Single-user Database Systems vs. Multi-user Database Systems

Single-user database systems:
Desktop database management systems such as Microsoft Access, Foxpro, etc., are aimed toward single-user (although they may be shared by several users) database applications that usually are stored on a single user's desktop computer, or a client workstation.

Multi-user database systems:
A multi-user database is a database in which more than one user processes the database at the same time.

Client-Server Systems
Unlike mainframe systems, client-server computing involves multiple computers connected in a network. Some of the computers process application programs and are designated as clients. Another computer processes the database and is designated as the server. Theoretically, the client computers can be mainframes or microcomputers. Because of cost, however, in almost all cases the client computers are microcomputers. Similarly, any type of computer can be the server, but again, because of cost, the server is most often a microcomputer. The clients and server are generally connected using a local area network (LAN).

Client/server databases such as ORACLE split the DBMS and applications accessing the DBMS into a "process" running on the server and the applications running on the client. The client application sends data requests across the network.

How do single-user database system (e.g., Access) and client/server database system (e.g., Oracle) handle the followings differently?
- Data requests: data processing location, the size of network traffic
- Failure recovery: does it affect other clients? In client/server database, no clients lock records.
- Transaction processing
- Concurrency control (competing user transactions)

Because of above differences:
- Client/server databases are preferred for database applications that retrieve and manipulate small amounts of data from databases containing large numbers of records because they minimize network traffic and improve response times.
- Client/server database systems are essential for mission-critical applications because of their failure handling, recovery mechanisms, and transaction management control.

The Oracle Environment

- **SQL*Plus** for creating and testing command-line SQL queries used to create, update, and delete database tables, views, and sequences, and so on.

- **Developer/2000** (**Oracle Navigator** for creating new tables, **Oracle Browser** for creating query-by-example(QBE) and SQL queries, **Oracle Forms** for creating graphical forms and menus for user applications, **Oracle Reports** for creating reports, **Oracle Graphics** for creating graphics charts based on database data)

- **Designer/2000**

- **Enterprise Manager** for managing and tuning the database; the enterprise manager uses the following utilities:
  - **Security manager** for creating and managing user accounts
  - **Storage manager** for creating and managing tablespaces
  - **Instance manager** for starting, shutting down, and tuning the database

- **Oracle Webserver** for creating a WWW site that allows users to access Oracle databases and create dynamic Web pages.

In addition, Oracle sells its server technology with a number of add-on options that enhance the server capabilities that include:

Video, Spatial Data, OLAP, Advance Networking, Parallel Server, etc.

Logging into Oracle through Telneting to Titan

Oracle can run on several platforms. In this lab, we will use Oracle that runs on Titan (Unix system). From the lab computers (or from any computers that have a connection to Internet including ***home computers***) you Telnet to Titan and run SQL*Plus.

Logging into Titan and Oracle

1. Establish a ***Telnet*** connection to ***Titan*** using the following command:

   telnet titan

   You can type this command either at DOS prompt or Windows (e.g, Start in Taskbar → Run → Telnet titan)

2. Login into Titan by entering your *user-id* and *password*. This will put you at the UNIX command shell.
3. Type the following to set up your environment:

   source /usr/local/bin/*coraenv*

source is a Unix command to execute Unix script files. The script file, '*coraenv'* was written by Oracle which makes all the necessary environment settings for the SQL*Plus client to run.

If you don't want to type this 'source' command whenever you login into Oracle, you may want to put this command line in the '*.login*' or '*.cshrc*' file which is executed automatically when you login into titan Unix system.

4. Type the following command to start the SQL*Plus client:

   sqlplus

5. Now, Oracle system will prompt for your user-id and password. For this class, initially your Oracle user-id and password is the same as your Titan user-id and password.

6. The prompt should be now "SQL>". You are now in SQL*Plus and ready to enter Oracle commands (e.g., SQL programs or commands).

Logging out from the Oracle

When you are finished, be sure to log out by typing one of the following commands at the "SQL>" prompt:

'*quit*' or '*exit*'

Changing password

- To change your password for *Titan account*, you can use '*passwd*' command.
- To change your password for *Oracle account*, you can use the following Oracle command:

ALTER USER *your-user-name* IDENTIFIED BY *your-new-password*;

OR

If you are using the SQL*Plus 8.0, there is a ***PASSWORD*** command that works similarly to UNIX '*passwd*' command.

Logging into Oracle directly from MS Windows

In Windows in Lab.,

1) Click Start (Windows taskbar)
2) Click Program
3) Click Oracle
4) Click Oracle for Windows
5) Click SQL*Plus 3.3 or **8.0** (8.0 is preferred!)
6) Enter *your-user-name*, *password*, and ***CS*** as Database

This is the ***recommended way*** of logging into Oracle System because you can easily use Windows95's powerful **GUI** features such Oracle *forms, Reports*, *Graphics*, and other GUI tools supported through Developer/2000.

If you have a **Personal Oracle** at your home computer, and want to access to the Oracle in Titan, you may need to configure the **Oracle*Net** (Oracle's Network Support Tool) assuming you already installed this Oracle*Net. To configure your Oracle*Net,

In Windows at home or any remote place that has Oracle installed,

1) **Connect your computer to Internet** (e.g., using Titan Access)
2) Click Start (Windows taskbar)
3) Click Program
4) Click Oracle for Windows
5) Click **SQL Net Easy Configuration**
6) Select Add **Database Alias** and click Ok
7) Enter your Database Alias Name (let's say ***TitanDB***, but any name is ok. This is just an ***alias*** for the actual database in Titan, which is '***CS***') and click Ok
8) Select **TCP/IP** (this is the network protocol) and click Ok
9) Enter '**sql.ecs.fullerton.edu**' for the TCP/IP host name and Enter ***'CS'*** for the database instance name, and click Ok.
10) Click Yes
11) Select Exit SQL*Net Easy Configuration and click Ok
12) Click Ok for rest of the messages from the system

Now, you configured your system so that you can access the ***CS*** database in Titan. This configuration process is **necessary only once**.

To access the *CS* database in the Titan system directly from Windows at home,

13) Click Start (Windows taskbar)
14) Click Program
15) Click Oracle for Windows
16) Click SQL*Plus 3.3 or 8.0 (preferred)
17) Enter *your-user-name*, *password* for Titan account and the **Database Alias Name** you created when you configure your Oracle*Net (e.g., ***TitanDB***) as Database, not necessarily *CS*.

✍ This is a much easier way to access a database in remote site.

Note that once you created a remote database alias through Oracle*Net Easy Configuration tool, you can use this alias as an actual database name (or database instance). Therefore, whenever you use other Oracle tools like Developer/2000 (Oracle's GUI tool like Microsoft Access or SQL Server), you can use this alias to access the remote database, which is really nice.

SQL*Plus Editing Commands

How to enter commands at SQL*Plus

SQL*Plus provides a **line-editor** (not a screen editor). You can take as many lines as you want to enter a command. For example, if you type something and press ENTER key, you will be given a new line to continue the command.

You can **run** or **terminate the command** by entering one of the following:

- Press ENTER until you see 'SQL>' prompt, then type '**run**' or '/' command. The slash '/' is the same as 'run' command in Oracle.

- If you end your command with a semicolon '**;**' Oracle will execute the command in the buffer. Actually, the semicolon, ';' in Oracle is the statement separator like in C language.

✓ **Note** SQL Plus Command line uses an Edit buffer, which means that Oracle stores only **one** command in the edit buffer. Therefore, if you have many commands to be executed and want to keep all those commands, you need to store those commands in a separate text file (using any text editor such as 'vi' editor, Notepad, Wordpad, and so on).

✓ All the commands you entered are stored in the Edit buffer that is in your client machine. Run (or /, ;) command sends the buffer content to the Oracle server for processing.

Command Reference

Edit Commands

L [list]	Display the buffer content. The line marked with the asterisk * is the current line.
n	Display the command in the *n*th line (current line)
A[ppend] *text*	Append the *text* to the current line
C[hange] /oldtxt/newtxt/ (or **$**oldtxt**$**newtxt)	Change the text
DEL[ete]	Delete the current line
I	insert more lines after the current one
I(nsert) *text*	Insert the *text*
CL(ear) BUFF(er) (to clear buffer)	
EDIT filename	Edits the filename using the defined editor using the DEFINE command.

✓ Although you can use above Oracle editor commands, I also **encourage** you to use a regular text editor such as **NotePad**, WordPad, etc. and use the Windows **Copy and Paste**.

✓ **Save** the SQL programs you wrote to a **text file**. We call this text file SQL script file, e.g., test.sql (query script name 'test' with extension 'sql')

Run Commands

/ or **R**(UN)	Execute the contents of the edit buffer
START *filename*	Execute the commands stored in the *filename* like executing a batch file. The *filename* is usually SQL script file.

File I/O Commands

SAVE *filename*	Save the buffer to the specified *filename*.
GET *filename*	Read the *filename* into the buffer.
SPOOL *filename*	Spool output to the *filename*
SPOOL OFF	Ends spooling and closes the spool file. SPOOL and SPOOL OFF command is used as a pair.

Operating System Command

HOST or **!** *OS-command*	Execute the host *OS-command*

Logout

EXIT or **QUIT**	Log out from the Oracle

Environment Setting Commands

Set feedback on	Turn on the feedback
Set feedback off	Turn off the feedback
Set feedback 25	Feedback at least 25 rows
Set sqlprompt 'name'	Change the SQL prompt
Set sqlnumber on	
Set numwidth 5	
Set pagesize 24	
Set linesize 79	
Set pause on (or Set pause 'More..')	Makes screen display stop between pages
Define _editor="vi"	Set vi as the default editor

- ✓ If you logged into Oracle directly from Windows, the default editor is **Notepad**. You can use **EDIT** command to edit commands. You can also change some other edit options.

- ✓ You can put the environment setting commands in the above in a login script file called ***login.sql***, which will be executed whenever you login into Oracle, just like a Unix login shell script.

To see the environment settings, use SHOW command
e.g., show numwidth or show feedback, etc.

Creating and Modifying Database Tables

Creating a Database Table using SQL*Plus

Syntax:

```
CREATE TABLE tableName
(fieldName DataType [CONSTRAINT iConstraintName constraint_decl],
                    [CONSTRAINT vConstraintName constraint_decl], …);
```

optional

- Comma **,** is the field separator.
- Semicolon **;** is the SQL statement terminator.

✓ Refer the Oracle manual for the complete syntax of CREATE TABLE.

✓ Double dash -- **or REMARK** is the single line comment operators and /* */ pair is the multiple line comment operator.

Table Names and Properties

- Table and field names can be from 1 to 30 characters long (alphanumeric, $, _, #)
- Oracle database table specification:
  Table names, Field names, Data types, Sizes, Constraints (integrity constraints and value or domain constraints).

Data types

- Variable-length characters (up to 2,000 char), VARCHAR2: e.g., VARCHAR2(30)
- Fixed-length characters (up to 255 char): CHAR e.g., CHAR(2)
- Integer: e.g., NUMBER(5), or INTEGER (4 bytes), SMALLINT (2 bytes)
- Fixed-point number: e.g., NUMBER(5,2), or DECIMAL(5,2)
- Floating-point number: e.g., NUMBER
- Date: DATE, e.g., DD-MM-YY
- LONG (used to store large amount of variable-length character data)
- RAW and LONG RAW (used to store binary data, sound, images, etc.)

```
CREATE TABLE customer
(custid NUMBER(5), last VARCHAR2(30), first VARCHAR2(30), balance NUMBER);
```

Integrity constraints

Key constraints (primary key or candidate key), Referential integrity constraints (foreign key constraints), Domain constraints (or value constraints)

Primary key definition (specifying Primary Key constraint)

```
CREATE TABLE customer
(cid NUMBER(8) CONSTRAINT customer_cid_pk PRIMARY KEY,
 name VARCHAR2(20), address VARCHAR2(30));
```

Remark: the primary key field is assumed to be NOT NULL and UNIQUE

Composite primary key

```
CREATE TABLE works_on
(ssn CHAR(14),  pno CHAR(2), hours number(2),
 CONSTRAINT customer_cid_pk PRIMARY KEY (ssn, pno));          -- composite key
```

Candidate key definition

```
CREATE TABLE bank_account
(acc# CHAR(10) PRIMARY KEY, name VARCHAR2(20), address VARCHAR2(30),
 ssn CHAR(9) UNIQUE);              -- this is a candidate key
```

Composite candidate key

```
CREATE TABLE city_weather
(city VARCHAR2(13) NOT NULL,
sdate DATE NOT NULL, noon_temp NUMBER (4,1),
UNIQUE (city, sdate));                          /* composite candidate key */
```

Foreign key definition (Specifying Referential Integrity constraint)

```
CREATE TABLE cust_order
(ordid NUMBER(4) PRIMARY KEY, odate DATE,
custid NUMBER(8) CONSTRAINT cust_order_custid_fk REFERENCES customer(cid));
```

Composite foreign key

```
CREATE TABLE custwork
(ssn CHAR(14) NOT NULL,
pno NUMBER(4),
name VARCHAR2(30),
CONSTRAINT custwork_fk FOREIGN KEY(ssn, pno) REFERENCES works_on)
```

Remark: Before you define a foreign key, first you need to define the foreign key field as a primary key in other table!

Remark: Either one of the key fields should be NOT NULL but it can be NOT UNIQUE

Value (or domain) Constraints

- To restrict what data can be entered into a given field and to specify a default value.

```
CREATE TABLE customer
(cid NUMBER(5) CONSTRAINT customer_cid_pk PRIMARY KEY,
ccredit NUMBER(2) CONTRAINT customer_ccredit_cc
                                    CHECK ((ccredit > 0) AND (ccredit < 12));
```

Other value constraints: NOT NULL, UNIQUE, CHECK (condition), DEFAULT

Listing the tables you own (like DIR command in DOS or ls command in Unix)

SELECT *table_name* FROM *user_tables*; -- *table_name* and *user_tables* (KEY words) are system defined fields and tables

✓ *user_tables* is a Oracle dictionary that contains information about tables and *table_name* is the column name in the *user_tables* table.

Viewing Table structure information

DESC[RIBE] table_name;

e.g., DESC customer;

Viewing Constraint Information

SELECT *constraint_name* FROM *user_constraints* WHERE *table_name* = 'CUSTOMER';

Note that the upper-case table name stored in *user_constraints* table (Oracle store the meta information in upper-case).

✓ *User_constraints* is a Oracle dictionary table that contains the user constraint information and the *constraint_name* is a column name in the *user_constraints* table.

To see the information about column name and the corresponding column name and table names, query the *user_cons_column* dictionary table.

Modifying Tables using ALTER command and their Restrictions

- Adding a new field to a table
- Adding a primary key only **if** the field values are unique
- Adding UNIQUE or CHECK condition constraints only **if** the current field values satisfy it
- Adding a foreign key only **if** current field values are NULL or exist in the reference table
- Deleting a primary key constraint (also delete any foreign key references)
- Deleting a foreign key constraint
- Changing a field's data type, size, and default value only if no data in the column

✓ Changing/deleting a **field name (column name)** are NOT normally allowed!

Examples

- **Adding a new field**

ALTER TABLE table_name ADD (field_name type, …);

e.g., ALTER TABLE customer ADD (cname VARCHAR2(20));

- **Modifying a field**

ALTER TABLE table_name MODIFY (field_name new_declaration);

e.g., ALTER TABLE customer MODIFY (cname VARCHAR2(15));

- **Adding a constraint**

ALTER TABLE table_name ADD CONSTRAINT constraint_name constraint_declaration;

e.g., ALTER TABLE faculty ADD PRIMARY KEY (fid);

e.g., ALTER TABLE customer ADD FOREIGN KEY(ordid) REFERENCES corder(ordid);

e.g., ALTER TABLE orders ADD FOREIGN KEY(cust_num) REFERENCES customer;

e.g., ALTER TABLE faculty ADD CONSTRAINT faculty_frank_cc
CHECK ((frank = 'ASSO') OR (frank = 'FULL'));

e.g., ALTER TABLE part ADD CHECK (item_class IN ('AP', 'HW', 'SG'));

- To **change** an existing check, **drop** the check condition first and **add** the new condition

ALTER TABLE table_name DROP CONTRAINT constraint_name;
ALTER TABLE table_name ADD CONSTRAINT constraint_name new_check_condition

e.g., ALTER TABLE student DROP CONSTRAINT student_sclass_cc;
ALTER TABLE student ADD CONSTRAINT student_sclass_cc
CHECK ((sclass = 'FR') OR (sclass = 'SO'));
ALTER TABLE student MODIFY (age NULL);

- **Changing a default value**

ALTER TABLE table_name MODIFY (field_name field_type DEFAULT default_value);

e.g., ALTER TABLE student MODIFY (sclass CHAR(2) DEFAULT 'FR');

Removing/Renaming Tables

To remove a table
DROP TABLE table_name;

Note that in order to remove a table, you need to remove the foreign key constraints first. To remove the table regardless of constraints use this syntax:
DROP TABLE table_name [CASCADE CONSTRAINTS] -- this will delete records and
-- table definition

To rename a table
RENAME old_table TO new_table;

Oracle Navigator (GUI) to create and modify tables

- Creating Tables using Table wizard or manually
- Modifying Tables