

**Object-Relational Database Technology Concept:
A Comparison of Oracle 8 with Other
Database Products**

By

Preedarat Sukanwattanchai

A PROJECT

**Presented to the Faculty of Graduate School of
Computer and Engineering Management**

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER AND ENGINEERING MANAGEMENT

ASSUMPTION UNIVERSITY

December, 1998

MS (CEM)

ABAC
GRADUATE SCHOOL LIBRARY

OBJECT-RELATIONAL DATABASE TECHNOLOGY CONCEPT:
A COMPARISON OF ORACLE8 WITH OTHER DATABASE PRODUCTS

by

PREEDARAT SUKANWATTANACHAI

A PROJECT

Presented to the Faculty of the Graduate School of
Computer and Engineering Management

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER AND ENGINEERING MANAGEMENT

ASSUMPTION UNIVERSITY

December 1998

Project Title : Object-Relational Database Technology Concept: A
Comparison of Oracle 8 with Other Database Products

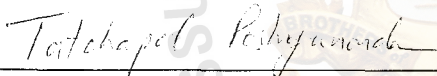
Name : Ms.Preedarat Sukanwattanachai

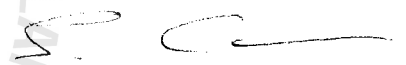
Project Advisor : Dr.Tatchapol Poshyanonda

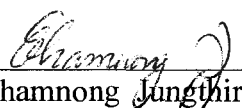
Academic Year : 1998


The Graduate School of Assumption University has approved this final report of the three-credit course, CE 6998 Project, submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer and Engineering Management.


Approval Committee:



(Dr.Tatchapol Poshyanonda)
Advisor


(Prof.Dr.Srisakdi Charmonman)
Chairman


(Dr.Chamnong Jungthirapanich)
Dean and Co-advisor


(Assist.Prof.Dr.Boonmark Sirinaovakul)
Member


(Assoc.Prof.Dr.Prapon Phasukyud)
Member


(Assoc.Prof.Somchai Thayarnyong)
MUA Representative

December 1998

ABSTRACT

This project is a descriptive survey describes the concepts and characteristics of Object-Relational technology which drove Oracle8 database to industry leadership. As a key component of Network Computing Architecture, Oracle8's advanced features and functionality, unique architecture for n-tiered computing coupled with the company strength positions positively against any competitor. This paper intends to take you through the Oracle8 features and compare them to the offerings and plans of the most frequent competitors in the database, systems management, and cartridge areas.

With the best plays offense namely: Java and Application Development which Oracle8's Java-in-the-database not only provides a powerful, platform-independent development environment, but also provides a highly scalable execution platform for web-based applications. Oracle's Internet strategy is to put the software on a couple application servers which dramatically reduces overhead maintenance costs. Web Information Management, Oracle8's extensible framework and wide variety of cartridges provide robust support for all multimedia data and web information management requirements. Oracle enables customers to store and retrieve files into the database server. The additional flexibility of WebDB makes it easy for applications to be built and deployed. Data can be stored directly in the database, it can be accessed by a URL or it can even be loaded from the file system. Enterprise Application Integration. Leveraging Java, Operational Simplicity which Oracle8 provides ease of use, superior manageability coupled with interoperability and choice, Operational Simplicity, Enterprise Java Beans, sophisticated business messaging, events, data replication and gateways, Oracle8 is a platform which

enables organizations to integrate and extend their legacy applications in today's leading enterprise applications. Data Warehousing which Oracle8's new set of partitioning methods coupled with powerful materialized views, summary provides unsurpassed supports for all aspects of data warehousing. High Availability/Scalability which Oracle8 includes many enhancements for Oracle Parallel Server, the industry leading optimizer, improved archiving, better recovery, and many OLTP enhancements, all which provide highly scalable environments with better resource usage and many different proven availability solutions.

This comparison will not be dictated based on features and functions. Many of sophisticated corporations plan to re-engineer their business processes to enable them to compete in fundamentally new ways by making their decisions subjected to a pure technology discussion and may not really care. It will be won based on how decision makers perceive the relative benefits of the products and the value which they feel the products provide back to the business and for their own careers.

ACKNOWLEDGEMENTS

I am indebted to the following people and organizations. Without them, this project would not have been possible.

I wish to express sincere gratitude to my advisor, Dr. Tatchapol Poshyanonda. His patient assistance, guidance, and constant encouragement has led me to the project completion. I would like to express appreciation to my Advisory Committee members: Prof. Dr. Srisakdi Chamonman, Dr. Boonmark Sirinaovakul, Dr. Prapon Pasukyud, Dr. Chamnong Jungthirapanich and Assoc. Prof. Somchai Thayarnyong for their constructive comments and advice throughout the project.

I would like to thank Dr. Kanonkluk Vanapipat of Oracle Systems (Thailand) Company Limited for her help in further information and recommendation.

Special appreciation is due to my family for their fervent and continuous encouragement. Above all, I am forever grateful to my parents whose willingness to invest in my future has enabled me to achieve my educational goal.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
LIST OF ILLUSTRATIONS	xii
LIST OF TABLES	xiii
CHAPTER	
I. INTRODUCTION	1
A. An Overview	1
B. Position of Objective and Strategy	1
C. Oracle8 – Heard on the Street	2
II. TERMINOLOGY AND FUNCTIONALITY	5
A. What is an Object-Relational Databases	5
B. Benefits of Object Technology	5
C. Object-Relational Features	6
D. Motivation for Oracle8’s Object Functionality	6
1. To Manage Complexity in Their Business Process	6
2. To Respond Quickly to Changes	7
3. To Make Better Use of Information	7
4. To Leverage Their Investments in People and Information Technology	8

	Page
E. Requirements on Database Infrastructure	8
1. Better Modeling of Business Processes	9
2. Integrate Multimedia Data in a Single Store	9
3. Exploit the Web and Component-Based Architectures	10
4. Evolve Existing Application with New Technology	10
F. Defining Oracle's Approach	11
1. Evolutionary	11
2. Integrated	13
3. Pragmatic	14
4. Open	14
5. Comprehensive	15
III. SOLUTION AND COMPARISON	17
A. Oracle Solution	17
1. Partitioned Workload	17
2. Object Types	18
3. Object Views	19
4. Large Objects	20
5. Multimedia Cartridges	21
6. Application Development Interfaces and Tools	22
7. Deployment Considerations—Performance and Scalability	23
8. Oracle8 Addresses Customer Needs	24
9. Change Management	25

	Page
B. Oracle8 Solution Versus Competitors' Solutions	25
1. Oracle8 Solution Versus Informix Universal Sever	25
a. Single Product Versus Many Products	27
b. Mainstream Applications Versus Niche Applications	27
c. Complete Application Development Architecture and Tools Versus Low Level Interfaces, No Tools	28
d. Tight Integration of Object/Relational Data Versus No Integration	28
e. Proven Scalable Deployment Platform Versus Untested, Experimental Deployment Platform	29
f. Logical Identity Versus Physical Identity	29
2. Oracle's Strategy Versus Microsoft	29
a. Single Database Versus Federated Databases	30
b. Scalable Database and Application Server Versus Application Server Only	30
c. Multiple Platform Versus Single Platform	31
d. Open Standard Versus Proprietary Standards	31
3. Oracle8 ORDBMS Versus Object-Oriented Databases	31
a. Robust vs. Fragile, Non-Scalable Deployment Platform	35
b. Simple Queries Versus Lack of Query Ability	35
c. Tight Object-Relational Integration Versus No Integration	35

	Page
d. Complete Application Development Solution Versus No Infrastructure	36
e. Operationally Viable Solution Versus Operationally Non-Viable Solution	36
IV. RISK ANALYSIS	46
A. Company Vision Risk	46
B. Platform Risk	46
C. Technology Risk	46
D. Partner Risk	47
E. Product Risk	47
F. Scalability Risk	47
V. FUTURE DIRECTION AND CONCLUSION	48
A. IT Challenges	48
B. Oracle8 and Database Extensibility	49
C. Oracle8 and Network Computing Architecture	51
D. Enterprise-Class Transaction Processing	58
E. Superior Scalability for Transaction Processing	58
F. High Performance for Transaction Processing	59
G. Object-Views	59
1. Advantages of Object Views	60

	Page
2. Defining Object Views	61
3. Using Object Views	63
4. Updating Object Views	64
H. User-Defined Datatypes (Object-Option)	65
I. Complex Data Models	65
1. An Example	66
2. Multimedia Datatypes	67
3. User-Defined Datatypes	68
4. Object Types	68
a. Purchase Order Example	69
b. Methods	70
c. Object Type Constructor Methods	71
d. Comparison Methods	72
5. Object Tables	73
6. Row Objects and Column Objects	75
7. REFs	75
a. Scoped REFs	76
b. Dangling REFs	76
c. Dereferencing REFs	76
d. Obtaining REFs	77
8. Collection Type	78
9. VARRAYs	78
10. Nested Tables	79

	Page
J. Application Interfaces	81
1. SQL	81
2. PL/SQL	81
3. Pro*C/C++	82
4. OCI	83
5. OTT	84
K. Conclusion	84
BIBLIOGRAPHY	



LIST OF ILLUSTRATIONS

Figure	Page
2.1. Objects simplify Application Development, Especially multimedia applications	10
3.1. Partitioned Workload Solution	17
5.1. Oracle8 and NCA define how software components communicate with each other and database. The database provides a safe, secure mechanism for invoking external code.	54



LIST OF TABLES

Table	Page
3.1. Summary of Comparison in Java Initiatives	38
3.2. Summary of Comparison in Web Enablement	39
3.3. Summary of Comparison in Enterprise Application Integration	40
3.4. Summary of Comparison in Data Warehousing	41
3.5. Summary of Comparison in Database Administration And Operational Simplicity	42
3.6. Summary of Comparison in Availability and Scalability Solution	44



I. INTRODUCTION

A. An Overview

The Oracle8 object-relational database provides the most complete solution to address object functionality needs of the mainstream business market. It combines the management of complexity in developing applications for complicated business processes, the management and manipulation of any data in a single data store, and the industry's most robust, scalable, and high performance deployment platform. In doing so, Oracle8 defines the next generation of database technology. This has aroused considerable excitement within the database community about the object functionality of the Oracle8 object-relational database.

This paper is a descriptive survey which clearly defines the benefits provided by the object features and concepts, more specifically in Oracle8 and explains how Oracle's evolutionary strategy to deliver object technology to the mainstream is superior to the strategies of other vendors.

B. Position of Objective and Strategy

This paper clarifies the position of the Oracle8 object strategy and its message in five important steps:

- Business needs: Define the kinds of business needs that motivate the requirement to store and access objects in a database.

- Oracle's strategy: Defines the strategic principles Oracle adopted in addressing these business needs.
- The Oracle8 solution: Provides a high level technical overview of the Oracle8 features and discusses how each business needs these features.
- The Oracle8 solution compared to competitor solution: Compares the approach Oracle adopted with that taken by other vendors and demonstrates the clear superiority of the Oracle approach.
- Future directions: Discuss how Oracle8 reinforces and fits into Oracle's overall strategic direction within the context of Network Computing Architecture (NCA).

The Oracle Objects Option is designed to leverage the new capabilities of Oracle8. Oracle8, the next-generation universal data server, delivers tremendous advances over the characteristics which drove the Oracle universal data server to industry leadership. As a key component of Network Computing Architecture, Oracle8 is designed to support all of a customer's users and data, providing a high-performance and cost-effective system for running business applications. (Wiseth, 1998)

C. Oracle8 – Heard on the Street

These are the examples from various point of views regarding to Oracle8's features and functionalities:

- “Oracle previewed a new "Internet database" called Oracle8, with

groundbreaking features that include the ability to drag and drop desktop and server files directly into a central database, as well as to search this central storage place via browser over the World Wide Web.”

- Newsbytes, September 16, 1998

http://firstnews.us.oracle.com/1stbin/read_story/FIRST/980916/0/11/86/1

- “Oracle8 (for "Internet") database will be a boon to database administrators and Java programmers..... The new file system should make it easier for administrators to use the database to manage all kinds of files through the use of new data types”.

- Network World, September 16, 1998

http://firstnews.us.oracle.com/1stbin/read_story/FIRST/980916/0/11/86/7

- "Oracle8 completes the distributed applications story for Oracle because business logic can now be developed in a single, powerful and no proprietary language [Java] and deployed at the client, the application server or the database,"

- Elton Barrendse, CEO of Quintessence Systems, Oracle turns up 'Net database adds Java support'

http://firstnews.us.oracle.com/1stbin/read_story/FIRST/980916/0/11/86/7

- “The file system, iFS, has a Web-based user interface that makes data look the same whether it's accessed through a browser, E-mail, the desktop, or other means.”

- Oracle Unveils Internet-Based Database, Jeff Sweat – InformationWeek

- “This browser-based development tool does a great job of providing simple to-use tools to manipulate data, build data-driven Web applications, and create entire Web sites. Administrators will also appreciate the built-in management and security features.

<http://www.infoworld.com/scoop/sc?980915rv1> - InfoWorld Reviews

- “The Redwood Shores, Calif., company's goal of distributed Internet computing gets a kick with Oracle8.”

- Mark Hammond, PC Week Online, September 15, 1998

<http://www.zdnet.com/pcweek/news/0914/15aorac.html>

- "The iFS helps make it easier to unify content".

- Frank Gillett, Forrester Research Inc.

<http://www.zdnet.com/pcweek/news/0914/15aorac.html>

- “Oracle's got a good point. Windows is a horrible collection of components shielded beneath a debatably usable interface..... [while] iFS has the makings of an ingenious device.”

- John Taschek, PC Week Online, September 15, 1998

<http://www.zdnet.com/pcweek/opinion/0914/15mtash.html>

- “Analysts said Oracle's Business On-line plan is a good one. ‘This is an enormous opportunity and nobody has really exploited it yet.’” - Merv Adrian, Giga Information Group”

<http://www.techweb.com/wire/story/TWB19980826S0002>

II. TERMINOLOGY AND FUNCTIONALITY

A. What is an Object-Relational Database?

Object-Relational Database is a paradigm for complex application with the special features as followings:

- Allows creation of user-defined data types.
- Supports multimedia and large data objects.
- Includes compatibility with standard SQL object standards.
- Contains database server features of high quality.

B. Benefits of Object Technology

Now that we are familiar with the Object-Relational Database terminology. The benefits that provide back to the business as survey are the followings:

- Easy maintenance of applications which saves money on maintenance.
- Faster product development which given company competitive advantage.
- Higher quality of code at a lower cost which lead to more satisfied customers.
- Reusability of code which allows multiple programmers to share a code repository, which saves time and money.

C. Object-Relational Features

Oracle8 provides an object-relational paradigm for complex applications. This improved way of defining data structures allows developers to directly define their business documents, such as purchase orders, inventory items, and data warehouse information, within Oracle8. This allows developers of mainstream commercial applications to better manage their business documents.

D. Motivation for Oracle Object Functionality

Oracle decided to add object functionality to Oracle8 because of the changing business environment in which its customers compete and because of the needs placed on application development and deployment by these changes. Today's networked economy requires corporations to be very flexible and agile formulating and executing their business strategies. Corporations face far greater competitive pressures than ever before. The barriers to entry into existing and new markets are lower and product cycles are considerably shorter, constantly rendering existing products obsolete. Traditional business delivery system and value chains are fundamentally altered as entire industries converge. New entrants exploit this period of change and convergence by attempting to redefine value propositions to customers. As a result, corporations increasingly find the traditional sources of competitive advantage ineffective and transient, compelling them to find new ways to compete. To succeed, corporations face four fundamental challenges:

1. To Manage Complexity in Their Business Process

Corporations find that their traditional business processes are far too complex and inflexible to let them reach their customers effectively. Rather than be circumscribed by them, successful companies are attempting to manage the complexity in their business processes by continually re-engineering until these processes were structured. Now, many sophisticated corporations plan to re-engineer their business processes to enable them to compete in fundamentally new ways.

2. To Respond Quickly to Changes

Corporations need to manage complexity in developing applications, manage and manipulate any data in a single data store, and deploy their applications on a robust, scalable database platform. The convergence of many previously disparate markets and pressure from new market entrants compels corporations to discover new ways they can better identify new market opportunities and be more agile in moving quickly to capture them. Corporations find these qualities are essential to defend their existing markets from attack. Time-to-market pressures have rapidly increased as product cycles have been reduced by orders of magnitude.

3. To Make Better Use of Information

The Internet is accelerating the need to combine multimedia data with other kinds of information. The explosion of the Internet and the World Wide Web provides corporations new ways to disintermediate multiple layers in their distribution systems to directly reach their customers and suppliers. Direct customer contact requires corporations to transform the data in their business information systems into information

which customers can use to make purchase decisions. To perform this transition, corporations need to combine rich, multimedia information (such as pictures and sound clips) with financial and other technical information on their products and services. The Internet provides a standards-based environment in which direct information exchange can occur easily. By embracing the Web and the opportunities it offers, leading edge corporations have begun to build webs of alliances with their most important customers and suppliers to shelter them from aggressive competitors.

4. To Leverage Their Investments in People and Information Technology

Corporations' traditional sources of competitive advantage, such as huge capital bases and fixed assets such as manufacturing facilities, are increasingly less important in the new economy. New entrants can raise the capital they need in the financial markets and can out-source many of their business operations at lower costs than incurred by their entrenched competitors. Corporations find they need to leverage their investments in two critical areas: their people and their information infrastructure. Corporations have already invested many billions of dollars in information technology: purchasing products and training people how to use them. It is necessary that they continue to leverage these investments to compete in the new economy.

E. Requirements on Database Infrastructure

Corporations participating in this competitive landscape are finding new ways to leverage information technology to create a sustainable competitive advantage. Their needs translate into four fundamental requirements of their database infrastructure:

1. Better Modeling of Business Processes

Corporations constantly search for ways to manage the complexity of modeling their complicated and constantly changing business processes. They must provide application developers a single, consistent model for the objects used to represent their business processes. An example is the set of the business objects used in their application and their data store. Corporations need to be able to treat these objects as complete entities when accessed from the database or manipulated in the application. This requires the database to:

- Let customers more easily represent business objects as native types in the database.
- Provide simple ways to query over these objects.
- Provide a mechanism for the database to map these stored business objects to client-side application constructs and marshal the data appropriately.
- Provide a client-side environment in which to use these client-side mapping.

2. Integrate Multimedia Data in a Single Store

The Internet provides an infrastructure over which rich, multimedia information can be easily exchanged. As corporations exploit the Web, they are synthesizing objects by combining rich, multimedia information with their business information. To facilitate such applications, the database must provide a single store where customers can store and query on all the various kinds of data they use to compose their business

objects. These objects include structured data and unstructured rich multimedia data, such as text, spatial and image data.

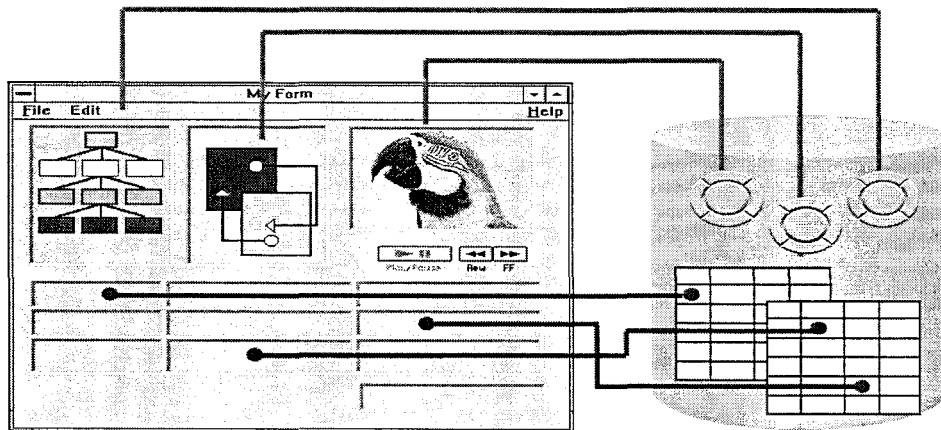


Figure 2.1. Objects simplify Application Development,
Especially Multimedia Applications

3. Exploit the Web and Component-Based Architectures

Corporate intranets and the public Internet are rapidly becoming the deployment environments for most mission-critical applications. Spurred by the proliferation of the Internet and the growing adoption of object-oriented middleware products (such as Object Request Brokers), corporations are now beginning to build applications by assembling prefabricated components. As they develop this new breed of network-centric enterprise applications, they are increasingly pushing the applications complexity into the network. This lets clients look to be extremely thin, simple, and cheap to purchase and maintain. Databases remain the central information store for these network-centric applications. In addition to storing information, they must also provide

an application development architecture that facilitates component-based development and Web-bases deployment.

4. Evolve Existing Application with New Technology

To protect and leverage customers' existing investments in information technology, their existing relational databases must be enriched with object technology. In evolving their relational databases, customers must be provided with a smooth evolutionary path that meets three requirements:

- It must allow customers to exploit their knowledge of existing products while gradually introducing new products to meet new needs.
- It must provide them with the predictable performance, scalability, reliability, and manageability they need for their mission-critical applications.
- It must conform to open industry standards.

F. Defining Oracle's Approach

To meet the needs of the business community, Oracle chose to evolve the industry-leading Oracle7 universal data server into the Oracle8 object-relational database. Oracle's approach was founded on five basic pillars: evolutionary, integrated, pragmatic, open, and comprehensive.

1. Evolutionary

Oracle has taken an evolutionary approach to bringing new technology to the mainstream market, building technological advances on its existing products and prioritizing its solutions to address the most important needs of its customers. Oracle facilitated client/server computing by introducing stored procedures with Oracle7 Release 7.0. It offered customers the use of extended datatypes with Oracle7 Release 7.3. With Oracle8, Oracle is bringing the next generation of relational database technology to the mainstream market by providing users the ability to define and store business objects in the database and by extending support for rich, multimedia datatypes. Oracle8 integrates object data and relational data, allowing customers to treat object data relationally. For example, Oracle allows users to access objects using SQL and to compose objects from relational data using object views. New features designed for use with objects, such as fully updatable views, can be used with relational tables. Oracle8 does not require users to migrate any of their existing Oracle7 applications. Therefore, Oracle8 provides customers with a very smooth evolutionary path that allows object types and relational tables to coexist in the same database and existing Oracle7 applications and tools to coexist with Oracle8 applications. It leverages customers' existing investments in four important ways:

- Integrates relational data with objects in the same data store.
- Exploits proven Oracle technology to provide customers with a robust scalable deployment platform.
- Allows customers to use their huge base of existing applications without needing to migrate them.
- Leverages customers' knowledge of existing application development products and tools.

In choosing this evolutionary approach, Oracle avoided strategies such as adding persistence to object-oriented programming languages. Rather, Oracle leveraged its expertise in data storage and manipulation to provide an easy way to store business objects in the database and access them through a number of client-side language interfaces (including OO programming languages).

2. Integrated

Oracle has chosen to offer a tightly integrated object solution with Oracle8. Object technology permeates the Oracle8 server and is not offered as a thin veneer on top of an existing relational database. Oracle8 is an integrated product in three important ways:

- It has a single architecture for both objects and relational data.
- It is a single, integrate product, integrating object functionality with advances for data warehousing and OLTP. Customers can purchase and invest in learning a single product rather than separate products for each of their needs.
- It has a single, consistent, and interchangeable interface to access both relational data and objects.

Choosing to offer an integrated solution, Oracle avoided short-cut solutions such as:

- Wrapping its existing relational database in an object wrapper.

- Delivering a persistent object store.
- Providing a middle-tier object server gateway to a relational database.
- Attempting to purchase and integrate a third-party object database with its existing relational database.

The resulting Oracle8 solution offers customers a much more tightly integrated solution and a complete application development architecture.

3. Pragmatic

Oracle has taken a very pragmatic approach to delivering object functionality in the server. It has chosen to deliver this functionality in phases, addressing the most important and clearly understood customer needs first, while giving customers sufficient time to absorb and effectively use the advances in the server. As a result, with Oracle8, Oracle lets users store business objects and multimedia data in the server. In subsequent releases of Oracle8, Oracle will carefully phase in features such as inheritance, polymorphism, and extensibility interfaces. Moreover, Oracle ensured that the Oracle8 solution is operationally viable and can be used as a deployment platform for mission-critical applications in a production environment.

4. Open

Oracle8 leverages your investment in existing tools and applications by providing open, data access through variety of standard, data-access methods. These include ODBC, Oracle Object for OLE, JDBC, and native Oracle drivers. By offering

open connectivity with open deployment, Oracle8 fits in any environment regardless of existing corporate standards. Oracle Objects for OLE provides a custom control (OCX or ActiveX) combined with an OLE in-process server that lets you plug native Oracle8 functionality into your Windows applications.

Oracle continues its commitment to open standards with Oracle8 in a number of different areas:

- SQL3: SQL3 is not yet a standard. Oracle is taking a leading role in moving the SQL3 specification forward and intends to comply with the SQL3 specification when it is officially accepted as a standard.
- JDBC, JSQL: Oracle is providing a comprehensive suite of Java products with Oracle8. It provides two JDBC drivers both of which comply with the standard JDBC specification from Javasoft at both the .binary and .class levels. Further, Oracle is working closely with a number of vendors including IBM, Tandem, and Javasoft to define JSQL, an embedded SQL in Java specification.
- CORBA: Oracle's future direction is based on strong support for the open, cross-platform, CORBA standard in its database, web application server, and tools products.

As a further commitment to providing the most open database platform for application development, Oracle provides a number of interfaces with which its partners can deliver solutions. A large group of industry-leading tools vendors, value-added resellers, and solution providers are working closely with Oracle to offer a variety of solutions with Oracle8.

5. Comprehensive

There are two important aspects to the comprehensiveness of the Oracle8 solution:

- In bringing object technology to the mainstream market, Oracle has adopted comprehensive company-wide effort integrating the features in the Oracle8 server with application development products and tools. It specifically designed features in Oracle8, for example Object Type Translator and the object cache, to offer the best application development infrastructure in the industry.
- Oracle8 will provide the widest choice in the industry for application development: existing Oracle8 application development products, including PL/SQL, Precompilers, Developer2000, all support object functionality. New products, such as C++, JDBC, JSQL, Object Designer, and Sedona will further address new market needs.
- Oracle8 is functionally rich, containing a number of important technological advances, such as object types, object IDs, object views, and object cache, and Java in the database. These bring object technology to the mainstream market and are discussed in more detail in the next section.

III. SOLUTION AND COMPARISON

A. Oracle Solution

Having identified the kinds of business needs that Oracle aimed to solve and the strategy it adopted to solve them, this section follows with a description of the fundamental technical components of Oracle8's object functionality.

1. Partitioned Workload

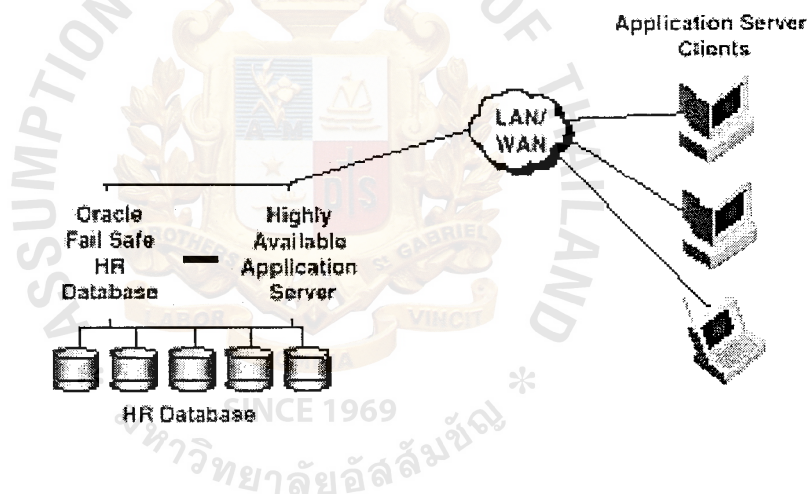


Figure 3.1. Partitioned Workload Solution

The partitioned workload solution, shown in Figure 3.1, is a variant of an active/active solution. The cluster provides high availability, for example, to both an application server and a data server. Node 1 serves an application server (such as SAP R/3, Baan, PeopleSoft, or Oracle Applications), while Node 2 serves an Oracle Fail Safe database. Each node backs up the other in the event of failure. Generally, application users connect only to the application server and never directly to the data server. The

data server. If the private heartbeat network connection between the nodes has high bandwidth, then the application server can further optimize database transaction processing by using the private network, rather than the public network, to communicate with the database. Because the bandwidth requirement for internode heartbeat communication is small, the application server can take advantage of what is effectively a dedicated network link to the database.

2. Object Types

Oracle has extended SQL (DDL, and DML) to allow users to define their own types (that represent their business objects), store them as base or native types within the database either within a column of a table or as tables themselves, and query, insert, and update them. They can contain one business object inside of another, point from one business object to another (using a pointer called a REF) and access and manipulate collections or sets of these objects, using structures called VARRAYS and Nested Tables. Users can define operations on business objects as methods of the objects. Methods can be implemented as PL/SQL, stored procedures. Objects have globally unique identifiers called Object IDs that capture references between objects.

Oracle8 allows a user to treat object data relationally. For example, users can query on object data using SQL, in the same way they access relational data. Users can access an object using SQL DML for the query, the object types attributes and methods with extended path expressions (for example, object.attribute), and perform explicit joins between objects in tables using SQL. In addition, Oracle8 lets users perform implicit joins between objects by traversing or navigating the REF from one object to

the other. Object types are indexable using MAP or ORDER methods to convert them to scalar values, which can then be indexed.

Oracle8's object constructs have a close correspondence with the relational constructs that Oracle's customers are familiar with. For example, a REF is very similar to a foreign key, methods are really PL/SQL stored procedures and perhaps most importantly, the security model that operates on object types is exactly the same that Oracle defined for relational tables. Of greatest significance, Oracle8 provides the same transactional semantics and behavior on objects as it does on relational data.

3. Object Views

Object Views allow customers to synthesize a business object from data that continues to be stored either in relational tables or as object types. Object Views provide an extremely easy way for customers to:

- Define objects they can use in their applications without migrating any of their data from relational tables.
- Combine objects developed for one application in different ways to use with other applications. Objects in an object view have much of the functionality as those in object tables. They can have methods, be part of collections, point to one another, have object identity, and be accessed from SQL or via pointer traversal. Further, Oracle has extended the view mechanism to provide fully updatable views using special types of triggers called INSTEAD OF triggers. Oracle is the only vendor that provides an object view mechanism.

4. Large Objects

Application developers who use object types and collections to store their structured business object data in Oracle8 also want to store rich, multimedia data as text, graphic images, still video clips, full motion video, and sound waveforms in the Oracle database. The Large Object (LOB) data types store blocks of such rich, and unstructured multimedia data in the Oracle database. Oracle8 supports two types of LOB: those that are stored in the database either in-line or in a separate tablespace (such as BLOB, CLOB, and NCLOB) and those types of LOB that are stored as operating system files (such as BFILEs). Oracle8's support for LOB data types improves upon the support for LONG and LONG RAWs in Oracle7 in several ways:

- **LOB Capacity:** With Oracle8, LOBs can store up to 4GB of data. This doubles the 2GB of data that LONG and LONG RAW data types could store.
- **Number of LOB columns per table:** An Oracle8 table can have multiple LOB columns. Each of these LOB columns in the same table can be of a different type. Oracle7 Release 7.3 tables are limited to a single LONG or LONG RAW column.
- **Random piece-wise access:** LOBS support random access to data, but LONGs support only sequential access. Further, to improve the speed with which a LOB can be brought from the server-side to the client, the LOB can be broken into chunks that can then be brought in a single round trip back to the client.

LOB types store values, called locators, that specify the location of large objects stored out-of-line or in an external file. Database columns of type BLOB, CLOB,

NCLOB, or BFILE store only the locators. BLOB, CLOB and NCLOB data is stored out-of-line inside the database. BFILE data is stored in operating system files outside the database. Oracle8 provides programmatic interfaces and PL/SQL support for access to and operations on LOBs.

5. Multimedia Cartridges

While LOBs provide the infrastructure within the database to store multimedia data, Oracle8 also provides developers with additional functionality for the most commonly used multimedia types. The multimedia types include text, image, video, spatial data and time series data. Oracle8 offers a suite of cartridges, one for each of the specialized data types. Each cartridge allows users to store and access the specific type of data. Users can access objects of the type using efficient SQL queries, manipulate its contents (trim an image), read and write its contents, and convert the data from one format to another.

The cartridges in turn use Oracle8's infrastructure to define the object types, methods, and LOBs necessary to represent these specialized types of data in the databases. Rather than require each development team to define their own sets of objects to represent the most common types of multimedia data, Oracle8's cartridges provide a predefined set of objects and operations. This facilitates application development with these types. The text cartridge is used in Web search engines, document repositories, and online help systems. The spatial cartridge can be used in route planning and asset management applications. Video data is used in online training, video documentation, and sophisticated Web applications.

6. Application Development Interfaces and Tools

Oracle has taken a comprehensive company-wide approach to make sure that customers can very easily build applications using the object functionality in the server. Oracle8 provides a robust infrastructure to facilitate application development and a broad set of application development products (languages, programmatic interfaces, and tools).

Specifically, Oracle8 provides the Object Type Translator—a utility that maps objects in the database to application constructs and marshals the data between the application and data store automatically. Oracle also provides an object cache into which objects can be fetched from the server, operated on as entities, and flushed back to the server while minimizing network traffic.

Oracle offers three kinds of development products with Oracle8:

- Evolved existing languages and interfaces. To leverage its customers' investment and training in familiar application development products, Oracle has extended PL/SQL, the Oracle Precompilers, and the Oracle Call Interface (OCI) support objects.

- New class-oriented language interfaces: For object-oriented application developers, Oracle provides a C++ solution and a comprehensive set of Java products, including JDBC and JSQL (embedded SQL in Java), that provide object-oriented language bindings to Oracle8 objects.

- Application development tools: Oracle is also evolving its existing tools, such as Developer 2000 and Designer 2000, to support object types. For object-oriented application developers who are very familiar with component models, Oracle will offer Sedona, a new component-centric application development tool.

In addition, Oracle is working closely with a number of vendors to ensure that it offers customers the widest possible choice in application development. Oracle works closely with market leading object-oriented analysis and design tools, with Smalltalk, C++, and Java vendors, and with a number of other solution providers. Oracle's efforts with Oracle8 reinforces Oracle's company-wide approach to deliver object functionality to its customers through a wide range of its products.

7. Deployment Considerations—Performance and Scalability

With Oracle8, Oracle integrated object technology with a proven, robust, high performance, scalable deployment platform. Oracle8 continues to provide many advances that Oracle first brought to market and that customers have come to take for granted such as scalable connectivity, read consistency, Oracle Parallel Server (OPS), and views.

Oracle8 delivers many technological advances that ensures it continues to provide industry leading performance and scalability. Among these are: Net8 Connection Manager, advances in the new OCI, and better internal memory management. Oracle has also integrated a number of features into Oracle8 specifically

to improve performance, security, and scalability when applications using Oracle8 objects are deployed:

- Improved concurrency: Oracle8 provides object-level locking, permitting applications using Oracle8 objects to support many more concurrent users.
- Improved performance: Oracle8 provides Complex Object Retrieval capability. This allows an application to fetch a complete graph of connected objects from the database to the application in a single round trip.
- Guaranteed security: Oracle8 provides the same, proven security model with objects as it does for relational tables and stored procedures.

8. Oracle8 Addresses Customer Needs (Buam, 1998)

Now that we have described the object functionality in Oracle8, consider now for a moment how Oracle8 satisfies each of the business needs it aimed to address:

- Using Oracle8 objects, customers can use a consistent model for their business objects in their applications and data store. The Oracle8 application architecture lets users manage complexity when modeling complex business processes.
- Oracle8 provides users with a single data store in which they can store relational data, structured data representing business objects, and unstructured, rich, multimedia data.
- Oracle8 provides a robust and scalable deployment platform for mission-critical applications.

9. Change Management

Change is a necessity for companies wanting to compete successfully. The ability to anticipate, prepare for, and quickly implement change in the business world can provide a compelling competitive advantage. In the IT environment, as database application become more widely used, their performance can suffer. Tuning and performance improvements can require changes in the schema objects supporting the application. If database administrators (DBAs) cannot respond quickly to make the necessary schema changes, business productivity can decline.

B. The Oracle8 Solution Versus Competitors' Solutions

According to how Oracle8 provides a solution that addresses the business needs of its customers. Let's now compare the Oracle solution with those other vendors. It will become clear why Oracle's evolutionary strategy is superior to those of Informix, Microsoft, and other OODB vendors.

Oracle8 represents an evolutionary approach to bringing object functionality to its huge base of relational database customers. By doing so, Oracle avoided the hasty and poorly conceived strategies adopted by other vendors for addressing the same business needs. As a result, Oracle's approach is far more comprehensive, better integrated, and more completely addresses customer needs.

1. Oracle8 Versus Informix Universal Server

Oracle recognized the need to tightly integrate object functionality in the database. Therefore, Oracle chose to evolve Oracle7's universal data server into Oracle8. In contrast, Informix attempted to expedite a solution by purchasing and integrating the Illustra Object Relational Database into its existing database. The resulting product, the Informix Universal Server (IUS), has several limitations, providing a far less complete solution to customer needs than Oracle8.

Informix Universal Server is the industry's extensible object-relational database management system that is designed explicitly to handle rich, complex datatypes. Informix Universal Server combines the power and scalability of Informix's Dynamic Scalable Architecture (DSA) with the extensibility of enabling the intelligent management of complex data while preserving the superior performance and manageability that's required for OLTP-intensive and data warehousing applications.

Informix Dynamic Server is the parallel-everywhere database that delivers the performance, scalability, and extensibility for meeting all information challenges facing companies today. It is ideal for databases of any size, from small development databases to large data warehouses. It is capable of supporting thousands of concurrent users, centrally or distributed across multiple sites. Regardless of your processing (OLTP), decision support, packaged applications, Web/content management, etc. It is the most effective database solution available for addressing all your information needs, delivering the competitive advantage you require to success in business. As your business involves to demand more complex processing, accommodating large scale data warehouses, analytical data marts, sophisticated Web sites, content and knowledge management, and innovative OLTP environments. Informix offered various options

including with adding the ability to perform more complex decision support operation such as bitmap indexes. Extensibility of the ability to run on cluster and massively parallel processing architectures.

Six of the most important differences are discussed below. You will see that Oracle provides a more integrated and comprehensive solution to the needs of mainstream customers than Informix.

a. Single Product Versus Many Products

Oracle chose to maintain a single, unified platform rather than compelling its customers to choose between various platforms: DSA for OLTP applications on SMP hardware, XPS for Data Warehousing applications on cluster and MPP hardware, and IUS for multimedia applications. Each platform contains a collection of features not available on the others.

Oracle's approach has three important advantages:

- Simplifies customer's investment decisions.
- Requires them to invest in learning only on product rather than three.
- Introduces the large base of data warehousing and OLTP customers to objects and the benefits of objects far sooner.

b. Mainstream Applications Versus Niche Applications

Informix focused most of its efforts on providing customers the ability to store new multimedia and specialized datatypes, instead of addressing the far greater market

need of storing and accessing business objects easily. As a result, IUS addresses only a small niche as early as Oracle7 Release 7.3 with its Context and Image products. It has further extended its support for multimedia and specialized datatypes with Oracle8.

c. Complete Application Development Architecture and Tools Versus Low Level

Interfaces, No Tools

Oracle8 provides a complete application development architecture: a client-side object cache, the object type translator, and automatic marshaling of data between the application and database. IUS does not provide any of this functionality. IUS requires users to write complex, low level code to server application programming interfaces (APIs) to applications. As a result, IUS makes it difficult to treat a business object consistently in the application and database.

In addition to providing the infrastructure for application development, Oracle8 provides a number of languages, interfaces, and tools to facilitate application design and development. Informix, on the other hand, provides no object-oriented analysis, no design tools, nor any application development tools for the IUS.

d. Tight Integration of Object/Relational Data Versus No Integration

Oracle8 lets users use SQL to query on objects, use object types in stored procedures, and compose objects from relational data using object views. IUS, on the other hand, does not allow a user to use object types in stored procedures and lacks object views. As a result, IUS compels customers to migrate their data in order to use object functionality.

e. Proven Scalable Deployment Platform Versus Untested, Experimental Deployment Platform

By maintaining a single product, Oracle ensured that its customers can take performance and scalability for granted when they deploy their applications. Furthermore, Oracle8 objects will run efficiently on SMP, clustered, and MPP hardware. This is in sharp contrast to IUS which lacks scalability and does not run efficiently on clusters and MPP hardware.

f. Logical Identity Versus Physical Identity

Objects in Oracle8 have logical identity through an object ID that is not tied to a specific database table. Objects in IUSs however have ROWIDs which are tied to a specific database table. This causes two major problems. First, if the table structure is reorganized, all applications using any ROWID in that table will break. Second, since ROWIDs are tied to a specific tables in IUS are not distributable across multiple databases.

The issues listed above illustrate clearly why Oracle8 is a far more complete solution to satisfy the business requirements of the mainstream market than the Informix Universal Server.

2. Oracle's Strategy Versus Microsoft

Oracle provides a more integrated, open, cross-platform, scalable solution than Microsoft. Due to SQL Server's inability to scale and perform, and its consistent failure

to win acceptance as a database for mission-critical enterprise applications, Microsoft's strategy is in stark contrast to Oracle in a number of important ways.

a. Single Database Versus Federated Databases

Microsoft has attempted to shift its message away from SQL Server to focus on Federated Databases that can be accessed through interfaces such as ODBC and OLE-DB. Oracle8 addresses the market requirements clearly indicating Microsoft's flawed strategy by:

- Preventing applications from leveraging the capabilities built into databases.
- Requiring specialized databases to handle specialized types of data.
- Making it difficult to simply and easily partition applications.
- Placing much of the burden of coordinating transactions on the application developer.

Federated Databases also significantly raise a customer's investment requirements significantly.

b. Scalable Database and Application Server Versus Application Server Only

Microsoft attempts to address application performance and scalability solely by migrating all of the application logic out of the database into Microsoft's proprietary Microsoft Transaction Server [MTS]. This is unnecessary, making application development more complex, and runs counter to customer experience over the past several years. For example, consider the limited penetration of TP Monitors and the poor reception of MTS. Oracle has chosen a two-part strategy:

- Continue to provide industry-leading performance and scalability with Oracle8.
- Provide those customers who own applications that need the capability of an application server or TP-Monitor for operational reasons, with the widest choice of TP-Monitor partners and Oracle's Web Application Server, which will soon incorporate Transaction Services.

c. Multiple Platform Versus Single Platform

Microsoft's SQL Server and its application development products are only available on a single, proprietary platform—Windows. Oracle8, on the other hand, will be ported along with many of Oracle's application development products to the nearly 90 platforms that Oracle currently supports.

d. Open Standard Versus Proprietary Standards

Microsoft's strategy is based on a proprietary set of standards D/COM and OLE-DB. Oracle's strategy has focused consistently on supporting open, industry standards such as CORBA, SQL, JDBC, and JSQL.

As a result, enterprise customers will find Oracle's solution far more easy to use more scalable, and with higher performance than that provided by Microsoft.

3. Oracle8 ORDBMS Versus Object-Oriented Databases

The cornerstone of the object-oriented approach is that it provides a more natural way to model many real-world situations. The important point here is that the model obtained by using an object-oriented approach will be a more direct representation of the situation, providing a better framework for understanding and manipulating the complex relationships which may exist. A number of key characteristics can be identified. These are:

- Support for object instances and classes: In the object-oriented model the entities of interest are called objects. These objects have in common which are data properties associated with its record the date design which was last changed and who changed it? As well as recording the design itself, and can be manipulated with a fixed set of editing commands and tools which extend the design, verify its correctness, and so on. In relational database systems for example, this problem arises and is overcome by associating a special property (or set of properties) called the primary key with each relational tuple. For any one relation, no two tuples of that relation are allowed to be assigned the same value simultaneously for the primary key, while this approach is satisfactory in some situations (e.g. a student entity can be given a unique registration number property). In object-oriented systems a distinction is made between an object's identity and its properties. As a way of organizing objects in a more manageable way, objects which have the same kinds of properties, and can be manipulated using similar operators, are classified to form distinct object classes. For example, the digits 7 and 8 response to the same operators (adding, subtracting, etc.) as well as when we have instances of employee classes and instances of a software engineer class, we will be able to use both classes in common such as name, date of birth or salary. They will be also having similar operators for manipulation such as hire or fire. The definition of the

object class can act as a template for creating instances of the class. Each instance of a class has a unique identity, but has the same set of data properties (or state variables) and responds to the same set of operators (or methods).

- Encapsulation of operations with data: The important of encapsulation is that the operators form an interface to assignment objects which provide the only way to amend the state of assignment objects. The user of an object has no way to access that object other than through the defined set of operators. This is the approach which one server object provides a fixed set of services to another client object. While the client object can't manipulate or make change to the server object other than through the set of services provided by server. For example, we can record information of programmers and their assignment to projects by encapsulating the data structures modeling. An assignment object can represent the fact that a particular programmer has been assigned to a project, with the state recording the name of programmer, project name, etc. It means that the definition of an assignment object tells us not only what is it, but also what you are allowed to do with it.

- Specialization Objects: According to the characteristics of the object classification, we can use the database terminology to define a specialization more over which is a subclass hierarchy. We can think of a new class as an existing class amended to fit our new requirements. Especially, we can define subclass as inheriting the behavior of its superclass, which derived behavior from the superclass.

- Advantages of Object-oriented: (Martin, 1997)

- Data hiding
- Data independence
- Modularity
- Reuse

- Disadvantages of Object-oriented:

- No formal semantics. Object-oriented data model has developed in a much more ad hoc way, mainly through extensions to existing programming languages.

It is only as a response to the existing problems.

- Loss of relational simplicity. In some applications the simplicity of the relational language as a modeling notation has led to the database schema begin to use as a communication medium between systems analyst and customer, allowing the customer to help in the verification of the design before and during implementation. For large extent the object-oriented model has compromised this simplicity help providing the greater modeling capability.

- Navigational queries. Object-oriented model made use of direct object references to represent relationships between objects. It has the consequence that access to data is essentially by navigating the object references (i.e. pointers) to move from one object to another.

- No general query language. Object-oriented database systems are specifically aimed at fixed interaction with complex structures; it is not easy to define ad hoc queries on the data.

- Lack of support for dynamic processes. Object-oriented systems describe encapsulated data structures and operations. So, the behavior of each object is defined by the operations which have been assigned to. (*Brown*)

Oracle provides a more robust, tightly integrated, and fully functional solution than OODBs. Object-Oriented databases (OODBs) attempt to provide persistence to a set of object, defined as classes in an object-oriented language such as C++. This contrasts with Oracle's strategy which is to provide an easy

way to store business objects in the database and access them through a number of client-side language interfaces (including OO programming languages). Since OODBs focus on adding persistence to object-oriented languages not designed with multiple users in mind, OODBs face several important limitations.

a. Robust Versus Fragile, Non-Scalable Deployment Platform

OODBs have repeatedly failed to scale in multi-user enterprise setting. Further their performance is extremely uneven as concurrent user loads on the database increase. In contrast, Oracle8, in a recent demonstration, supported over 17,000 concurrent users. As a result, after nearly 10 years, OODBs are still confined to a small niche primarily for experimental purposes.

b. Simple Queries Versus Lack of Query Ability

OODBs use a different kind of query language, OQL, that does not conform with, and lacks many important facets of industry-standard SQL. It lacks, for example, nested subqueries, set queries, aggregation functions, and joins of multiple classes. As a result, OODBs meet only a small part of a customer's needs. OODBs allow users to store business objects easily, but these objects cannot be queried or retrieved easily from the database once they have been stored.

c. Tight Object-Relational Integration Versus No Integration

Unlike Oracle8, OODBs provide no facilities to support relational data. As a result, customers must migrate their data and build new applications in order to use an OODB. With Oracle8, existing applications can run against the relational tables, while

new applications using object functionality can be developed using object views without migrating any data.

d. Complete Application Development Solution Versus No Infrastructure

OODBs have very limited support for application development, query, and reporting tools. In contrast, Oracle8 offers a very wide range of development products and tools to facilitate application development. Many industry-leading tools vendors are also actively developing new products or evolving their existing products to work with Oracle8.

e. Operationally Viable Solution Versus Operationally Non-Viable Solution

Most OODBs lack important pieces of functionality that preclude them from being viable in a mission-critical environment. Three issues need to be addressed here:

- **Functionality gaps:** Many OODBs lack multi-domain query ability, view mechanisms, and automatic lock management. This places these burdens on the application developer.

- **Inadequate security:** Many OODBs lack adequate database security, preventing businesses from completely centralizing the administration and enforcement of their business rules. Oracle8 provides the same robust security model with objects that it does with relational data.

- **Administrative issues:** OODBs have consistently failed to provide adequate facilities to carry out important, but elementary administration tasks, such as backup, recovery, and performance tuning.

Oracle8's object-relational capability achieves the main purpose of OODBs; it can store and access business objects easily. In addition, it can overcome all the performance, scalability, and functionality shortcomings of today's OODBs. With Oracle8's release, Oracle simultaneously brings object functionality to mainstream applications and eliminates the need from OODBs.

The following tables listed some common features for the database products against market functionality.



Table 3.1. Summary of Comparison in Java Initiatives

<i>Features for Java Support</i>	<i><u>Oracle8i</u></i>	<i><u>Informix</u></i>	<i><u>Microsoft</u></i>
JDBC Driver Support	Yes	<i>Yes</i>	<i>Yes</i>
SQLJ Translator	Yes	<i>No</i>	<i>No</i>
Using SQLJ with Database	Yes	<i>No</i>	<i>No</i>
Support for Enterprise Java Beans	Yes	<i>No</i>	<i>No</i>
Java Virtual Machine	Yes	<i>Yes</i>	<i>No</i>
Java Stored Procedures	Yes	<i>No</i>	<i>No</i>
Java Component Development	Yes	<i>Yes</i>	<i>No</i>

Other database products view Java as a programming language. They also provide Java database connectivity in client-side but no Java Virtual Machine in the database.

Table 3.2. Summary of Comparison in Web Enablement

<i>Features for Web Information Management</i>	<i><u>Oracle8i</u></i>	<i><u>Informix</u></i>	<i><u>Microsoft</u></i>
Extensible Indexing/Clustering	Yes	Yes	No
Extensible Optimizer	Yes	Yes	No
Cartridge Services	Yes	Yes	No
Spatial Cartridge	Yes	Yes	No
Image Cartridge	Yes	Yes	No
Time Series Cartridges	Yes	Yes	No
Data-Server Managed Audio Cartridge	Yes	Yes	No
Data-Server Managed Video Cartridge	Yes	Yes	No
Extensibility Partners (Cartridge Providers)	Yes	Yes	No
WebDB Capabilities	Yes	Yes	No

Oracle8 provides features for all multimedia data types and enable web information management by a tool for publishing data to the Web comparable with WebDB. Other products, Web and other contents are retrieved by keyword searching which in the full-text indexing of database feature or maintained the Web pages only.

Table 3.3. Summary of Comparison in Enterprise Application Integration

<i>Features for Enterprise Application Integration</i>	<i><u>Oracle8i</u></i>	<i><u>Informix</u></i>	<i><u>Microsoft</u></i>
Rules based Publish/Subscribe messaging	Yes	No	No
Database Events	Yes	Yes	No
Advanced Replication	Yes	Yes	No
Support for Mass Deployment	Yes	Yes	No
Row-level locking	Yes	Yes	Yes
Snapshot Security	Yes	No	No
Distributed Query Optimizations	Yes	Yes	No
MS Transaction Server integration	Yes	Yes	Yes
Fine-grained Access Control	Yes	No	No
N-tier Authentication/Authorization	Yes	Yes	Yes
Multi-threaded Database Server	Yes	Yes	No

All database products are actively developing their features to be enabled the organizations to integrate or extend their legacy application's with today's leading enterprise applications. Mostly claim the integration with enterprise applications such as SAP, PeopleSoft or Baan in their platform.

Table 3.4. Summary of Comparison in Data Warehousing

<i>Features for Data Warehousing</i>	<i><u>Oracle8i</u></i>	<i><u>Informix</u></i>	<i><u>Microsoft</u></i>
ROLAP operators	Yes	Yes	Yes
Direct-path Load API	Yes	Yes	No
Parallel Query	Yes	Yes	Yes
Terabyte Database	Yes	Yes	Yes
VLDB Architecture	Yes	Yes	No
Scalable SMP/MPP Support	Yes	Yes	No
Parallel User Defined Functions	Yes	Yes	No
Parallel DML	Yes	Yes	No
Function-based Indexing	Yes	Yes	No
Data Partitioning	Yes	Yes	No
In Memory Database (64 bit)	Yes	Yes	No

By product features, Oracle also provides support of all aspects for data warehousing and data mart solutions. For others with the reason of attempting to address some of data warehousing shortcomings, included now are: parallel query, larger page size, more table joining, Data Transformation Service Support, etc. By providing OLAP technology which bundled with their products. This separation means more expensive administration and maintenance.

Table 3.5. Summary of Comparison in Database Administration and Operational Simplicity

<i>Features for Database Administration and Operational Simplicity</i>	<i><u>Oracle8i</u></i>	<i><u>Informix</u></i>	<i><u>Microsoft</u></i>
Database Creation/Removal	Yes	Yes	Yes
Graphical Query Analyzer	Yes	Yes	Yes
Runs in Silent Mode	Yes	Yes	No
Remote Database Creation	Yes	Yes	No
Helpful Error Messages	Yes	Yes	No
Pre-Tuned Starter Database	Yes	Yes	Yes
Hardware Detection	Yes	Yes	No
Sample Schemas	Yes	Yes	Yes
Heterogeneous Platform Support	Yes	Yes	No
Replication Management	Yes	Yes	Yes
DROP Column (ANSI SQL92 Transitional)	Yes	Yes	Yes
Locally-Managed Tablespaces for reduced fragmentation and no need for table reorgs	Yes	Yes	No
Online Index Rebuilds for better availability	Yes	No	No
Monitoring / Scheduling	Yes	Yes	Yes
Online Backup and Recovery	Yes	Yes	Yes
Read-Only Tablespaces	Yes	Yes	No

All database products provide the perception in easy-to-use by providing Wizards to help in performing tasks. The ease-of-use features they market include:

Dynamic Self Management, Multi-site and Alert/response management, Job scheduling and execution, Distributed management objects, DBA profiling/tuning tools, Installation and upgrade with seamless migration or Major contribution to Zero Administration. Oracle believes in the thin-client model, the software must not be managed on all the PC servers that is not easy to manage for the software required copied to each PC server.



Table 3.6. Summary of Comparison in Availability and Scalability Solution

<i>Features for High Availability/Scalability</i>	<i><u>Oracle8i</u></i>	<i><u>Informix</u></i>	<i><u>Microsoft</u></i>
Database Resource Manager	Yes	<i>Yes</i>	<i>Yes</i>
TerraServer Support	Yes	<i>Yes</i>	<i>Yes</i>
Parallel Server	Yes	<i>Yes</i>	<i>No</i>
Parallel Server Enhancements			
Consistent Read Server	Yes	<i>Yes</i>	<i>No</i>
User-Mode IPC Support	Yes	<i>Yes</i>	<i>No</i>
High Availability Job Submission	Yes	<i>Yes</i>	<i>No</i>
Dynamic Load Balancing	Yes	<i>Yes</i>	<i>No</i> <i>(promising as a part of NT 5)</i>
Multiple, Remote Archival of Redo Logs	Yes	<i>No</i>	<i>No</i>
Standby Database Enhancements	Yes	<i>Yes</i>	<i>No</i>
OLTP Enhancements			
Optimizer Plan Stability (bypasses rewrite for well-tuned apps)	Yes	<i>Yes</i>	<i>No</i>
Stored Plan Outlines	Yes	<i>Yes</i>	<i>No</i>
Copy Optimizer Statistics to other Databases	Yes	<i>Yes</i>	<i>No</i>

With Internet computing and e-commerce availability requirement is not longer just for the enterprise. This is becoming a mass requirement. Oracle includes many enhancements for Oracle Parallel Server, the industry leading optimizer, improved archiving, better recovery, and many OLTP enhancements, all which provide highly scalable environments with better resource usage and many different proven availability solutions.



IV. RISK ANALYSIS

A. Company Vision Risk

Oracle has a clear vision and track record of incorporating technology for advancement while maintaining backward compatibility for those customers whose business need not yet require the newer technology. If IT manager chooses the products that mostly developed based on their own best interests, he is supposed to be charged for replacement software, required purchase of other software parts to achieve functionality.

B. Platform Risk

Oracle8 runs on not only Windows NT and Windows98, but almost any other operating systems available today. What if any kind of operating system is not the end-all platform? IT decision makers needs the solution which could be going to cover them if the selected platform is the wrong choice.

C. Technology Risk

Standard Oracle8 provides features such as Java-in-the-database that companies can use as well as continued investments in technology advancements, due to the industry moving quickly and to stay competitive. With Internet computing, some 'Enterprise-type' features are now becoming business requirements. For those features, Java is now the language of the Internet.

D. Partner Risk

Oracle continuously works closely with key partners to smoothly upgrade their products to work when products are released. Oracle is the leader with SAP as 70% of SAP's installations are on Oracle. Decision makers find the products which have partner products available that complement their business needs and have more stability.

E. Product Risk

Oracle continues to have compatible software and has a track record of consistency to prove it. Major changes to the database internals make for predictable performance, scalability and reliability.

F. Scalability Risk

Oracle8 can provide the necessary scalability for enterprise as well as small to mid-size business. Decision makers realized that if their development's application become so outstanding that the company needs to integrate with it, they suppose that the applications and databases should be able to grow incrementally with added users, processors and data.

V. FUTURE DIRECTION AND CONCLUSION

A. IT Challenges

In today's demanding business world, operational systems are relied upon more than ever. User communities are growing in number and they are demanding higher levels of application performances, reliability, and availability. Business environments are also more dynamic than ever before, with continually decreasing time lags for their applications to respond to changing business requirements. Also, strategic decision must be made quickly, so businesses can react and adapt to reorganizations and process changes, regulation revisions, and competitors' directions. These decisions need to be based on accurate data and thorough analysis. The ability to analyze and exploit operational data in a data warehouse becomes a key competitive weapon.

To meet these challenges, IT organizations need an enterprise software strategy for managing any data, in any application, at any scale. Oracle8 is the only open systems solution that meets the demands of high-transaction on-line transaction processing (OLTP) systems, query processing in large-scale data warehouses, and manageability requirements of businesses that are distributing data throughout the corporate enterprise and the Internet. Oracle8 also makes a major leap in data management technology with the introduction of an object-relational paradigm for complex applications. This improved way of defining data structures allows developers to directly define their business objects, such as purchase orders, inventory items, and data warehouse information, within Oracle8. This allows developers of mainstream commercial applications to better manage their business objects. Oracle8 leverages

investment in development, deployment, and maintenance of applications by providing a single product to deliver the maximum benefits of open, multimedia systems across your enterprise, while minimizing the risks, complexity, and costs of moving to both centralized and distributed application environments.

So how does Oracle8 fit into Oracle's future strategic roadmap? Oracle8 provides the foundation for Oracle's future strategy as defined by its recently announced Network Computing Architecture (NCA). To clarify the relationship between the object functionality in Oracle8 and the NCA, this section explores two important areas:

- Database extensibility: It defines the concept of database extensibility, illustrates how the object functionality in Oracle8 lays the foundation for extensibility, and describes how Oracle plans to extend Oracle8 in future releases.
- Network Computing Architecture: It clearly defines how the object functionality delivered by Oracle8 and the concept of database extensibility form a fundamental component of NCA.

B. Oracle8 and Database Extensibility

Oracle8 provides the foundations of database extensibility which will be enhanced with subsequent releases. Database extensibility allows users to define software cartridges consisting of:

- Objects and methods defined by the user to represent new data types.
- Operators and aggregates defined by the user on these objects.

- Special ways of indexing on these objects that are implemented by the user.
- A consistent scheme to report errors on these data types.

Therefore, database extensibility requires opening up several important components of the database engine: its SQL execution engine, indexing mechanism, and optimizer. This allows users to build new data types and to treat them as types native to the server. These new types may be multimedia types such as image, text and video, or domain-specific types which represent data that is unique to a particular industry or vertical market. Oracle8 provides several important elements of the database extensibility infrastructure which will be further extended during future releases. These are:

- Extensibility in Oracle8: Oracle8 provides several important elements of the infrastructure required for database extensibility. It allows users to define their own objects and methods and treat them as native types in the database. It allows users to index on these object types. External procedures provide a mechanism through which the database engine can call out to cartridges that have been registered with the database. Further, with Oracle8, Oracle and some of its partners already provide a series of cartridges that cover all the fundamentally important multimedia data types: text, image, spatial, time series and video data. With subsequent release of Oracle8, Oracle will further standardize the set of interfaces it uses internally to support these cartridges and provide them as a complete framework, enabling a number of third parties to easily develop these cartridge types.

- Extensions to extensibility infrastructure: Oracle will provide an extensibility

framework that will extend key components of its database infrastructure: the SQL engine, its indexing mechanism, and optimizer. This will allow users to define new data types with their own operators and aggregates, index them in unique way implemented by the user, and optimally query over these types.

- Packaging: Extensibility is facilitated by the ability to package object types defined by the user as cartridges that can be accessed through standardized interfaces. With Oracle8, a user can define and store business objects in the server. With future releases of Oracle8, Oracle will provide additional functionality to let these business objects be packaged as cartridges that can then be externalized through standard CORBA IDL interfaces.

C. Oracle8 and Network Computing Architecture

This allows IT organizations to spend less time struggling with interoperability and more time focusing on deploying solutions. Oracle8, a major component of NCA, is designed to meet the demand of network-centric computing and object-oriented development methods. NCA provides maximum extensibility and is based on open industry standards such as CORBA and IIOP. Whether for traditional enterprise applications or electronic commerce on the Web, Oracle8 and NCA provide the power, robustness, network integration, and flexibility to support the most demanding applications. Oracle8 is a central component of NCA with its integration of a Java VM and a CORBA ORB. NCA represents Oracle's strategic vision for distributed objects and network computing. It provides a complete architectural framework consisting of thin clients, an extensible application server, and an extensible database server which communicate with each other through standards-based CORBA/IIOP protocols.

The NCA framework provides a standards-based environment in which applications can be easily assembled from distributed components and deployed in a thin client environment. Oracle8 is a fundamental component of NCA and reinforces each of the important messages of NCA:

- Simplified application partitioning: The NCA strategy is aimed at further simplifying application partitioning to optimize application performance and scalability. Oracle8's support for a simplified and consistent representation of business objects in the application and the database is a critical underpinning to NCA's direction of simplified partitioning. Couple these benefits with Oracle8's advances in performance and scalability and Oracle8 becomes the ideal choice as the database for network computing applications.
- Enabling thin client and Web applications: Oracle has been the champion of thin clients and Web technology, supporting these as ways to enable a new generation of network centric applications while making it possible for customers to significantly reduce their information technology costs.

NCA provides a simple framework for customers to build Web-enabled applications that can execute in a thin client environment. To enable these applications, Oracle has adopted a comprehensive Java strategy with support for Java in all three tiers of NCA. Oracle is providing a complete set of JDBC drives and JSQL (embedded SQL in Java) for Java application development.

NCA defines an infrastructure for developing, and managing multi-tier, Web-enabled applications. As such, three of the key platforms of NCA comprise the majority of its applications:

- Clients: Clients provide the user interface at the first tier of NCA's applications. A broad range of client types is supported.
- Application Service: At the middle tier are a set of shared modular services that implement the logic, or business rules, of applications.
- Database: Oracle's Universal Server forms the database tier of NCA. It has been enhanced to become an open, extensible database. Through NCA, it will be able to support complex types and encapsulate database-specific behavior.

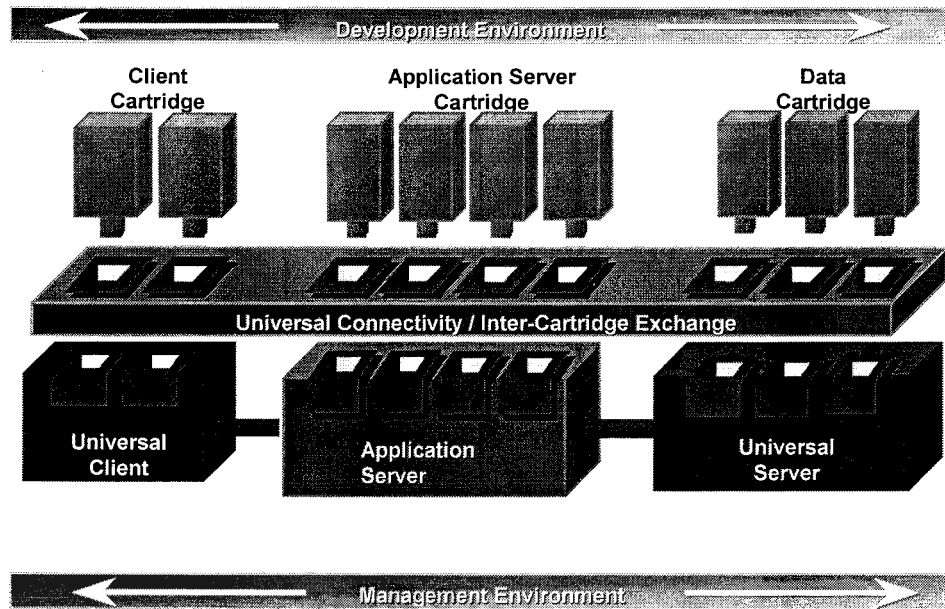


Figure 5.1. Oracle8 and NCA define how software components communicate with each other and database. The database provides a safe, secure mechanism for invoking external code.

In addition to these application components, NCA also includes the interfaces through which applications components communicate; the protocols that define how those communications take place. NCA built around the concept of a software bus, a well-proven architecture for supporting large scale, distributed applications. The bus is a logical mechanism that all NCA application platforms—clients, services, and the database—used to communicate with each other. They “plug” into it using a common set of interfaces and communicate using a common protocol set:

- **Cartridges:** NCA’s pluggable components are called cartridges. Cartridges are based on CORBA objects. They may implement application functionality or provide support services at any of NCA application platforms.

- Inter-Cartridge Exchange: NCA's software bus is called the Inter-cartridge Exchange, or ICX. Application components access ICX through a set of libraries. The interface provided by these libraries will support both request/response and messaging and queuing interactions between clients and servers. The protocols supported by these libraries are HTTP and CORBA/IIOP.

- Support Services: Oracle has defined three sets of support services for NCA. They provide a rich infrastructure for developing, deploying, executing, and managing all aspects of the NCA environment.

- Universal Cartridge Services will provide facilities to deploy, administer, and manage application cartridges.

- Scalable Cartridge Services will provide support for transactions, messaging and queuing, and data access.

- Specialized Cartridge Services will provide support facilities at each of the three tiers of NCA applications: client, application server, and database.

The key benefits of ICX are that it both simplifies and standardizes client/server interactions. Simplification is achieved through an abstracted, high-level interface that insulates developers from the details of operating systems and networks. The interface enables consistent inter-component communication independent of platform, component type, or component location. Standardization is achieved by using a common interface for all interactions. Developers have to learn only one high-level set of commands for all types of application components for all kinds of applications. The bottom line is that ICX can reduce the time to build distributed applications and can help developers build higher-quality application components.

Most importantly, Oracle is building a scalable Java Virtual Machine in the database server to provide support for Java stored procedures, methods, and triggers. Oracle's comprehensive Java solution will allow customers to build and easily partition client/server and multi-tier applications completely in Java. These products will further accelerate Java's momentum and make the primary choice for enterprise application development.

NCA focuses on CORBA and IIOP as standard ways for clients, middle-tiers, and database servers to communicate. With future releases of Oracle8, Oracle will integrate a CORBA Object Request Broker into the server. The ORB will allow applications to communicate with the database using IIOP. Business objects, store procedures, and packages defined in the server can be published with standard CORBA interfaces defined in Interface Definition Language (IDL). Finally, the database engine can use the ORB's infrastructure to call out from the database, extending the external procedure mechanism that is provided with Oracle8.

NCA will be very attractive to corporate developers. It will provide and infrastructure that can support new distributed and new Web-based applications while integrating existing client/server and legacy resources. It will deliver the maintainability, extensibility, and reuse benefits of component-based applications.

NCA will also deliver the following benefits to corporate developers:

- A Complete Solution: NCA will be a complete middleware solution. It will

address all requirements for the development, deployment, and management of distributed component-based applications in a tightly integrated manner—standard systems integrators, a have many prior approaches to distributed computing. No longer will the developers have to build an infrastructure in piece-parts fashion from sets of incompatible or poorly integrated products.

- **Leverage Existing Resources:** New applications, especially large-scale applications, are never implemented in a vacuum. They must fit smoothly into existing environments. In fact, this integration has become one of the top middleware products. NCA is particular strong in this area. It will incorporate ActiveX and PL/SQL clients and will integrate legacy resources. Corporate developers will not have to become systems integrators to implement NCA and applications built on it.

- **Leverage Existing Skills:** NCA cartridges are language independent. As a result, corporate developers will not have to learn new languages and new development tools to build NCA applications. The time and cost required to build NCA applications will thus be minimized, which is important because cycle time has become the key driver in corporate development activity.

- **A Big Push for Corba:** CORBA-compliant products have been available for several years, but their adoption rate has been almost painfully slow. Oracle's decision to base NCA on CORBA will instantaneously make CORBA a mainstream technology. It will give CORBA a mainstream technology. It will give CORBA a giant boost in credibility and will accelerate its adoption rate.

- **A Viable Distributed Computing Infrastructure:** NCA will become a viable infrastructure for large-scale, distributed, component-based applications. It is at once open, inclusive, and extensible. It will provide competition to Microsoft's DCOM approach and will complement the efforts of the OMG and the vendors that offer

CORBA products. NCA will also expand and enhance Oracle's position as a database vendor and will immediately make the company a major player in the middleware market. (Kramer, 1996)

D. Enterprise-class Transaction Processing

Oracle8's scalable, reliable architecture delivers unmatched scalability, availability, and performance needed for mission-critical OLTP systems. Integrated, dynamic facilities ensure that Oracle8 and Oracle8 Parallel Server make efficient use of all system resources on hardware ranging from uni-processors to symmetric multiprocessors (SMP), to clusters, to massively parallel processors (MPP).

E. Superior Scalability for Transaction Processing

The Oracle8 architecture provides OLTP applications with scalability to support large numbers of users and high-volume transaction workloads. Oracle8 provides exceptional scalability on SMP, clustered, and MPP machines. OLTP applications take advantage of Oracle8's parallel architecture by distributing tasks across multiple processors or machines, such as in a clustered environment, which improves individual transaction response times and overall system throughput. An automatic, dynamic self-tuning capability balances processing workload evenly across allocated hardware and operating system resources. Additional processors and/or nodes can be added to expand configuration incrementally as both organization and data volumes grow—with minimal disruption to existing environment—resulting in dramatic performance improvements and breakthrough price/performance. OLTP systems require high availability so your

business can continue to operate when a hardware failure occurs. The Oracle8 Parallel Server uniquely extends the reliability of open systems applications by transparently harnessing the power of clustered computers in a single logical processing “complex” that can tolerate individual machine and/or node failures without loss of data availability. The Oracle8 Parallel Server also supports hybrid configurations, combining elements of clustered and MPP architectures. Should a node in the parallel server fail, transparent application failover migrates the users’ connections and automatically re-establishes their session on another node. The users’ applications continue to run, and the users themselves may be unaware of the failure. This provides continuous availability in the event of scheduled and unscheduled and unscheduled outages.

F. High Performance for Transaction Processing

For high-performance transaction processing, the Oracle8 multithreaded, multiserver architecture coordinates thousands of simultaneous user requests. Individual requests are queued and serviced by a minimum of server processes. Sophisticated caching of database blocks, SQL execution plans, and executable stored procedures takes maximum advantage of available server memory. Available system resources can be precisely allocated with a high degree of control, optimizing performance to the capabilities of the system and to the system workload on a dynamic basis.

G. Object Views

Just as a view is a virtual table, an object view is virtual object table. Oracle provides object views as an extension of the basic relational view mechanism. By using object views, you can create virtual object tables from data-of either built-in or user-defined types-stored in the columns of relational or object tables in the database.

Object views allow use of relational data in object-oriented applications. They let users:

- Try object-oriented programming techniques without converting existing tables.
- Convert data gradually and transparently from relational tables to object-relational tables.
- Use legacy RDBMS data with existing object-oriented applications.

Object views provide the ability to offer specialized or restricted access to the data and objects in a database. For example, you might use an object view to provide a version of an employee object table that doesn't have attributes containing sensitive data and doesn't have a deletion method.

1. Advantages of Object Views

Using object views can lead to better performance. Relational data that make up a row of an object view traverse the network as a unit, potentially saving many round trips. You can fetch relational data into the client-side object cache and map it into C or

C++ structures so 3GL applications can manipulate it just like native structures. Object views provide a gradual migration path for legacy data.

Object views provide for co-existence of relational and object-oriented applications. They make it easier to introduce object-oriented applications to existing relational data without having to make a drastic change from one paradigm to another.

Object views provide the flexibility of looking at the same relational or object data in more than one way. Thus you can use different in-memory object representations for different applications without changing the way you store the data in the database.

2. Defining Object Views

Conceptually, the procedure for defining an object view is simple:

- Define an object type to be represented by rows of the object view.
- Write a query that specifies which data in which relational tables contain the attributes for objects of that type.
- Specify an object identifier, based on attributes of the underlying data, to allow REFs to objects (rows) of the object view.

The object identifier corresponds to the unique object identifier that Oracle generates automatically for rows of object tables. In the case of object views, however, the declaration must specify something that is unique in the underlying data (for

example, a primary key). If the object view is based on a table or another object view and you don't specify an object identifier, Oracle uses the object identifier from the original table or object view. If you wish to be able to update a complex object view, you may have to take another step. Write an INSTEAD OF trigger procedure for Oracle to execute whenever an application program tries to update data in the object view.

After these steps you can use an object view just like an object table. For example, the following SQL statements define an object view:

```
CREATE TABLE emp_table (  
  empnum      NUMBER (5),  
  ename       VARCHAR2 (20),  
  salary      NUMBER (9, 2),  
  job         VARCHAR2 (20) ;  
  
CREATE TYPE employee_t (  
  empno       NUMBER (5),  
  ename       VARCHAR2 (20),  
  salary      NUMBER (9, 2),  
  job         VARCHAR2 (20) ;  
  
CREATE VIEW emp_view1 OF employee_t  
  WITH OBJECT OID (empno) AS  
  
  SELECT      e.empnum, e.ename, e.salary, e.job  
  
  FROM        emp_table e
```

WHERE job = 'Developer';

The object view looks to the user like an object table whose underlying type is employee_t. Each row contains an object of type employee_t. Each row has a unique object identifier.

Oracle constructs the object identifier based on the specified key. In most cases it is the primary key of the base table. If the query that defines the object view involves joins, however, you must provide a key across all tables involved in the joins, so that the key still uniquely identifies rows of the object view.

Note: Columns in the WITH OBJECT OID clause-empno in the example-must also be attributes of the underlying object type-employee_t in the example. This makes it easy for trigger programs to identify the corresponding row in the base table uniquely.

3. Using Object Views

Data in the rows of an object view may come from more than one table, but the object still traverses the network in on operation. When the instance is in the client side object cache, it appears to the programmer as a C or C++ structure or as a PL/SQL object variable. You can manipulate it like any other native structure.

You can refer to object views in SQL statements the same way you refer to an object table. For example, object views can appear in a SELECT list, in an UPDATE-SET clause, or in a WHERE clause. You can also define object views on object views.

You can access object view data on the client side using the same OCI calls you use for objects from object tables. For example, you can use `OCIObjectPin()` for pinning a REF and `OCIObjectFlush()` for flushing a object to the server. When you update or flush to the server an object in an object view, Oracle updates the object view.

4. Updating Object Views

You can update, insert, and delete the data in an object view using the same SQL DML you use for object tables. Oracle updated the base tables of the object view if there is no ambiguity.

A view is not updatable if its view query contains joins, set operations, group functions, GROUP BY, or DISTINCT. If a view query contains pseudo columns or expressions, the corresponding view columns are not updatable. Object views often involve joins.

To overcome these obstacles Oracle provides INSTEAD OF triggers. They are called INSTEAD OF triggers because Oracle executes the trigger body instead of the actual DML statement.

INSTEAD OF triggers provide a transparent way to update object views or relational views. You write the same SQL DML (INSERT, DELETE, and UPDATE) statement as for an object table. Oracle invokes the appropriate trigger instead of the SQL statement, and the actions specified in the trigger body take place.

H. User-Defined Datatypes (Object Option)

The objects option allows users to define datatypes that model the structure and behavior of the data in their applications. Relational database management systems (RDBMSs) are the standard tool for managing business data. They provide fast, efficient, and completely reliable access to huge amounts of data for millions of businesses around the world every day.

The objects option makes Oracle and object-relational database management system (ORDBMS), which means that users can define additional kinds of data-specifying both the structure of the data and the ways of operating on it-and use these types within the relational model. This approach adds value to the data stored in a database.

Oracle with the objects option stores structured business data in its natural form and allows applications to retrieve it that way. For that reason it works efficiently with applications developed using object-oriented programming techniques. Oracle's support for user-defined datatypes makes it easier for application developers to work with complex data like images, audio, and video.

I. Complex Data Models

The Oracle Server allows you to define complex business models in SQL and make them part of your database schema. Applications that manage and share your data need only contain the application logic, not the data logic.

1. An Example

For example, your firm may use purchase orders to organize its purchasing, accounts payable, shipping, and accounts receivable functions.

A purchase order contains an associated supplier or customer an indefinite number of line items. In addition, applications of need dynamically computed status information about purchase orders. For example, you may need the current value of the shipped or unshipped line items.

Later sections show how you can define a schema object, called an object type, that serves as a template for all purchase order data in your applications. An object type specifies the elements, called attributes, that make up a structured data unit like a purchase order. Some attributes such as the list of line items, may be other structured data units. The object type also specifies the operations, called methods, you can perform on the data unit, such as determining the total value of a purchase order.

You can create purchase orders that match the template and store them in table columns, just as you would numbers or dates:

You can also store purchase orders in object tables, where each row of the table corresponds to a single purchase order and the table columns are the purchase order's attributes.

Since the logic of the purchase order's structure and behavior is in your schema, your applications don't need to know the details and don't have to keep up with most changes.

Oracle uses schema information about object types to achieve substantial transmission efficiencies. A client-side application can request a purchase order from the server and receive all the relevant data in a single transmission. The application can then, without knowing storage locations or implementation details, navigate among related data items without further transmissions from the server.

2. Multimedia Datatypes

Many efficiencies of database systems arise from their optimized management of basic datatypes like numbers, dates, and characters. Facilities exist for comparing values, determining their distributions, building efficient indexes, and performing other optimizations. Text, video, sound, graphics, and spatial data are examples of important business entities that don't fit neatly into those basic types. Oracle with the objects option supports modeling and implementation of the complex datatypes.

3. User-Defined Datatypes

The objects option adds two categories of user-defined datatypes: object types and collection types.

User-defined datatypes use the built-in datatypes and other user-defined datatypes as the building blocks for datatypes that model the structure and behavior of data in applications.

4. Object Types

Object types are abstractions of the real-world entities-for example, purchase orders-that application programs deal with. An object type is a schema object with three kinds of components:

- A name which serves to identify the object type uniquely within that schema.
- Attributes which model the structure and state of the real world entity.

Attributes are built-in types or other user-defined types.

- Methods which are functions or procedures written in PL/SQL and stored in the database, or written in a language like C and stored externally. Methods implement operations the application can perform on the real world entity.

An object type is a template. A structured data unit that matches the template is called an object.

a. Purchase Order Example

Here is an example of how you might define object types called `external_person`, `lineitem` and `purchase_order`.

The object types `external_person` and `lineitem` have attributes of built-in types. The object type `purchase_order` has a more complex structure, which closely matched the structure of real purchase orders.

The attributes of `purchase_order` are `id`, `contact`, and `lineitems`. The attribute `contact` is an object, and the attribute `lineitems` is a nested table.

```
CREATE TYPE external_person AS OBJECT (  
    name          VARCHAR2(30) ,  
    phone         VARCHAR2(20) ) ;  
  
CREATE TYPE lineitem AS OBJECT (  
    item_name     VARCHAR2(30) ,  
    quantity      NUMBER,  
    unit_price    * NUMBER(12,2) ) ;  
  
CREATE TYPE lineitem_table AS TABLE OF lineitem ;  
  
CREATE TYPE purchase_order AS OBJECT (  
    id            NUMBER,  
    contact       external_person,  
    lineitems     lineitem_table,  
  
    MEMBER FUNCTION  
    get_value     RETURN NUMBER) ;
```

This is a simplified example. It does not show how to specify the body of the method `get_value`. Nor does it show the full complexity of a real purchase order. See Oracle8 Server Application Developer's Guide for a complete purchase order example. An object type is a template. Defining it doesn't result in storage allocation. You can use `lineitem`, `external_person`, or `purchase_order` in SQL statements in most of the same places you can use types like `NUMBER` or `VARCHAR2`.

For example, you might define a relational table to keep track of your contacts:

```
CREATE TABLE contracts (  
  contact          external_person  
  date            DATE );
```

The `contact` table is a relational table with an object type defining one of its columns. Objects that occupy columns of relational tables are called column objects.

b. Methods

In the example, `purchase_order` has a method named `get_value`. Each purchase order object has its own `get_value` method. For example, if `x` and `y` are PL/SQL variables that hold purchase order objects and `w` and `z` are variables that hold numbers, the following statements can leave `w` and `z` with different values:

```
w = x.get_value();
```

```
z = y.get_value();
```


After those statements, `w` has the value of the purchase order referred to by variable `x`; `z` has the value of the purchase order referred to by variable `y`.

The term `x.get_value()` is an invocation of method `get_value`. Method definitions can include parameters, but `get_value` does not need them, because it finds all of its arguments among the attributes of the object to which its invocation is tied. That is, in the first of the sample statements, it computes its value using the attributes of purchase order `x`. In the second it computes its value using the attributes of purchase order `y`.

c. Object Type Constructor Methods

Every object type has a system-defined constructor method, that is, a method that makes a new object according to the object type's specification. The name of the constructor method is the name of the object type. Its parameters have the names and types of the object type's attributes. The constructor method is a function. It returns the new object as its value.

For example, the expression

```
purchase_order (
    1000376,
    external_person ("John Smith", "1-800-555-1212"),
    NULL )
```

Represents a purchase order object with the following attributes:

```
id          1000376

contactexternal_person ("John Smith","1-800-555-1212")

lineitems   NULL
```

The expression `external_person ("John Smith","1-800-555-1212")` is an invocation of the constructor function for the object type `external_person`. The object that it returns becomes the `contact` attribute of the purchase order.

d. Comparison Methods

Methods play a role in comparing objects. Oracle has facilities for comparing two data items of a given built-in type (for example, two numbers), and determining whether one is greater than, equal to, or less than the other. Oracle cannot, however, compare two items of an arbitrary user-defined type without further guidance from the definer. Oracle provides two ways to define an order relationship among objects of a given object type: map methods and order methods.

Map methods use Oracle's ability to compare built-in types. Suppose, for example, that you have defined an object type called `rectangle`, with attributes `height` and `width`. You can define a map method `area` that returns a number, namely the product of the rectangle's `height` and `width` attributes. Oracle can then compare two rectangles by comparing their areas.

Order methods are more general. An order method uses its own internal logic to compare two objects of a given object type. It returns a value that encodes the order

relationship. For example, it may return -1 if the first is small, 0 if they are equal, and 1 if the first is larger.

Suppose, for example, that you have defined an object type called address, with attributes street, city, state, and zip. The terms “greater than” and “less than” may have no meaning for addresses in your application, but you may need to perform complex computations to determine when two addresses are equal.

In defining an object type, you can specify either a map method or an order method for it, but not both. If an object type has no comparison method, Oracle cannot determine a greater than or less than relationship between two objects of that type. It can, however, attempt to determine whether two objects of the type are equal.

Oracle compares two objects of a type that lacks a comparison method by comparing corresponding attributes:

- If all the attributes are non-null and equal, Oracle reports that the objects are equal.
- If there is an attribute for which the two objects have unequal non-null values, Oracle reports them unequal.
- Otherwise, Oracle reports that the comparison is not available (null).

5. Object Tables

An object table is a special kind of table that holds objects and provides a relational view of the attributes of those objects.

For example, the following statement defines an object table for objects of the `external_person` type defined earlier:

```
CREATE TABLE external_person_t OF external_person;
```

Oracle allows you to view this table in two ways:

- A single column table in which each entry is an `external_person` object.
- A multi-column table in which each of the attributes of the object type `external_person`, namely `name` and `phone`, occupies a column.

For example, you can execute the following instructions:

```
INSERT INTO external_person_t VALUES (
```

```
    "John Smith",
```

```
    "1-800-555-1212");
```

```
SELECT VALUE(p) FROM external_person_t p
```

```
WHERE p.name = "John Smith";
```

The first instruction inserts a purchase order object into `external_person_t` as a multi-column table. The second selects from `external_person_t` as a single column table.

6. Row Objects and Column Objects

Objects that appear in object tables are called row objects. Objects that appear only in table columns or as attributes of other objects are called column objects.

7. REFs

In the relational model, foreign keys express many-to-one relationships. Oracle with the objects option provides a more efficient means of expressing many-to-one relationships when the “one” side of the relationship is a row object. Oracle gives every row object a unique, immutable identifier, called an object identifier. Oracle provides no documentation of or access to the internal structure of object identifiers. This structure can change at any time.

An object identifier allows the corresponding row object to be referred to from other objects or from relational tables. A built-in data type called REF represents such references. A REF encapsulates a reference to a row object of a specified object type.

An object view is a virtual object table. Its rows are row objects. Oracle materializes object identifiers, which it does not store persistently, from primary keys in

the underlying table or view. Oracle uses these identifiers to construct REFs to the row objects in the object view.

You can use a REF to examine or update the object it refers to. You can also use a REF to obtain a copy of the object it refers to. The only changes you can make to a REF are to replace its contents with a reference to a different object of the same object type or to assign it a null value.

a. Scoped REFs

In declaring a column type, collection element, or object type attribute to be a REF, you can constrain it to contain only references to a specified object table. Such a REF is called a scoped REF. Scoped REFs require less storage space and allow more efficient access than unscoped REFs.

b. Dangling REFs*

It is possible for the object identified by a REF to become unavailable-through either deletion of the object or a change in privileges. Such a REF is called dangling. Oracle SQL provides a predicate (called IS DANGLING) to allow testing REFs for this condition.

c. Dereferencing REFs

Accessing the object referred to by a REF is called dereferencing the REF. Oracle provides the Deref operator to do this. Dereferencing a dangling REF results in a null object.

Oracle provides implicit dereferencing of REFs. For example, consider the following:

```
CREATE TYPE person AS OBJECT (  
    name          VARCHAR2 (30),  
    manager       REF person );
```

If x represents an object of type person, then the expression

x.manager.name

Represents a string containing the name attribute of the person object referred to by the manager attribute of x. The above expression is a shortened form of:

y.name, where y = Deref(x.manager)

d. Obtaining REFs*

You can obtain a REF to a row object by selecting the object from its object table and applying the REF operator. For example, you can obtain a REF to the purchase order with identification number 1000376 as follows:

```
DECLARE OrderRef REF to purchase_order ;  
  
SELECT REF(po)  
  
INTO      OrderRef  
  
FROM      purchase_order_table po  
  
WHERE     po.id = 1000376 ;
```

8. Collection Types

The collection types are array types and table types. Each describe a data unit made up of indefinite number of elements, all of the same datatype. Array types and table types are schema objects. The corresponding data units are called VARRAYs and nested tables. When there is no danger of confusion, we often avoid circumlocution by referring to the collection types as VARRAYs and nested tables.

Collection types have constructor methods. The name of the constructor method is the name of the type, and its argument is a comma-separated list of the new collection's elements. The constructor method is a function. It returns the new collection as its value.

An expression consisting of the type name followed by empty parentheses represents a call to the constructor method to create an empty collection of that type. An empty collection is different from a null.

9. VARRAYs

An array is an ordered set of data elements. All elements of a given array are of the same datatype. Each element has an index, which is a number corresponding to the element's position in the array.

The number of element in an array is the size of the way. Oracle allows arrays to be of variable size, which is why they are called VARRAYs. You must specify a maximum size when you declare the array type.

For example, the following statement declares an array type:

```
CREATE TYPE prices AS VARRAY (10) OF NUMBER (12,2)
```

VARRAYs of type prices have no more than ten elements, each of datatype NUMBER(12,2).

Creating an array type does not allocate space. It defines a datatype, which you can use as:

- The datatype of a column of a relational table.
- An object type attribute.
- A PL/SQL variable, parameter, or function return type.

A VARRAY is normally stored in line, that is, in the same tablespace as the other data in its row. If it is sufficiently large, however, Oracle stores it as a BLOB.

10. Nested Tables

A nested table is an unordered set of data elements, all of the same datatype. It has a single column, and the type of that column is a built-in type or an object type. If

an object type, the table can also be viewed as a multi-column table, with a column for each attribute of the object type.

For example, in the purchase order example, the following statement declares the table type used for the nested tables of line items:

```
CREATE TYPE lineitem_table AS TABLE OF lineitem;
```

A table type definition does not allocate space. It defines a type, which you can use as:

- The datatype of a column of a relational table.
- An object type attribute.
- A PL/SQL variable, parameter, or function return type.

When a table type appears as the type of a column in a relational table or as attribute of the underlying object type of an object table, Oracle stores all of the nested table data in a single table, which it associates with the enclosing relational or object table.

For example, the following statement defines an object table for the object type `purchase_order`:

```
CREATE TABLE purchase_order_table OF purchase_order  
NATED TABLE lineitems STORE AS lineitems_table;
```

The second line specifies `linesitems_table` as the storage table for the `lineitems` attributes of all the `purchase_order` objects in `purchase_order_table`. A convenient way to access the elements of a nested table individually is to use a nested cursor.

J. Application Interfaces

Oracle provides a number of facilities for using user-defined datatypes in application programs.

1. SQL

Oracle SQL DDL provides the following support for user-defined datatypes:

- Defining object types, nested tables, and arrays.
- Specifying privileges.
- Specifying table columns of user-defined types.
- Creating object tables.

Oracle SQL DML provides the following support for user-defined datatypes:

- Querying and updating objects and collections.
- Manipulating REFs.

2. PL/SQL

PL/SQL is a procedural language that extends SQL. It offers modern software engineering features like packages, data encapsulation, information hiding, overloading, and exception handling. It is the language used for stored procedures.

PL/SQL allows use from within functions and procedures of the SQL features that support user-defined types.

The parameters and variables of PL/SQL functions and procedures can be of user-defined types.

PL/SQL provides all the capabilities necessary to implement the methods associated with object types. These methods (functions and procedures) reside on the server as part of a user's schema.

3. Pro*C/C++

The Oracle Pro*C/C++ precompiler allows programmers to use user-defined datatypes in C and C++ programs. Pro*C developers can use the Object Type Translator to map Oracle object types and collections into C datatypes to be used in the Pro*C application.

Pro*C provides compile time type checking of object types and collections and automatic type conversion from database types to C datatypes.

Pro*C includes an EXEC SQL syntax to create and destroy objects and offers two ways to access objects in the server:

- SQL statements and PL/SQL functions or procedures embedded in Pro*C programs.
- A simple interface to the object cache where objects can be accessed by traversing pointers, then modified and updated on the server.

4. OCI

The Oracle call interface (OCI) is a set of C language interfaces to the Oracle server. It provides programmers great flexibility in using the server's capabilities.

An important component of OCI is a set of calls to allow application programs to use a workspace called the object cache. The object cache is a memory block on the client side that allows programs to store entire objects and to navigate among them without round trips to the server.

The object cache is completely under the control and management of the application programs using it. The Oracle server has no access to it. The application programs using it must maintain data coherency with the server and protect the work space against simultaneous conflicting access.

OCI provides functions to:

- Access objects on the server using SQL.
- Access, manipulate and manage objects in the object cache by traversing pointers or REFs.
- Convert Oracle dates, strings and numbers to C data types.
- Manage the size of the object cache's memory.

OCI improves concurrency by allowing individual objects to be locked. It improves performance by supporting complex object retrieval. OCI developers can use the object type translator to generate the C datatypes corresponding to an Oracle object types.

5. OTT

The Oracle type translator (OTT) is a program that automatically generates C language structure declarations corresponding to object types. OTT facilitates using the Pro*C precompiler and the OCI server access package.

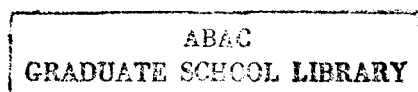
K. Conclusion

The Oracle8 object-relational database provides the most comprehensive solution to meet customer needs. In deciding to develop an object-relational database, Oracle identified a clearly defined set of business needs and application development requirements among mainstream customers.

Oracle chose an evolutionary, comprehensive, integrated, open, and pragmatic strategy to address these needs by evolving the Oracle7 universal data server. The resulting product, the Oracle8 object-relational database server, defines the next generation of relational database technology.

Oracle8 lets developers manage complexity in building mission-critical applications, provides a consistent way to manage and manipulate any data in a single database, such as: relational data, structured data representing business objects, and unstructured multimedia data, and provides a robust and highly scalable deployment platform.

Oracle's carefully considered evolutionary object strategy provides the best and most comprehensive solution to customer needs and leverages customer investment in relational technology and existing Oracle products.



BIBLIOGRAPHY

1. Baum, David. "Empower IT: KEEPING UP WITH CUSTOMER NEEDS," Oracle Magazine Vol.XII No. 6. Oracle Corporation, Redwood Shores, CA, 1998.
2. Brown, W. Alan. 'OBJECT-ORIENTED DATABASES Applications in Software Engineering'. The McGRAW-HILL International Series in Software Engineering.
3. Candace, C. Fleming, Barbara von Halle. 'Handbook of Relational Database Design'. Addison-Wesley Publishing Company.
4. Kramer, I. Mitchell. 'Network Computing Architecture: Oracle's Infrastructure for Distributed Applications'. Patricia Seybold Group, Boston, MA, November 1996.
5. Martin, James, and James J. Odell, OBJECT-ORIENTED METHODS: A Fundamental, Prentice Hall, Englewood Cliffs, NJ, 1997.
6. Wiseth, Kelli. "NEXT-GENERATION WEB," Oracle Magazine Vol.XII No. 6. Oracle Corporation, Redwood Shores, CA, 1998.

