, the ability of S/390 to excel at supporting various end-user access methods to warehouse data.

Additional background information about Workload Manager policies are in Chapter 7.

6.3.1 Web case study tests

We performed the following tests.

6.3.1.1 Web infrastructure

To fully test Web query workloads, a Web infrastructure is created consisting of several workstations on a token-ring LAN, along with a 2216 router as shown in Figure 54. The workstations run Netscape Navigator and Loadrunner Remote Command Launcher, with one workstation also running the Loadrunner Controller. The Loadrunner Controller initiates a “scenario” which routes scripts to each workstation’s Loadrunner Remote Command Launcher. Each script consists of HTTP requests - for static Web pages that invoke Net.Data to perform a DB2 query. A “scenario” can, therefore, run scripts concurrently on the workstations and simulate a number of users submitting DB2 queries from the Internet.

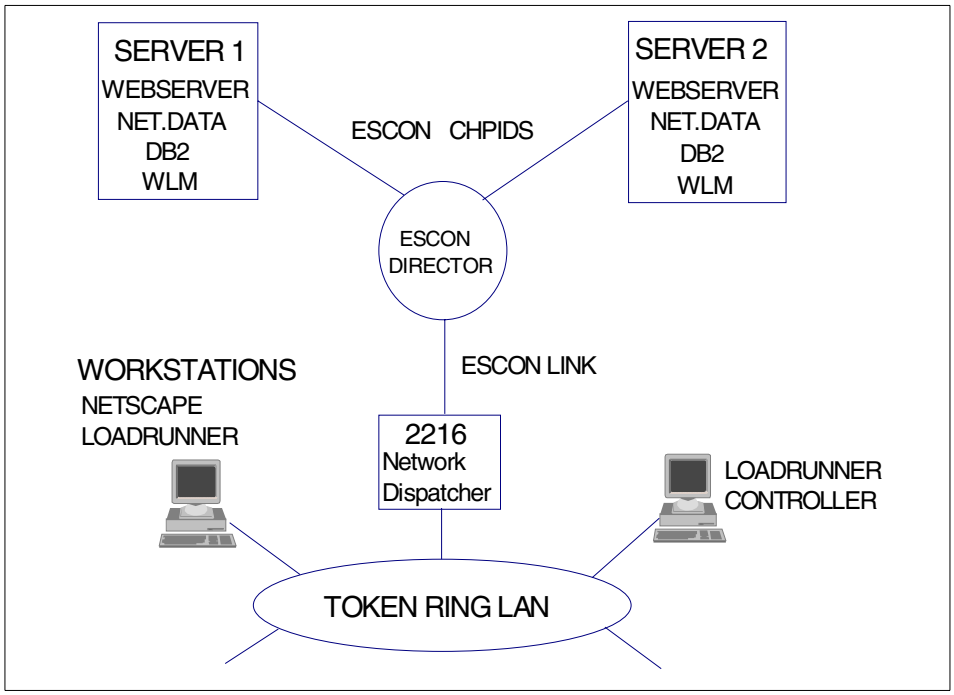


Figure 54. Web workload configuration

Each HTTP request contains an IP address that is recognized by the 2216 router as being a “cluster” of servers consisting of two S/390 servers. They share an ESCON adaptor on the 2216, via an ESCON director.

When the 2216 receives the HTTP request, the Network Dispatcher software running on the 2216 decides which of the two servers to route the request to, based on the capacity of each server to accept additional work; see Figure 55.

This HTTP request is received by the Web server Queue Manager on the S/390, a part of Domino Go Webserver V5.0. The Queue Manager selects which Application Environment will run the request as determined by the Web server configuration file. Based on the Workload Manager policy, the request is then passed to the appropriate Web server Queue Server, for execution in the Web Workload Configuration.

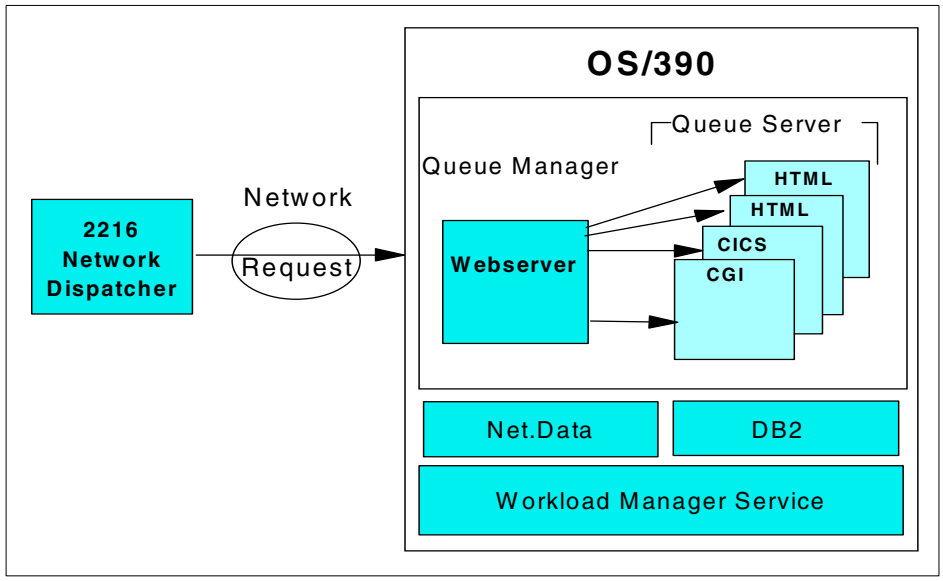


Figure 55. Single server connection to Network Dispatcher

We performed the following two tests:

6.3.1.2 Test case 1 (WEB-A-1)

This test was done for customer A, a Banking customer.

Objectives

- Show that trivial and small queries originating from the Web have little impact on the throughput of trivial and small queries being submitted by TSO users.
- Show that queries originating from the Web could be prioritized so that a *power* user could complete requests faster than a *normal* user.
- Show that the 2216 running Network Dispatcher, in combination with WLM on S/390, would balance Web requests across two fully loaded S/390 servers.

Methodology

The Network Dispatch settings were the following:

- Proportions
  - Current connections - 30%
  - New connections - 0%

    - System (capacity) - 70%
    - Input from a protocol advisor - 0%
- Smoothing Index - 2.0
- Sensitivity Threshold - 1%

A general ad hoc query workload of 50 TSO TPNS users is run for a 3-hour interval (SCA-A-1), as a base case; see 8.2.1, “User scaling (SCA-A-1)” on page 209. This workload is run again, but together with a Web query workload (WEB-A-1). During the first hour of WEB-A-1, a Loadrunner scenario of 50 Web users, each running a trivial query was executed twice. The number of TSO queries completed during the first hour (actually 54 min.) was compared to 50 user case of SCA-A-1(first 59 min.). The results are shown in Table 17.

Table 17. WEB-A-1 1-hour TSO query throughput

	SCA-A-1 (50 users)	WEB-A-1
Trivial	60	58
Small	17	15
Medium	31	24
Large	41	37
Total	149	134

The additional Web workload of 50 users had minimal impact upon the favored TSO trivial and small queries; query completions for the favored queries were almost equal even with the additional 50 Web users.

During the second and third hours, five Web query scenarios were run, each consisting of 50 to 80 users, running 25 trivial and small queries, generating a total of 12,250 queries (WEB-A-1). Again the number of TSO queries completed in hour two and three was compared to 50 user case of SCA-A-1. The results are shown in Table 18.

Table 18. WEB-A-1 3-hour TSO query throughput

	SCA-A-1 (50 users)	WEB-A-1
Trivial	192	196
Small	48	48
Medium	101	92
Large	162	128

	SCA-A-1 (50 users)	WEB-A-1
Total	503	464

Again, the favored TSO trivial and small queries were not effected by the additional Web workload.

During the second and third hours of the test case, the Web query scenarios consisted of two groups of users:

- A group of *normal* users running in a WLM service class (WEB) with velocity and importance values equal to the TSO TPNS users
- A group of *power* users running in a WLM service class (WEBHI) with a higher priority than the normal users

Two scenarios are run with 50 normal users and 30 power users. Three scenarios were run with 50 normal users and 5 power users. The WLM service class definitions used for WEB are shown in Table 19.

Table 19. Service Class WEB - Web user

#	Duration	Importance	Goal Description
1	5000	2	Execution velocity of 80
2	50000	2	Execution velocity of 60
3	1000000	3	Execution velocity of 40
4	10000000	4	Execution velocity of 30
5			Discretionary

The WLM service class definitions used for WEBHI are shown in Table 20.

Table 20. Service Class WEBHI- hi priority Web users

#	Duration	Importance	Goal Description
1		1	Execution velocity of 90

The WEBHI service class was given a higher priority than service class WEB, in which all other Web work ran. All Net.Data macros used in our tests were called webqry01.db2w through webqry25.db2w in the HFS file system on the S/390. To create queries that would run at a higher priority, these files were copied to files called hiqry01.db2w through hiqry25.db2w. So, if a URL invoked a macro starting with "hi", it would run in a higher priority service class than a macro starting with "web", even though these requests ran the same DB2 query.

The average elapsed time for normal users was then compared to the average elapsed time for power users in several Loadrunner scenarios run during the test. The overall average elapsed times are shown in Table 21.

Table 21. Overall average elapsed times

normal elapsed time	9.9 min.
power elapsed time	8.35 min.
% improvement	15%

The shorter elapsed time for the higher priority power users indicates Workload Manager allocated resources as defined in the policies properly.

To measure how evenly the Web-originated queries were balanced across the two S/390 servers RMF workload reports were obtained for the test case interval. The number of transactions ended for service classes WEB and WEBHI for each system gave the total number of Web requests (Web static page requests + normal queries + power queries) routed to that system. The number of queries (normal queries + power queries) routed to the server was determined by the RMF report by the number of transactions completed in the OMVS forked children service class, which equates to the number of Net.Data CGI programs run.

Additional test case settings are shown in Table 22.

Table 22. System setting of the test

	SCA-A-1 (50 users)	WEB-A-1
No. of servers	2	2
No. of trivial TSO users	5	5
No. of small TSO users	5	5
No. of medium TSO users	5	5
No. of large TSO users	35	35
No. of total TSO users	50	50
Measurement interval	1 Hr./3 Hr.	1 Hr./ 3 Hr.
RMF interval	15 min.	15 min.

Of the 12,336 Web-originated queries measured by RMF, 7018, or 57 percent, ran on Server #1, while 5318, or 43 percent, ran on Server #2. Of the

27,043 total Web requests (static pages + queries), 13629 ran on Server #1, while 13414 ran on Server #2, very nearly a 50-50 split.

Table 23. WEB-A-1 Web request distribution

	% Run on Server #1	%Run on Server #2
Web-originated queries	57 %	43 %
All Web Requests	50 %	50 %

Analysis and Conclusions

- The Web query work had no impact to the TSO trivial and small queries.
- The medium and large queries were impacted by Web work as was expected, since these queries age to low priority levels.
- On average, the power users ran their scripts in shorter elapsed times than the regular users.
- The total number of Web requests was nearly evenly balanced between the two S/390 servers. Neither servers were running importance 1 work, so WLM reported high capacities for both and Network Dispatcher weighted them equally.

6.3.1.3 Test case 2 (WEB-A-2)

This test was done for customer A, a banking customer.

Objective

- Demonstrate the ability of Network Dispatcher, working with WLM, to direct incoming Web requests to the server in an S/390 sysplex with the most available capacity

Methodology

A large query (see Appendix B, “Query information” on page 235) was submitted via a batch job on S/390 Server #1, to drive it to “busy”. With Server #1 busy, a Loadrunner scenario of 50 Web users running from two workstations was begun. Each Web user ran a script of 25 trivial and small queries, plus static Web page requests. The Network Dispatcher settings were the following:

- Proportions
  - Current connections - 0%
  - New connections - 0%
  - System (capacity) - 100%
  - Input from a protocol advisor - 0%

- Smoothing Index - 2.0
- Sensitivity Threshold - 1%

These Network Dispatcher settings were adjusted several times before we settled on the values shown, which produced satisfactory results for our environment. These settings would naturally have to be tuned to achieve similar results in another environment.

Web-originated queries ran in service class WEBHI, while static Web page requests ran in WEB.

Results

The elapsed time for all 50 Web users to complete their scripts was 8 minutes and 45 seconds (17:37:08 - 17:45:53). The RMF report is for the 10-minute interval 17:39:11 - 17:49:22. The CPU busy numbers over this interval are shown in Table 24.

Table 24. WEB-A-2 - Server 1 CPU busy

TIME	CPU BUSY (%)
17.39.13	100
17.40.15	100
17.41.15	100
17.42.15	100
17.43.15	100
17.44.15	100
17.45.15	100
17.46.15	100
17.47.15	100
17.48.15	100
17.49.15	100

NUMBER OF INTERVALS 11

AVERAGE BUSY 100

Table 25. WEB-A-2 - Server 2 CPU busy

TIME	CPU BUSY (%)
17.39.22	1.8
17.40.23	35.4
17.41.23	84
17.42.23	89
17.43.23	67.7
17.44.23	99.9
17.45.23	100
17.46.23	100
17.47.23	100
17.48.23	48.5

NUMBER OF INTERVALS 10

AVERAGE BUSY 72.5

Number of transactions that ended on each system for Web service classes, where WEB = static page requests and WEBHI= queries using Net.Data

Table 26. WEB-A-2 - Web transactions per server

	Server #1	Server #2
WEBB	15	1,385
WEBHH	12	1,288

Analysis and Conclusions

The RMF data shows that the overwhelming majority of Web requests were routed to S/390 Server #2, idle at the start of the run. The system soon reached 100 percent busy and remained busy for several intervals. Network Dispatcher, working with WLM, correctly routed the Web work to the server with the most available capacity.

6.3.2 OLAP case study test

This test was done for case study B, the health insurance customer, and we used claim data for the OLAP analysis.

Introduction

For our case study test, we used a ROLAP tool from Microstrategy. This tool directly interfaced with the data mart instead of the data warehouse. The data mart contained a subset of the data warehouse data, namely 1 year and 36 states. See 5.5 for more details on the data mart's design. This insurance customer only deals with claims.

Objectives

- Demonstrate to the Health Insurance customer an end-user access method that returns data faster than it does in their current production environment
- Demonstrate that query results are not obtainable before can now be obtained
- Demonstrate the advantage of having a data mart on DB2 for OS/390
- Demonstrate the ability to analyze the data from multiple perspectives by drilling up and down, under and over, and across multiple levels of hierarchies or aggregations of data

Configuration

The Microstrategy product suite runs on a Windows NT workstation. Also running on the workstation is DB2 Connect and ODBC. The ODBC driver is used to access the insurance data mart. The data mart physically resides on the S/390 in the same DB2 subsystem as the data warehouse, and in the same LPAR. See Figure 56 on page 157.

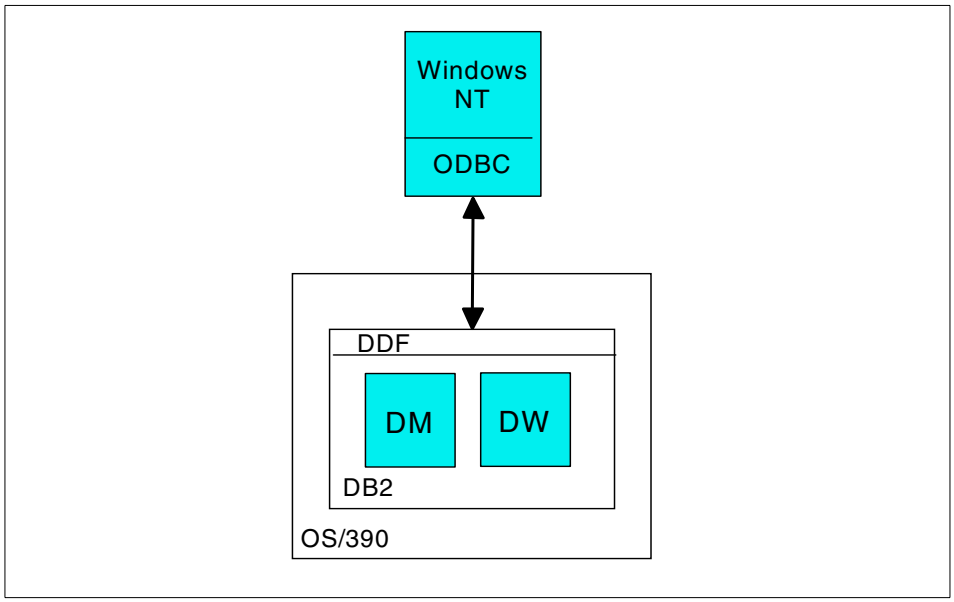


Figure 56. Configuration of test environment

Data mart Overview

The logical design of the data mart is used to set up the OLAP environment for data access and drill down processing. It is important that the design address the query requirements of the customer. In this case, the data mart contains a subset of the data warehouse data, namely 1 year and 37 states and is defined using the star schema (see Figure 40 on page 110).

The central box of the star consists of the Fact table and contains 223 million rows of “final action” claims. The fact table is made up of a multi-column partitioned key and each column of the key is a foreign key to an entire dimension. Its columns are known as *facts*. *Dimensions* are subject areas or lines of business and are an integral part of an organization. There are nine dimensions shown in the diagram which allow the user to view the data in multiple dimensions during the drill down process. Each dimension has columns associated with it that map to the Fact tables. These columns are known as *attributes*.

Methodology

Initially, we created a Project using DSS Architect. Within a project we defined the Meta Data and set up the ODBC connection to point to DB2 on S/390. We also defined the Fact tables, the facts and all nine dimensions, as well as the

attributes and any relationships that exist between them. The hierarchies within each dimension were also defined.

We started DSS Agent to create user reports. A variety of templates were created which contain the names of columns that may appear on the reports. We then created metrics, which are specific column computations that may also be used for reports, followed by filters. They supply the conditions for your queries and can be associated with the WHERE clause of an SQL statement. Lastly were the reports themselves. The reports require a template and a filter. You can then choose any number of fact columns and various metrics to generate a report. A report may be in grid, graph or map form. An example is shown in Figure 57.

---------- YEAR ----------					
---------- STATE ----------					
		FACT1	FACT2	FACT3	FACT4
CLAIM PART	CLAIM TYPE				

Figure 57. DSS report layout

The diagram demonstrates the ability of an OLAP tool. As you can see, the report layout shows a three-dimensional output, that is, all claim types per part by state by year. Once the report is generated, you can click a specific claim, and drill up, down, or across a dimension. The resulting information is now a level deeper in detail. The resulting report is shown in Figure 58.

---------- 1998 ----------					
---------- TEXAS ----------					
Part of Claim	Type of Claim	Fact1	Fact2	Fact3	Fact4
PART A	CLAIM1 CLAIM2 CLAIM3				
PART B	CLAIM1 CLAIM2 CLAIM3				

Figure 58. DSS report output

We generated over 30 reports that are frequently posed in the insurance industry. We found that the initial report execution always took the longest,

but once the report was created it could be cached, thereby reducing subsequent access to the same report. This is a common OLAP feature. Caching abilities depend on the memory capacity of your workstation.

One of the key objectives of this case study test was to demonstrate to our insurance customer the ability to generate reports in less time than was currently taking place. The customer previously generated all its reports against a tape-based file system. This resulted in a two-to-four week cycle time to obtain any reports. We addressed this problem foremost in the data warehouse design (see Chapter 2) and in the use of a data mart, both being DB2-based. The data mart contained a subset of the data warehouse data and focused on selective queries. Four summary tables were also created to improve response times. When interfaced with an OLAP tool, query performance improved three-fold. Most queries ran under two minutes. The query reports are shown in Table 27, along with their elapsed times.

Table 27. Query report response times

REPORT NAME	ET (secs)
Graph Reimbursement By Claim Type	1.00
Providers by State	2.50
Beneficiaries by State	1.30
Monthly Summary (All Claim Types)	1.60
Monthly Summary (Select Claim Types)	1.30
AVG Reimbursements by ClaimType	1.60
AVG Reimbursements by Claim Type, State	1.80
AVG Reimbursements by Claim Type, Month	4.90
BQ1(A) - Utilization by All Claims	13.70
BQ3 - MCO Disenrolled Beneficiaries	1096.40
Top 25 Diagnosis	128.20
Top 25 Diagnosis by Frequency, State	13.20
Drill to Diagnosis "DMII WO CMP NT ST UNCNTR"	3.10
Top 25 Diagnosis by Reimbursement, State	16.10
Drill to Diagnosis "CRNRY ATHRCSL NATVESSL"	4.30
Top 25 DRG	52.00
Top 25 DRG by Frequency, State	68.50
Drill to DRG "DC 19M, PSYCHOSES"	4.20
Top 25 DRG by Reimbursement, State	68.20
Drill to DRG "DC 23M, REHABILITATION"	2.70
Top 25 Procedures	75.7
Top 25 Procedures by Frequency, State	21.4
Drill to Procedure "CATARAC PHACOEMULS/ASPIR	4.9
Top 25 Procedures by Reimbursement, State	15.3
Drill to Procedure "TEMPORARY TRACHEOSTOMY"	5

Another key objective for our customer was to show that there was no limitation as to the type of question posed against the data. In the customer's

tape-based environment, files were generated that contained a subset of the data warehouse data. Users posed their queries against these files. However, any question that required data outside the contents of a file could not be posed.

Using a DB2-based data mart eliminates the need for these intermediate files and allows users to access cross dimensions, thereby eliminating the problem. This is possible because the data is accessed directly from the data mart.

For those users who do not want to access the report facility for drill-down data processing, we experimented with the tool's user interface, DSS Exec. We were able to create a menu with buttons, where each button corresponded to a report that was defined in the report facility. Selecting the button ran the report.

We found this facility very basic, but it had enough features to create a nice, simple-looking interface and it was easy to learn to use. For example, we were able to copy in graphical images to enhance the menu, create multiple menus, and add backgrounds. This interface is very practical for the user who only requires a set amount of information.

Results

- We were able to run various complex queries in a shorter amount of time than the customer could in their existing environment.
- We proved that a data mart stored on DB2 in a S/390 environment can improve query performance.
- We were able to access all levels of data detail.

6.4 Summary

Selecting the proper type of end-user access to your data warehouse is critical to your business. It is important to understand your computing environment and the objectives of your end-users early in the data warehouse design. This preparation will enable you to make the best choices in end-user solutions.

We presented two types of end-user access methodologies in this chapter, DB2 Web access and DB2 distributed access using DRDA, in terms of available software products and tools used in our case study tests. We demonstrated their use in a scalable S/390 server environment and proved

that together with DB2, they could meet the challenges and objectives of our Banking and Health Insurance customers.

The Web-related tests proved S/390's capabilities in the areas of throughput, workload balancing and query prioritization. It was shown that the additional Web workload had no impact to small and trivial queries running concurrently. Medium and large queries were impacted as expected since these queries aged to a lower priority. Because of S/390's Workload Manager feature, users could be categorized as *power* users, thereby running in a shorter amount of time than *normal* users. Our tests also showed the benefits of S/390 in terms of workload balancing. We were able to successfully balance Web requests evenly between two S/390 servers with the assistance of WLM and Network Dispatcher. Network Dispatcher correctly routed Web work to the server with the most available capacity until both were running 100 percent busy.

In our case study test using DB2 distributed data access, we showed the customer that with the proper database design and end-user tool, complex queries can run in a much shorter time than in their existing environment. This was proven by designing a DB2 data mart, which isolated specific warehouse data catered to selective queries, and structuring it such that when used with an OLAP tool, we were able to access data quickly and at various levels of detail. These levels of detail were not easily attained before, thereby providing customers with a viable solution to their business requirements.

Chapter 7. Workload management in data warehouse environment

Why is workload management important? Installations today process different types of work with different completion and resource requirements. Every installation wants to make the best use of its resources, maintain the highest possible throughput and achieve the best possible system responsiveness. On the OS/390 platform, Workload Manager (WLM) makes this possible.

With Workload Manager for OS/390, you define performance goals for each type of workload and assign a business importance to each goal. You define the goals for the work based on business priorities, and the system decides how much resource, such as CPU and storage, should be allocated to meet the goal.

This chapter covers a variety of areas related to workload management and WLM as they pertain to a data warehouse environment. The first section introduces the key workload management issues, followed by a section covering the basic concepts of Workload Manager for OS/390 for those readers relatively new to it. The next section focuses on how WLM resolves the workload management issues mentioned previously and is followed by a section that steps through an example with recommendations on WLM implementation. The last section presents a series of case study tests that were performed to demonstrate the resource management ability of Workload Manager for OS/390 in a true Enterprise Data Warehouse (EDW) environment.

7.1 Workload management issues

In an EDW environment with very large databases (VLDB) where queries run in parallel and resource consumption continuously fluctuates, managing the workload becomes extremely important. The challenges of doing this are best captured in the following set of questions that should be asked (and answered) when making decisions regarding the selection of the platform infrastructure for an enterprise data warehouse:

- Can a variety of query workloads run concurrently?
- Is it possible to dynamically re-prioritize work as business needs change? Can high priority work be isolated such that it is not impeded by a continuous flow of lower priority work that is using up large amounts of system resources?
- Is it possible to maintain query throughput even though the system has reached saturation?

- Can system resources be managed such that killer queries cannot monopolize these resources and lock out other work?
- Will the elapsed time of short queries be affected by workload? Is it necessary to pre-classify queries to ensure good performance? Will end-user response time remain relatively consistent with an increase in concurrent queries?
- Can normal maintenance occur while users are online without impacting their work?
- Can the system respond quickly to changes in business priorities? Is it possible to manage resource usage based on business needs?
- Can the full capacity of the system be utilized efficiently?

The above questions (issues) fall into the following categories:

Query concurrency

When large volumes of concurrent queries are executing, the system must be flexible enough to handle it. The amount of concurrency is not always predictable and often fluctuates based on business dynamics, so the system should be able to adjust to variations in concurrency without having to reconfigure system parameters. This situation is often overlooked, but is critical to maintaining a stable environment.

Prioritization of work

High priority work has to be able to complete in a timely fashion without totally impeding other users. In other words, the system should be able to accept new work and shift resources to accommodate workload. If not, high priority jobs may run far too long or not at all.

System saturation

System saturation can be reached quickly if a large number of users suddenly inundate system resources. The system must be able to maintain throughput with reasonable response times to maintain consistent throughput. This is especially important in an EDW environment where the number of users may vary considerably.

System resource monopolization

Queries which consume huge amounts of system resources and exclude other work from access to those resources for an extended period time are known as *killer queries*. If killer queries enter your system without proper

resource management controls in place, they can quickly consume system resources and cause drastic consequences on the performance of other work in the system. Not only will time be wasted by administrative personnel to cancel the killer query, but these queries result in excessive waste of system resources. Usually, unsophisticated governors are used to control the effects that such queries have on the system, but they do not totally solve the problem. Therefore, more sophisticated controls must be put in place to ensure good system resource utilization.

Consistency in response time

Performance is key to any EDW environment. Data throughput and response time must always remain as consistent as possible in spite of the workload. Queries that are favored because of a business need should not be impacted by an influx of concurrent users. The system should be able to balance the work load to ensure that the critical work gets done in a timely fashion.

Maintenance

What about EDW maintenance? Refreshes and/or updates must occur regularly, yet end users need the data to be available as much as possible. It is therefore important to be able to run online utilities that do not cause a negative impact. The ability to dynamically adjust priorities should be available, especially at times when a refresh may be more important than query work.

Resource utilization

It is important to utilize the full capacity of the system as efficiently as possible, reducing idle time to a minimum. The system should be able to respond to dynamic changes to address immediate business needs.

The next section introduces a way to address the previously mentioned situations: Workload Manager for OS/390 (WLM).

7.2 Workload Manager concepts

The purpose of workload management is to balance the available system resources to meet the demands of S/390 subsystems and processes, such as DB2, Web server, batch, UNIX system services, CICS and TSO, in response to incoming work requests.

OS/390 achieves the needs of workloads (response time) by appropriately distributing system resources such that all resources are fully utilized. Equally

important, OS/390 maximizes system use (throughput) to deliver maximum benefit from the installed hardware and software.

In the VLDB data warehouse environment, OS/390 Workload Manager (WLM), an integrated function of the OS/390 operating system, provides highly sophisticated industry-unique features for managing complex query workloads without the need to employ external work flow control mechanisms such as query governors.

WLM introduces a new approach to system management. Before WLM, the main focus was on resource allocation, not work management. WLM changes that perspective by introducing controls that are business goal-oriented, work related, and dynamic.

The most important step in WLM implementation planning is properly classifying your workloads, in other words, establishing an overall workload policy for your entire system called a *service definition.* The service definition contains performance goals, the business importance of the work, resource requirements and reporting information. The goals are defined in business terms and the system determines the amount of resources, CPU and storage that is required. WLM constantly monitors these rules and dynamically adjusts its processing to meet the goals.

WLM, in conjunction with DB2, provides the capability to categorize and execute work in terms of relative importance. Query prioritization can be assigned at a user level, plan level or package level to allow you, for example, to specify the priority of work at very granular levels.

Figure 59 on page 167 shows a very simplistic view of WLM's ability to classify work in a mixed workload environment.

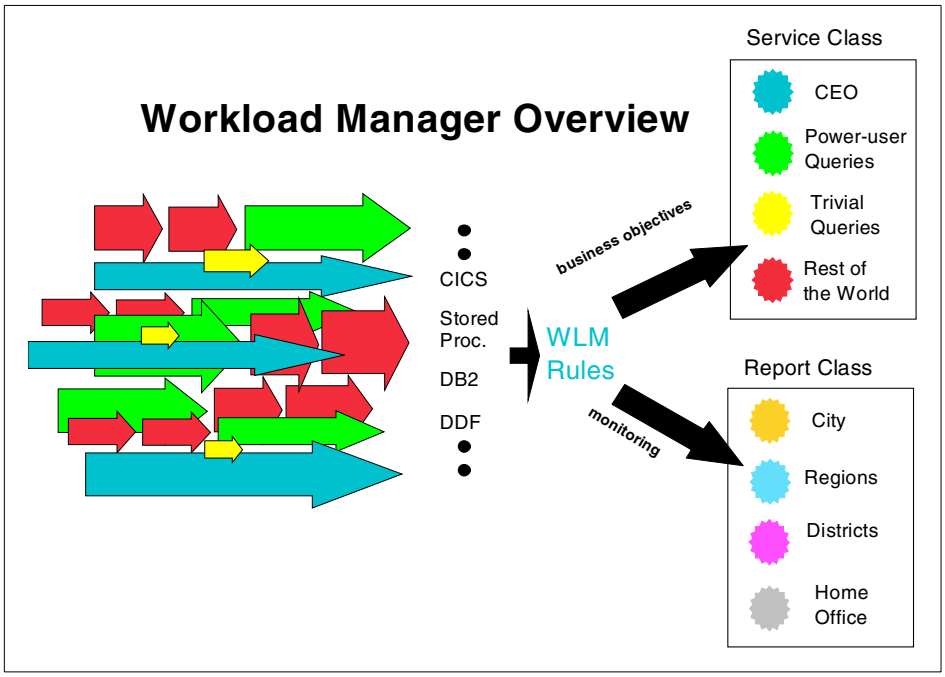


Figure 59. Workload Manager classification

Business intelligence workloads are heterogeneous in nature and decision support systems derive excellent benefits from Workload Manager. Incoming queries of different sizes and complexities arrive at the system and are assigned to a *service class*. They are categorized by a particular subsystem, depending on their point of origin, and are assigned some level of priority based on objectives established by the business community and IT. Classification rules contain the assignment criteria which is matched to the specifications of the incoming query.

Additionally, report classes may be defined in order to monitor resource consumption, which is aggregated in different ways than service classes, and to assist in capacity planning. WLM reports are used to determine how much work is run, response time/completion distributions, whether or not performance goals are met and delays which the work incurred. Resource Management Facility (RMF), the reporting tool for WLM, has been enhanced to become the reporting component of MVS workload management.

One option of WLM, *resource groups*, is important to mention here. They define processor capacity boundaries across the sysplex. Minimum and maximum CPU service units can be assigned to specific work by placing the

associated service class into a resource group. All work in that service class will be bound to those CPU limits.

For more information on Workload Manager for OS/390, refer to *OS/390 Workload Manager Implementation and Exploitation,* SG24-5326.

7.3 WLM strategy

In this section, we describe the typical WLM strategies which are required to manage the workload of each request, as well as utilize system resources effectively in VLDB environment. They are:

- Favoring short-running queries
- Favoring critical queries
- Preventing system resource monopolization
- Enabling continuous computing

7.3.1 Favoring short-running queries

Certainly, in most installations, very short-running queries will need to execute with longer-running queries.

- How do you manage response time when queries with such diverse processing characteristics compete for system resources at the same time?
- How do you prevent significant elongation of short queries for end users who are waiting at their workstations for responses, when large queries are competing for the same resources?

These are important questions. Let's explore this a bit further.

What is the impact to the user community?

As shown in Figure 60 on page 169, short-running queries can be likened to an end user who is waiting at a workstation for the response to their question, while long-running queries can be likened to an end user who will check query results at a later time (perhaps later in the day, or even the next day).

Given this typical scenario, it is apparent that the submitter of trivial and small queries is expecting to get an answer immediately, in real time. Delays in response times are going to be most noticeable to them. On the other hand, someone who submits a query that is expected to take some time is less likely to notice an increase in response time.

By favoring the short queries over the longer, we can maintain a consistent expectancy level across user groups, depending on the type of work and the volume of work competing for system resources at the same time.

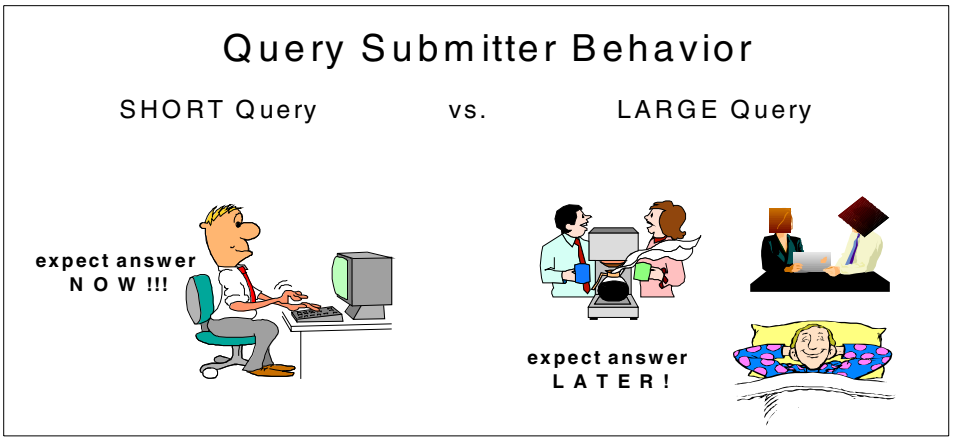


Figure 60. Query submitter behavior

The first challenge in managing this situation obviously becomes one of identification of short- and long-running work. Unsophisticated resource managers may try to deal with this identification process through the use of external resource prediction tools and governors. OS/390's Workload Manager takes a far more sophisticated and accurate approach to this kind of problem with the period aging method.

Period Aging: As queries consume additional resources, *period aging* is used to step down the priority of resource allocations. Basically, the concept is that all work entering the system begins execution in the highest priority performance period. When it exceeds the duration of the first period, its priority is stepped down. At this point the query will still receive resources, but at lower priority than when it was in the first period. The definition of a period is flexible, as the duration and number of periods established are totally at the discretion of the system administrator. Through the establishment of multiple periods, priority will naturally be established for shorter-running work without the need of pre-identifying short and long running queries. *It is important to note that implementation of period aging does not in any way prevent or cap full use of system resources when only a single query is present in the system. These controls only have effect when multiple concurrent queries are executing.*

The clear advantage that period aging provides is that work does not need to be pre-screened before entering the system. Many customers indicated that using external resource prediction tools and governors did not always properly screen short- and long-running work and thus resulted in adverse performance effects when improperly categorized queries were allowed in the system. Period aging prevents this from happening.

Preventing short query elongation is a subject of great interest to many businesses. Therefore, in our case study tests, we explore the use of WLM's period aging to show how shorter queries can be given priority over longer-running queries, thus preventing their elongation.

7.3.2 Favoring critical queries

In addition to the ability to favor short-running queries, most customers are also interested in the ability to prioritize individual pieces of work regardless of size considerations.

Every environment has critical users, queries and applications that need to run at a higher priority than other work. Critical work requirements, however, may change as workloads change, so modifications may have to be made on an ad hoc basis. OS/390's Workload Manager lets you favor critical work over other work in the system and also lets you dynamically change the priority as needed so critical work will always flow through the system seamlessly, thus maintaining a balanced workload.

The ideal Workload Manager policy for data warehousing is to provide consistent favoring of shorter-running work through WLM period aging and/or with select favoring of critical business users through WLM explicit prioritization of critical users.

Figure 61 on page 171 shows how business importance can be honored in a data warehouse environment with or without period aging by WLM in order to provide consistent throughput and response time, and to maximize the use of system resources.

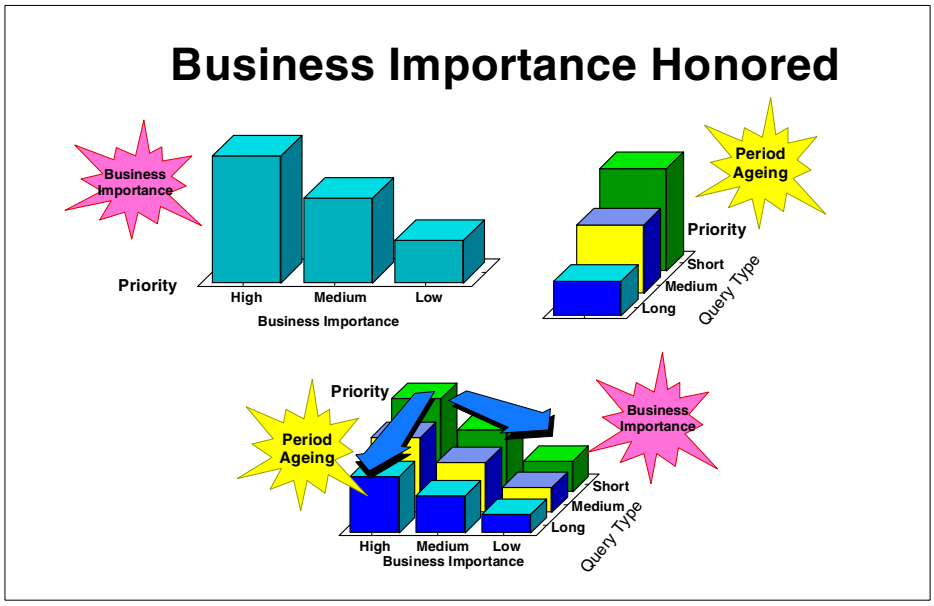


Figure 61. Business importance

Dynamic prioritization: WLM policy changes can be made dynamically through a simple “alter” command. These changes are immediate and are applicable to all new work, as well as currently executing work. There is no need to suspend, cancel or restart existing work in order to affect priority changes. While prioritization of specific work is not unique to OS/390, the ability to dynamically change the priority of work that is already in flight is an OS/390 exclusive. Business needs may dictate shifts in priority at various times of the day/week/month and as crises arise. WLM allows you to make changes “on the fly” in an unplanned situation, or automatically, based on the predictable processing needs of different times of the day.

Because of the importance of being able to favor critical workloads for maximum benefit to your system environment, we investigate this area further in our case study tests by analyzing the response times of various queries when favored and not favored.

7.3.3 Preventing system resource monopolization

Another key attribute often overlooked is how well a system is capable of dealing with large killer queries which may enter the system either intentionally or unwittingly. Such queries, without proper resource management controls, can have drastic consequences on the performance of

other work in the system and, even worse, result in excessive waste of system resources.

These queries may have such severe impact on the workload that a predictive governor would be required to identify the killer queries before they enter the system. Once they are identified, they could be placed in a batch queue for later execution when more critical work would not be affected. Such preprocessing of queries can add much overhead and, in spite of these precautions, however, such queries could still slip through the cracks and get into the system.

Work can literally come to a standstill as these queries monopolize all available system resources. Once the condition is identified, the offending query would then need to be canceled from the system before normal activity is resumed. When this situation happens, significant system resources are lost because of the cancellation of such queries. Additionally, an administrator's time is consumed identifying and removing these queries from the system.

OS/390's Workload Manager can prevent this situation from happening.

Such killer queries are not even a threat to your system when using Workload Manager, because of WLM's ability to quickly move such queries into a discretionary performance period. Aging such queries into a discretionary or low priority period allows such queries to still consume system resources, but only after or when higher priority work is not using the resources.

Table 28 on page 173 shows a WLM policy. In this case, velocity as well as period aging and importance is used. *Velocity* is a rate at which you expect work to be processed. The smaller the number, the longer it must wait for available resources. Therefore, as work ages, its velocity decreases.

The longer a query is in the system, the further it drops down in the periods. In period 5 it becomes discretionary, that is, it only receives resources when there is no other work using them. Killer queries quickly drop to this period, rendering them essentially harmless. WLM period aging and prioritization keeps resources flowing to the normal workload, and keeps the resource consumers under control. System monopolization is prevented and the killer query has essentially become neutralized.

Table 28. WLM service definition

Period	Velocity	Duration	Importance
1	80	5000	1
2	60	50000	2
3	40	1000000	3
4	30	10000000	4
5		Discretionary	5

We can summarize the significance of such a resource management scheme as follows:

- Killer queries can get into the system, but cannot impact the more critical work.
- There is no longer a need to batch queue these killer queries for later execution. In fact, allowing such queries into the system would allow them to more fully utilize the processing resources of the system as these killer queries would be able to soak up any processing resources not used by the critical query work. As a result, idle cycle time is minimized and the use of the system's capacity is maximized.
- Fewer system cycles would be lost due to the need to immediately cancel such queries. End users, instead of administrators, can now decide whether or when queries should be canceled.

We demonstrate in case study tests that OS/390 Workload Manager provides monopolization protection by adding four killer queries to a workload in progress and showing how they quickly move to the discretionary period. Virtually no effect on the base high priority workload was even noticed.

7.3.4 Enabling continuous computing

Globalization today has made the requirement for availability of decision support systems across 24 time zones 365 days per year, making continuous computing in the data warehouse environment a critical factor, and Workload Manager for OS/390 plays a key role. Workload Manager manages all aspects of data warehouse processes including batch and query workloads. Batch processes must be able to run concurrently with queries without adversely affecting the system. Such batch processes include database refreshes and other types of maintenance. Workload Manager is able to

manage these workloads so there is always a continuous flow of work executing concurrently.

Another factor that is important to continuous computing is Workload Manager's ability to favor one workload over the other. This is accomplished by dynamically switching policies. In one instance, query work may be favored over a database refresh so as to not impact online end users. On the other hand, there may be times a database refresh needs to be placed at a higher priority should end users require up-to-date data as soon as possible, or if the refresh needs to finish within a specified window. These ad hoc change requests can easily be fulfilled with Workload Manager.

We explore this area further in the case study tests documented in this redbook.

7.3.5 Data warehouse and data mart coexistence

Many customers today, struggling with the proliferation of dedicated servers for every business unit, are looking for an efficient means of consolidating their data marts and data warehouse on one system while guaranteeing capacity allocations for the individual business units (workloads). To perform this successfully, systems must be shared such that work done on the data mart does not impact work done on the data warehouse, and vice versa. This resource sharing can be accomplished using advanced resource sharing features of S/390, such as LPAR and Workload Manager resource groups.

Using these advanced resource sharing features of S/390, multiple or diverse workloads are able to coexist on the same system. These features can provide significant advantages over implementations where each workload runs on separate dedicated servers. Foremost, these features provide the same ability to dedicate capacity to each business group (workload) that dedicated servers provide without the complexity and cost associated with managing common software components across dedicated servers.

Another significant advantage comes as a result of the "idle cycle phenomenon" that can exist where dedicated servers are deployed for each workload. Inevitably there will be periods in the day where the capacity of these dedicated servers is not fully utilized, hence "idle cycles". A dedicated server for each workload strategy provides no means for other active workloads to tap or "co-share" the unused resources. These unused cycles in effect are wasted and generally go unaccounted for when factoring overall cost of ownership. Features such as LPAR and WLM resource groups can be used to virtually eliminate the "idle cycle phenomenon" by allowing the co-sharing of these unused cycles across business units (workloads) automatically, without the need to manually

reconfigure hardware or system parameters. The business value derived from minimizing these idle cycles is maximization of your dollar investment (that is, goal = zero idle cycles=maximum return on investment).

The final advantage comes from the fact that not only can capacity be guaranteed to individual business units (workloads), but there is potential through this intelligent resource sharing for business units to take advantage of additional cycles unused by other business units. The result is a guarantee to each business unit of X MIPs with a potential for X+ MIPs, bonus MIPs if you will, that come from the sophisticated resource management features available on S/390.

To demonstrate that LPAR and WLM resource groups distribute system resources effectively when a data warehouse and a data mart coexist on the same S/390 LPAR, a case study test was performed. Details of the test are outlined in Figure 62.

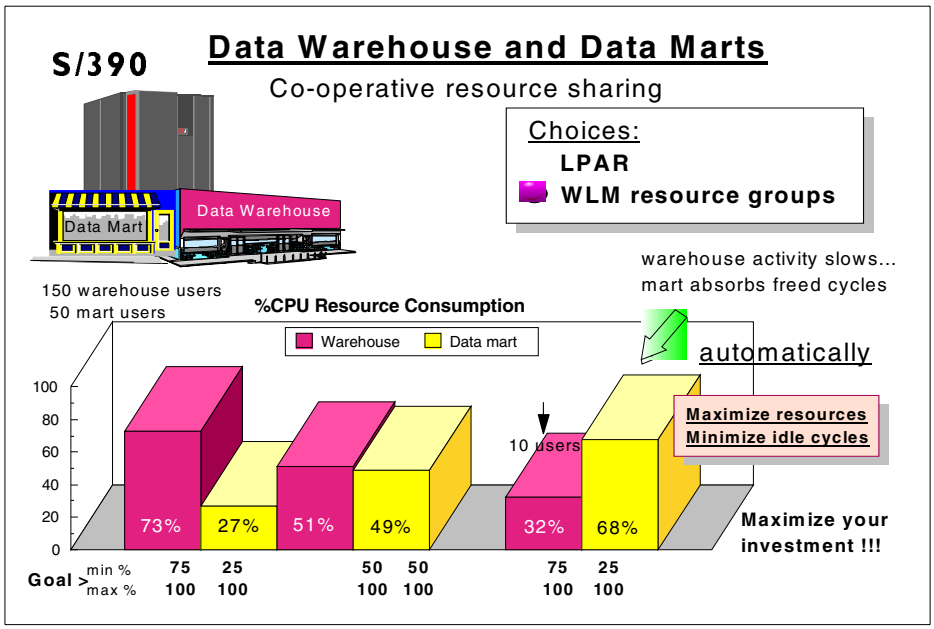


Figure 62. Coexistence of data warehouse and data mart

Two WLM resource groups were established, one for the data mart and one for the data warehouse. Each resource group was assigned specific minimum and maximum capacity allocations. 150 warehouse users were running concurrently with 50 data mart users.

In the first test, 75 percent of system capacity is available to the warehouse and 25 percent to the data mart. Results indicate a capacity distribution very close to the desired distribution. In the second test, the capacity allocations are dynamically altered a number of times, also with little effect to the existing workload. Again, the capacity stays close to the desired 50/50 distribution. The final test demonstrates the ability of one workload to absorb unused capacity of another workload in which workload activity subsides on the data warehouse. The idle cycles are absorbed by the data mart, accelerating the processing while maintaining constant use of all available resources.

7.3.6 Managing external sources of work

When businesses use a data warehouse as a source for Web or client/server access, external workload balancing must be considered.

With Workload Manager for OS/390, Web work can be managed together with other work running on the system and, if necessary, given priority over other workloads. WLM is able to balance the system load no matter the type of work.

Once Web access is made available, a new type of user is introduced into the system. The resource requirements for work associated with these users need to be isolated in order to manage them properly. With WLM, this can easily be accomplished by creating a service policy specifically for Web users.

Another important point to make is that WLM not only balances loads within a single server, but can also balance Web requests and other external requests across multiple servers in a sysplex. This is a very powerful feature of WLM.

External workload balancing using WLM is discussed further in Chapter 6. In that chapter Lotus Domino Go Web Server, Net.Data, and Network Dispatcher are used to demonstrate that together with Workload Manager, incoming requests can be redirected to the proper server for processing depending on existing server workloads. A detailed description of the test is outlined in 6.3.1, “Web case study tests” on page 147.

Figure 63 on page 177 shows the possible access methods to a data warehouse through Web and client/server, and the throughput of concurrent execution of traditional queries (small, medium, large, and trivial queries) and a new type of query from the Web.

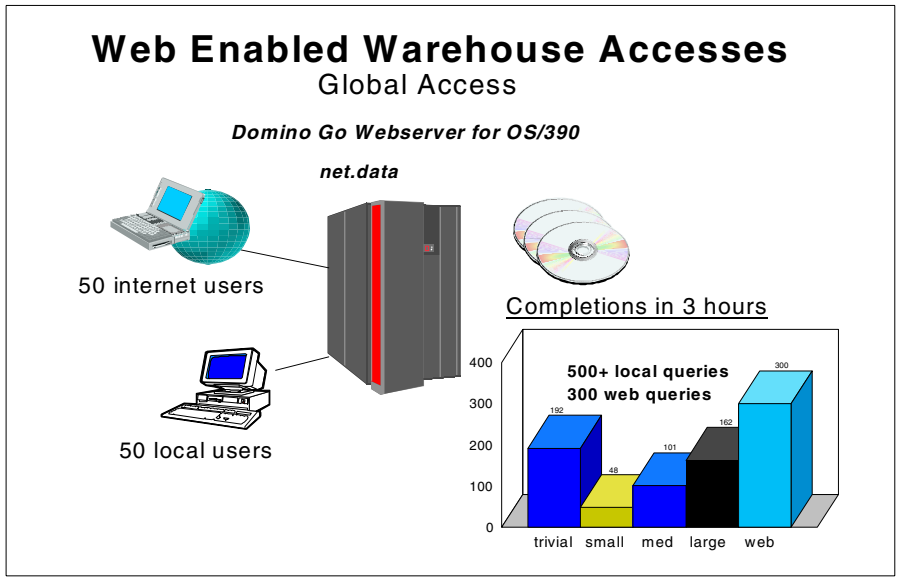


Figure 63. External workload balancing

7.4 Implementing WLM for business intelligence

Once the general functionality of Workload Manager for OS/390 is understood, you can easily set up your environment to take advantage of WLM's powerful capabilities

This section provides general guidance on the initial implementation of WLM for BI systems. It gives you a starting point by stepping you through the workload analysis process which includes requirements gathering and the business importance selection process. An example is then used to demonstrate how to create a service definition from the requirements. The focus is on WLM policies, service classes, and classification rules.

You can define all WLM policies using ISPF WLM administrative dialog in interactive mode.

For more detailed information on WLM along with general guidance for a full WLM implementation, refer to *OS/390 Workload Manager Implementation and Exploitation,* SG24-5326, and *OS/390 V2R7 MVS Planning: Workload Management*, GC28-1761.

7.4.1 Considerations when implementing WLM

The basic premise of implementing WLM for any system is to first understand the individual application requirements and their relative importance to the business. The WLM service definition needs to be able to meet those requirements, be flexible enough to handle new applications, and be smart enough to capture information to understand how the application and its requirements are changing and growing.

As an administrator of WLM, you must get answers to the requirements and business importance questions. Individuals who can assist you are database administrators, systems programmers, end-user managers, and capacity planners. Minimally, a list of all application and application requirements should be created, maintained, and be approved by all, including management. When a new project comes in, it needs to be properly placed within these defined requirements.

Keep in mind that periodically you should review your requirements to determine if they still meet your company's objectives. The first implementation does not have to be perfect. Put a definition in place and then monitor the RMF Monitor I Workload Activity reports and tune accordingly.

7.4.2 Business intelligence workload characteristics

Unlike an OLTP environment, requirements of a BI application can be very diverse. Some applications might be very similar in nature to OLTP, with each query (or transaction) requiring approximately the same small amount of resource. Others will vary dramatically from one query requiring only 500 service units of work to another query requiring 1,000,000 service units. This is why it is so crucial to understand the workload requirements and establish rules accordingly.

Figure 64 on page 179 contrasts resource utilization curves found in OLTP and BI environments. OLTP workloads typically consist of a high concurrent rate of short, well-understood transactions lending themselves to response time goals, thus allowing WLM to manage them in the most effective way. Query workloads for BI systems, however, consist of both predefined queries with well-understood resource requirements, and ad hoc queries, that range from trivial resource consumption to very large resource consumption. For the ad hoc workloads, we recommend WLM's *period aging* with velocity goals. It allows the longer running queries to age as run-time increases, giving them less priority than other work, but still allocating CPU time. This is why *period aging* is a natural fit for many data warehouse workloads.

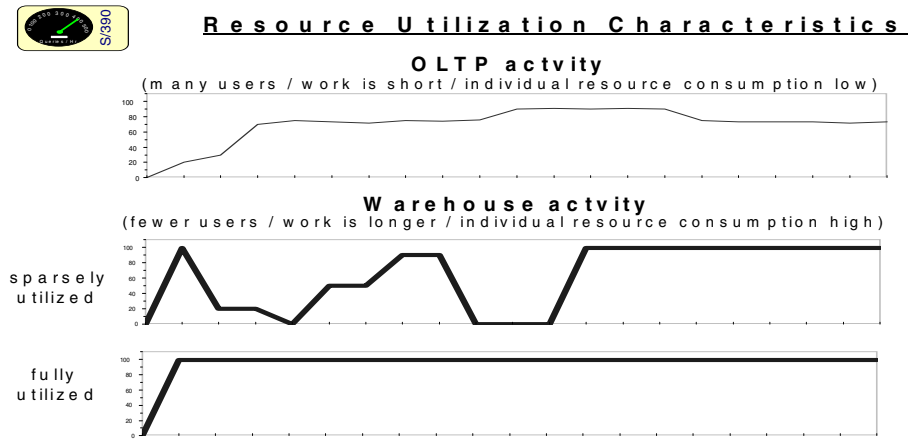


Figure 64. Resource utilization: OLTP versus BI

Additionally, within many corporations, there are likely to be different classes of users executing ad hoc queries, from those posing sophisticated queries to novices who may pose less sophisticated, untuned queries. It is, therefore, suggested that these groups of users be classified into their own service classes, providing the ability to manage one group differently from the other.

7.4.3 Thoughts on classification

Planning is the key to a successful WLM implementation and adequate time should be allocated to this activity. Classification is important foremost in order to get the incoming work into the right service class, so it can be managed properly (that is, in line with the business demands). In addition to managing the work, it is typically very important to understand how much resource the work is consuming. Consider the different ways a business could track work in the system: by department, by application, by user, or by geography. Probably some combination of these is the solution. These questions should be asked early on so classification rules can be put in place to accommodate not only management, but tracking. Classification goes hand-in-hand with determining naming conventions for user IDs, DB2 plan names, DB2 package names, batch job names, and so on. So start to think about classification at the same time naming conventions are being discussed.

When work arrives into the system through several OS/390 subsystems, each subsystem has a set of qualifiers associated with it to classify the incoming work to WLM. It is therefore important to decide what qualifiers should be

used to classify the workload. Use *OS/390 MVS Planning: Workload Management*, GC28-176, to understand all the options available to you as early as possible in the process.

Capacity planning for a data warehouse is one of the challenges facing capacity planners today. The ability to understand usage by departments, applications, and projects becomes much easier with careful planning of classification rules.

7.4.4 Sample data warehouse analysis

We put together a table with sample data for a data warehouse scenario that represents a simple mix of applications and projects representative of an Insurance data warehouse, as shown in Table 29 on page 181. The table contains various applications with descriptions of their workloads, general requirements and business importance. Its goal is to help you recognize the type of information that is important to collect up front as part of WLM preparation. This sample table will be referenced by other examples throughout the rest of this chapter.

The scenario assumes that no Service Level Agreement (SLA) was in place, therefore, meetings were held with management and the business users to gather the business importance and goal requirements for each application/project.

Additional information on the sample BI system follows:

- The system contains one data warehouse and three data marts (one for marketing, one for sales, and one for the fraud and abuse group).
- The marketing and sales marts are fed from the data warehouse on a monthly basis. The *fraud and abuse* mart also comes directly from the warehouse, but it has undergone transformations to prepare it for data mining. Additionally, there is one data mart on an alternate platform which is fed by the data warehouse, monthly as well.
- There are three DB2 subsystems, one for data warehouse, one for test DB2, and one is shared by the two data marts (the data marts are both star schemas and share some common dimension tables, and hence are good candidates for sharing DB2 subsystems).

Table 29. Requirements/business importance information

Application / Project	End-users	Inter-face	Sub-sys	Importance to Business	Goal	Target Data
Marketing Group (General)	30 Regional reps. 5 Headquarter reps., of which 2 (Frank, Kathy) are considered key cotributors to bottom line. 2 novice users	QMF for Windows	DDF	Med-High, High for queries originating w/ Frank or Kathy	Trivial queries should have 3 secs. or less response time. Generally favor the Short consumption queries over the Medium and Medium over the Large and so on. Novice users treat with care.	Mrktng. DM
Marketing Group Daily Reports	All Marketing.	Batch	JES	High	All reports complete by each Mon. AM	Mrktng. DM
Actuarial Analysis	4 Actuarials at Hdqrs.	PC /SAS	TSO	Med-High		EDW
Data Mining Fraud and Abuse	3 Statistical Analysts from Fraud and Abuse	FAMS / IM	OMVS	Med	When the mine is ready to be mined, complete the mining within 48 hours.	Data-mining mart
DW/DM Daily Refresh	N/A	Native Batch	JES2	High	Refresh to be done by 7AM every day.	All
DW Maintenance	N/A	Native Batch	JES2	Med	Maint. to applied 48 has priority Tues and Wed. PM	All
DW Performance/ System and Database Management	System Performance, Analysts, Programmers, DBAs, Operations	Native TSO	TSO	High	Trivial TSO transactions require 1 second response times all the time. As the work gets larger, favor small over medium and medium over large.	All

7.4.5 Creating a service definition

Given the information in Table 29 we are now able to put together a WLM service definition. In this example we only focus on the WLM setup associated with the BI applications themselves.

A good reference for creating a WLM service definition from scratch is the sample QUICKDEF service definition supplied in *OS/390 Workload Manager Implementation and Exploitation,* SG24-5326. It covers all the basic systems and subsystem address spaces and recommended service class and report class definitions for them.

7.4.6 WLM policies

We suggest our service definition contain three policies:

- Prime shift: This policy would be in effect 7:00 A.M. - 6:00 P.M. every day of the week. It would favor interactive work over long-running, large

consumption, and batch type work. Additionally, it would give higher priority to the critical business users and applications

- Second shift: This policy would be in effect between 6:00 P.M. and 11:00 P.M. It would favor some of the longer, large consumption work, such as data mining and warehouse maintenance.
- Third shift: This policy could be titled the REFRESH policy. During third shift the focus shifts to completing the warehouse and data mart refreshes along with any canned daily reporting, prior to 7:00 A.M.

These policies are discussed further in the rationale sections.

7.4.7 Workloads and service classes

We now create appropriate workloads and service classes for each of the application and projects in our sample BI system. We recommend four WLM workloads to represent the application/project requirements provided in Table 29. To recap, the concept of a WLM workload is simply to logically group work you would like to track as an aggregate. For each workload we will suggest service class definitions and the rationale behind them. Additionally, some service classes will take on different goals/importance depending on what policy is in place at the time. We cover those policy changes when applicable.

7.4.7.1 Marketing workload

First shift policy

We created four service classes for the marketing workload (see Table 30), three for the marketing group analysis team and one for the marketing group daily reports.

Table 30. Marketing workload service classes - first shift

Workload	Application / Project (Interface)	Sub-sys	Service class	Period	Duration	Impt	Goal Description
Marketing	Mrktng Group Analysis (QMF Windows)	DDF	QUERY	1	5,000	2	90 percent complete within 3 secs.
				2	20,000	3	Velocity = 70
				3	1,000,000	4	Velocity = 50
				4	15,000,000	5	Velocity = 30
				5			Discretionary
		DDF	QUERYHI	1	5,000	1	90 percent complete within 3 secs.
				2	20,000	2	Velocity = 80
				3	1,000,000	3	Velocity = 60
				4	15,000,000	4	Velocity = 40
				5		5	Velocity = 20
		DDF	QUERYLO	1	5,000	3	90 percent complete within 3 secs.
				2	15,000,000	4	Velocity = 20
				3			Discretionary
	Mrktng. Group Daily Reports (Batch)	JES	DAILYBAT	1	100,000	4	Velocity = 30
						5	Velocity = 10

For the analysis team, we created QUERYHI for critical end-users, Frank and Kathy, QUERYLO for novice users, and QUERY for the rest of the marketing team. Per the requirements, the trivial queries (which we defined as requiring 5,000 service units or less) for all three service classes have a *goal* of three seconds. We utilized period aging to favor the shorter consumption work over medium, and the medium over large. We did this by decreasing the velocity and importance as work moves into the lower periods. The period durations were derived from performance data that was available on prototype queries. If no such data is available, we suggest starting with something similar to what is being used in this example. Once the policy is utilized in production, RMF Workload Activity reports can then be referenced to highlight potential areas for improvement. Optimally, you are looking to get a nice spread of query completions in all periods. See “Setting Durations” for more information.

Also, notice how we overlapped the period definitions between the QUERY and QUERYHI service classes. This ensured, for similar type work (similar resource consumption) that we always favored the critical user over the normal user. Additionally, notice the QUERYHI work never moves to a discretionary goal, yet the QUERY work will. The QUERYLO work starts at an Importance of “3” and quickly moves to a Discretionary goal. We want to ensure everyone else is happy before expending resource on the less critical work.

For the daily reports team, we created DAILYBAT. The goal for this work is a completion time of 7:00 A.M. every morning. But, since this work is dependent upon the daily refresh being complete, it only requires a high priority during third shift, past the daily refresh. Hence, during primary shift this work is assigned a low Importance and a relatively low velocity. This allows the marketing team to run new batch reports during prime shift at a low priority without having to wait for the outputs until the following morning.

Response time goals versus velocity goals

WLM recommends that whenever there is an opportunity, response time goals should be utilized over velocity goals. For response time goals to be effective, the work being assigned a response time must have sufficient completions. As a rule of thumb, WLM assumes 10 completions within a 20-minute period should be sufficient.

For our sample, we expect a high rate of completions in period 1, are but not quite sure what it will be as we move to the lower periods, therefore we used velocity goals for the rest.

Second shift policy
It is same as the first shift policy with the identical rationale.

Third shift policy
For the third shift policy, we have increased the priority of the daily reports, which must be completed by 7:00 A.M. every day. The remaining marketing service classes were been left the same, still providing the very short trivial work enough priority to get in and out of the system while the daily reports are running; see Table 31.

Table 31. WLM policy for marketing workload - third shift

Workload	Application / Project (Interface)	Sub-sys	Serv class	Period	Duration	Impt.	Goal Description
Marketing	Mrktng. Group Daily Reports (Batch)	JES	DAILYBAT	1		1	Velocity = 80

7.4.7.2 Actuarial workload

First shift policy

Table 32 shows a service class, ACTSAS, which has the exact characteristics of the QUERY service class. We could have re-used the QUERY service class, but in order to better understand the actuarial workload at a period level we decided to use a unique service class for this work.

Table 32. Actuarial workload service class - first shift

Workload	Application / Project (Interface)	Sub-sys	Serv class	Period	Duration	Impt	Goal Description
Actuarial	Actuarial Analysis (PC SAS)	TSO	ACTSAS	1	5,000	2	90% complete within 3 secs.
				2	20,000	3	Velocity = 70
				3	1,000,000	4	Velocity = 50
				4	15,000,000	5	Velocity = 30
				5			Discretionary

The rationale for the period characteristics is the same as it was for QUERY. If at some point management determines that this work was less or more important than the marketing QUERY work, then adjustments can easily be made.

Second and third Shift

The actuarial workload for second and third shift is same as that of first shift.

7.4.7.3 Fraud and Abuse

First Shift policy

We created a service class, DATAMINE for the FAMS intelligent miner application (see Table 33).

Table 33. Fraud and Abuse workload service class - first shift

Workload	Application / Project (Interface)	Sub-system	Serv class	Period	Duration	Impt	Goal Description
Fraud and Abuse	Fraud and Abuse Data Mining (FAMS IM)	OMVS	DATAMINE	1	100,000	4	Velocity = 30
				2			Discretionary

For the first shift policy, we treat this work similar to the way we treated the daily marketing reports work. The required turnaround time is 48 hours, therefore during prime shift, when most of the interactive work is running, we will let work come through, but at a lower priority.

Second shift policy

For second shift we provide a greater level of service to the data mining applications; see Table 34.

Table 34. Fraud and Abuse workload service class - second shift

Workload	Application / Project (Interface)	Subsys	Serv class	Period	Duration	Impt	Goal Description
Fraud and Abuse	Fraud and Abuse Data Mining (FAMS IM)	OMVS	DATAMINE	1		2	Velocity = 80

Resource requirements for a data mining application are very difficult to predict, therefore if providing the level of service this policy provides is insufficient to meet the 48 hours requirement, this definition will have to be reconsidered.

Third shift policy

For third shift we continue to provide the data mining application higher priority than what we provided it on first shift, but not quite as much as we provided it on second shift; see Table 35.

Table 35. Fraud and Abuse workload service class - third shift

Workload	Application / Project (Interface)	Sub-system	Serv class	Period	Duration	Impt	Goal Description
Fraud and Abuse	Fraud and Abuse Data Mining (FAMS IM)	OMVS	DATAMINE	1		3	Velocity = 60

This allows some of the larger consumption interactive work to run above the data mining application.

7.4.7.4 DW/DM management workload

First shift policy

We created three service classes for the DM/DW management workload; see Table 36.

Table 36. DW/DM management workload service classes - first shift

Workload	Application / Project (Interface)	Sub-sys	Serv class	Period	Duration	Impt	Goal Description
DW/DM Mangmnt.	DW/DM Daily Refresh	JES	REFRESH	1	50,000	4	Velocity = 10
				2			Discretionary
	DW/DM Maintenance	JES	BATMAINT	1	50,000	4	Velocity = 10
							Discretionary
	DW/DM Performance System Management Maintenance	TSO	TSOMAINT	1	1,000	1	90 percent complete within 1 sec.
				2	10,000	2	90 percent complete within 10 sec.
				3	1,000,000	3	Velocity = 50
				4	15,000,000	4	Velocity = 30
				5		5	Velocity = 10

For first shift, we treated the daily refresh and the batch maintenance as low priority work. For TSOMAINT, the early period is set to a one-second response time as requested. Since we expect a lot of transactional type requests in this service class, a response time goal was also set for the second period. We then chose velocity goals for the last three periods.

Second shift policy

During second shift, a higher level of service is desired for the maintenance of the data warehouse and data mart; see Table 37.

Table 37. DW/DM management workload - second shift

Workload	Application / Project (Interface)	Sub-sys	Serv class	Period	Duration	Impt	Goal Description
DW/DM Mangmnt.	DW/DM Daily Refresh	JES	REFRESH	1	50,000	4	Velocity = 10
				2			Discretionary
	DW/DM Maintenance	JES	BATMAINT	1		2	Velocity = 80
	DW/DM Performance System Management Maintenance	TSO	TSOMAINT	1	1,000	1	90 percent complete within 1 sec.
				2	10,000	2	90 percent complete within 10 sec.
				3	1,000,000	3	Velocity = 50
				4	15,000,000	4	Velocity = 30
				5		5	Velocity = 10

We have increased the importance and velocity to equal that of the data mining work. All other work in the workload was left the same.

Third shift policy

During third shift, the Data Warehouse Refresh becomes the most important work, hence we change its velocity to 90; see Table 38.

Table 38. DW/DM management workload - third shift

Workload	Application / Project (Interface)	Sub-sys	Serv class	Period	Duration	Impt	Goal Description
DW/DM Mangmnt .	DW/DM Daily Refresh	JES	REFRESH	1	50,000	2	Velocity = 90
				2			Discretionary
	DW/DM Maintenance	JES	BATMAINT	1		3	Velocity = 50
	DW/DM Performance System Management Maintenance	TSO	TSOMAINT	1	1,000	1	90 percent complete within 1 sec.
				2	10,000	2	90 percent complete within 10 sec.
				3	1,000,000	3	Velocity = 50
				4	15,000,000	4	Velocity = 30
				5		5	Velocity = 10

We continue to provide a high level of service to the maintenance work, if there are spare cycles beyond the Refresh.

7.4.8 Setting durations

WLM recommends the following settings for the service coefficients:

- IOC = 0.1
- MSO = 0.001
- CPU = 1.0
- SRB = 1.0

IOC is for I/O subsystem, MSO for storage, and CPU and SRB for processor. You can check these values in *WLM administrative dialog* from ISPF panel. These recommendations essentially base total service unit consumption on the actual CP consumption.

Notes

Service units are a measure of resources productively consumed by an address space or enclave. They are calculated based on the task time (CPU), SRB time (SRB), I/O, and storage (MSO) service consumed. Installations can assign weight to one kind of service over another. This weight is called a service definition coefficient (SDC).

We followed these recommendations for our sample warehouse. We derived our period durations from converting the CP seconds, for our prototype queries, into service units. On a G5 10-way server, one CP second is approximately 5070 service units. This information is available in WLM planning documentation as well as the SU/SEC filed in an RMF workload activity report.

7.4.9 Classification rules

Table 39 on page 190 depicts how we classifed the incoming work into the appropriate defined service classes. We utilized userid (UI) for most of the workloads due to the ability to qualify work at the most granular level (a given user). We used a simple naming convention to differentiate the critical business user from the novice user within the Marketing Analysis group. This same technique could have been utilzed for the other interactive workloads, including the performance/system maintenance group. For the batch workloads we utilized jobname (TN), also referred to as transaction name, which is a very common approach. For instance any job starting with REF* will get classified into the REFRESH service Class. As mentioned earlier, consider classification early on when first building a data warehouse.

Table 39. Sample classification rule

Application	Interface	Sub System	Work Qualifer	Name	Service Class
Marketing Analysis	QMF / WIN	DDF	UI	DW*	QUERY
				DWHI*	QUERYHI
				DWNEW*	QUERYLO
Marketing Daily Reports	Batch	JES	TN	DREP*	DAILYBAT
Actuarial Analysis	PC / SAS	TSO	UI	ACT*	ACTSAS
Fraud and Abuse Data Mining	FAMS / IM	OMVS	UI	DM*	DATAMINE
DW/DM Refresh	Batch	JES	TN	REF*	REFRESH
DW/DM Maintenance	Batch	JES	TN	MNT*	BATMAINT
Performance/ System Maintenance	Batch	JES	UI	USR*	TSOMAINT

7.5 Summary of Workload Manager

Key attributes of OS/390's Workload Manager are the abilities to do the following:

- Manage resource consumption based on established business objectives
- Run diverse mixed query workloads concurrently
- Prevent monopolization of system resources by killer applications or queries
- Dynamically alter existing work priorities as business needs demand
- Fully utilize all system capacity minimizing idle cycles
- Respond instantaneously to shifts in business priorities
- Prevent elongation of short queries
- Avoid the need to pre-identify short- and long-running queries
- Provide more consistent response times for end users
- Favor high priority work

- Provide effective sharing of system resources across data marts and data warehouse
- Extend prioritized functionality to Web-based accesses as well
- Oversee all work components of a data warehouse from maintenance to query workloads

7.6 Case study tests for WLM

Three tests address Workload Manager capabilities. They include the following:

- Protect smaller queries from large (killer) queries
- Explicitly prioritize critical queries
- Manage data warehouse refresh concurrently with query activity

7.6.1 Protect smaller queries from larger killer queries (WLM-A-1)

Customer A wants to ensure WLM handles the killer ad hoc query workload appropriately.

Objectives

Ensure that the throughput of the general ad hoc query workload is not impacted when large killer queries enter the system

Methodology

Two tests were conducted:

- The base test, WLM-A-11, consisted of 50 users running the general ad hoc query workload for a measurement interval of three hours. This is the same run as the 50-user case of SCA-A-1, see 8.2.1, “User scaling (SCA-A-1)” on page 209.
- The delta test, WLM-A-12, consisted of the same 50-user workload in WLM-A-11 plus four additional killer queries. The killer queries were submitted as batch jobs (2 per server) 15 minutes into the three-hour measurement interval.

Each of the four killer queries had the capability of consuming the complete processing capacity of the sysplex. The SQL for the killer queries can be found in Appendix B. "Query information" on page 235.

Both tests used the TSO WLM policy, see A.4 "WLM settings" on page 230.

Table 40. WLM-A-1 test configurations

	WLM-A-11	WLM-A-12
No. of servers	2	2
No. of trivial users	5	5
No. of small users	5	5
No. of medium users	5	5
No. of large users	35	35
No. of total users	50	50
No. killer queries	0	4
WLM policy	TSO	TSO
Measurement interval	3 hours	3 hours

Results

Table 41 compares the number of queries completed for the two test cases.

Table 41. WLM-A-1 query results

Test case	Total queries completed	Trivial queries completed	Small queries completed	Medium queries completed	Large queries completed
WLM-A-11	503	192	48	101	162
WLM-A-12	491	195	46	95	155

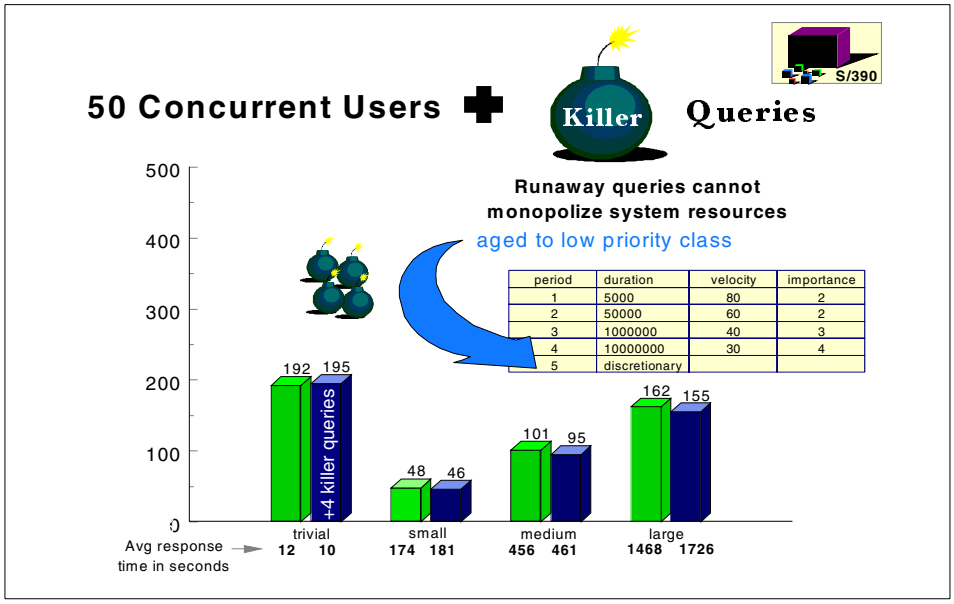


Figure 65. Average response time of queries without/with killer queries

Figure 65 not only indicates the number of queries completed for WLM-A-11 and WLM-A-12, but also the average response times.

Analysis and conclusions

With WLM, large killer queries can be prevented from monopolizing system resources. The killer queries in WLM-A-12 quickly aged to the lowest priority class (discretionary) and therefore did not impact the more important ad hoc query workload. Practically the same number of queries completed in each class of work, both with and without the killer queries present. The throughput for WLM-A-12 was 98 percent of that of WLM-A-11, thus surpassing the success criteria of 85 percent. It should also be noted that the response time remains consistent in spite of the killer queries. Very little deviation in response time is noted.

7.6.2 Prioritize critical query (WLM-A-2)

This test case demonstrates Workload Manager's ability to prioritize critical users/queries above the rest of the workload.

Objective

Prove that the critical users are provided with as much resource as they demand while running concurrently with the general ad hoc query workload.

Methodology

Two tests were conducted:

The base test, WLM-A-21, consisted of four single query executions (one trivial, one small, one medium, and one large) executed on one of the servers, with no other work in the system.

The delta test, WLM-A-22, consisted of the 100-user general ad hoc query workload plus the same four queries executed in WLM-A-21. The four queries were submitted as batch jobs, one hour into the three-hour measurement interval. The four queries were executed under a user ID that was considered a critical user, and therefore had a higher priority than any of the general ad hoc query work. This is actually the same run as the 100-user test case of SCA-A-1, so refer to 8.2.1, “User scaling (SCA-A-1)” on page 209 for more details.

Both tests used the TSO WLM policy, see A.4 "WLM settings" on page 230.

Table 42. WLM-A-2 test configurations

	WLM-A-21	WLM-A-22
No. of servers for ad hoc query workload	N/A	2
No. of trivial users	N/A	10
No. of small users	N/A	10
No. of medium users	N/A	10
No. of large users	N/A	70
No. of total users	N/A	100
No. of critical queries (single threaded)	4	4
No. of servers for critical queries	1	1
WLM policy	TSO	TSO
Measurement interval	3 hours	3 hours

Results

In Figure 66, the top bar of each graph shows the response time when the query is run alone. The middle bars represent the response time when the query is given favored status in a WLM policy and is run with 100 other queries. The bottom bars show the response time of the query when run in not-favored status with 100 other queries.

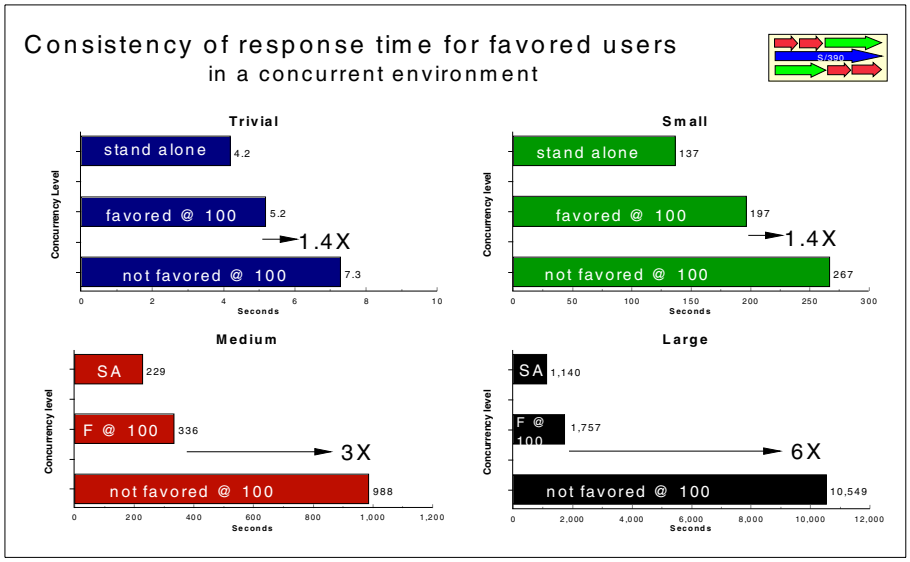


Figure 66. Consistency of response time for favored users

Analysis and conclusions

When running favored in a contention environment, the response time (RT) for the trivial query is increased, but not significantly enough to be perceived negatively by the end user. When run as not favored, the response time is increased 1.4 times. A similar comparison can be made with the other query types. The large query is the most noticeably affected by being not favored. Its response time is six times longer than when favored. But, you will notice that when favored, the RT increase is almost imperceptible. In other words, elongation of the large query was minimized by favoring it using WLM.

7.6.3 Manage DW refresh concurrent with query activities (WLM-A-3)

In a data warehouse environment, continuous computing is a critical factor. In this case study we look at concurrent utility execution and resource management.

Objectives

- Manage data warehouse refresh concurrent with query activity (continuous computing) in an efficient manner.
- Specifically prove WLM-A-3 provides the resource management capabilities necessary to favor one or the other workload and at the same time allow the less favored workload the ability to consume any spare resource without impact to the favored workload.

Methodology

Four separate tests comprised this continuous computing test case. The configurations and methodologies are described in 3.6, “Concurrency” on page 83.

Analysis and conclusions

Data warehouse refreshes and queries can run concurrently with little impact to the utilities or the query workload, and one workload can be favored over the other. When the query workload had higher priority, there was minimal impact on query throughput. Basically, the system supported the same query throughput levels for query workload by itself and queries plus refresh.

When the LOAD/RUNSTATS were the highest priority work in the system (WLM-A-34), there was practically no change in the elapsed time of these utilities compared to when they were run stand-alone (WLM-A-31). There was a slight impact to the query workload when query workload was favored (WLM-A-33), but again not a great deal, as this was an expected result. The large queries suffered more than the other classes of queries because, being the lowest priority work in the system, most CPU resources were taken away from them to allow the LOAD/RUNSTATS to run.

One of the key factors that allowed the queries and refreshes to run concurrently with hardly any impact to the running work and, more importantly with no show-stopping problems, was the fact that the table spaces are partitioned by date, and type 2 indexes are used.

Because of the concurrent query and load activity, there was inter-DB2 R/W interest, and the page sets/partitions became group buffer pool-dependent in the data sharing environment. From the results achieved this did not seem to cause any impact to the query response time; for details see *DB2 MVS/ESA V4 Data Sharing Performance Topics*, SG24-4611.

7.7 Summary

The S/390 environment with a DB2-based EDW together with OS/390 and Workload Manager makes an ideal team for managing a very large EDW. Our tests prove not only the importance of workload balancing, but the capabilities that an S/390 environment can provide.

It is important to keep the workload issues mentioned in this chapter a priority with your EDW designs. They are key to any successful implementation.

Regarding scalability, our tests focused on throughput scalability, one of the key customer issues. Others not mentioned are server scalability with an increase in users and servers, server scalability using CP-bound queries, and data scalability.These types of scalability are difficult to scale because they are less predictable than throughput. Data growth, server growth, and user counts can fluctuate and are difficult to predict. Testing can be performed but with some type of estimation.

We conclude this chapter with some general WLM recommendations. They may come in helpful.

- Use multiple service classes/multiple periods (consider response time goals for early periods and velocity goals for later periods).
- Set response time goals for canned queries/reports.
- Utilize dynamic policy changes.
- Consider resource groups for political battles.
- Create DISCRET and HIVEL service classes.
- Utilize report classes to assist capacity planning.
- Monitor workload activity reports.

Chapter 8. Scalability

What is scalability in a data warehouse environment? It represents how much work can be accomplished on a system when there is an increase in the number of users, the volume of data and/or processing power.

Why are most customers seeking improved scalability solutions for their VLDB data warehouse? Published technology industry reports show three typical workload trends in the data warehouse area.

The first trend is database growth. By the end of 1999, about 30 percent of data warehouse sites will exceed one terabyte of data, and many companies are mentioning that annual growth rates in database sizes are anywhere from 8 percent to more than 200 percent.

The second trend is that significant growth in user populations is also expected. One company is expecting the number of in-house BI users to increase more than three times, and another is expecting to expand its user base from 200 to more than 2000 within 18 months. Data warehouse access through Web connections offered to the suppliers, business partners, contractors and general public, has been started in many companies.

The third trend is that there are also pressures to improve query response time even as workloads are expanding.

This chapter explains the scalability of S/390 to cope with workload trends based on the experience of the customer case studies, which prove that S/390 has the ability to provide proportional performance to the growth of BI users, data volume and processing power.

Because of the growth in the number of business intelligence (BI) users, amount of data, and complexity of processing, the building of BI solutions requires not only careful planning and administration, but also the implementation of scalable products (both in hardware and software) that can deal with the growth in the use of these systems.

8.1 Scalability issues

There are a number of situations in a VLDB data warehouse environment which should be addressed in managing these workload trends. They are posed in the following questions:

- When the number of users increases, what will happen to query response time and throughput?

- When the volume of data grows, is there a disproportional performance degradation in the response time of queries?
- As processing power is expanded, will performance improvement in query response time and query throughput be achieved? And is it possible to support more end users?
- Does the current platform provide a path to scale up in order to prevent a warehouse from being outgrown?
- Can the server and database provide adequate I/O bandwidth to move large amounts of data within a given duration?
- Does data loading time increase linearly as data volume increases?
- Is it possible to update or refresh a data warehouse or data mart within given batch windows as data volume grows?
- Can normal maintenance such as backup or reorganization occur during batch windows as the volume of data grows?

In order to answer those questions, you need to understand several scalability attributes of S/390 for a VLDB data warehouse. In the following sections we discuss these attributes:

- User scalability
- Processor scalability
- Data scalability
- I/O scalability

8.1.1 User scalability

As users are added, is each individual query's response time elongated proportionally to the increase in concurrency?

User scalability ensures performance (response time) scales proportionally as the number of active concurrent users grows. An undesirable impact would be showing a greater than N-times increase in query response time with an N-times increase in a number of users.

As the number of concurrent users grows, a scalable BI system should maintain consistent system throughput while individual query response time is increasing proportionally. Data warehouse access through Web connection makes this scalability even more essential in a VLDB environment.

It is also a desirable system attribute to provide optimal performance at both low and high levels of concurrency without altering the hardware configuration

or system parameters or reserving capacity to handle an expected level of concurrency.

S/390 is able to provide both optimal single query response times and optimal throughput for all concurrency points, without the need for system parameter or hardware alterations. This sophistication is built into the DB2 optimizer, in its determination of parallel degrees. Workload Manger also provides for intelligent use of system resources, as well as the flexibility to yield optimal performance for both single query and fluctuating concurrency levels without the need for re-configuration or capping. See 5.5 "Resource management: intelligent resource sharing" on page 119 for more information.

Figure 67 shows how a mixed query workload scales while the concurrency levels are scaling up.

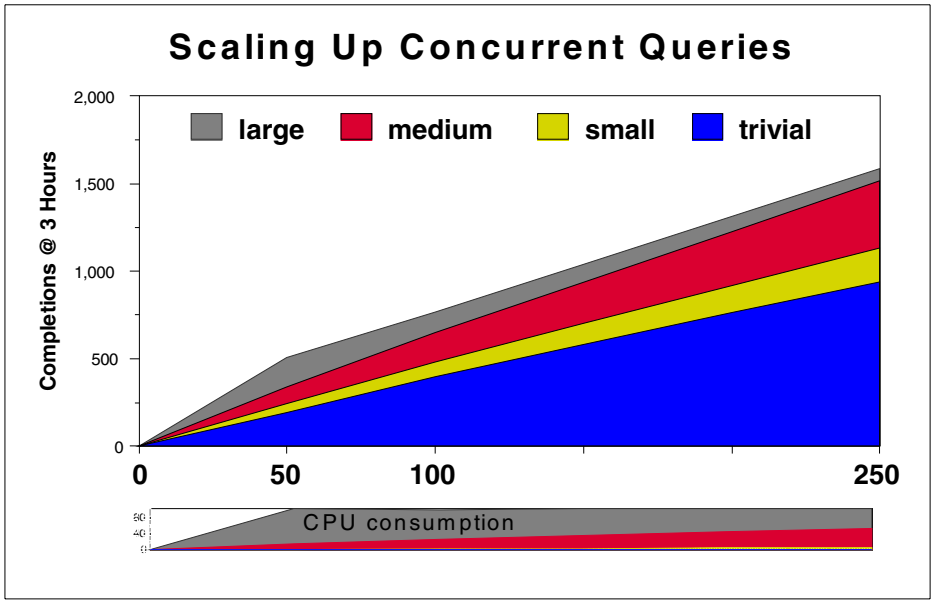


Figure 67. Throughput and CPU consumption per query class

In the top graph, the X-axis shows the number of queries, which grows from 20 to 250 (20, 50, 100, 250), and the Y-axis represents the number of completed queries within 3 hours. See Appendix B, “Query information” on page 235 for the description of query type, such as trivial, small, medium and large query.

The bottom graph simply shows how resources are consumed by the different workloads over the same 3 hours. You can see that the large queries continue to use the largest amount of resource, but decreasingly so as the system becomes busier with a concurrency level increase. And, as you would expect, the small and trivial queries use the smallest amount of resource.

Notice that we added more users even after the system saturation point (in this case, when the number of concurrent queries reached 50), which resulted in a growth of overall system throughput proportionally as the total number of completed queries increases. The behavior is due to the workload management policy which is designed to fit the query environment.

Figure 68 shows the same result, but the bottom graph depicts a response time point of view.

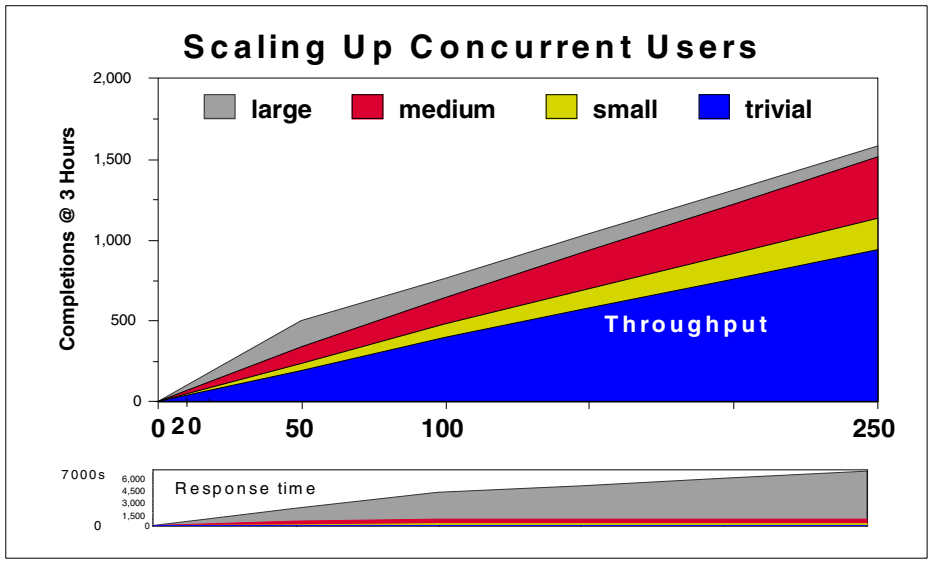


Figure 68. Throughput and response time per query class

The response time for trivial, small and medium queries increases less than linearly even after the system saturation point, and for large queries, response time increases in a worse than linear fashion after the system saturation point.

S/390 is able to keep query response times consistent for small queries with WLM period aging functions, which prevent large queries from monopolizing

the system. For more details about period aging, see 7.3.1 "Favoring short-running queries" on page 168.

Figure 68 on page 202 shows the overall affect of period aging and prioritization on response times. The large queries are the most affected, and the other queries are minimally affected. This behavior is desirable, because while end users of trivial and small queries expect to get an answer immediately, end users of large queries expect longer response times in acceptable ranges.

8.1.2 Processor scalability

Processor scalability attributes beyond a single SMP node, with an increased number of users, is one of a data warehouse customer's considerations. This ensures performance (such as throughput and response time) scales appropriately when multiplying the processing power.

Hardware processor scalability can be defined as the ability of a workload to fully exploit the capacity within a single SMP node, as well as the ability to utilize the capacity beyond a single SMP. Many people want to know whether any significant increase in performance would occur as processors are increased within an SMP instance as well as across multiple SMP nodes in a Parallel Sysplex.

The Parallel Sysplex configuration is able to achieve a near 2X increase in system capacity by adding another server to the configuration. This means that when more processing power is added to the sysplex, more end-user queries could be processed with a constant response time, more query throughput can be achieved, and the response time of a CP-intensive query would decrease in a linear fashion. S/390 servers can be increased to a maximum of 32X in system capacity, with 32 nodes of up to 12 processors.

Throughput: How many more queries are completed when we scale processor power up twice? Figure 69 on page 204 shows the test results when twice the number of users are running queries on a two-server system which doubles computing power. Parallel Sysplex provides almost perfect scalability in a BI environment. The number of completed queries during 3 hours shows 96 percent throughput scalability, which is from 293 on SMP to 576 on Parallel Sysplex. The figure shows that S/390 with Parallel Sysplex is capable of keeping linear scalability in processors.

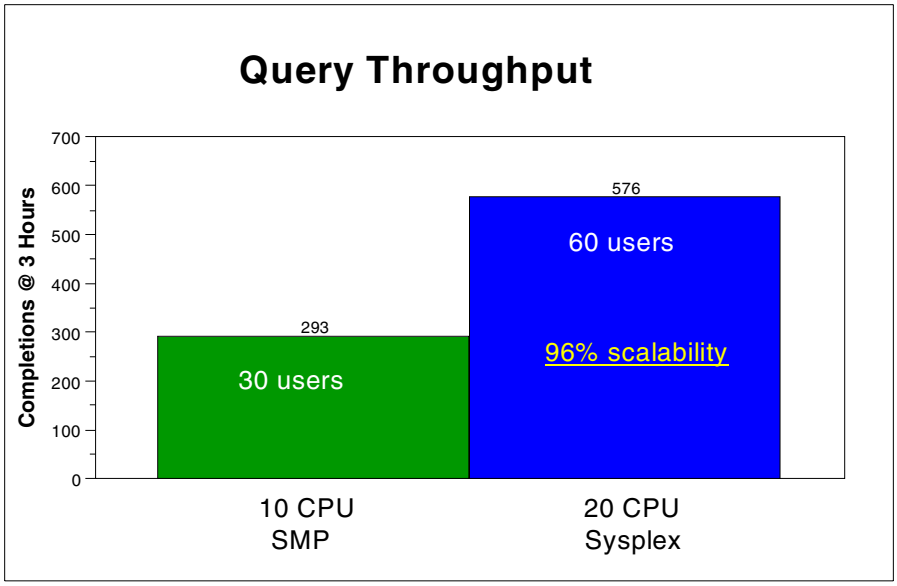


Figure 69. Processor scalability

Response time: How much faster will the CPU-intensive queries run if we get more CPU power by adding more SMP node into Parallel Sysplex? Is there any significant performance loss or overhead to maintain Parallel Sysplex?

Figure 70 on page 205 shows how soon a CPU-intensive query finished after the processor scaled up twice. We have 92 percent scalability in response time perspective (1874 sec. to 1020 sec.), while 937 sec. is the perfect scalability of 100 percent.

.

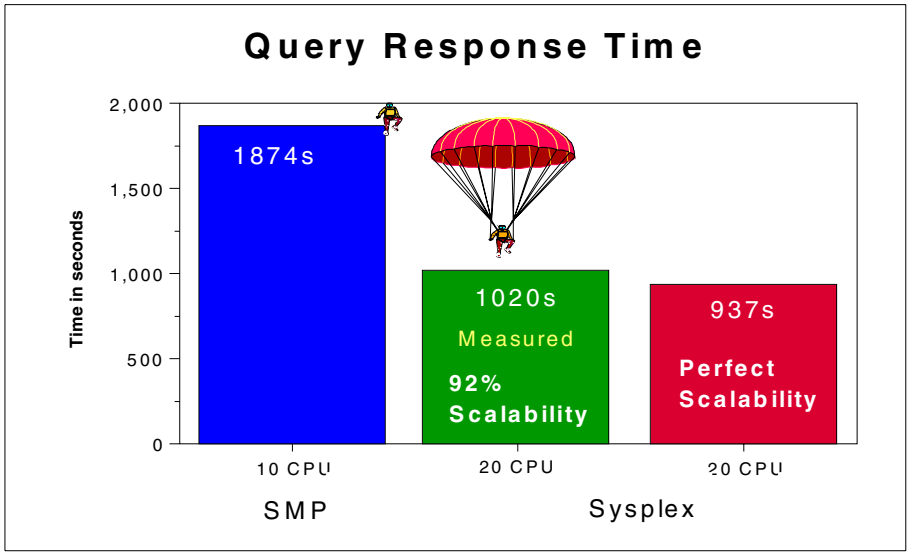


Figure 70. Server scalability

8.1.3 Data scalability

This data scalability examines the ability of a system to maintain expected performance in query response time and data loading time as database size increases. S/390 clearly provided this ability in a number of studies at the Teraplex center.

Consider these two cases. Data volume of a query may increase twice because of following:

1. Doubling data is needed in a given month due to a change in its business model.
2. The query needs an additional month of data (from one month to two months).

Even though the total volume of data is almost same in these two cases, the results for throughput and response time might be different. The difference between “Doubling data” and “Two months data” follows.

Doubling data: There is no change in SQL statements for the data doubling case. A SQL statement continues to point to a single month of data:

```
where date = 'month1'
```

To double data in a month, a company has to go through some change in its business model, such as adding more applications to its data warehouse, or merging with another company. Doubling data requires careful planning in a production system and the additional data must be added slowly.

Figure 71 shows the ability of a S/390 platform to manage increased data volume effectively.

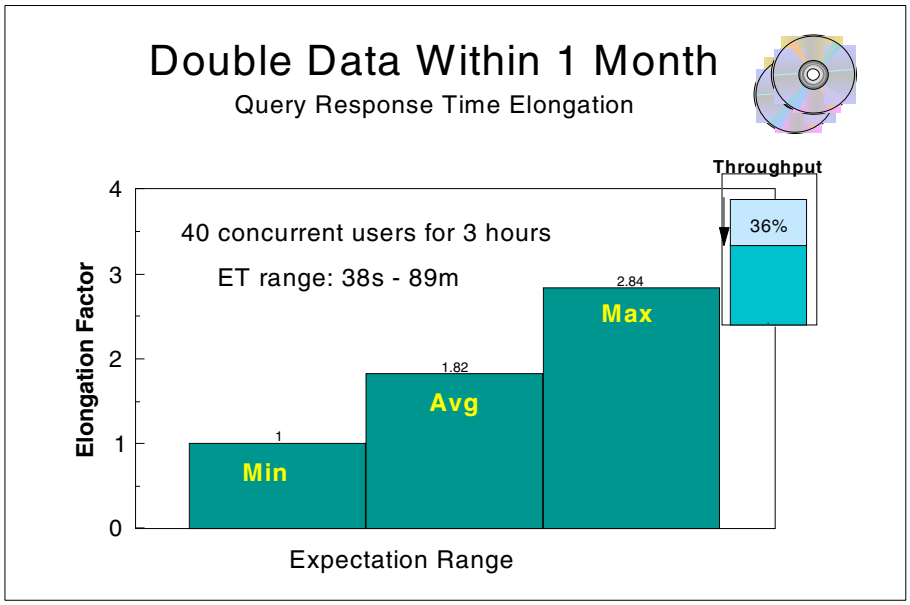


Figure 71. Data scalability

The doubling of the data in a given month impacted the response times for most queries. The minimum response time elongation shows 1X, the average is 1.82X and the maximum response time elongation is 2.84X. Running the sort function for a large number of qualified rows caused this elongation; that is, sort is not a linear function. The total throughput of queries when doubling data decreased by 36 percent. For detailed information, see 8.2.4 "Data scalability: growth within a month (SCA-A-4)" on page 214.

Two months data: Accessing two months of data implies a user is interested in using two months of data for his analysis. It simply requires rewriting a SQL statement to use two months of data.

There are two ways to write SQL statements to use two months of data. The more common technique is to specify the following:

```
where date in ('month1', 'month2')
```

Alternatively, you can use a between clause instead of an in clause.

It is more common to observe the “two months data” situation. A business user can ask for more than one month of data by simply changing the query predicate. For detailed information, see 8.2.5 "Data scalability test 2 (SCA-B-1)" on page 216.

Load time for database is another point to be addressed when you are considering data scalability in a VLDB data warehouse environment. S/390 with DB2 for OS/390 has been shown to have linear scalability on loading.

In one customer case, shown in Figure 72, loading 1.1 billion rows took 110 minutes, while loading 1.9 billion rows took only 200 minutes. The amount of data increased 1.75 times and the load times increased 1.81 times, which shows near-linear data scalability in database loading, and is an acceptable increase in elapsed time. This shows 97 percent scalability, which is very good. For more detailed information, see *Data Warehousing with DB2 of OS/390*, SG24-2249.

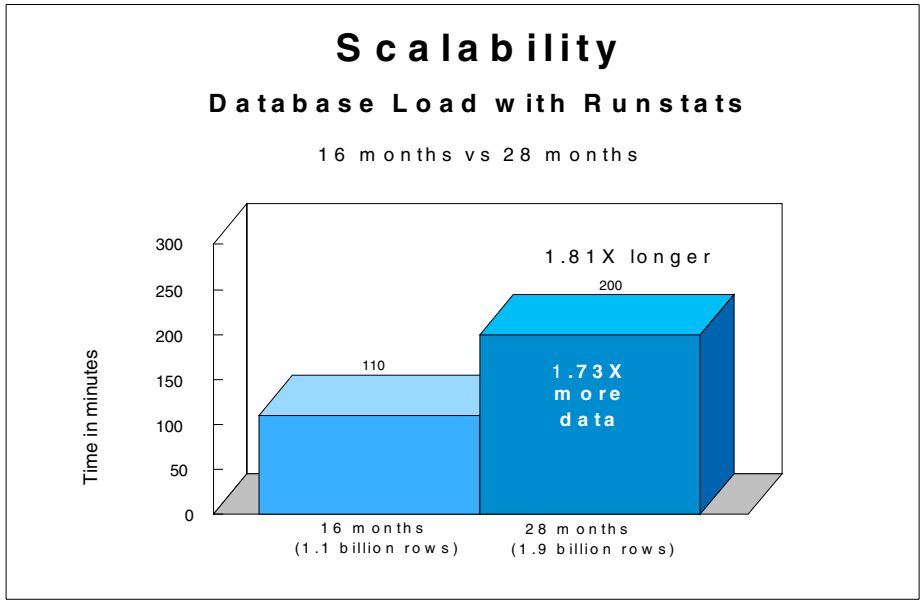


Figure 72. Data scalability in load time

8.1.4 I/O scalability

Another important aspect is the S/390 I/O capability to scale as a database grows in a VLDB data warehouse environment. If the server platform and database could not provide adequate I/O bandwidth, a progressively greater amount of time would be required for data movement tasks, and query response time elongation would occur.

Figure 73 shows the scalability characteristics of the I/O subsystem on an I/O-intensive query. We could get more than 93 percent of I/O scalability in response time of I/O-intensive queries, which means a 7.5 times speed-up of the query across the I/O subsystem configurations when we increased the number of CUs from 1 to 8. For detailed information, refer to *Data Warehousing with DB2 for OS/390,* SG24-2249.

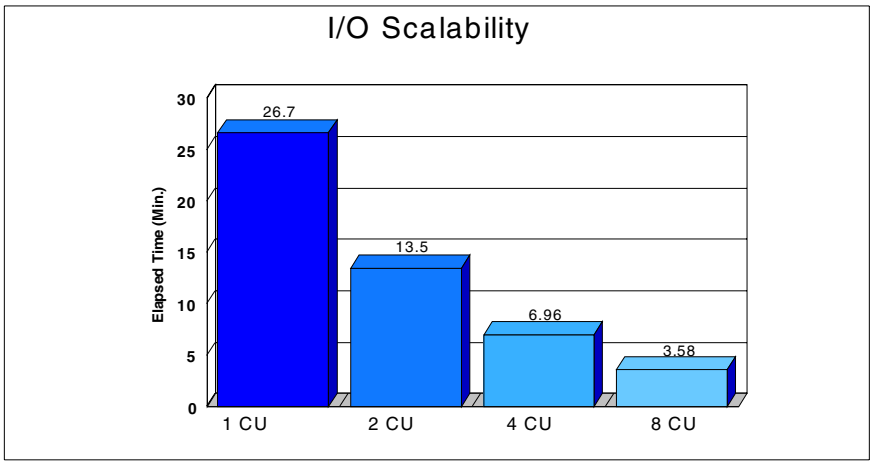


Figure 73. I/O scalability

Under S/390 architecture, you can add I/O devices independently to get better I/O performance without having to upgrade CPU capacity, and vice versa.

8.2 Case study tests

We tested five items in the categories of user scalability, processor scalability and data scalability for customer A, from the Banking industry, and also tested data scalability for customer B, from the Health Insurance industry.

While this does not represent a comprehensive set of scalability tests, such as we have conducted in other testing, these items represent some key attributes that a majority of customers want a better understanding of.

8.2.1 User scaling (SCA-A-1)

This test is designed to ensure performance scales appropriately with an increase in the number of active users. In addition, it is designed to understand the impact that different levels of concurrency have on response time and on key DB2 resources such as DB2 buffer pools and DB2 work space.

Objectives:

Prove query throughput remains consistent and the response time scales linearly with increased concurrency in accordance with the workload management policy.

Provide throughput and response times at various levels of concurrency for capacity planning of the data warehouse.

Provide data necessary for understanding the impact of concurrency on key DB2 resources.

Methodology:

Execute the general ad hoc query workload using the Tele-Processing Network Simulator (TPNS) driver at the following levels of active concurrent users: 20, 50, 100, 250.

Results:

Four separate measurements comprised the user scalability test case. Each measurement evenly distributed the users across the two servers. The configuration for each test varied in the number of users. The RMF interval was 15 minutes for each case. For 250 users, the measurement interval increased to 6 hours. See Table 43 on page 210 for detailed information about this test environment.

Table 43. User scalability test configuration

No. of total users	20	50	100	250
No. of servers	2	2	2	2
No. of trivial query users	2	5	10	25
No. of small query users	2	5	10	25
No. of medium query users	2	5	10	25
No. of large query users	14	35	70	175

Throughput results of each case are shown in Table 44 on page 210.

Table 44. No. of queries during three hours

No. of total users	20	50	100	250
Trivial queries	77	192	398	942
Small queries	20	48	87	195
Medium queries	66	101	161	380
Large queries	137	162	121	68
Total	300	503	767	1,585

Analysis/Conclusions:

For this particular workload, the point of system saturation is around 50 users. Figure 67 shows that even after system saturation, the throughput for the total number of completed queries increases as does the trivial, small, and medium classes.

As shown Table 45 on page 211, however, the large query class decreases as the number of users increase. Because workload policy was defined to favor small workloads, as the system becomes over-saturated (50 users), WLM period aging has an affect on the large queries. As there are more and more

shorter-running queries to consume the system resources, there are not as many CPU cycles left in the system for the longer-running queries.

Table 45. Number of queries per hour

No. of total users	20	50	100	250
Trivial	26	64	133	314
Small	7	16	29	65
Medium	22	34	54	127
Large	46	54	40	23
Total	101	168	256	529

A similar explanation can be made for query response time. Table 46 on page 211 shows the average response time of each query at the different levels of concurrency. For the trivial, small and medium queries, the response time increases less than linearly and even remains flat in some cases. This is a direct result of period aging and of S/390 capabilities for keeping response times consistent for shorter-running work, not letting the longer-running work monopolize the system.

Table 46. Average query response time (min.)

No. of total users	20	50	100	250
Trivial queries	0.07	0.12	0.08	0.12
Small queries	1.25	2.65	4.36	4.81
Medium queries	4.09	7.11	9.20	8.63
Large queries	16.06	29.36	58.84	101.41

This test shows the S/390's capabilities of providing performance scaled proportionally with high-level concurrency.

From the DB2 system point of view, for concurrent query execution, enough DB2 sort work space is needed.

8.2.2 Server scalability of a CP-intensive query (SCA-A-2)

This test case was designed to demonstrate that as more processing power was added to the sysplex, the response time of a CP-intensive query would decrease in a linear fashion.

Objectives:

Prove that the response time of a CP-intensive query decreases linearly as more servers are added to the sysplex.

Methodology:

Two tests comprised this test case. A description of each of their methodologies follows.

For test case SCA-A-21, the CP-intensive query executed via a batch job on one server. The bufferpools' assisting parallel sequential threshold (VPXPSEQT) was equal to 0 on the second server.

The CPU-intensive query got a parallel degree of 23.

For test case SCA-A-22, the same query then executed across two servers exploiting Sysplex Query Parallelism (SQP). To exploit SQP, the bufferpools' assisting parallel sequential threshold (VPXPSEQT) was equal to 100 on the second server. Table 47 on page 212 shows the system configuration for server scalability test.

Table 47. Server scalability test configuration

	SCA-A-21	SCA-A-22
No. of servers	1	2
VPXPSEQT (assisting parallel sequential threshold)	0 %	100 %
Measurement interval	Start to finish of query	Start to finish of query

Results:

Table 48 on page 212 shows the elapsed and CPU times for the two measurements.

Table 48. Server scalability test result

Test case	Parallelism type	Elapsed time (sec.)	CPU time (hh:mm:ss)
SCA-A-21	CP	1,874	04:34:28
SCA-A-22	Sysplex	1,020	04:35:03

Analysis/Conclusions:

An elapsed time of 937 seconds for the two-server run would have resulted in perfect scalability of 100 percent. The actual result yielded a scalability of 92 percent.

The overhead of using Sysplex Query Parallelism is minimal. In this test case there was only a 35-second increase in CPU time between test case SCA-A-21 and SCA-A-22.

8.2.3 Processor scaling with users (SCA-A-3)

This test is designed to ensure performance (throughput and response time) scales appropriately when doubling the number of active users and processing power. The processing power was doubled by adding another server to the configuration.

Objectives:

Prove query throughput grows linearly and query response times remain flat.

Methodology:

Execute the general ad hoc query workload to saturate the system (in this case, 30 users on one server). Afterward, double the number of active concurrent users across two servers, 30 users on each server.

Results:

The test configurations for both test cases are in Table 49 on page 213. The RMF interval was 15 minutes for each case.

Table 49. Processor scalability test configuration

	Single server	Double server
No. of servers	1	2
No. of trivial users	3	6
No. of small users	3	6
No. of medium users	3	6
No. of large users	21	42
No. of total users	30	60
Measurement interval	3 hours	3 hours

The measurements with 30 active concurrent users on one server and with 60 active users on two servers are shown in Table 50 on page 214.

Table 50. Processor scalability: query throughput and response time

	Single server		Double server	
Type of query	Completed queries	Average ET (min.)	Completed queries	Average ET (min.)
Trivial	120	0.10	240	0.06
Small	30	1.85	60	1.98
Medium	67	6.22	114	7.55
Large	76	35.04	162	32.91
Total	293	N/A	576	N/A

The distribution of queries across servers was excellent in the double server case. Each server ran the same queries at the same time. The second server got the large queries started before the first server. If the measurement interval had been slightly longer, two more queries from server one would have completed.

Analysis/Conclusions:

96 percent total throughput scalability was achieved ((576-293)/293*100); see Figure 69 on page 204. Throughput scalability of the trivial and small queries seems to be in line. However, the medium queries resulted in an increase in elapsed time, which resulted in only 82 percent scalability, while the large had a reduction in elapsed time, 106 percent scalability.

In the single server test case, CPU was 88 percent busy for the first 15-minute interval and 100 percent busy for the rest of the time. In the double server test case, CPU was 96 percent busy for the first 15-minute interval and 100 percent busy for the rest of the time.

8.2.4 Data scalability: growth within a month (SCA-A-4)

This data scalability test was designed to demonstrate that overall query throughput and response times show a near-linear behavior as database size grows and system resources remain constant.

Objectives:

Prove query Throughput and Response Time scale linearly as the data within a month grows.

Methodology:

To accomplish this test, forty TPNS users run queries for a three-hour interval. The query mix includes trivial, small, medium and large queries. This query activity is enough to saturate the sysplex. The following two tests were executed:

- Testcase SCA-A-41: queries access one month of data
- Testcase SCA-A-42: queries access one month of “double data”

Table 51. Data scalability1: data growth within a month test configuration

	SCA-A-41	SCA-A-42
Data size	One month	One month of double data
No. of servers	2	2
No. of trivial users	10	10
No. of small users	10	10
No. of medium users	10	10
No. of large users	10	10
No. of total users	40	40
Measurement Interval	3 hours	3 hours

Results

Table 52 contains the total number of query completions during the measurement interval for each query type.

Table 52. Data scalability1: query completion totals

Query type	SCA-A-41	SCA-A-42
Trivial	373	236
Small	632	619
Medium	106	43
Large	142	126
Total	1,253	1,024

The doubling of the data within one month impacted the response times for most queries. The minimum response time elongation was 1X. The average response time elongation was 1.82X, and the maximum response time elongation was 2.84X.

Analysis/Conclusions:

The query throughput for test case SCA-A-42 decreased by 36 percent when compared to test case SCA-A-41. This is better than expected when double the data is being accessed. The worst case for query throughput when doubling the data would be a 50 percent reduction.

The query response time elongation (or lack thereof) must be analyzed on an individual query basis. Some query response times did not change between the two tests. With these queries, the addition of more data within the partition did not affect them because they only access part of the data. Other queries experienced a 2X response time elongation, due to the fact that they were scanning 2X the data. Still other queries experienced up to a 2.84X elongation due to a large sort of qualified rows, which is not a linear function.

8.2.5 Data scalability test 2 (SCA-B-1)

This test case demonstrated that as the amount of data accessed for several of business questions increased, the elapsed time of each query increased in a less-than-linear fashion. All business questions consists of multiple SQL statements. This test was done for customer B from the Health Insurance industry.

Objectives:

Demonstrate the effects on business question elapsed time when increasing the mount of data accessed.

Methodology:

The business questions are run batch via DB2 sample programs such as DSNTEP2 and DSNTIAUL. The measurement was when the business question goes into execution until the final answer is written out.

Results:

The degrees of parallelism of all queries remained between 32 and 40.

The response time elongation multiple is always much less than the data increase multiple. The main reason for this is that more qualified rows are contained in each index and data page, thus avoiding significant increases in I/O.

BP1 contains data pages for large tables, while BP2 has index pages for large indexes.

Figure 74 shows the query response time elongation that occurred with an n-fold of volume of data accessed increase. The Y-axis shows the n-fold of data and query response time elongation, respectively.

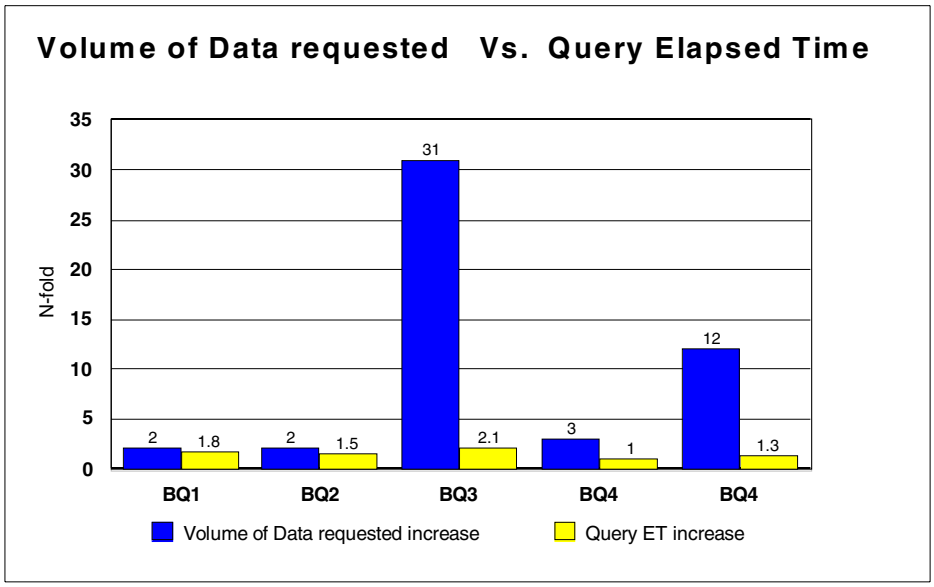


Figure 74. Data scalability test with query elapsed time elongation

Business Question 1: The amount of data accessed was increased from 1 year of data to 2 years of data (2X more data). There were 7133 rows in the base query table and 14482 rows in the “double data” table.

Business Question 2: The volume of data accessed was increased twice from 1 year to 2 years (2X data). There were 119159 rows in the one-year table and 188750 rows in the “double data” table.

Business Question 3: The data volume was increased to 31 times by changing the region (from *small* STATE to *large* STATE). Although there is an increase of 31-fold in data volume, the query response time increase is small because there were more rows with a *large* state in each index and data page, when compared to a *small* state.

This avoids having the number of getpages increase by a multiple of 31, which in turn reduces asynchronous reads significantly. All these factors contribute to a lesser response time elongation.

Business Question 4: We tested Business Question 4 twice using different increases in data volume accessed.

The volume of data accessed was increased to 3X (from one month of data to three month’s of data), and 12X (from one month to one year).

Although there is 3 times more data in the “increased data” run, elapsed time elongation is minimal because the additional qualified rows are contained in approximately the same number of pages. In the case of 12X more data, we scanned more pages.

With a good partitioning strategy based on the business questions, minimal query elapsed time elongation can be achieved even when there is an n-times increase in data volume.

8.3 Summary

Many data warehouses are exceeding one terabyte with rapid annual growth rates in database size. User populations are also increasing by broadening user groups such as marketers, salespeople, financial analysts and customer service staffs internally, as well as access by their business partners and customers through Web connections externally.

Scalability in conjunction with workload management is a key issue for any data warehouse platform.

The test results of case studies with S/390 show that queries, in any category of work, do not experience a disproportional increase in elapsed time as query volumes or data volumes are increased. This is mainly due to the use of OS/390 Workload Manager's functions.

S/390, as a data warehouse environment, can manage workload scalability in both an SMP and Parallel Sysplex environment with extremely high system utilization. The DB2 optimizer and Workload Manager make VLDB data warehouses easier to scale up in order to cope with business objectives.

We conclude this chapter by summarizing S/390's capability in each scalability attribute, based on the results of case studies.

- S/390 is able to linearly scale query response times and query throughput when user populations or concurrent queries increase.
- S/390 ensures performance scales proportionally when multiplying processing power.
- S/390 is able to maintain both the user's expectation in query response time and data load time as database size increases.
- S/390's I/O subsystem ensures adequate I/O bandwidth scalability to process an I/O-intensive query.

Chapter 9. S/390 strengths for BI applications

In this chapter, we summarize the strengths of S/390 with appropriate software for BI applications, such as parallelism for query and data population, workload management, scalability with user/data growth and effective resource management described in previous chapters.

The primary benefits from those strengths are:

- Faster deployment
- Better performance
- Lower costs

Performance and throughput
VLDB business intelligence solutions require high-end query performance and extensive capabilities for handling large volumes of data, as well as the high levels of robustness demanded by business-critical operations. The S/390 platform excels at combining these elements:

- Fast processors combined with parallel technologies in hardware and software as well as unique hardware compression provides unmatched performance in database load and refresh.
- DB2 query parallelism and cost-based optimization, together with OS/390 Workload Manager, achieve high performance of complex queries as well as consistently high levels of throughput with large mixed query workloads.

Parallelism
A key factor in achieving performance in the data warehouse environment is parallelism and S/390 provides the best parallelism:

- Parallelism of individual batch processes for initial data load and data refresh with SmartBatch, DB2 parallel utilities, as well as Independent Software Vendors, enable installations to meet shrinking batch windows.
- Data partitioning for parallel query processing, up to 254 partitions
- Application-level parallelism support with S/390 parallel open editor support.

Workload management
OS/390 Workload Manager controls resources such as CPU, I/O and storage within a single system or within a sysplex environment based on business priorities set by the installation. WLM helps eliminate bottlenecks and ensure

consistent throughput using workload prioritization techniques, such as period aging or workload prioritization.

With WLM, you can realize benefits in a mixed workload environments as follows:

- Business importance is honored through dynamic prioritization of critical work as it enters the system.
- Long-running resource-intensive work can complete without impacting short-running queries.
- System resources are managed efficiently beyond system saturation, maintaining throughput and performance requirements.

Leverage of existing S/390 assets and skills

For customers that have S/390-based operating systems, the ability to leverage existing S/390 investments and skills can reduce implementation time and cost of an enterprise data warehouse and data marts.

If a large portion of the source data on S/390 is in the form of DB2 databases, IMS/DB, VSAM and sequential files, deployment of BI solutions is a faster, less expensive process with lower levels of risk.

- Lower risk
- Faster deployment
- Lower cost

Scalability

When planning the capacity requirements for an enterprise data warehouse, there is no fool-proof method for estimating growth. Therefore, it is critical to be able to understand how the data warehouse infrastructure will scale to meet predictable and *unpredictable* growth. A successful data warehouse depends on an infrastructure that is able to respond consistently to unforeseen demands that may result from unanticipated changes in the business environment, as well as respond predictably to the planned growth associated with a well thought-out roll out and expansion plan. S/390 is uniquely qualified to address the challenged of predictable as well as unpredictable growth.

- Simple upgradability across all processor families and models, with dynamic growth options such as Capacity on Demand.
- Incremental growth from a small standalone department data mart or logical partition (LPAR) to data sharing across a Parallel Sysplex without a system or database migration.

- Dynamic workload management to insure consistent throughput and workload prioritization beyond 100 percent effective processor utilization.
- Unlimited connectivity of processors to external storage devices.
- Full utilization of computing power is provided, which means MVS systems typically run with processor utilization of 90 to 100 percent, with minimal impact on system operation. Most other systems cannot approach this without introducing response time/elapsed time bottlenecks.

Synergy of S/390, OS/390 and DB2

Interoperability between DB2 as the database manager, OS/390 operating system and S/390 hardware provides a powerful, high-performance and cost-effective data warehouse solution that offers the following:

- Support for large tables: each table can be up to 16 TB in DB2 V6.
- Hardware-assisted data compression.
- I/O parallelism, CPU parallelism, sysplex parallelism.
- Hiperspace to keep large amounts of data in memory to reduce the I/O.
- Architectural integration: many products functioning as one system.
- A DB2 optimizer to build effective access paths for sophisticated queries from business questions with robust catalog statistics.

Availability

- Unmatched reliability of S/390 hardware and software makes 24 x 365 operation possible
- Parallel utilities allow for refresh/update/backup the data warehouse and data mart concurrently with SQL access
- Workload Manager ensures high priority work receives required resources
- Parallel Sysplex gives you the ability to do hardware or software maintenance/upgrades while still having full access to the data.

Connectivity

S/390 is no longer an island in an enterprise's network. The open systems capabilities and infrastructure for applications allow S/390 to fully participate in enterprise and public network solutions in a two- or three-tiered application architecture.

- Web-enabled data access through Net.Data
- Client/server support through DRDA with TCP/IP and SNA with or without ODBC, JDBC and SQLJ
- Native TSO

- A wide range of specific vendor-supplied and customized tools and applications
- A LAN/WAN connection with UNIXs, NTs, PCs in an enterprise network

I/O subsystem

High I/O throughput and system-wide data compression have emerged as major strengths of the S/390 for VLDB data warehouse because of the following:

- Several I/O alternatives, including SmartBatch, HiperBatch, LSPR Batch, SMS Data Striping and so forth, enable applications to more fully utilize the processor, minimizing elapsed time such as batch windows and response times.
- SMS (ACS routines), DFHSM, RVA/Snapshot/Data Striping and high volume tape data processing in a S/390 environment are stronger I/O alternatives than any other platform provides.
- Backup/Recovery with the RVA Snapshot feature substantially reduces the elapsed time needed to back up and release data sets for further processing.
- Disaster recovery with, for example, XRC.
- Unlimited connectivity to processors.

System management

- WLM dynamic management of workloads to meet business priorities.
- Db2 utility suite on S/390.
- A wealth of diagnostic and performance monitoring tools exist.
- A wealth of third-party products are available which enable robust capacity planning capability, monitoring, and management of I/O devices.
- Robust security management is available with RACF as the security server and the Domino Go Webserver as the secure Web server.

Appendix A. System settings

This appendix contains the definitions of system environments such as operating system, DB2 system, and WLM definitions.

A.1 DB2 settings

Table 53 shows DB2 parameters (called DSNZPARM) that we used in the case studies.

Table 53. DSNZPARM used in case studies

	Case study A	Case study B
CTHREAD	500	750
EDMPOOL (4 KB)	51,200,000	307,200,000
SRTPOOL (4 KB)	65,536,000	65,536,000
MAXRBLK (4 KB)	63,881,216	63,881,216
Bufferpool 0/Hiperpool	1000/0	1000/0
Bufferpool 1/Hiperpool	60,000/120,000	30,000/150,000
Bufferpool 2/Hiperpool	60,000/120,000	30,000/150,000
Bufferpool 3/Hiperpool	40,000/80,000	20,000/100,000
Bufferpool 4/Hiperpool	40,000/80,000	20,000/100,000
Bufferpool 5/Hiperpool	3,000/10,000	3,000/10,000
Bufferpool 6/Hiperpool	3,000/10,000	3,000/10,000
Bufferpool 7/Hiperpool	100,000/200,000	50,000/250,000
3990 Cache (SEQCACH)	SEQ	SEQ
LOGLOAD	50,000	50,000
SMF ACCT class	1,2,3	1,2,3
SMF STAT class	1,3,4,5	1,3,4,5
STATIME (min.)	5	5
Data Sharing Enabled	Yes	Yes
Parallelism ASSIST	Yes	Yes
Parallelism COORDNTR	Yes	Yes

A.2 RVA definition

IOCP

```
CHPID PATH=01,TYPE=CNC,SWITCH=01
CHPID PATH=21,TYPE=CNC,SWITCH=02
CHPID PATH=61,TYPE=CNC,SWITCH=03
CHPID PATH=81,TYPE=CNC,SWITCH=04
CHPID PATH=A1,TYPE=CNC,SWITCH=05
CHPID PATH=C1,TYPE=CNC,SWITCH=06
CHPID PATH=E1,TYPE=CNC,SWITCH=07
CHPID PATH=F1,TYPE=CNC,SWITCH=08

*IOCP
 CNTLUNIT CUNUMBR=2D00,PATH=(01,21,61,81),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=0,                     +
          LINK=(C0,C0,C0,C0)
 CNTLUNIT CUNUMBR=2D01,PATH=(A1,C1,E1,F1),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=0,                     +
          LINK=(C0,C0,C0,C0)
 IODEVICE ADDRESS=(2D00,64),CUNUMBR=(2D00,2D01),UNIT=3390,          +
          STADET=Y,FEATURE=(SHARED,ALTCTRL),UNITADD=00
*IOCP
 CNTLUNIT CUNUMBR=2D40,PATH=(01,21,61,81),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=1,                     +
          LINK=(C0,C0,C0,C0)
 CNTLUNIT CUNUMBR=2D41,PATH=(A1,C1,E1,F1),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=1,                     +
          LINK=(C0,C0,C0,C0)
 IODEVICE ADDRESS=(2D40,64),CUNUMBR=(2D40,2D41),UNIT=3390,          +
          STADET=Y,FEATURE=(SHARED,ALTCTRL),UNITADD=00
*IOCP
 CNTLUNIT CUNUMBR=2D80,PATH=(01,21,61,81),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=2,                     +
          LINK=(C0,C0,C0,C0)
 CNTLUNIT CUNUMBR=2D81,PATH=(A1,C1,E1,F1),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=2,                     +
          LINK=(C0,C0,C0,C0)
 IODEVICE ADDRESS=(2D80,64),CUNUMBR=(2D80,2D81),UNIT=3390,          +
          STADET=Y,FEATURE=(SHARED,ALTCTRL),UNITADD=00
*IOCP
 CNTLUNIT CUNUMBR=2DC0,PATH=(01,21,61,81),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=3,                     +
          LINK=(C0,C0,C0,C0)
 CNTLUNIT CUNUMBR=2DC1,PATH=(A1,C1,E1,F1),                          +
          UNITADD=((00,064)),UNIT=3990,CUADD=3,                     +
          LINK=(C0,C0,C0,C0)
 IODEVICE ADDRESS=(2DC0,64),CUNUMBR=(2DC0,2DC1),UNIT=3390,          +
          STADET=Y,FEATURE=(SHARED,ALTCTRL),UNITADD=00
*IOCP
```

Figure 75. IOCP statements of RVA subsystem (partial)

HCD Definition

```
Configuration ID . .: ITSO          ITSO
Device number  . . : 2D00        Device type  . . . : 3390
Generic / VM device type  . . . . : 3390

ENTER to continue.

Parameter/
Feature      Value   Req.  Description
OFFLINE       No           Device considered online or offline at IPL
DYNAMIC       Yes          Device supports dynamic configuration
LOCANY                     UCB can reside in 31 bit storage
ALTCTRL       Yes          Separate physical control unit path
SHARED        Yes          Device shared with other systems
SHAREDUP      No           Shared when system physically partitioned
```

Figure 76. HCD statements used for each RVA device

A.3 2216 definition

IOCP

```
CHPID PATH=3C,TYPE=CNC,SWITCH=73
*IOCP
 CNTLUNIT CUNUMBR=A00,PATH=(3C),LINK=(CE),UNIT=3172,UNITADD=((00,16))
 IODEVICE ADDRESS=(A00,16),CUNUMBR=A00,UNIT=3172,UNITADD=00
*IOCP
```

Figure 77. IOCP definition for 2216

HCD Definition

```
Device number . . . . . : 0A00
Device type . . . . . : 3172
Serial number . . . . . :
Description . . . . . :
Volume serial number . . . . :      (for DASD)
Connected to CUs :

Parameter/
Feature    Value   Req.  Description
OFFLINE    No           Device considered online or offline at IPL
DYNAMIC    Yes          Device has been defined to be dynamic
LOCANY     No           UCB can reside in 31 bit storage
```

Figure 78. HCD definition for 2216

TCP/IP Profile

```
; FOR SYSTEM #1 - NOTE PORT VALUE IS 0 - MUST AGREE WITH DEF ON 2216
; Device A00,A01 is 2216
;
 DEVICE DEVA00 LCS  A00
 LINK LNKA00  IBMTR 0 DEVA00

; FOR SYSTEM #2 - NOTE PORT VALUE IS 1 - MUST AGREE WITH DEF ON 2216
; Device A00,A01 is 2216
;
DEVICE DEVA00 LCS  A00
LINK LNKA00  IBMTR 1 DEVA00

; To get the 2216 routing correctly between S/390s, one has to  put a loopback
 address in the home ; statement, where the loopback address is the cluster address
specified on the 2216.
; For example,
HOME
  192.168.42.2   LNKA00
  192.168.41.100 LOOPBACK

;   192.168.41.100 is the address to which to point the webbrowser.  It's
;   the cluster address specified in the 2216.  The server addresses
;   are 192.168.42.2 (QP01) and 192.168.43.2 (QP02)

GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value
 192.168.42.1 =      LNKA00   2000     HOST
 9.117.59.251 =      OSA900   4000     HOST

; Indirect Routes - Routes that are reachable through routers on my
;             network.
;
; Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value

 192    192.168.42.1  LNKA00   2000     0
 9      9.117.59.251  OSA900   4000     0
 32     9.117.59.251  OSA900   4000     0
 33     9.117.59.251  OSA900   4000     0

; Default Route - All packets to an unknown destination are routed
;            through this route.
;
; Network  First Hop  Link Name Packet Size  Subnet Mask  Subnet Value

DEFAULTNET 192.168.42.1  LNKA00     2000    0

;START DEVICE STATEMENT
START DEVA00
```

Figure 79. TCPIP profile for 2216

A.4 WLM settings

The following sections contain several examples of the WLM application definitions and service class definitions.

A.4.1 Service class definitions associated with TSO policy

Table 54. Normal TSO data warehouse and data mart users

Period	Duration (Service units)	Duration (Apprx. CP sec.)	Importance	Goal
1	5,000	1	2	Execution velocity of 80
2	50,000	10	2	Execution velocity of 60
3	1,000,000	200	3	Execution velocity of 40
4	30,000,000	6000	4	Execution velocity of 30
5			5	Discretionary

Table 55 shows that duration was eliminated from the service class definition.Therefore all queries (trivial, small, medium and large) were treated equally, essentially giving service in an unsophisticated round-robin fashion.

Table 55. TSO data warehouse users without period aging for queries

Period	Duration (Service units)	Duration (Apprx. CP sec.)	Importance	Goal
1			2	Execution velocity of 70

Table 56. Critical TSO users

Period	Duration (Service units)	Duration (Apprx. CP sec.)	Importance	Goal
1			2	Execution velocity of 80

Table 57. Normal data warehouse refresh utilities

Period	Duration (Service units)	Duration (Apprx. CP sec.)	Importance	Goal
1			5	Execution velocity of 10

Table 58. Favored data warehouse refresh utilities

Period	Duration (Service units)	Duration (Apprx. CP sec.)	Importance	Goal
1			2	Execution velocity of 80

A.4.2 Web server configuration file

```
APPLENV /web-cgi/db2www/hi* WEBHI WEBHI
APPLENV /* WEBHTML
```

Figure 80. Web server configuration file

A.4.3 Workload Manager panel

WLM application environment panel for WEBHI:

```
Appl Environment Name . . WEBHI
 Description . . . . . . . . . . . . WEB Power Users
 Subsystem type . . . . . . . . . IWEB
 Procedure name . . . . . . . . .IMWHI
 Start parameters . . . . . . . IWMSN=&IWMSSNM,IWMAE=WEBHI

Limit on starting server address spaces for a subsystem instance: No limit
```

Figure 81. WLM application environment for WEBHI

WLM application environment panel for WEBHTML:

```
Appl Environment Name . . WEBHTML
 Description . . . . . .HTTP Environment
 Subsystem type  . . . . . IWEB
 Procedure name  . . . . . IMWIWM
 Start parameters  . . . . . IWMSN=&IWMSSNM,IWMAE=WEBHTML

Limit on starting server address spaces for a subsystem instance:  No limit
```

Figure 82. WLM application environment for WEBHTML

WLM classification rule panel for IWEB:

```
Subsystem Type IWEB - Web Server Subsystem

Created by user USER01 on 1999/04/20 at 09:46:41
Last updated by user USER01 on 1999/04/27 at 12:20:32

Classification:

  Default service class is WEB
  Default report class is WEB

     Qualifier  Qualifier     Starting     Service     Report
  #   type       name         position     Class       Class
  -  ----------  ----------   ---------    ----------  ----------
  1   TC         WEBHI                     WEBHI       WEBHI
```

Figure 83. Example of Web classification rule

WLM service class panel for service class WEBHI:

```
Service Class WEBHI - Hi priority Web Users

 Created by user USER01 on 1999/04/20 at 13:32:12
 Base last updated by user USER01 on 1999/05/05 at 17:06:40

 Base goal:

  #  Duration   Imp   Goal description
  -  ----------  ----  ---------------------------------------
  1              1     Execution velocity of 90
```

Figure 84. Example of service class for high priority Web users

WLM service class panel for service class WEB:

```
Service Class WEB - Web Users

 Created by user USER01 on 1999/04/20 at 09:40:47
Base last updated by user MIKET on 1999/05/05 at 14:49:37

 Base goal:

  #  Duration   Imp  Goal description
  - ---------     -   ---------------------------------------
  1  5000        2   Execution velocity of 80
  2  50000       2   Execution velocity of 60
  3  1000000     3   Execution velocity of 40
  4  10000000    4   Execution velocity of 30
  5                  Discretionary
```

Figure 85. Example of service class for Web users

Appendix B. Query information

The query workloads used in the case study tests are representative of actual queries the two customers expected to run in their production environments. The query suite consisted of multi-table joins, correlated queries, unions, group by, and order by processing, to name a few. Each query accepted a set of input variables which, when applied to the SQL, produced hundreds of unique queries used throughout the evaluation.

Queries were classified into three major work categories based on the number of rows qualified and returned for processing. Queries that fell into the small category qualified 1000 rows, medium queries qualified 50,000 rows and large queries qualified 250,000 rows.

Standalone (individual query) executions, as well as various combinations of mixed query workloads, were examined. Mixed workloads were comprised of various percentages of different query types. Table 59 shows the standalone execution time of different query types.

Table 59. Workload classification standalone execution times

Query type	Elapsed Time	CPU	# Queries
Trivial	0 - 22s	0 -10s	145
Small	22s - 160s	10s - 100s	49
Medium	160s - 360s	100s - 240s	20
Large	360s -	240s -	38

Table 60 shows the number of service units of the G5 processor for each query type.

Statistical Overview:

Table 60. Workload classification for G5 processor (1 sec = 5,070 SUs)

Query Type	CPU	Service Units
Trivial	0 -10s	0 - 50,700
Small	10s -100s	50,701 - 507,000
Medium	100s - 240s	507,001 - 1,216,800
Large	240s -	1,216,800 -

B.1 Queries

We have four sample typical queries here. In order to help you understand the workload of each query, Table 61 shows the database statistics for each table we used. Examples of different-sized queries follow:

Table 61. Banking database statistics DB 1998

Table name	Npages	Cardf	Record Length	Parti-tions	Bpool	SIZE (GB)
TESTA.TACCOUNT	11,627,804	111,278,325	1724	240	BP2	44.18
TESTA.TRELATION1	1,161,207	37,394,345	322	240	BP2	4.41
TESTA.TRELATION2	1,214,304	54,762,768	208	240	BP2	4.61
TESTA.TARRANGE	65,383,877	651,658,062	1549	240	BP2	248.45

Trivial query

```
SELECT  COUNT(*) FROM TESTA.TARRANGE
 WHERE COLA = '1998-09-30'
 AND COLB  = 'DDA'
 AND  COL1  BETWEEN '4330000000000000' AND '4340000000000000';
```

Figure 86. Example of trivial query

Small query

```
SELECT  C. COL1,  C.COL2,  C.COL3,
     C.COL4,  C.COL5,   A.COL4,  A.COL5
FROM   TESTA.TACCOUNT  A,
     TESTA.TRELATION1  B,
     TESTA.TRELATION2  C
WHERE   A.COLA = B.COLA2
 AND   B.COLA = C.COLA
 AND   C.COLA = '1998-08-31'
 AND   A.COLB = B.COLB
 AND   B.COLB = 'DDA'
 AND   B.COLC = 295
 AND   A.COLC = B.COLC
 AND   A.COL1 = B.COL1
 AND   A.COLD = B.COLD
 AND   B.COLE = C.COLE
 AND   B.COLF = C.COL1
ORDER BY COL1  ;
```

Figure 87. Example of small query

Medium query

```
SELECT C.COL1,
    C.COL2, C.COL3,
    C.COL4, C.COL5,
    A.COL4, A.COL5
FROM   TESTA.TACCOUNT A,
    TESTA.TRELATION1 B,
    TESTA.TRELATION2 C
WHERE  A.COLA = B.COLA2
 AND  B.COLA = C.COLA
 AND  C.COLA = '1998-08-31'
 AND  A.COLB = B.COLB
 AND  B.COLB = 'DDA'
 AND  B.COLC = 75
 AND  A.COLC = B.COLC
 AND  A.COL1 = B.COL1
 AND  A.COLD = B.COLD
 AND  B.COLE = C.COLE
 AND  B.COLF = C.COL1
ORDER BY COL1 ;
```

Figure 88. Example of medium query

Large query

```
SELECT COLC
  ,COL1 ,COL2 ,COL3 ,COL4
  ,COL5 ,COL6 ,COL7 ,COL8
  ,COL9 ,COL10 ,COL11 ,COL12
  ,COL13 ,COL14 ,COL15 ,COL16
  ,COL17 ,COL18 ,COL19 ,COL20
  ,COL21 ,COL22 ,COL23 ,COL24
  ,COL25 ,COL26 ,COL27 ,COL28
  ,COL29 ,COL30
FROM TESTA.TACCOUNT
WHERE COLB = 'DDA'
AND  COLC BETWEEN 173 AND 2000
AND COLA = '1998-09-30'
ORDER BY COL5 ;
```

Figure 89. Example of large query

"Killer" query

This is an example of a typical "killer" query.

```
SELECT COLB, COLC
     ,MIN(COL11) , MIN(COL12), MIN(COL13),
     ,MIN(COL14), MIN(COL15), MIN(COL16),
     ,MIN(COL17), MIN(COL18), MIN(COL19),
     ,MIN(COL20), MIN(COL21), MIN(COL22),
     ,SUM (CASE WHEN COL23 > '1998-08-31' THEN 1 ELSE 0 END),
     ,MIN(COL24), MIN(COL25), MIN(COL26),  ,MIN(COL27)
     ,MIN(COL28  * COL29)  ,MIN(COL30  * COL31)
     ,MIN(COL32  * COL33)  ,MIN(COL34  * COL35)
     ,MIN(COL36  * COL37)  ,MIN(COL38  * COL39)
     ,MIN(COL40  * COL41)  ,MIN(COL42  * COL43)
     ,MIN(COL44  * COL45)  ,MIN(COL46  * COL47)
    , MIN(COL48  * 10)
     ,MIN(COL49 * COL50)  ,MIN(COL51  * COL52)
     ,MIN(COL53  * COL54) ,MIN(COL55  * COL56)
     ,MIN(COL57  * COL58) ,MIN(COL59  * COL60)
     ,MIN(COL61  * COL62)
     ,MAX(COL11), MAX(COL12), MAX(COL13)
     ,MAX(COL14), MAX(COL15), MAX(COL16)
     ,MAX(COL17), MAX(COL18), MAX(COL19)
     ,MAX(COL20), MAX(COL21), MAX(COL22)
     ,MAX(COL24), MAX(COL25), MAX(COL77)
     ,MAX(COL28  * COL29)  ,MAX(COL30  * COL31)
     ,MAX(COL32  * COL33)  ,MAX(COL34  * COL35)
     ,MAX(COL36  * COL37)  ,MAX(COL38  * COL39)
     ,MAX(COL40  * COL41)  ,MAX(COL42  * COL43)
     ,MAX(COL44  * COL45)  ,MAX(COL46  * COL47)
    , MAX(COL48  * 10)
     ,MAX(COL49  * COL50)  ,MAX(COL51  * COL52)
     ,MAX(COL53  * COL54)  ,MAX(COL55  * COL56)
     ,MAX(COL57  * COL58)  ,MAX(COL59  * COL60)
     ,MAX(COL61  * COL62)
 FROM   TESTA.TARRANGE
 WHERE  COLA  BETWEEN '1998-07-31' AND '1998-09-30'
 GROUP BY  COLB, COLC
 ORDER BY  COLB, COLC ;
```

Figure 90. Example of killer query

Appendix C. Case study test scenario

In this appendix we list some test items that are needed for the customers who will build a very large database for BI applications, in order to validate that system's configuration, system resources, performing processes and end-user applications prior to implementing a data warehouse production system.

Not all test items in this section were performed for our case studies, since each test has its own objectives and the customers who participated in the tests had their own business problems, system environments, data characteristics and end-user requirements.

However, these tests are valuable, for example, in estimating the elapsed time of each task, verifying the SmartBatch for data movement, exploiting WLM functions in complex query environments, optimizing system resources, and developing required procedures for each task.

C.1 Raw data load test

Run the initial raw data load utility to determine the data load rate and compression ratio of a medium-sized customer databases with several non-partitioned indexes.

This test is performed to help establish some guidelines on sizing the system for each customer.

Objectives:

Measure the elapsed time, data load rate and compression rate when system resources are fully available.

Observe and document the optimal approach to load very large tables with more than 100 million rows.

Determine the viability of data compression.

Methodology:

Using IBM utilities or business partners' utilities (for example: BMC, Platinum and so on), run the initial raw data load.

While partitioning indexes are built during the load, non-partitioned indexes are built using RECOVER utilities. (With DB2 Version 6, you should use REBUILD INDEX utilities.)

The total elapsed time measured could be the time to load the data and the time to build the partitioning indexes and all NPIs for the tables.

Key metrics:

Elapsed time, data load rate and compression ratio.

Success criteria

Observe the standalone data load for *one* table which has non-partitioned indexes and record the data load rate and compression ratio. Then observe and record the data load rate and compression of *other* tables.

Measurement:

With SMF records and RMF intervals captured, the statistics would be elapsed time, CPU time, memory usage, %CPU utilization and %DASD utilization.

Data load is an I/O-bound job. If there are bandwidth constraints, it is not easy to get optimal load rates for the large tables. Extrapolation might be needed.

C.2 Single query performance test

Since there are multiple queries running concurrently in a production environment, the single query test is not of vital importance but the purpose of this test is to get a *baseline* performance run of each query with no resource contention.

Objectives:

Measure the response time of each business query when all system resources are available.

Methodology:

Run an individual query serially while all system resources are available. The response time should be measured from the time the query goes into execution till the final report is generated. Use QMF or DSNTEP2 (which is a

DB2 sample program). Review the optimizer's decisions such as the degree of parallelism, access path selection, and so on.

Key metrics:

Query response time measured from the start of query to end of query.

Success Criteria:

Better response time than obtained on the current implementation.

Measurement:

With SMF records and RMF intervals captured, the statistics would be elapsed time, CPU time, memory usage, %CPU utilization and %DASD utilization.

The queries are expected to be CPU-bound.

DB2 performance trace data could be captured, but the performance impact of the task should be considered.

C.3 Throughput test

After you run the baseline test (C.2, "Single query performance test" on page 240), you can perform the following complex tests and compare the results with the baseline's results.

C.3.1 Run multiple queries as a single unit of work

This test requires concurrent execution of queries. The elapsed time should be measured from the start of the first query to the end of the last query. This should not be a stress test.

Objectives:

Determine the shortest elapsed time to execute the multiple queries when the queries are run concurrently.

Determine the system utilization and performance during the test.

Achieve the best throughput rate.

Methodology:

Using the WLM, run multiple queries concurrently (which may be assigned priorities as necessary).

The WLM should be used in a normal way to control the level of concurrency and sustain the optimal throughput rate until the batch work unit of multiple queries is completed.

Key metrics:

Elapsed time from the start of the first query to the end of the last query.

System utilization and performance via SMF, RMF and DB2 performance trace records.

Success criteria:

The best result from several iterations should be acceptable. The result of this test is useful in sizing the customer's system.

Measurement:

The elapsed time from the start of the first query to the end of the last query should be measured. Several iterations with various concurrency levels may be necessary to achieve the best elapsed time.

C.3.2 Protect small resource queries from a large resource query

Objectives:

Ensure that the throughput of a small query workload is not impacted when a large query enters the system.

Methodology:

Create an ad hoc service class in the WLM policy.

Two measurements are required:

1. Case A: 50 users (the number of users depends on the complexity of the query and the system capacity) submitting trivial/small consumption queries over X minutes measurement interval.
2. Case B: 50 users submitting trivial/small consumption queries plus the killer query run for the same X minutes measurement interval.

Key metrics:

Throughput (number of queries completed) of the small query workload.

Success criteria:

The throughput of the small query workload in measurement in Case B should be >=95 percent of that of the throughput of the small query workload in measurement in Case A.

C.3.3 Prioritize a critical query over a small query workload

Objectives:

Ensure that a critical large consumption query is provided with as much resource as it demands while running concurrently with the small query workload.

Methodology:

Create a critical service class in the WLM policy.

Create a critical large consumption query.

Two measurements are required.

1. Case A: A critical large consumption query to run single-threaded to completion.
2. Case B: A critical large consumption query to run concurrently with the 50 small consumer query workload. The measurement ends upon completion of the critical large consumption query.

Key metrics:

Elapsed time of the critical consumption query

Throughput (number of queries completed) of the small query workload

Success criteria:

The elapsed time of the critical large query should be minimally impacted when running it with the small consumer query workload, as compared to running it single-threaded.

C.4 Scalability test

You might need to ensure performance scales appropriately with an increase in concurrent queries in terms of throughput and response time. You would want to understand the level of concurrency the system can support while contrasting that with the effect on response time.

The scalability test measures the throughput of multiple streams at various saturation points in contrast to C.3, “Throughput test” on page 241, which measures the throughput of a single unit of work. A *stream* may be defined as a batch of multiple queries executed serially in any sequence.

Objectives:

Determine the elapsed time to execute multiple streams at saturation points.

Determine the system utilization and performance.

Determine scalability and whether the result would be useful in sizing the customer’s system.

Methodology:

Keep data volumes and the configuration unchanged and use the result of the query performance test to determine the saturation point.

Once the saturation point determined, varying the number of streams measures the best throughput rate at each saturation point. Usually tests are run at three saturation points such as .5X saturation, saturation and 2X saturation.

Key metrics:

Throughput (number of queries completed) and the elapsed time from the start of the first query to the end of the last query.

System utilization and performance should be measured via SMF and via RMF records.

DB2 statistics for internal resources.

Success criteria:

Beyond points of system saturation, as the number of concurrent users grows, *flat throughput and linear response time within 85 percent* of perfect scalability.

Measurement:

Measure the elapsed time at each saturation point.

Only SMF records and RMF intervals should be captured.

C.5 Monthly update test

This test is the insert or append of one month of data (the duration depends on the customer environment, so it could be one week, one month or three months and so forth) into the existing tables. This test is designed to determine the insert rate. Since the monthly updates are usually run in a shrinking batch window, multiple jobs for multiple partitions should be run concurrently.

Objectives:

Determine the elapsed time to insert one month of data.

Methodology:

Using the DB2 LOAD utility or solution developer's load utility (BMC, Platinum and so on), append one month of data into the existing data warehouse database.

To achieve parallelism during this test, multiple utility jobs on multiple partitions should be run concurrently.

Key metrics:

Elapsed time from the start of the first monthly update job to the end of the last monthly update job.

System utilization and performance should be measured via SMF and via RMF records and DB2 performance records.

Success criteria:

Measure the best elapsed time to perform monthly updates.

The test results should be used evaluate what it would take in terms of hardware, software and update methodology to insert a much larger volume of data within the given batch window.

Measurement:

WLM may be used as necessary to control the concurrency level. Several iterations with various concurrency levels may be necessary to get the best elapsed time.

C.6 Data warehouse refresh concurrent with query activity

Objectives:

Ensure data warehouse refresh (refer to C.5, “Monthly update test” on page 245) can run concurrently with query activity in an efficient manner. What you need to prove is that WLM provides the resource management capabilities necessary to favor one or the other workload and at the same time allow the less favored workload the ability to consume any spare resource without impact to the favored workload.

Methodology:

Using the DB2 LOAD utility or a solution developer’s load utility, append one month of data into the existing data warehouse database while an X number of users are running the ad hoc query concurrently. X represents the number of concurrent users to bring the servers to saturation. You can find the X number in C.4, “Scalability test” on page 244.

Create a load service class in the WLM policy.

Four measurements are required:

- One month of load data. You can use the results of C.5, “Monthly update test” on page 245.
- Ad hoc query workload run. You can use the results of the C.3, “Throughput test” on page 241.
- Run load and ad hoc queries concurrently and favor the ad hoc query workload.
- Run load and ad hoc queries concurrently and favor the load.

Key metrics

Elapsed time of the load.

Throughput (number of queries completed) of the ad hoc query workload.

Success Criteria:

Ensure that queries and data warehouse refresh can be executed concurrently and that one workload can be favored over the other.

C.7 Data warehouse and data mart on the same OS/390 image

For customers who need to consolidate workload and manage system resources effectively, the coexistence of a data warehouse and a data mart should be considered as the solution.

Objectives:

Ensure that the data warehouse workload can run concurrently with the data mart workload in an efficient and manageable environment. Prove that WLM provides the resource management capabilities necessary for both workloads to meet the customer's goals.

Prove the efficient use of total system resources by allowing the data mart to utilize any additional resource that might be available when data warehouse activity is not using its total share of the system.

Prove the ability to cap the amount of processing resource obtainable by a given workload.

Methodology:

X users (enough to saturate the sysplex standalone) running the ad hoc query workload will be run concurrently with Y users (enough to saturate the sysplex standalone) running the data mart workload into the following scenarios:

Dedicate 75 percent of processing resource to the data warehouse and 25 percent of processing resource to the data mart.

Dedicate 50 percent of the processing resource to the data warehouse and 50 percent of the processing resource to the data mart.

Cut the number of ad hoc query users (X users) in half.

Cap the data mart processing resource at 25 percent and cut the number of ad hoc query users for the data warehouse in half.

You can use QMF or TPNS in order to simulate workload.

Key metrics:

Processor consumption by workload with an RMF workload activity report.

Throughput (that is, the number of queries completed) of the ad hoc and data mart workloads.

Success criteria:

Actual processor consumption of the data mart and the data warehouse is +/-10 percent of their expected processor consumption.

The data mart is capable of consuming available processing resource which is not utilized by the data warehouse.

Capability of capping the data mart.

C.8 Reorganization test

This test should be done after data warehouse refresh. You should run the reorganization utility for a partition and an index.

Objectives:

Determine the reorganization rates of a partition and an index.

Determine the data compression ratio of a partition when reorganized.

Determine the impact of updates on data, indexes and compression ratio.

Methodology:

Run the REORG utility on the tablespaces before the refresh test. Run the REORG utility on a selected partition and on a selected index after the refresh test (see C.5, “Monthly update test” on page 245) so a comparison ratio can be made with the LOAD utility.

Key metrics:

Elapsed time measured from the start of the reorganization to the end of the reorganization.

C.9 Unload test

This test is to unload partition data onto tape for further processing.

Objectives:

Determine the unload rate of a selected partition.

Methodology:

Run the unload program (DSNTIAUL or your own program) or utility on a selected partition to tape.

Key metrics:

Elapsed time measured from the start of the unload job to the end of unload job.

C.10 Backup/recovery test

Back up the medium size table to either tape or disk. Then run the recovery utility to restore that table.

This test item might not be focused on for testing. But for the production environment, you should build a backup/recovery strategy through this kind of test.

Objectives:

Determine the backup and recovery rates for the table.

Determine which products to use for table backup and recovery, such as the DB2 utility, solution developer's utilities (BMC, Platinum and so on), DFDSS volume backup, DFSMS concurrent copy and/or RVA SnapShot. For additional information on RVA SnapShot, refer to 4.2, "Backup" on page 95.

Methodology:

In order to determine the elapsed time to backup/recover a table, you can test with several products. If you encounter any bandwidth constraint due to use of tapes, you could extrapolate instead of backing up entire tables.

Key metrics:

Elapsed time measured from the start of the backup/recovery job to the end of the backup/recovery job.

Appendix D. Introduction to piping

We use a very simple job flow to explain what piping means.

The use of pipes enables an increase in parallelism and provides data-in-memory exploitation by overlapping jobs and eliminating the I/Os incurred in passing data sets between them. Additionally, if a tape data set is replaced by a pipe, tape mount can be eliminated.

Figure 91 shows the traditional step-by-step data flow for data-dependent steps.

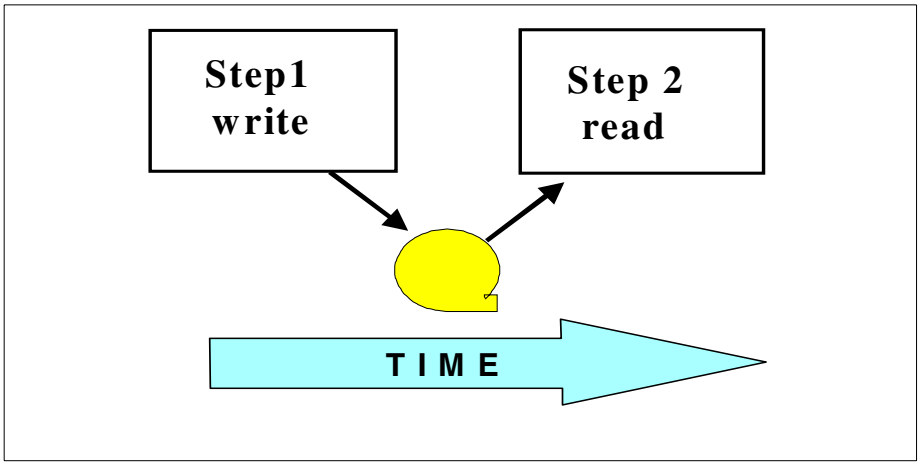


Figure 91. Two steps in a traditional batch job stream

In Figure 91, Step 1 writes data to tape (or DASD) sequentially. When all the records have been written, the data set is closed and Step 1 terminates. Then Step 2 reads the data from tape. Step 2 cannot start until Step 1 has closed the data set, because to start earlier could lead to Step 2 trying to read a record before Step 1 has written it. To prevent such problems, *MVS allocation does not allow a sequential data set to be read by one job while being updated by another.*

After Step 2 has finished processing all records, the data set is no longer required and can be deleted. In effect, the transfer of data from one step to the other is at a *data set level*.

Figure 92 on page 252 shows those same two steps through a pipe. Notice that the external storage device is eliminated and replaced by a storage buffer (a pipe) that holds a small portion of the data set's content.

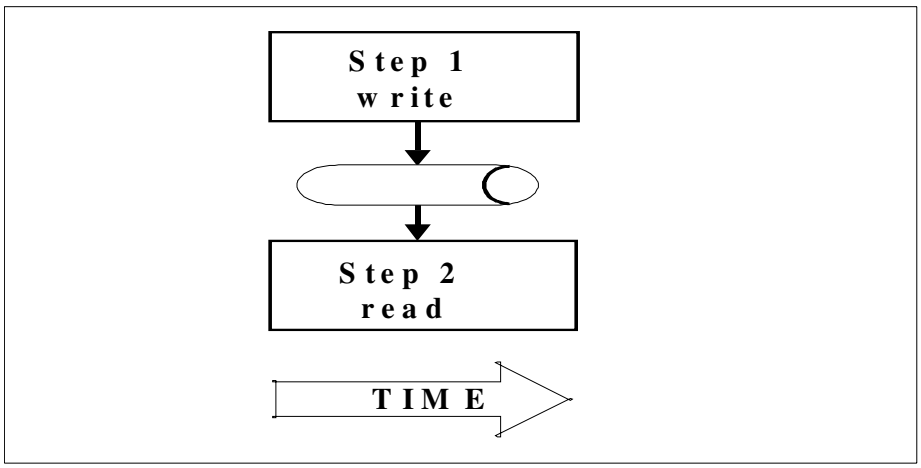


Figure 92. Two steps - using a pipe

A pipe can be thought of a FIFO queue: as each block of records is written by Step 1, it is placed on the queue. As each block is read by Step 2, it is removed from the queue in the order in which it was placed in the pipe.

In Figure 92, Step 1 writes blocks to the pipe and Step 2 reads data from the pipe. Step1 and Step 2 run concurrently.

Step 2 can obtain data from the pipe when Step 1 writes the first block. Output from Step 1 becomes immediately available as input to Step 2. The writer cannot get more than the depth of the pipe ahead of the reader. The writer and the reader process the piped data at the same speed; if either job is delayed, both jobs are delayed to same extent.

You can think of the data as “flowing” from Step 1 to Step 2 through a pipe. As the data is transferred through processor storage, the I/Os to read and write the data set are eliminated, potentially leading to elapsed time reductions for both Step 1 and Step 2. By reducing DASD contention or tape mounts, a pipe might speed up steps that do not use the pipe.

In a pipe, data flows always in one direction: *from a writer to a reader*. A pipe can exist in processor storage or in a coupling facility. A writer-pipe-reader set is known as a *pipeline*.

OS/390 SmartBatch is a software that enables the piping technology in the S/390 platform. The flowchart symbols in Figure 93 on page 253 are used to represent the *pipes* and *filters* of SmartBatch.

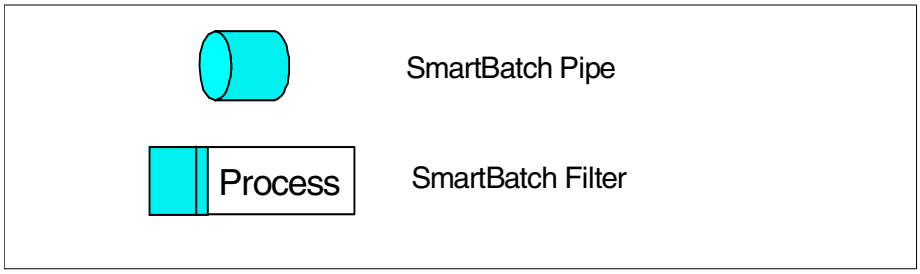


Figure 93. SmartBatch symbols

For more information about piping, refer to *System/390 MVS Parallel Sysplex Batch Performance,* SG24-2557.

Appendix E. Reasons for choosing S/390

This appendix summarizes reasons for choosing S/390 as a data warehouse platform, based on survey results provided by the International Technical Group (ITG).

Overall, 42 of the 58 S/390 user organizations surveyed had evaluated alternative platforms. The other 13 had not done so, typically because existing investments and skills, data synergies, or both determined choices at an early stage. Three organizations declined to provide details of the platforms reviewed.

Combined responses from these groups are summarized in Figure 94 on page 256.

Although respondents cited generally similar reasons for deciding to host BI applications on S/390 systems, some variations were apparent. This was particularly the case in the following areas:

E.1 Large systems

Among organizations which had implemented, or were in the process of implementing, BI solutions with databases of over 1 TB, S/390 strengths in availability (cited by 9 out of 12 organizations), data movement, query performance and workload management emerged as more critical concerns than in smaller installations.

There were, it was reported, unique challenges in combining very large databases with large mixed workloads and high availability — in all of these organizations, BI systems operated on a 24 x 7 basis, or were expected to do so in the future. One respondent noted that while other servers could meet some of these requirements, the S/390 platform running DB2 could handle *all* of them effectively.

E.2 Scalability

S/390 and DB2 scalability, and the ability to support future workload growth, were cited less frequently by organizations reporting databases of over 1 TB.

E.3 Leverage

The ability to leverage existing S/390 and DB2 skills and investments, and synergies with operational data generated by S/390 systems, were cited more frequently in smaller installations. In most cases, organizations were reluctant to add additional servers, databases and staff skills for applications that could be implemented in a less disruptive manner on existing S/390 systems.

Data synergies were cited as a particularly important factor in 13 organizations which reported significant overlap between operational and BI systems. In these cases, there was a clear perception that hosting both on S/390s resulted in reduced data complexity, faster implementation and lower cost than the use of multiple servers and databases.

“Other” responses included strategic vendor and systems integrator relationships (eight cases), better service and support for S/390 systems (six cases) and superior tools for data management, optimization, and other functions (four cases).

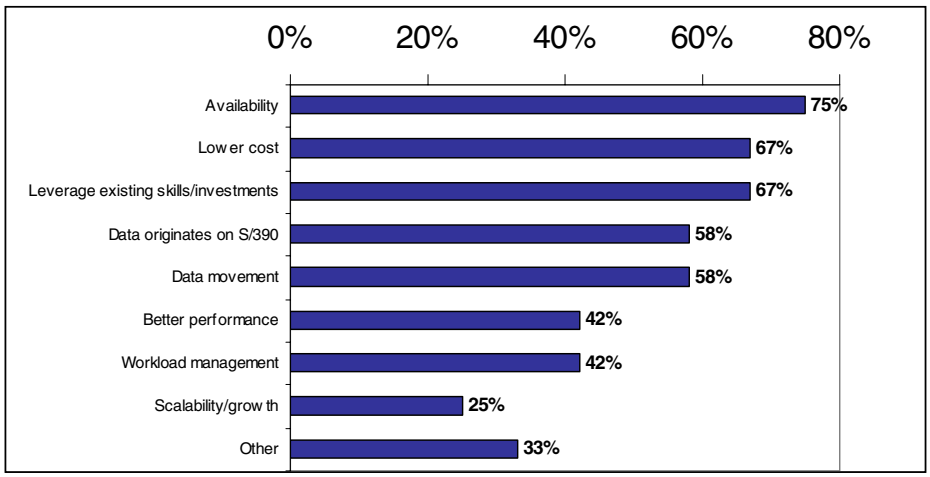


Figure 94. ITG market research: reasons for choosing S/390

Activities like ad hoc queries and even some planned queries can create situations where the complete processing capacity of the entire computer system is used at once. What type of mechanism is suitable to use in order to avoid system overload? How does one insure that the data mart workload can be executed with minimal impact to the data warehouse and vice versa? How many DB2-subsystems are needed?

The answer lies with Workload Manager (WLM). The user workload must be choked and balanced. Workload Manager is a product which has the capability to manage workloads by policies; for more information, see *OS/390 Workload Manager Implementation Exploitation,* SG24-5326 and *System/390 MVS/ESA Version 5 Workload Manager Performance,* SG24-4352.

The workload can also be balanced by using several DB2 subsystems. Our experience indicates that the data warehouse and the data mart should reside on separate subsystems. Together with WLM, this is a powerful combination that will help utilize the system resources; see 5.6.2, “Intelligent resource sharing between DM and DW” on page 125 for more information.

Appendix F. Special Notices

This publication is intended to help information technology (IT) managers and technical people undertaking the build of a VLDB data warehouse using DB2 for OS/390. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390 or DB2 for OS/390 Server. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 and DB2 for OS/390 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

DATABASE 2
DB2
DB2 OLAP Server
DFSMS
DFSORT
Domino Go Webserver
DRDA
eNetwork Communications Server
ESCON
Hiperspace
IBM
Net.Data
OS/390
Parallel Sysplex
QMF
QMF for Windows
RACF
RAMAC
SmartBatch
S/390
IBM ®

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix G. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

G.1 IBM Redbooks publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 267.

- *Data Warehousing with DB2 for OS/390*, SG24-2249
- *Using RVA and SnapShot for BI Applications with OS/390 and DB2,* SG24-5333
- *OS/390 Workload Manager Implementation and Exploitation,* SG24-5326
- *System/390 MVS/ESA Version 5 Workload Manager Performance,* SG24-4352
- *DB2 Server for OS/390 Recent Enhancements - Reference Guide*, SG24-5421
- *OS/390 MVS Parallel Sysplex Capacity Planning,* SG24-4680
- *DB2 for OS/390 V5 Performance Topics,* SG24-2213
- *DB2 for OS/390 Application Design Guidelines for High Performance,* SG24-2233
- *DB2 for OS/390 Data Compression*, SG24-5261
- *DB2 for MVS/ESA Version 4 Data Sharing Implementation,* SG24-4791
- *DB2 for MVS/ESA Version 4 Data Sharing Performance Topics,* SG24-4611
- *DB2 on the MVS Platform: Data Sharing Recovery,* SG24-2218
- *Accessing DB2 for OS/390 data from the WWW,* SG24-5273
- *Managing Multidimensional Data Marts Visual Warehouse DB2 OLAP Server*, SG24-5270
- *Data Modeling Techniques for Data Warehousing*, SG24-2238
- *S/390 MVS Parallel Sysplex Batch Performance,* SG24-2557
- *IBM 2216 Network Utility Host Channel Connections,* SG24-5303
- *A Comprehensive Guide to Virtual Private Networks, Volume II: IBM Nways Router Solutions,* SG24-5234

- *IBM Nways 2216 Multiaccess Connector Description and Configuration Scenarios - Volume I*, SG24-4957
- *IBM 2210 Nways Multiprotocol Router and BM 2216 Nways Multiaccess Connector Description and Configuration Scenarios - Volume II,* SG24-4956

G.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

G.3 Other resources

These publications are also relevant as further information sources:

- *OS/390 MVS Planning: Workload Management,* GC28-1761
- *IBM SmartBatch for OS/390 Overview,* GC28-1627
- *DB2 for OS/390 V6 Release Planning Guide,* SC26-9013
- *DB2 for OS/390 V6 Release Utilities Guide and Reference*, SC26-9015
- *DB2 for OS/390 V6 What's New?,* GC26-9017
- *DB2 for OS/390 V5 Administration Guide,* SC26-8957
- *DB2 for OS/390 V5 Application Programming and SQL Guide,* SC26-8958
- *DB2 for OS/390 V5 Call Level Interface Guide and Reference,* SC26-8959
- *DB2 for OS/390 V5 Command Reference,* SC26-8960

- *DB2 for OS/390 V5 Data Sharing: Planning and Administration,* SC26-8961
- *DB2 for OS/390 V5 Installation Guide,* GC26-8970
- *DB2 for OS/390 V5 Messages and Codes,* GC26-8979
- *DB2 for OS/390 V5 SQL Reference,* SC26-8966
- *DB2 for OS/390 V5 Reference for Remote DRDA Requesters and Servers,* SC26-8964
- *DB2 for OS/390 V5 Utility Guide and Reference,* SC26-8967
- *DB2 for OS/390 V5 Release Guide,* SC26-8965
- *DB2 for OS/390 V5 What's New?,* GC26-8971
- *Installing and Managing QMF for Windows,* GC26-9583
- *Domino Go Webserver Release 5.0 for OS/390 Web Master's Guide,* SC31-8691
- *OS/390 MVS Planning: Workload Management* , GC28-1761
- *Nways MAS Using and Configuring Features Version 3.4,* SC30-3993
- Yevich, Richard, "Why to Warehouse on the Mainframe"*, DB2 Magazine,* Spring 1998

G.4 Referenced Web sites

- META Group, "1999 Data Warehouse Marketing Trends/Opportunities", 1999, go to the META Group Website and visit “Data Warehouse and ERM”.

  `http://www.metagroup.com/`

- International Technology Group,"Enterprise Solutions for Business Intelligence", Spring 1999

  `http://www.itg-inc.com/`

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

BI	Business Intelligence
BP	Bufferpool
CAE	Client Application Enabler
CCSID	Coded Character Set ID
CFRM	Coupling Facility Resource Manager
CGI	Common Gateway Interface
C/S	Client/Server
DA	Data Analyst
DBA	Database Administrator
DBCS	Double Byte Character Set
DB2PM	DB2 Performance Monitoring
DDF	Distributed Data Facility
DDL	Data Definition Language
DFSMS ACS	DFSMS Automated Class Selection
DM	Data Mart
DML	Data Manipulation Language
DR	Disaster Recovery
ERD	Entity Relationship Diagram
EDW	Enterprise Data Warehouse
ETL	Extract/Transformation/Load
GBP	Group Buffer Pool
GWAPI	Go Web server Application Programming Interface
HTML	HyperText Markup Language
ITSO	International Technical Support Organization
JDBC	Java Database Connectivity
LAN	Local Area Network
LOB	Large Objects
LPAR	Logically Partitioned mode
MOLAP	Multi-Dimensional On-Line Analytical Processing
MPP	Massive Parallel Processing
ODBC	Open Database Connectivity
ODS	Operational Data Systems
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
QEP	Query Execution Plan
RACF	Resource Access Control Facility
RLF	Resource Limit Facility
ROLAP	Relational On-Line Analytical Processing
RVA	RAMAC Virtual Array
SLA	Service Level Agreement
SMS	System Managed Storage
SNMP	Simple Network Management Protocol
TPNS	Tele-Processing Network Simulator
UCB	IOS Unit Control Block (OS/390 Control Block)
VLDB	Very Large Database
WLM	Workload Manager

Index

Numerics

A

B

C

D

IBM Redbooks evaluation

Building VLDB for BI Applications on OS/390: Case Study Experiences
SG24-5609-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other Redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

SG24-5609-00
Printed in the U.S.A.

Building VLDB for BI Applications on OS/390: Case Study Experiences

SG24-5609-00