LO7 – SELECT Statement

Select is used to query the database. The result set is simply another table (dynamically created from our database).

* SELECT – Simplest form syntax:
  + SELECT \*

FROM TableName;

* + Selects all columns in the order created; all rows from the specified table
  + It’s generally best practice in industry to avoid the use of (\*)
* Specifying columns
  + SELECT expression [, expression]

FROM TableName;

* + Expression can be the name of a column, or a constant, or an expression (function, calculations, etc)
  + Can use \* for all columns, can also put columns in any order
  + Can also give columns an **alias**: SELECT col1 AS someName FROM TableName;
* Full Syntax:
  + SELECT [DISTINCT | ALL] select\_list

FROM the\_tables\_envolved

[WHERE search\_conditions]

[GROUP BY group\_by\_list [HAVING group\_search\_condition]]

[ORDER BY order\_by\_list];

* Operators in SQL
  + Typical operators (\*, /, +, -)
  + String concatenation: ||
  + Relational operators:
    - = (not == like in Java, or === like in JavaScript)
    - >, <, >=, <=, <> (not equal to, Oracle also allows !=)
    - Relational operators can be applied to character strings (city > ‘London’), but how the evaluation is done is dependent on the DBMS.
      * Oracle compares character strings position by position, starting at the left end, and up to the first character that is different. Whichever string has the “greater” character in that position is considered the greater string. The collation sequence of the characters is dependent on the computer’s OS (ASCII or EBCDIC).
      * If two VARCHAR2’s are identical up to the end of the shorter one, the longer string is considered greater.
  + Boolean operators – use words (don’t use !, &&, ||, etc)
    - NOT, AND, OR
  + Range Operators
    - Column BETWEEN low AND high
      * Is inclusive = equivalent to: column >= low AND column <= high
      * Ex: Age BETWEEN 18 AND 65
    - Column NOT BETWEEN low AND high
  + Nulls
    - Cannot test for NULL values using equality/inequality
    - ColumnName IS NULL
    - ColumnName IS NOT NULL
  + Comparison Operators:
    - ColumnName IN (constant [, constant…])
    - ColumnName NOT IN (constant [, constant…])
    - ColumnName LIKE pattern [ESCAPE ‘character’]
      * Wild card characters are, by default:
        + % - any 0 or more characters
        + \_ - any 1 character
      * The ESCAPE clause can be used if you need to search ‘%’ or ‘\_’
* Sorting the results
  + Use ORDER BY
  + Only affects the display order. Can order on multiple columns.
  + Oracle treats null values, by default, as high values. In other words, the default behaviour is as follows:
    - NULLS LAST is the default for ASC
    - NULLS FIRST is the default for DESC.
  + Default is ascending order. Can use ASC or DESC.

# Single Row Functions

* Powerpoint available on Brightspace

# Group or Aggregate Functions

* Aggregate functions return a single result row based on groups of rows, rather than on single rows.
* These functions apply to groups of rows – for a group of rows, you can get a single value
  + Example, add up all the order amounts for a giving group of rows (orders)
* Mixing aggregate results and single row results doesn’t work in most implementations of SQL (including Oracle).
  + Usually you would have to use some kind of reporting tool to see individual records and summary records.
* Aggregate functions take some type of expression. The expression is usually a column name, but it can also be a constant or calculated value.
* The use of DISTINCT with the above means the system with SUM/AVG/COUNT on the distinct values and not include repeated values more than once.
* Aggregates cannot be used as conditions in the WHERE clause.
* Aggregates and NULLS
  + If there are any null values in the column on which the aggregate function is operating, they are ignored for the purpose of the function, except for COUNT(\*), which counts each row.
  + Can use **NVL** or **COALESCE** in the argument of an aggregate to avoid nulls by substituting null for something else (“NA”, or 0, etc). **COALESCE** is standardized across many RDBMS’s. **NVL** is Oracle-specific. Keep in mind, depending on the RDBMS you are using, you may have different SQL syntax and operations available to you.
* The functions are:
  + SUM(expression)
    - Provides the total of the expression over the group of rows. The expression must evaluate to a number or NULL
  + AVG(expression)
    - Averages the expression over the group of rows
  + COUNT(expression)
    - Proves the number of non-null values in the expression over the group of rows.
  + COUNT(\*)
    - Provides the number of rows in the group, *regardless* of nulls.
  + Max/Min(expression)
    - Provides the highest (or lowest) value in the group of rows.
* GROUP BY and HAVING Clauses
  + Specify the GROUP BY clause if you want Oracle to group the selected rows based on the value of expression(s) for each row and return a single row of summary information for each group.
  + Use the HAVING clause to restrict the groups of returned rows to those groups for which the specified condition is TRUE
  + GROUP BY Syntax:
    - When you specify columns or expressions in the GROUP BY clause, then those columns / expressions **MUST** appear in the SELECT clause.
    - Aliases do not work in Oracle’s GROUP BY
    - Multiple expressions can appear in the GROUP BY
    - NULL and groups
      * If the grouping columns contains more than one NULL, all of them are put into a single group.

# Joining Tables

* With normalization, we break related information into multiple tables. We use joins to put the tables back together.

General rules to get meaningful results with joins:

* All tables must be listed in order for the system to perform the join
* The columns must have related meaning. For example, you could the author’s age to numbers in the salesDetail table, but the results would be meaningless.
* Columns that we ***typically*** join will be through the Primary Key of the parent, and the Foreign Key in the child.

Join Syntax:

SELECT <selectList>

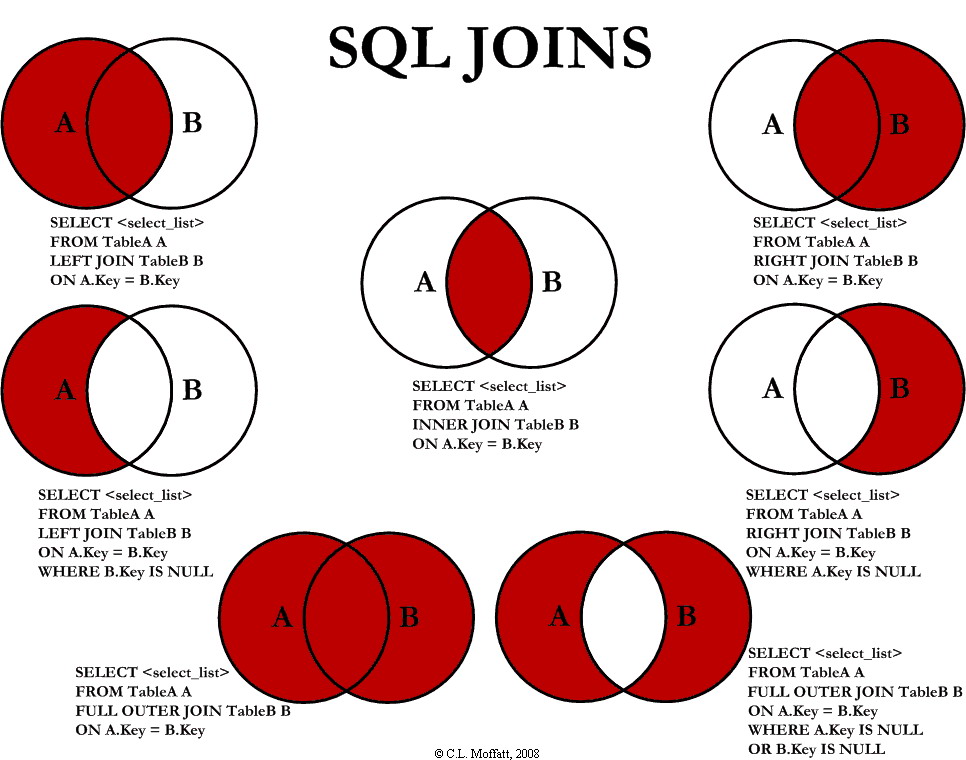
FROM <table1> JOIN <table2> ON condition [JOIN <table3> ON condition]

Join Operator (Part of **condition**)

* Usually equal, can be any relational operator
* Join columns do not have to appear in the SELECT list portion of the query.
* ERD Diagrams will help us figure out what join conditions are necessary.

Types of joins:

* CROSS JOIN – A cross-product.
* INNER JOIN – Or Equijoin, and the default
* OUTER JOIN
  + Data can be lost performing an INNER JOIN that can be regained using an OUTER JOIN
  + LEFT JOIN – Includes all the rows from the “first” table (the table on the left side of the JOIN operator), and all the matching records in the second table.
  + RIGHT JOIN – Sames as LEFT, but uses the right side table.
  + FULL JOIN – include all rows from both tables and all matching rows between the two tables
* SELF JOIN
  + To join a table to itself, the table must appear more than once in the FROM clause. At least one table must be given an alias, so that we can distinguish one instance of the table from another.



# Set Operators – UNION, INTERSECT, and MINUS