**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=1)**] What is SQL and where does it come from?**

Structured Query Language (SQL) is a language that provides an interface to relational database systems. The proper pronunciation of SQL, and the preferred pronunciation within Oracle Corp, is "sequel" and not "ess cue ell".

SQL was developed by IBM in the 1970s for use in System R, and is a de facto standard, as well as an ISO and ANSI standard.

In common usage SQL also encompasses DML (Data Manipulation Language), for INSERTs, UPDATEs, DELETEs and DDL (Data Definition Language), used for creating and modifying tables and other database structures.

The development of SQL is governed by standards. A major revision to the SQL standard was completed in 1992, called SQL2. SQL3 support object extensions and are (partially?) implemented in Oracle8 and 9i.

Example SQL statements:

CREATE TABLE table1 (column1 NUMBER, column2 VARCHAR2(30));

INSERT INTO table1 VALUES (1, 'XYZ');

SELECT \* FROM table1 WHERE column2 = 'XYZ';

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=2)**] What are the difference between DDL, DML and DCL commands?**

**DDL** - Data Definition Language: statements used to define the database structure or schema. Some examples:

* CREATE - to create objects in the database
* ALTER - alters the structure of the database
* DROP - delete objects from the database
* TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
* COMMENT - add comments to the data dictionary
* RENAME - rename an object

**DML** - Data Manipulation Language: statements used for managing data within schema objects. Some examples:

* SELECT - retrieve data from the a database
* INSERT - insert data into a table
* UPDATE - updates existing data within a table
* DELETE - deletes all records from a table, the space for the records remain
* MERGE - UPSERT operation (insert or update)
* CALL - call a PL/SQL or Java subprogram
* EXPLAIN PLAN - explain access path to the data
* LOCK TABLE - controls concurrency

**DCL** - Data Control Language. Some examples:

* GRANT - gives user's access privileges to database
* REVOKE - withdraw access privileges given with the GRANT command

**TCL** - Transaction Control: statements used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

* COMMIT - save work done
* SAVEPOINT - identify a point in a transaction to which you can later roll back
* ROLLBACK - undo the modification I made since the last COMMIT
* SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use
* SET ROLE - set the current active roles

DML are not auto-commit. i.e. you can roll-back the operations, but DDL are auto-commit

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=3)**] Difference between TRUNCATE, DELETE and DROP commands?**

The [DELETE](http://www.orafaq.com/wiki/DELETE) command is used to remove **some or all** rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. After performing a DELETE operation you need to [COMMIT](http://www.orafaq.com/wiki/COMMIT) or [ROLLBACK](http://www.orafaq.com/wiki/ROLLBACK) the transaction to make the change permanent or to undo it. Note that this operation will cause all DELETE triggers on the table to fire.

SQL> SELECT COUNT(\*) FROM emp;

COUNT(\*)

----------

14

SQL> DELETE FROM emp WHERE job = 'CLERK';

4 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> SELECT COUNT(\*) FROM emp;

COUNT(\*)

----------

10

[TRUNCATE](http://www.orafaq.com/wiki/TRUNCATE) removes **all rows** from a table. The operation cannot be rolled back and no triggers will be fired. As such, TRUNCATE is faster and doesn't use as much undo space as a DELETE.

SQL> TRUNCATE TABLE emp;

Table truncated.

SQL> SELECT COUNT(\*) FROM emp;

COUNT(\*)

----------

0

The [DROP](http://www.orafaq.com/wiki/DROP) command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. No DML triggers will be fired. The operation cannot be rolled back.

SQL> DROP TABLE emp;

Table dropped.

SQL> SELECT \* FROM emp;

SELECT \* FROM emp

\*

ERROR at line 1:

ORA-00942: table or view does not exist

DROP and TRUNCATE are DDL commands, whereas DELETE is a DML command. Therefore DELETE operations can be rolled back (undone), while DROP and TRUNCATE operations cannot be rolled back.

From [Oracle 10g](http://www.orafaq.com/wiki/Oracle_10g) a table can be "undropped". Example:

SQL> FLASHBACK TABLE emp TO BEFORE DROP;

Flashback complete.

PS: DELETE will not free up used space within a table. This means that repeated DELETE commands will severely fragment the table and queries will have to navigate this "free space" in order to retrieve rows.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=4)**] How does one escape special characters when writing SQL queries?**

**Escape quotes**

Use two quotes for every one displayed. Examples:

SQL> SELECT 'Frank''s Oracle site' AS text FROM DUAL;

TEXT

--------------------

Franks's Oracle site

SQL> SELECT 'A ''quoted'' word.' AS text FROM DUAL;

TEXT

----------------

A 'quoted' word.

SQL> SELECT 'A ''''double quoted'''' word.' AS text FROM DUAL;

TEXT

-------------------------

A ''double quoted'' word.

Use Q expression:

SQL> SELECT q'[Frank's Oracle site]' AS text FROM DUAL;

TEXT

-------------------

Frank's Oracle site

SQL> SELECT q'[A 'quoted' word.]' AS text FROM DUAL;

TEXT

----------------

A 'quoted' word.

SQL> SELECT q'[A ''double quoted'' word.]' AS text FROM DUAL;

TEXT

-------------------------

A ''double quoted'' word.

**Escape wildcard characters**

The LIKE keyword allows for string searches. The '\_' wild card character is used to match exactly one character, while '%' is used to match zero or more occurrences of any characters. These characters can be escaped in SQL. Examples:

SELECT name FROM emp

WHERE id LIKE '%/\_%' ESCAPE '/';

SELECT name FROM emp

WHERE id LIKE '%\%%' ESCAPE '\';

**Escape ampersand (&) characters in SQL\*Plus**

When using SQL\*Plus, the DEFINE setting can be changed to allow &'s (ampersands) to be used in text:

SET DEFINE ~

SELECT 'Laurel & Hardy' FROM dual;

**Other methods:**

Define an escape character:

SET ESCAPE '\'

SELECT '\&abc' FROM dual;

Don't scan for substitution variables:

SET SCAN OFF

SELECT '&ABC' x FROM dual;

Another way to escape the & would be to use concatenation, which would not require any SET commands -

SELECT 'Laurel ' || '&' || ' Hardy' FROM dual;

Use the 10g Quoting mechanism:

Syntax

q'[QUOTE\_CHAR]Text[QUOTE\_CHAR]'

Make sure that the QUOTE\_CHAR followed by an ' doesn't exist in the text.

SELECT q'{This is Orafaq's 'quoted' text field}' FROM DUAL;

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=5)**] Can one select a random collection of rows from a table?**

The following methods can be used to select a random collection of rows from a table:

**The SAMPLE Clause**

From Oracle 8i, the easiest way to randomly select rows from a table is to use the SAMPLE clause with a SELECT statement. Examples:

SELECT \* FROM emp SAMPLE(10);

In the above example, Oracle is instructed to randomly return 10% of the rows in the table.

SELECT \* FROM emp SAMPLE(5) BLOCKS;

This example will sample 5% of all formatted database blocks instead of rows.

This clause only works for single table queries on local tables. If you include the SAMPLE clause within a multi-table or remote query, you will get a parse error or "ORA-30561: SAMPLE option not allowed in statement with multiple table references". One way around this is to create an inline view on the driving table of the query with the SAMPLE clause. Example:

SELECT t1.dept, t2.emp

FROM (SELECT \* FROM dept SAMPLE(5)) t1,

emp t2

WHERE t1.dep\_id = t2.dep\_id;

If you examine the execution plan of a "Sample Table Scan", you should see a step like this:

TABLE ACCESS (SAMPLE) OF 'EMP' (TABLE)

**ORDER BY dbms\_random.value()**

This method orders the data by a random column number. Example:

SQL> SELECT \* FROM (SELECT ename

2 FROM emp

3 ORDER BY dbms\_random.value())

4 WHERE rownum <= 3;

ENAME

----------

WARD

MILLER

TURNER

**The ORA\_HASH() function**

The following example retrieves a subset of the data in the emp table by specifying 3 buckets (0 to 2) and then returning the data from bucket 1:

SELECT \* FROM emp WHERE ORA\_HASH(empno, 2) = 1;

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=6)**] How does one eliminate duplicates rows from a table?**

Choose one of the following queries to identify or remove duplicate rows from a table leaving only unique records in the table:

**Method 1:**

Delete all rowids that is BIGGER than the SMALLEST rowid value (for a given key):

SQL> DELETE FROM table\_name A

2 WHERE ROWID > ( SELECT min(rowid)

3 FROM table\_name B

4 WHERE A.key\_values = B.key\_values );

**Method 2:**

This method is usually faster. However, remember to recreate all indexes, constraints, triggers, etc. on the table when done.

SQL> create table table\_name2 as select distinct \* from table\_name1;

SQL> drop table table\_name1;

SQL> rename table\_name2 to table\_name1;

**Method 3:**

SQL> DELETE FROM my\_table t1

2 WHERE EXISTS ( SELECT 'x' FROM my\_table t2

3 WHERE t2.key\_value1 = t1.key\_value1

4 AND t2.key\_value2 = t1.key\_value2

4 AND t2.rowid > t1.rowid );

Note: One can eliminate N^2 unnecessary operations by creating an index on the joined fields in the inner loop (no need to loop through the entire table on each pass by a record). This will speed-up the deletion process.

Note 2: If you are comparing NULL columns, use the NVL function. Remember that NULL is not equal to NULL. This should not be a problem as all key columns should be NOT NULL by definition.

**Method 4:**

This method collects the first row (order by rowid) for each key values and delete the rows that are not in this set:

SQL> DELETE FROM my\_table t1

1 WHERE rowid NOT IN ( SELECT min(rowid)

2 FROM my\_table t2

3 GROUP BY key\_value1, key\_value2 );

Note: IF key\_value1 is null or key\_value2 is null, this still works correctly.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=7)**] How does one get the time difference between two date columns?**

Oracle allows two date values to be subtracted from each other returning a numeric value indicating the number of days between the two dates (may be a fraction). This example will show how to relate it back to a time value.

Let's investigate some solutions. Test data:

SQL> CREATE TABLE dates (date1 DATE, date2 DATE);

Table created.

SQL>

SQL> INSERT INTO dates VALUES (SYSDATE, SYSDATE-1);

1 row created.

SQL> INSERT INTO dates VALUES (SYSDATE, SYSDATE-1/24);

1 row created.

SQL> INSERT INTO dates VALUES (SYSDATE, SYSDATE-1/60/24);

1 row created.

SQL> SELECT (date1 - date2) FROM dates;

DATE1-DATE2

-----------

1

.041666667

.000694444

**Solution 1**

SQL> SELECT floor(((date1-date2)\*24\*60\*60)/3600)

2 || ' HOURS ' ||

3 floor((((date1-date2)\*24\*60\*60) -

4 floor(((date1-date2)\*24\*60\*60)/3600)\*3600)/60)

5 || ' MINUTES ' ||

6 round((((date1-date2)\*24\*60\*60) -

7 floor(((date1-date2)\*24\*60\*60)/3600)\*3600 -

8 (floor((((date1-date2)\*24\*60\*60) -

9 floor(((date1-date2)\*24\*60\*60)/3600)\*3600)/60)\*60) ))

10 || ' SECS ' time\_difference

11 FROM dates;

TIME\_DIFFERENCE

--------------------------------------------------------------------------------

24 HOURS 0 MINUTES 0 SECS

1 HOURS 0 MINUTES 0 SECS

0 HOURS 1 MINUTES 0 SECS

**Solution 2**

If you don't want to go through the floor and ceiling maths, try this method:

SQL> SELECT to\_number( to\_char(to\_date('1','J') +

2 (date1 - date2), 'J') - 1) days,

3 to\_char(to\_date('00:00:00','HH24:MI:SS') +

4 (date1 - date2), 'HH24:MI:SS') time

5 FROM dates;

DAYS TIME

---------- --------

1 00:00:00

0 01:00:00

0 00:01:00

**Solution 3**

Here is a simpler method:

SQL> SELECT trunc(date1-date2) days,

2 to\_char(trunc(sysdate) + (date1 - date2),

3 'HH24 "Hours" MI "Minutes" SS "Seconds"') time

4 FROM dates;

DAYS TIME

---------- ------------------------------

1 00 Hours 00 Minutes 00 Seconds

0 01 Hours 00 Minutes 00 Seconds

0 00 Hours 01 Minutes 00 Seconds

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=8)**] How does one add a day/hour/minute/second to a date value?**

The SYSDATE pseudo-column shows the current system date and time. Adding 1 to SYSDATE will advance the date by 1 day. Use fractions to add hours, minutes or seconds to the date. Look at these examples:

SQL> select sysdate, sysdate+1/24, sysdate +1/1440, sysdate + 1/86400 from dual;

SYSDATE SYSDATE+1/24 SYSDATE+1/1440 SYSDATE+1/86400

-------------------- -------------------- -------------------- --------------------

03-Jul-2002 08:32:12 03-Jul-2002 09:32:12 03-Jul-2002 08:33:12 03-Jul-2002 08:32:13

The following format is frequently used with Oracle Replication:

select sysdate NOW, sysdate+30/(24\*60\*60) NOW\_PLUS\_30\_SECS from dual;

NOW NOW\_PLUS\_30\_SECS

-------------------- --------------------

03-JUL-2005 16:47:23 03-JUL-2005 16:47:53

**Here are a couple of examples:**

|  |  |
| --- | --- |
| **Description** | **Date Expression** |
| Now | SYSDATE |
| Tomorow/ next day | SYSDATE + 1 |
| Seven days from now | SYSDATE + 7 |
| One hour from now | SYSDATE + 1/24 |
| Three hours from now | SYSDATE + 3/24 |
| A half hour from now | SYSDATE + 1/48 |
| 10 minutes from now | SYSDATE + 10/1440 |
| 30 seconds from now | SYSDATE + 30/86400 |
| Tomorrow at 12 midnight | TRUNC(SYSDATE + 1) |
| Tomorrow at 8 AM | TRUNC(SYSDATE + 1) + 8/24 |
| Next Monday at 12:00 noon | NEXT\_DAY(TRUNC(SYSDATE), 'MONDAY') + 12/24 |
| First day of the month at 12 midnight | TRUNC(LAST\_DAY(SYSDATE ) + 1) |
| The next Monday, Wednesday or Friday at 9 a.m | TRUNC(LEAST(NEXT\_DAY(sysdate, 'MONDAY'), NEXT\_DAY(sysdate, 'WEDNESDAY'), NEXT\_DAY(sysdate, 'FRIDAY'))) + 9/24 |

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=9)**] How does one code a matrix/crosstab/pivot report in SQL?**

Newbies frequently ask how one can display "rows as columns" or "columns as rows". Look at these example crosstab queries (also sometimes called transposed, matrix or pivot queries):

SELECT \*

FROM (SELECT job,

sum(decode(deptno,10,sal)) DEPT10,

sum(decode(deptno,20,sal)) DEPT20,

sum(decode(deptno,30,sal)) DEPT30,

sum(decode(deptno,40,sal)) DEPT40

FROM scott.emp

GROUP BY job)

ORDER BY 1;

JOB DEPT10 DEPT20 DEPT30 DEPT40

--------- ---------- ---------- ---------- ----------

ANALYST 6000

CLERK 1300 1900 950

MANAGER 2450 2975 2850

PRESIDENT 5000

SALESMAN 5600

Here is the same query with some fancy headers and totals:

SQL> ttitle "Crosstab Report"

SQL> break on report;

SQL> compute sum of dept10 dept20 dept30 dept40 total on report;

SQL>

SQL> SELECT \*

2 FROM (SELECT job,

3 sum(decode(deptno,10,sal)) DEPT10,

4 sum(decode(deptno,20,sal)) DEPT20,

5 sum(decode(deptno,30,sal)) DEPT30,

6 sum(decode(deptno,40,sal)) DEPT40,

7 sum(sal) TOTAL

8 FROM emp

9 GROUP BY job)

10 ORDER BY 1;

Mon Aug 23 page 1

Crosstab Report

JOB DEPT10 DEPT20 DEPT30 DEPT40 TOTAL

--------- ---------- ---------- ---------- ---------- ----------

ANALYST 6000 6000

CLERK 1300 1900 950 4150

MANAGER 2450 2975 2850 8275

PRESIDENT 5000 5000

SALESMAN 5600 5600

---------- ---------- ---------- ---------- ----------

sum 8750 10875 9400 29025

Here's another variation on the theme:

SQL> SELECT DECODE(MOD(v.row#,3)

2 ,1, 'Number: ' ||deptno

3 ,2, 'Name: ' ||dname

4 ,0, 'Location: '||loc

5 ) AS "DATA"

6 FROM dept,

7 (SELECT rownum AS row# FROM user\_objects WHERE rownum < 4) v

8 WHERE deptno = 30

9 /

DATA

--------------------------------------- ---------

Number: 30

Name: SALES

Location: CHICAGO

From Oracle 11g, we can use pivot option

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=10)**] Can one retrieve only rows X to Y from a table?**

SELECT \* FROM (

SELECT ename, rownum rn

FROM emp WHERE rownum < 101

) WHERE RN between 91 and 100 ;

Note: the 101 is just one greater than the maximum row of the required rows (means x= 90, y=100, so the inner values is y+1).

SELECT rownum, f1 FROM t1

GROUP BY rownum, f1 HAVING rownum BETWEEN 2 AND 4;

Another solution is to use the MINUS operation. For example, to display rows 5 to 7, construct a query like this:

SELECT \*

FROM tableX

WHERE rowid in (

SELECT rowid FROM tableX

WHERE rownum <= 7

MINUS

SELECT rowid FROM tableX

WHERE rownum < 5);

"this one was faster for me and allowed for sorting before filtering by rownum. The inner query (table A) can be a series of tables joined together with any operation before the filtering by rownum is applied."

SELECT \*

FROM (SELECT a.\*, rownum RN

FROM (SELECT \*

FROM t1 ORDER BY key\_column) a

WHERE rownum <=7)

WHERE rn >=5;

Please note, there is no explicit row order in a relational database. However, this query is quite fun and may even help in the odd situation.

The generic solution to get full information of rows between x and y

SELECT \* FROM emp WHERE empno in (SELECT empno FROM emp GROUP BY rownum,empno HAVING rownum BETWEEN &x AND &y);

"select particular rows from a table : select for rownum = 4, 15 and 17."

select \* from (

select rownum myrownum, emp.\* from employees emp

) mytable

where myrownum in (4,15,17);

"selecting row between range of rownum: select for rownum between (12, 20)."

select \* from (

select rownum myrownum, emp.\* from employees emp

) mytable

where myrownum between 12 and 20;

"Replace 12 and 20 with &x and &y respectively to assign range dynamically."

select \* from (

select rownum myrownum, emp.\* from employees emp

) mytable

where myrownum between &x and &y;

"Combined query to give complete flexibility to pick particular rows and also a given range."

select \* from (

select rownum myrownum, emp.\* from employees emp

) mytable

where myrownum between 12 and 17

or myrownum in ( 3, 18, 25);

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=11)**] Can one retrieve only the Nth row from a table?**

SELECT \* FROM t1 a

WHERE n = (SELECT COUNT(rowid)

FROM t1 b

WHERE a.rowid >= b.rowid);

SELECT \* FROM (

SELECT ENAME,ROWNUM RN FROM EMP WHERE ROWNUM < 101 )

WHERE RN = 100;

Note: In this first query we select one more than the required row number, then we select the required one. Its far better than using a MINUS operation.

SELECT f1 FROM t1

WHERE rowid = (

SELECT rowid FROM t1

WHERE rownum <= 10

MINUS

SELECT rowid FROM t1

WHERE rownum < 10);

SELECT rownum,empno FROM scott.emp a

GROUP BY rownum,empno HAVING rownum = 4;

Alternatively...

SELECT \* FROM emp WHERE rownum=1 AND rowid NOT IN

(SELECT rowid FROM emp WHERE rownum < 10);

Please note, there is no explicit row order in a relational database. However, this query is quite fun and may even help in the odd situation.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=12)**] How can one dump/ examine the exact content of a database column?**

Table data can be extracted from the database as octal, decimal or hex values:

SELECT DUMP(col1, 10)

FROM tab1

WHERE cond1 = val1;

DUMP(COL1)

-------------------------------------

Typ=96 Len=4: 65,66,67,32

For this example, type=96 is indicating a CHAR column. The last byte in the column is 32, which is the ASCII code for a space. This tells us that this column is blank-padded.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=13)**] How does one add a column to the middle of a table?**

Oracle only allows columns to be added to the end of an existing table. Example:

SQL> CREATE TABLE tab1 ( col1 NUMBER );

Table created.

SQL> ALTER TABLE tab1 ADD (col2 DATE);

Table altered.

SQL> DESC tab1

Name Null? Type

----------------------------------------- -------- ----------------------------

COL1 NUMBER

COL2 DATE

Nevertheless, some databases also allow columns to be added to an existing table after a particular column (i.e. in the middle of the table). For example, in MySQL the following syntax is valid:

ALTER TABLE tablename ADD columnname AFTER columnname;

Oracle does not support this syntax. However, it doesn't mean that it cannot be done.

**Workarounds:**

1. Create a new table and copy the data across.

SQL> RENAME tab1 TO tab1\_old;

Table renamed.

SQL> CREATE TABLE tab1 AS SELECT 0 AS col1, col1 AS col2 FROM tab1\_old;

Table created.

2. Rename the table and create a view upon it with its former name and with the columns in the order you want.

3. Use the DBMS\_REDEFINITION package to change the structure on-line while users are working.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=14)**] How does one code a hierarchical tree-structured query?**

The SCOTT/TIGER database schema contains a table EMP with a self-referencing relation (EMPNO and MGR columns). This table is perfect for testing and demonstrating tree-structured queries as the MGR column contains the employee number of the "current" employee's boss.

The LEVEL pseudo-column is an indication of how deep in the tree one is. Oracle can handle queries with a depth of up to 255 levels. Look at this example:

SQL> SELECT level, empno, ename, mgr

2 FROM emp

3 CONNECT BY PRIOR empno = mgr

4 START WITH mgr IS NULL

5 /

LEVEL EMPNO ENAME MGR

---------- ---------- ---------- ----------

1 7839 KING

2 7566 JONES 7839

3 7788 SCOTT 7566

...

One can produce an indented report by using the level number to substring or lpad() a series of spaces, and concatenate that to the string. Look at this example:

SQL> SELECT LPAD(' ', LEVEL \* 2) || ename

2 FROM emp

3 CONNECT BY PRIOR empno = mgr

4 START WITH mgr IS NULL;

LPAD(*,LEVEL\*2)||ENAME*

------------------------------------------------------

KING

JONES

SCOTT

...

Use the "start with" clause to specify the start of the tree. More than one record can match the starting condition. One disadvantage of having a "connect by prior" clause is that you cannot perform a join to other tables. The "connect by prior" clause is rarely implemented in the other database offerings. Trying to do this programmatically is difficult as one has to do the top level query first, then, for each of the records open a cursor to look for child nodes.

One way of working around this is to use PL/SQL, open the driving cursor with the "connect by prior" statement, and the select matching records from other tables on a row-by-row basis, inserting the results into a temporary table for later retrieval.

NOTE: Tree-structured queries are definitely non-relational (enough to kill Codd and make him roll in his grave). Also, this feature is not often found in other database offerings.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=15)**] How does one count/sum data values in a column?**

**Count/sum FIX values:**

Use this simple query to count the number of data values in a column:

select my\_table\_column, count(\*)

from my\_table

group by my\_table\_column;

A more sophisticated example...

select dept, count(decode(sex,'M',1)) MALE,

count(decode(sex,'F',1)) FEMALE,

count(decode(sex,'M',null,'F',null,1)) OTHER,

count(\*) TOTAL

from my\_emp\_table

group by dept;

**Count/sum RANGES of data values in a column:**

A value x will be between values y and z if GREATEST(x, y) = LEAST(x, z). Look at this example:

select f2,

sum(decode(greatest(f1,59), least(f1,100), 1, 0)) "Range 60-100",

sum(decode(greatest(f1,30), least(f1, 59), 1, 0)) "Range 30-59",

sum(decode(greatest(f1, 0), least(f1, 29), 1, 0)) "Range 00-29"

from my\_table

group by f2;

For equal size ranges it might be easier to calculate it with DECODE(TRUNC(value/range), 0, rate\_0, 1, rate\_1, ...). Eg.

select ename "Name", sal "Salary",

decode( trunc(f2/1000, 0), 0, 0.0,

1, 0.1,

2, 0.2,

3, 0.31) "Tax rate"

from my\_table;

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=16)**] How does one drop/ rename a column in a table?**

**Drop a column**

From [Oracle 8i](http://www.orafaq.com/wiki/Oracle_8i) one can DROP a column from a table. Look at this [sample script](http://www.orafaq.com/scripts/sql/dropcol.txt), demonstrating the *ALTER TABLE table\_name DROP COLUMN column\_name;* command.

**Workarounds for older releases:**

SQL> update t1 set column\_to\_drop = NULL;

SQL> rename t1 to t1\_base;

SQL> create view t1 as select >specific columns> from t1\_base;

SQL> create table t2 as select >specific columns> from t1;

SQL> drop table t1;

SQL> rename t2 to t1;

**Rename a column**

From [Oracle 9i](http://www.orafaq.com/wiki/Oracle_9i) one can RENAME a column from a table. Look at this example:

ALTER TABLE tablename RENAME COLUMN oldcolumn TO newcolumn;

**Workarounds for older releases:**

Use a view with correct column names:

rename t1 to t1\_base;

create view t1 >column list with new name> as select \* from t1\_base;

Recreate the table with correct column names:

create table t2 >column list with new name> as select \* from t1;

drop table t1;

rename t2 to t1;

Add a column with a new name and drop an old column:

alter table t1 add ( newcolame datatype );

update t1 set newcolname=oldcolname;

alter table t1 drop column oldcolname;

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=17)**] How does one implement IF-THEN-ELSE logic in a SELECT statement?**

One can use the [CASE](http://www.orafaq.com/wiki/CASE) expression or functions like [DECODE](http://www.orafaq.com/wiki/DECODE), [NVL](http://www.orafaq.com/wiki/NVL), [NVL2](http://www.orafaq.com/wiki/NVL2), [NULLIF](http://www.orafaq.com/wiki/NULLIF), [COALESCE](http://www.orafaq.com/wiki/COALESCE), etc.

Here is the syntax for the CASE-statement:

CASE exp WHEN comparison\_exp1 THEN return\_exp1

[WHEN comparison\_exp2 THEN return\_exp2

WHEN comparison\_exp3 THEN return\_exp3

ELSE else\_exp

]

END

And for DECODE:

DECODE( col | exprn, srch1, rslt1 [, srch2, rslt2,...,] [,default] )

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=18)**] How does one prevent Oracle from using an Index?**

In certain cases, one may want to disable the use of a specific, or all indexes for a given query. Here are some examples:

**Adding an expression to the indexed column**

SQL>select count(\*) from t where empno+0=1000;

COUNT(\*)

----------

1

Execution Plan

--------------------------------------------- ----- --------

0 SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=1 Bytes=3)

1 0 SORT (AGGREGATE)

2 1 TABLE ACCESS (FULL) OF 'T' (Cost=2 Card=1 Bytes=3)

**Specifying the FULL hint to force full table scan**

SQL>select /\*+ FULL(t) \*/ \* from t where empno=1000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO GRADE

---------- ---------- --------- ---------- --------- ---------- ---------- ---------- ----------

1000 Victor DBA 7839 20-MAY-03 11000 0 10 JUNIOR

Execution Plan

--------------------------------------------- ----- --------

0 SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=1 Bytes=41)

1 0 TABLE ACCESS (FULL) OF 'T' (Cost=2 Card=1 Bytes=41)

**Specifying NO\_INDEX hint**

SQL>select /\*+ NO\_INDEX(T) \*/ count(\*) from t where empno=1000;

COUNT(\*)

----------

1

Execution Plan

--------------------------------------------- ----- --------

0 SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=1 Bytes=3)

1 0 SORT (AGGREGATE)

2 1 TABLE ACCESS (FULL) OF 'T' (Cost=2 Card=1 Bytes=3)

**Using a function over the indexed column**

SQL>select count(\*) from t where to\_number(empno)=1000;

COUNT(\*)

----------

1

Execution Plan

--------------------------------------------- ----- --------

0 SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=1 Bytes=3)

1 0 SORT (AGGREGATE)

2 1 TABLE ACCESS (FULL) OF 'T' (Cost=2 Card=1 Bytes=3)

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=19)**] How does one select EVERY Nth row from a table?**

One can easily select all even, odd, or Nth rows from a table using SQL queries like this:

**Method 1:** Using a [subquery](http://www.orafaq.com/wiki/Subquery)

SELECT \*

FROM emp

WHERE (ROWID,0) IN (SELECT ROWID, MOD(ROWNUM,4)

FROM emp);

**Method 2:** Use dynamic views (available from Oracle7.2):

SELECT \*

FROM ( SELECT rownum rn, empno, ename

FROM emp

) temp

WHERE MOD(temp.ROWNUM,4) = 0;

**Method 3:** Using GROUP BY and HAVING

SELECT rownum, f1 FROM t1

GROUP BY rownum, f1 HAVING MOD(rownum,n) = 0 OR rownum = 2-n;

Please note, there is no explicit row order in a relational database. However, these queries are quite fun and may even help in the odd situation.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=20)**] How does one select the LAST N rows from a table?**

From Oracle 9i onwards, the RANK() and DENSE\_RANK() functions can be used to determine the **LAST N** or **BOTTOM N** rows. Examples:

**Get the bottom 10 employees based on their salary**

SELECT ename, sal

FROM ( SELECT ename, sal, RANK() OVER (ORDER BY sal ASC) sal\_rank

FROM emp )

WHERE sal\_rank <= 10;

**Select the employees getting the lowest 10 salaries**

SELECT ename, sal

FROM ( SELECT ename, sal, DENSE\_RANK() OVER (ORDER BY sal) sal\_dense\_rank

FROM emp )

WHERE sal\_dense\_rank <= 10;

For [Oracle 8i](http://www.orafaq.com/wiki/Oracle_8i) and above, one can get the **bottom N** rows using an inner-query with an ORDER BY clause:

SELECT \*

FROM (SELECT \* FROM my\_table ORDER BY col\_name\_1)

WHERE ROWNUM < 10;

Use this workaround for older (8.0 and prior) releases:

SELECT \*

FROM my\_table a

WHERE 10 >= (SELECT COUNT(DISTINCT maxcol)

FROM my\_table b

WHERE b.maxcol <= a.maxcol)

ORDER BY maxcol;

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=21)**] How does one select the TOP N rows from a table?**

From Oracle 9i onwards, the RANK() and DENSE\_RANK() functions can be used to determine the TOP N rows. Examples:

**Get the top 10 employees based on their salary**

SELECT ename, sal

FROM ( SELECT ename, sal, RANK() OVER (ORDER BY sal DESC) sal\_rank

FROM emp )

WHERE sal\_rank <= 10;

**Select the employees making the top 10 salaries**

SELECT ename, sal

FROM ( SELECT ename, sal, DENSE\_RANK() OVER (ORDER BY sal DESC) sal\_dense\_rank

FROM emp )

WHERE sal\_dense\_rank <= 10;

For Oracle 8i and above, one can get the Top N rows using an inner-query with an ORDER BY clause:

SELECT \*

FROM (SELECT \* FROM my\_table ORDER BY col\_name\_1 DESC)

WHERE ROWNUM < 10;

Use this workaround for older (8.0 and prior) releases:

SELECT \*

FROM my\_table a

WHERE 10 >= (SELECT COUNT(DISTINCT maxcol)

FROM my\_table b

WHERE b.maxcol >= a.maxcol)

ORDER BY maxcol DESC;

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=22)**] How to generate a text graphs (histograms) using SQL?**

SELECT d.dname AS "Department",

LPAD('+', COUNT(\*), '+') as "Graph"

FROM emp e, dept d

WHERE e.deptno = d.deptno

GROUP BY d.dname;

Sample output:

Department Graph

-------------- --------------------------------------------------

ACCOUNTING +++

RESEARCH +++++

SALES ++++++

In the above example, the value returned by COUNT(\*) is used to control the number of "\*" characters to return for each department. We simply pass COUNT(\*) as an argument to the string function LPAD (or RPAD) to return the desired number of \*'s.

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=23)**] Map/ concatenate several rows to a column**

This FAQ will demonstrate how row values can be concatenated into a single column value (similar to MySQL's [i]GROUP\_CONCAT[/i] function).

Start by creating this function:

SQL> CREATE OR REPLACE FUNCTION rowconcat(q in VARCHAR2) RETURN VARCHAR2 IS

2 ret VARCHAR2(4000);

3 hold VARCHAR2(4000);

4 cur sys\_refcursor;

5 BEGIN

6 OPEN cur FOR q;

7 LOOP

8 FETCH cur INTO hold;

9 EXIT WHEN cur%NOTFOUND;

10 IF ret IS NULL THEN

11 ret := hold;

12 ELSE

13 ret := ret || ',' || hold;

14 END IF;

15 END LOOP;

16 RETURN ret;

17 END;

18 /

Function created.

This function returns a string result with the concatenated non-NULL values from a SQL statement. It returns NULL if there are no non-NULL values.

Here is an example of how to map several rows to a single concatenated column:

SQL> SELECT rowconcat('SELECT dname FROM dept') AS departments

2 FROM dual;

DEPARTMENTS

--------------------------------------------------------------------------------

ACCOUNTING,RESEARCH,SALES,OPERATIONS

This example is more interresting, it concatenates a column across several rows based on an aggregation:

SQL> col employees format a50

SQL> SELECT deptno,

2 rowconcat('SELECT ename FROM emp a WHERE deptno='||deptno) AS Employees

3 FROM emp

4 GROUP BY deptno

5 /

DEPTNO EMPLOYEES

---------- --------------------------------------------------

30 ALLEN,WARD,MARTIN,BLAKE,TURNER,JAMES

20 SMITH,JONES,SCOTT,ADAMS,FORD

10 CLARK,KING,MILLER

**[**[**edit**](http://www.orafaq.com/wiki/index.php?title=SQL_FAQ&action=edit&section=24)**] What is the difference between VARCHAR, VARCHAR2 and CHAR data types?**

Both CHAR and VARCHAR2 types are used to store character string values, however, they behave very differently. The VARCHAR type should not be used:

**CHAR**

CHAR should be used for storing fixed length character strings. String values will be space/blank padded before stored on disk. If this type is used to store variable length strings, it will waste a lot of disk space.

SQL> CREATE TABLE char\_test (col1 CHAR(10));

Table created.

SQL> INSERT INTO char\_test VALUES ('qwerty');

1 row created.

SQL> SELECT col1, length(col1), dump(col1) "ASCII Dump" FROM char\_test;

COL1 LENGTH(COL1) ASCII Dump

---------- ------------ ------------------------------------------------------------

qwerty 10 Typ=96 Len=10: 113,119,101,114,116,121,32,32,32,32

Note: ASCII character 32 is a blank space.

**VARCHAR**

Currently VARCHAR behaves exactly the same as VARCHAR2. However, this type should not be used as it is reserved for future usage.

SQL> CREATE TABLE varchar\_test (col1 VARCHAR2(10));

Table created.

SQL> INSERT INTO varchar\_test VALUES ('qwerty');

1 row created.

SQL> SELECT col1, length(col1), dump(col1) "ASCII Dump" FROM varchar\_test;

COL1 LENGTH(COL1) ASCII Dump

---------- ------------ ------------------------------------------------------------

qwerty 6 Typ=1 Len=6: 113,119,101,114,116,121

**VARCHAR2**

VARCHAR2 is used to store variable length character strings. The string value's length will be stored on disk with the value itself.

SQL> CREATE TABLE varchar2\_test (col1 VARCHAR2(10));

Table created.

SQL> INSERT INTO varchar2\_test VALUES ('qwerty');

1 row created.

SQL> SELECT col1, length(col1), dump(col1) "ASCII Dump" FROM varchar2\_test;

COL1 LENGTH(COL1) ASCII Dump

---------- ------------ ------------------------------------------------------------

qwerty 6 Typ=1 Len=6: 113,119,101,114,116,121