## ***SQL/Oracle***

SQL stands for **Structured Query Language**. It was first introduced by IBM as early as 1973. Since then, SQL has undergone a number of changes and is now formally recognized as the standard relational database query language.

Oracle, on the other hand, is a **database management system** (DBMS). Users interact with a relational DBMS primarily through SQL. Without a DBMS, it would be very difficult to efficiently organize, store, secure, maintain, and query data, especially when supporting many concurrent users. Thus, a DBMS is a collection of programs to facilitate all of these functions (and more).

Besides Oracle, there are a number of different relational database management systems in the industry, including IBM's DB2, Microsoft's SQL Server, Sybase, Informix, and others. Microsoft's Access is another relational database system, but it is for smaller-sized applications, and is meant to be used by one user at a time.

Oracle comes with a number of products, a few of them being:

* *SQL\*Plus*

An interactive front-end program for entering SQL statements and Oracle commands. You will be using this for your homework.

* *SQL\*Graph*

A graphing utility which uses results based on SELECT statements

* *SQL\*Calc*

A spreadsheet, compatible with Lotus 1-2-3, which holds results of SELECT statements

* *Easy\*SQL*

A utility for inexperienced SQL users who do not know SQL syntax

* *SQL\*Forms*

A form generating utility for use with SQL applications

The most useful tool for our purposes would probably be *SQL\*Plus*, which is an interactive SQL command interpreter. With SQL\*Plus, you can enter an SQL statement at the terminal and immediately see the results of the command.

**SQL\*Plus**

SQL\*Plus accepts pure SQL statements (also called SQL commands, in some of the literature) and special SQL\*Plus commands. First, before giving any examples, we have to note the difference between SQL statements and SQL\*Plus commands. An SQL\*Plus command is a command which only SQL\*Plus understands. For example, the SQL\*Plus command DESCRIBE <table-name> outputs the schema of a table which was created using SQL statements. SQL\*Plus commands are NOT SQL statements. SQL statements are those which have been taught in class, and are understood by any DBMS which uses SQL. SQL\*Plus, on the other hand, is a tool which comes with Oracle. Each vendor has its own tools.

The other distinction between SQL\*Plus commands and SQL statements is that in SQL\*Plus, an SQL statement can span multiple lines, and needs to be followed by a semicolon, whereas an SQL\*Plus command does not need to be followed by a semicolon, and can only span multiple lines if each line is ended with the "-" character.

# **SQL\*Plus Basics**

Oracle's SQL\*Plus is a command line tool that allows a user to type SQL statements to be executed directly against an Oracle database. SQL\*Plus has the ability to format database output, save often used commands and can be invoked from other Oracle tools or from the operating system prompt.

In the following sections, the basic functionality of SQL\*Plus will be demonstrated along with sample input and output to demonstrate some of the many features of this product.

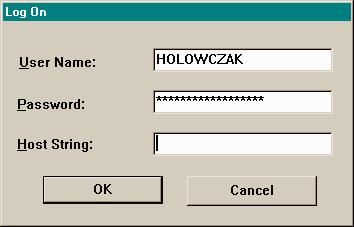
## ***3.1 Running SQL\*Plus***

In this section, we give some general directions on how to get into the SQL\*Plus program and connect to an Oracle database. Specific instructions for your installation may vary depending on the version of SQL\*Plus being used, whether or not SQL\*Net or Net8 is in use, etc.

Before using the SQL\*Plus tool or any other development tool or utility, the user must obtain an Oracle account for the DBMS. This account will include a username, a password and, optionally, a host string indicating the database to connect to. This information can typically be obtained from the database administrator.

The following directions apply to two commonly found installations: Windows 95/98 or NT client with an Oracle server, and a UNIX installation.

### **3.1.1 Running SQL\*Plus under Windows 95/98 and Windows NT**

To run the SQL\*Plus command line program from **Windows 95/98** or **Windows NT**, click on the [Start]button, Programs, Oracle for Windows 95 and then SQL\*Plus. The SQL\*Plus login screen will appear after roughly 15 seconds.  


In the **User Name:** field, type in your Oracle username.  
Press the TAB key to move to the next field.  
In the **Password:** field, type your Oracle password.  
Press the TAB key to move to the next field.  
In the **Host String:** field, type in the *Service Name* of the Oracle host to connect to. If the DBMS is Personal Oracle lite then this string might be ODBC:POLITE. If the DBMS is Personal Oracle8, then the host string might be beq-local. For Client/Server installations with SQL\*Net or Net8, this string will be the service name set up by the SQL\*Net assistant software.

Finally, click on the OK button to complete the Oracle log in process. SQL\*Plus will then establish a *session* with the Oracle DBMS and the SQL\*Plus prompt (SQL> ) will appear.

The following figure shows the results of logging into Oracle using SQL\*Plus:

Once a session has been established using the SQL\*Plus tool, any SQL statements or SQL\*Plus Commands may be issued. In the following section, the basic SQL\*Plus Commands are introduced.

**SQL\*Plus Commands**

SQL\*Plus commands allow a user to manipulate and submit SQL statements. Specifically, they enable a user to:

* Enter, edit, store, retrieve, and run SQL statements
* List the column definitions for any table
* Format, perform calculations on, store, and print query results in the form of reports
* Access and copy data between SQL databases

# **The SQL Language**

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

In the relational model, data is stored in structures called relations or *tables*. Each table has one or more attributes or *columns* that describe the table. In relational databases, the table is the fundamental building block of a database application. Tables are used to store data on Employees, Equipment, Materials, Warehouses, Purchase Orders, Customer Orders, etc. Columns in the Employee table, for example, might be Last Name, First Name, Salary, Hire Date, Social Security Number, etc.

SQL statements are issued for the purpose of:

* Data definition - Defining tables and structures in the database (DB).
* Data manipulation - Inserting new data, Updating existing data, Deleting existing data, and Querying the Database ( Retrieving existing data from the database).

Another way to say this is the SQL language is actually made up of 1) the Data Definition Language (DDL) used to create, alter and drop scema objects such as tables and indexes, and 2) The Data Manipulation Language (DML) used to manipulate the data within those schema objects.

## **Creating Tables**

You must create your tables before you can enter data into them. Use the Create Table command.

Syntax:

Create table tablename using filename  
(fieldname fieldtype(length),   
fieldname fieldtype(length),   
fieldname fieldtype(length));

Explanation:

* Table names cannot exceed 20 characters.
* Table names must be unique within a database.
* Field names must be unique within a table.
* You may specify the data file to use. If you do not specify a data file, Scalable SQL will create one, using a .dat extension.
* The list of fields must be enclosed in parentheses.
* You must specify the field type.

Examples:   
Char -- a character string    
Float -- a number    
Date -- a date field    
Logical -- a logical field

* You must specify the field length.
* The field length must be enclosed in parentheses.
* You must separate field definitions with commas.
* You must end each SQL statement with a semicolon.

## **Retrieving All Data**

Select statements are used to retrieve data from  SQL tables. The Select statement illustrated below retrieves all of the columns and rows  from the named table.

Syntax:

Select \* from tablename;

Explanation:

* A Select statement is a SQL statement that begins with the word "select."
* Select statements are used to retrieve data from SQL tables.
* An asterisk after the word "select" means retrieve all fields (columns).
* The name of the table from which you are retrieving data is specified in the From clause.
* Use a semicolon to signify the end of a SQL statement.

## **Retrieving a Single Column**

You can use SQL to retrieve a single column of data.

Syntax:

Select fieldname from tablename;

Explanation:

* Instead of an asterisk, the name of the field is specified in the Select clause.
* Field names are not case-sensitive. You can type field names in uppercase or lowercase letters.

## **Retrieving Multiple Columns**

You can use SQL to retrieve multiple columns.

Syntax:

Select fieldname1, fieldname2, fieldname3 from tablename;

Explanation:

* When retrieving multiple columns, specify each field name.
* A comma must separate field names.
* The columns will display in the order you select them.
* Numbers display in scientific notation.

## **Ordering Rows**

Data is stored in Scalable SQL in no particular sequence. If you want to see your data displayed in sequence, you must add an Order By clause to your Select statement.

Syntax:

Select fieldname from tablename order by fieldname;

Explanation:

* The Order By clause tells SQL you want the specified fields displayed in ascending order (ordered from A to Z, 1 to 100).

## **Displaying Rows in Descending Order**

If you would like to see fields displayed in descending order, follow the field name with "desc" in the Order By clause.

Syntax:

Select fieldname from tablename order by fieldname <desc> ...

Explanation:

* By default, the Order By clause tells SQL you want the field displayed in ascending order.
* Typing "desc" after the field name in the Order By clause tells SQL you want the data in the field displayed in descending order (Z to A, 100 to 1).

## **Ordering Multiple Columns**

When ordering your data, you can have multiple sort levels. For example, you can order your data by city and then by name within the city.

Syntax:

Select fieldname1, fieldname2, fieldname3 from tablename   
order by fieldname <desc>, fieldname <desc>, fieldname <desc>

Explanation:

* By default, the Order By clause orders the specified fields in ascending order.
* Typing "desc" after a field name in the Order By clause tells SQL you want the data in the specified field displayed in descending order (Z to A, 100 to 1).
* The first field name specified is the primary sort order, the second field name specified is the secondary sort order, and so on ...

**Using Distinct**

DISTINCT means unique. Using DISTINCT will filter out duplicates from your results table. If you wanted a list of all the individual titles without any repeating rows you would use DISTINCT to filter out the duplicates.

**SQL> SELECT DISTINCT JOB ←** DISTINCT verb

**FROM EMPLOYEE;**

## **Retrieving Specific Rows**

So far, you have been retrieving all of the rows in the table. You can, however, specify which rows you wish to retrieve. For example, you could retrieve only those vendors who are in Chicago.

Syntax:

Select fieldname from tablename   
where fieldname =/!=/<>/>/>=/</<=/in/not in/between/not between/begins with/contains/not contains/ is null/is not null/like/not/like value   
order by fieldname <desc>...

Explanation:

* You can use any of the following logical operators in your Where clause to restrict the rows you retrieve.

**Logical Operators**

= Equal to

!= or <> Not equal to

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

in Equal to any item in a list

not in  Not equal to any item in a list

between Between two values, greater than or equal to one and

less than or equal to the other

not between Not between two values

begins with Begins with specified value

contains Contains specified value

not contains Does not contain specified value

is null Is blank

is not null Is not blank

like Like a specified pattern.   
% means any series of characters.   
\_ means any single character.

* In the Where clause, when referring to variables in character fields, you must enclose the values in single quotes.

Example:   
where City = 'Chicago'

* Variables that refer to numeric fields should not be enclosed in quotes.

Example:   
where CurrBal > 1200

## **Multiple Conditions**

You can add multiple criteria to your Where clauses by using "and" or "or."

Syntax:

Select fieldname from tablename where fieldname =/!=/<> ... value   
and/or   
fieldname =/!=/<>... value   
and/or   
fieldname =/!=/<> ... value    
order by fieldname <desc> ...

Explanation:

* The and tells SQL to retrieve the record if both conditions are met.
* The or tells SQL to retrieve the record if either condition is met.
* The or is less restrictive and retrieves more records.
* If multiple ands and ors are used, the ands are evaluated first, and then the ors.
* Use parentheses to change precedence (the order of evaluation

## **Updating Tables**

So far, you have looked at several different ways to retrieve and review your data. In this section, you will learn how to update your data. In the following two sections, you will learn about deleting and inserting rows. When you update, delete, and insert, you change the data -- you should perform these operations very cautiously. Before performing any of these operations on a production database, make sure your data is backed up and use the Start Transaction command. If you use the Start Transaction command, all of your changes are temporary until you commit your work and can be rolled back . If you have issued the Start Transaction command, you can undo your changes simply by typing "rollback work."

NOTE: The exercises that follow should not be performed on a production database. Use a test or trial database.

Syntax:

Start transaction;

Update tablename   
set fieldname = value   
where fieldname = value;

Rollback work;

Commit work;

Explanation:

* Issue a Start Transaction command before updating your table. This will allow you to roll back the changes, if necessary. If you do not issue a Start Transaction command, you will not be able the roll back your work.
* If you find that you have updated a row in error, execute the Rollback Work command.
* When you are satisfied with your changes, issue the Commit Work command.
* Use a Where clause to specify which rows will be updated. If you do not include a Where clause, all rows will be updated.
* Remember to end each command with a semicolon.

## **Deleting Rows**

You can use Scalable SQL to delete rows of data.

Syntax:

Delete from tablename   
where fieldname =/<>/ ... value   
and/or ...    
fieldname =/<>/ ... value   
and/or   
fieldname =/<>/ ... value

### **Oracle SQL Functions**

The Oracle implementation of SQL provides a number of functions that can be used in SELECT statements. Functions are typically grouped into the following:

* **Single row functions** - Operate on column values for each row returned by a query.
* **Group functions** - Operate on a collection (group) of rows.

The following is an overview and brief description of **single row functions.** *x* is some number, *s* is a string of characters and *c* is a single character.

* Math functions include:  
  **ABS (*x*)** - Absolute Value of *x*  
  **CEIL (*x*)** - Smallest integer greater than or equal to *x*. **COS (*x*)** - Cosine of *x*  
  **FLOOR (*x*)** - Largest integer less than or equal to *x*. **LOG (*x*)** - Log of *x*  
  **LN (*x*)** - Natural Log of *x*  
  **ROUND (*x*, *n*)** - Round *x* to *n* decimal places to the right of the decimal point.  
  **SIN (*x*)** - Sine of *x*  
  **TAN (*x*)** - Tangent of *x*  
  **TRUNC (*x*, *n*)** - Truncate *x* to *n* decimal places to the right of the decimal point.
* Character functions include:  
  **CHR (*x*)** - Character for ASCII value *x*.  
  **INITCAP (*s*)** - String *s* with the first letter of each word capitalized.  
  **LOWER (*s*)** - Converts string *s* to all lower case letters.  
  **LPAD (*s*, *x*)** - Pads string *s* with *x* spaces to the left.   
  **LTRIM (*s*)** - Removes leading spaces from *s*.  
  **REPLACE (*s1*, *s2*, *s3*)** - Replace occurrences of *s1* with *s2* in string *s*.  
  **RPAD (*s*, *x*)** - Pads string *s* with *x* spaces to the right.   
  **RTRIM (*s*)** - Removes trailing spaces from *s*.  
  **SUBSTR (*s*, *x1*, *x2*)** - Return a portion of string *s* starting at position *x1* and ending with position *x2*. If *x2* is omitted, it's value defaults to the end of *s*.  
  **UPPER (*s*)** - Converts string *s* to all upper case letters.
* Character functions that return numbers include:  
  **ASCII (*c*)** - Returns the ASCII value of *c*  
  **INSTR (*s1*, *s2*, *x*)** - Returns the position of *s2* in *s1* where the search starts at position *x*.   
  **LENGTH (*s*)** - Length of *s*
* Conversion functions include:  
  **TO\_CHAR (*date*, *format*)** - Converts a date column to a string of characters. *format* is a set of Date formatting codes where:  
  YYYY is a 4 digit year.  
  NM is a month number.  
  MONTH is the full name of the month.  
  MON is the abbreviated month.  
  DDD is the day of the year.  
  DD is the day of the month.  
  D is the day of the week.  
  DAY is the name of the day.  
  HH is the hour of the day (12 hour clock)  
  HH24 is the hour of the day (24 hour clock)  
  MI is the minutes.  
  SS is the seconds.

**TO\_CHAR (*number*, *format*)** - Converts a numeric column to a string of characters. *format* is a set of number formatting codes where:  
9 indicates a digit position. Blank if position value is 0.  
0 indicates a digit position. Shows a 0 if the position value is 0.  
$ displays a leading currency indicator.  
**TO\_DATE (*s*, *format*)** - Converts a character column (string *s* to a date. *format* is a set of Date formatting codes as above.  
**TO\_NUMBER (*s*, *format*)** - Converts a character column (string *s* to a Number. *format* is a set of Number formatting codes as above.

* Date functions include:  
  **SYSDATE** - Returns the current date (and time if the TO\_CHAR function is used) from the system clock.

Some additional function are:   
 The following is an overview and brief description of **multiple row (group) functions.** *col* is the name of a table column (or expression) of type NUMBER.

* **AVG (*col*)** - Returns the average of a group of rows for *col*
* **MAX (*col*)** - Returns the maximum of a group of rows for *col*
* **MIN (*col*)** - Returns the minimum of a group of rows for *col*
* **STDEV (*col*)** - Returns the standard deviation of a group of rows for *col*
* **SUM (*col*)** - Returns the sum (total) of a group of rows for *col*
* **VARIANCE (*col*)** - Returns the variance of a group of rows for *col*

In addition the COUNT group function counts instances of values. These values can be any type (CHAR, DATE or NUMBER):

* **COUNT (*columns*)** - Returns the number of instances of a group of rows for (*columns*)

To use an aggregate function, a GROUP BY clause must be added to the SELECT statement.

## ***The LIKE Condition***

The LIKE condition is used to specify a search for a pattern in a column.

### **Syntax**

SELECT column FROM table

WHERE column LIKE pattern

A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

## ***Using LIKE***

The following SQL statement will return persons with first names that start with an 'O':

SELECT \* FROM Persons

WHERE FirstName LIKE 'O%'

The following SQL statement will return persons with first names that end with an 'a':

SELECT \* FROM Persons

WHERE FirstName LIKE '%a'

The following SQL statement will return persons with first names that contain the pattern 'la':

SELECT \* FROM Persons

WHERE FirstName LIKE '%la%'

**Constraints**

Data integrity ensures the consistency and correctness of data stored in a database. Such integrity can be enforced by incorporating business rules. *Constraints* are the rules that are enforced on data stored in a table.Constraints are used to do the following:

* Enforce rules on the data in a table whenever a row is updated, inserted, or deleted from that table
* Prevent the deletion of a table if there are dependencies from other tables

**Constraints can be enforced at two levels**:

* Column level
* Table level

**A constraint can be created with either of the following statements**:

* CREATE TABLE
* ALTER TABLE

With the ALTER TABLE statement, you can disable or enable the imposed constraint without dropping it or re-creating it:

* Disable a constraint by using the DISABLE clause.
* Enable a constraint by using the ENABLE clause.

**Types of Constraints**:

Several types of Oracle constraints can be applied to Oracle tables to enforce data integrity, including:

* **Oracle "Check" Constraint:**This constraint validates incoming columns at row insert time.
* **Not Null Constraint:**This Oracle constraint is used to specify that a column may never contain a NULL value. This is enforced at SQL *insert* and *update* time.
* **Primary Key Constraint:** This Oracle constraint is used to identify the primary key for a table. This operation requires that the primary columns are unique, and this Oracle constraint will create a unique index on the target primary key.
* **References Constraint:** This is the foreign key constraint as implemented by Oracle. A references constraint is only applied at SQL *insert* and *delete* times.  At SQL *delete* time, the references Oracle constraint can be used to ensure that an employee is not deleted, if rows still exist in the DEPENDENT table.
* **Unique Constraint:** This Oracle constraint is used to ensure that all column values within a table never contain a duplicate entry.