1

Basic Concepts

Physical concepts are free creations of the human mind, and are not, however it may seem, uniquely determined by the external world.
—ALBERT EINSTEIN, The Evolution of Physics

Before we begin our exposition of Oracle performance and scalability, we need to consider a few preliminaries. This would include a brief introduction to what SQL is, and then a comparison between relational and object-oriented databases. We'll also clarify the differences between the concept of an Oracle *instance* and that of an Oracle *database* (these two concepts will come up frequently throughout this text). This is a necessary preparation before we take off on optimizing and tuning Oracle performance and scalability.

To be specific, this chapter consists of the following main sections:

- Standard versus Flavored SQLs
- Relational versus Object-Oriented Databases
- An Instance versus a Database

Let's start with a brief overview of standard SQLs versus flavored SQLs next.

Oracle Database Performance and Scalability: A Quantitative Approach, First Edition. Henry H. Liu.

1.1 STANDARD VERSUS FLAVORED SQLS

SQL stands for Structured Query Language, which is pronounced as "sequel" or "ess cue ell." It was said that at Oracle the former has been the norm. SQL was originally invented in 1974 by Donald Chamberlin and Raymond Boyce at IBM. The concept of SQL was further refined by Edgar F. Codd in his influential paper "*A Relational Model of Data for Large Shared Data Banks*," published in 1970. SQL is a language for querying data, which is structured in relations, with both data and relations stored in a data store or database. SQL was later taken over and standardized by ANSI/ISO. The latest version of standard SQL is SQL: 2008, which is version 6 since the first version of SQL-86 released in 1986.

Most college students as well as computer and software professionals have had some exposure to SQL, and some of them are experts in this area. Our purpose here is not to delve deeply into SQL, but rather to review the components of SQLs divided into a series of subsets as described below:

- *DML (Data Manipulation Language) SQLs*. This subset of SQLs includes the SQL statements of SELECT, INSERT, UPDATE, DELETE, MERGE, and so on. Such SQLs allow users to query and manipulate data stored in a database.
- *DDL (Data Definition Language) SQLs*. This subset of SQLs includes the SQL statements of CREATE, ALTER, DROP, TRUNCATE, and so on. Such SQLs are used to create and modify tables, views, and indexes, and so on.
- *DCL (Data Control Language) SQLs*. This subset of SQLs includes GRANT, REVOKE, and so on. Such SQLs are used for controlling user access to data in a database, for example, granting/revoking certain privileges to/from certain users.
- *TCL (Transaction Control Language)*. This subset of SQLs includes COMMIT, ROLLBACK, SET TRANSACTION, and so on. Such SQLs are useful for defining and manipulating database transactions.

Since a SQL standard is a common specification, it's subject to implementations by any interested parties. That leads to various flavors of SQL database products such as IBM's DB2, Microsoft's SQL Server, Oracle's Oracle, and the open source MySQL, PostgreSQL, and so on. MySQL was acquired by Oracle in 2009 as part of the Sun Microsystems acquisition.

Each of those products mentioned above has its own specific procedural language for writing programs and scripts that can run on its database server, for example:

- IBM's SQL is termed SQL PL
- Microsoft's SQL is termed T-SQL
- MySQL's SQL is termed MySQL
- Oracle's SQL is termed: PL/SQL
- PostgreSQL's SQL is termed PL/pgSQL

Mostly, those various flavors of SQLs differ in data types, especially in time and date, as well as in schemas and stored procedures. One needs to learn the proper SQL with a given database product.

Next, let's briefly discuss the subject of relational database management systems (RDBMS) versus object-oriented database management systems (ODBMS).

1.2 RELATIONAL VERSUS OBJECT-ORIENTED DATABASES

Real database-centric enterprise applications are rarely coded in SQL directly or entirely. Instead, they are coded in object-oriented programming languages such as C++, Java, or Microsoft C#, and so on. This has created a disparity between the language used for coding object-oriented application logic and SQL for operating on relational data in a relational database. Thus, the need for storing objects rather than relational tables in a database arose accordingly. It is believed that by supporting data as objects in a database, the overhead of converting between objects and relations can be avoided, resulting in higher development efficiency and better performance as well.

Most major database products, including Oracle, started supporting objects a few years ago. However, it's beyond the scope of this text at this time. We will concentrate on the relational side of Oracle only, mainly because the relational model will remain the mainstream for many years to come. Standard technologies such as Hibernate in the Java camp exist today to take care of object to relational table mapping. Besides, most application development frameworks support issuing SQLs directly to relational databases using technologies such as JDBC (Java database connectivity), which has proven to be effective in alleviating the object-relational gap.

Next, let's clarify the distinctions between an instance and a database in Oracle's context.

1.3 AN INSTANCE VERSUS A DATABASE

Conceptually, an Oracle `database` and an Oracle `instance` are two different, complementary entities. An Oracle Server consists of an Oracle instance and an Oracle database. An Oracle `database` is a logical entity from the data perspective. For example, the constituents of a database include schemas, tables, indexes, views, triggers, stored procedures, dictionaries, as well as users, and so on. Nevertheless, an Oracle `instance` is more of a physical entity from the system resource perspective with such constituents as processes that perform various tasks, memory areas that hold various types of data, and data files residing on physical disks, etc. An instance can operate on one database only, whereas a database can be operated upon by more than one instance in a clustered environment for high-availability. We will elaborate more on the concepts of `database` and `instance` later when we discuss Oracle architecture.

To help you get to know about Oracle quickly, in the next few chapters, I'll walk you through on how to set up and get around an Oracle database server using the latest version of Oracle 11g. Then I'll guide you through a tour of an Oracle Server to help you learn various basic concepts and building blocks of an Oracle Server. There is no better way in learning a software product than actually getting your hands dirty and experimenting with the product with the guidance of the well-written product documents or a more pertinent text such as this book.

1.4 SUMMARY

This chapter briefly introduced some basic concepts such as standard versus flavored SQLs, relational versus object-oriented databases, and an instance versus a database in Oracle's context. The purpose is to help you see the *forests* before seeing the *trees* to which the remainder of this book will be devoted. If you are interested in knowing more about those subjects, refer to the recommended sources listed next.

RECOMMENDED READING

The ISO's Web site (http://www.iso.org) has the following SQL-related documents available, which are not free, however:

- SQL - Part 1: Framework (SQL/Framework)
- SQL - Part 2: Foundation (SQL/Foundation)
- SQL - Part 3: Call-Level Interface (SQL/CLI)
- SQL - Part 4: Persistent Stored Modules (SQL/PSM)
- SQL multimedia and application packages - Part 5: Still image
- SQL multimedia and application packages - Part 6: Data mining
- SQL - Part 9: Management of External Data (SQL/MED)
- SQL - Part 10: Object Language Bindings (SQL/OLB)
- SQL - Part 11: Information and Definition Schemas (SQL/Schemata)
- SQL - Part 13: SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)
- SQL - Part 14: XML-Related Specifications (SQL/XML)

Since Oracle's object-oriented features are far less widely used than its relational features, there are not many texts about them. If you are interested in learning more about object-oriented features of Oracle, refer to the following text:

W. Rahayu, *Object-Oriented Oracle*, IRM Press, Hershey, 2005.

EXERCISES

1.1 If you happen to have had experiences with all or some of those major database products, which product will you recommend for a new enterprise application that is to be settled down on a specific database product? Justify.

1.2 In general, which type of database will you be mostly likely to recommend for a new enterprise application: object-oriented or relational? Justify.

1.3 What are the criteria for distinguishing between physical and logical entities? Use examples to explain.

1.4 What's the major difference between an Oracle *instance* and an Oracle *database* conceptually?