Developing an Oracle Application

Client/Server Model

- In a traditional client/server program, the code of your application runs on a machine other than the database server.
- Database calls are transmitted from this client machine to the database server. Data is transmitted from the client to the server for insert and update operations, and returned from the server to the client for query operations. The data is processed on the client machine.
- Client/server programs are typically written using precompilers, where SQL statements are embedded within the code of another language such as C, C++, or COBOL.

Two-Tier Versus Three-Tier Models

- Client/server computing is often referred to as a **two-tier model**: your application communicates directly with the database server.
- In the **three-tier model**, another server (known as the **application server**) processes the requests. The application server might be a basic web server, or might perform advanced functions like caching and load-balancing.
- Increasing the processing power of this middle tier lets you lessen the resources needed by client systems, resulting in a **thin client** configuration where the client machine might need only a web browser or other means of sending requests over the TCP/IP or HTTP protocols.

User Interface

- The interface that your application displays to end users depends on the technology behind the application, as well as the needs of the users themselves.
- Experienced users might enter SQL commands that are passed on to the database. Novice users might be shown a graphical user interface that uses the graphics libraries of the client system (such as Windows or X-Windows).
- Any of these traditional user interfaces can also be provided in a web using HTML and Java.

Overview of PL/SQL

- PL/SQL is Oracle's procedural extension to SQL, the standard database access language.
- An advanced 4GL, PL/SQL offers seamless SQL access, tight integration with the Oracle server and tools,portability, security, and modern software engineering features such as data encapsulation, overloading, exception handling, and information hiding.
- With PL/SQL, you can manipulate data with SQL statements, and control program flow with procedural constructs such as IF-THEN and LOOP.
- You can also declare constants and variables, define procedures and functions, use collections and object types, and trap run-time errors.

Overview of PL/SQL

- Applications written using any of the Oracle programmatic interfaces can call PL/SQL stored procedures and send blocks of PL/SQL code to the server for execution.
- 3GL applications can access PL/SQL scalar and composite datatypes through host variables and implicit datatype conversion.
- Because it runs inside the database, PL/SQL code is very efficient for data-intensive operations, and minimizes network traffic in client/server applications.
- PL/SQL's tight integration with Oracle Developer lets you develop the client and server components of your application in the same language, then partition the components for optimal performance and scalability.

Built-In Packages for Application Development

- DBMS_PIPE is used to communicate between sessions.
- DBMS_ALERT is used to broadcast alerts to users.
- DBMS_LOCK and DBMS_TRANSACTION are used for lock and transaction management.
- DBMS_AQ is used for Advanced Queuing.
- DBMS_LOB is for your manipulation of large objects.
- DBMS_ROWID is used for employing ROWIDs.
- UTL_RAW is for the RAW facility.
- UTL_REF is for work with REFs.

Built-In Packages for Server Management

- DBMS_SESSION is for session management by DBAs.
- DBMS_SYSTEM is used to set events for debugging.
- DBMS_SPACE and DBMS_SHARED_POOL obtain space information and reserve shared pool resources.
- DBMS_JOB is used to schedule jobs in the server.

Overview of Java Stored Procedures, JDBC, and SQLJ

- Oracle9*i* embeds the OracleJVM, a J2SE 1.3-compliant JVM. Oracle can store Java classes and execute them inside the database, as stored procedures and triggers.
- These classes can manipulate data, but cannot display GUI elements such as AWT or Swing components.
- Running inside the database allows these Java classes to be called many times and manipulate large amounts of data, without the processing and network overhead that comes with running on the client machine.
- Oracle9*i* includes the core JDK libraries such as java.lang, java.io, and so on.
- Oracle9*i* supports client-side Java standards such as JDBC and SQLJ, and provides server-side JDBC and SQLJ drivers that allow data-intensive Java code to run within the database.

Overview of Writing Procedures and Functions in Java

- You write these named blocks and then define them using the loadjava command or SQL CREATE FUNCTION, CREATE PROCEDURE, or CREATE PACKAGE statements.
- These Java methods can accept arguments and are callable from:
  - SQL CALL statements.
  - Embedded SQL CALL statements.
  - PL/SQL blocks, subprograms and packages.
  - DML statements (INSERT, UPDATE, DELETE, and SELECT).
  - Oracle development tools such as OCI, Pro*C/C++ and Oracle Developer.
  - Oracle Java interfaces such as JDBC, SQLJ statements, CORBA, and Enterprise Java Beans.
  - Method calls from object types.

Overview of Writing Database Triggers in Java

- A database trigger is a stored procedure that Oracle invokes ("fires") automatically when certain events occur, for example, when a DML operation modifies a certain table.
- Triggers enforce business rules, prevent incorrect values from being stored, and reduce the need to perform checking and cleanup operations in each application.

Why Use Java for Stored Procedures and Triggers?

- Stored procedures and triggers are compiled once, are easy to use and maintain, and require less memory and computing overhead.
- Network bottlenecks are avoided, and response time is improved.
- Distributed applications are easier to build and use.
- Computation-bound procedures run faster in the server.
- Data access can be controlled by letting users only have stored procedures and triggers that execute with their definer's privileges instead of invoker's rights.
- PL/SQL and Java stored procedures can call each other.
- Java in the server follows the Java language specification and can use the SQLJ standard, so that non-Oracle databases are also supported.
- Stored procedures and triggers can be reused in different applications as well as different geographic sites.

Managing Schema Objects

Managing Tables: Creating Tables

```
CREATE TABLE Emp_tab (
Empno NUMBER(5) PRIMARY KEY,
Ename VARCHAR2(15) NOT NULL,
Job VARCHAR2(10),
Mgr NUMBER(5),
Hiredate DATE DEFAULT (sysdate),
Sal NUMBER(7,2),
Comm NUMBER(7,2),
Deptno NUMBER(3) NOT NULL,
CONSTRAINT dept_afkey REFERENCES Dept_tab(Deptno))
PCTFREE 10
PCTUSED 40
TABLESPACE users
STORAGE ( INITIAL 50K
NEXT 50K
MAXEXTENTS 10
PCTINCREASE 25 );
```

Managing Temporary Tables

- Oracle8*i* provides a special kind of table to hold temporary data. You specify whether the data is specific to a session or to a transaction.
- When the session or transaction finishes, the rows that it inserted are deleted.
- Multiple sessions or transactions can use the same temporary table, and each session or transaction only sees the rows that it created.

Managing Temporary Tables

- Temporary tables are useful any time you want to buffer a result set or construct a result set by running multiple DML operations.

Creating Temporary Tables

- You create a temporary table by using special ANSI keywords.
- You specify the data as **session-specific** by using the **ON COMMIT PRESERVE ROWS** keywords.
- You specify the data as **transaction-specific** by using the **ON COMMIT DELETE ROWS** keywords.
- **Creating a Session-Specific Temporary Table**

  CREATE GLOBAL TEMPORARY TABLE ...

  [ON COMMIT PRESERVE ROWS]

  Creating a Transaction-Specific Temporary Table

  CREATE GLOBAL TEMPORARY TABLE ...

  [ON COMMIT DELETE ROWS]

A Session-Specific Temporary Table

- The following statement creates a session-specific temporary table, FLIGHT_SCHEDULE, for use in an automated airline reservation scheduling system.
- Each client has its own session and can store temporary schedules. The temporary schedules are deleted at the end of the session.

```
CREATE GLOBAL TEMPORARY TABLE flight_schedule (
startdate DATE,
enddate DATE,
cost NUMBER)
ON COMMIT PRESERVE ROWS;
```