Cognixia™

Oracle SQL

Day 2

www.cognixia.com

Day 2

- Recap
- SELECT
  - Sorting
  - Conditional logic
  - Joins
  - Subqueries
  - Common table expressions
  - Aggregate functions
  - Aggregation using:
    - GROUP BY ROLLUP
    - GROUPING()
    - CUBE
    - PIVOT

Recap - SELECT

```
SELECT columnlist
FROM tablelist
WHERE condition
GROUP BY columnlist
HAVING condition
ORDER BY columnlist
```

This is the generic SELECT statement.

Sorting

- Order By • ASC • DESC
- Multiple Column Sorts
- Collation order (ASC): NULL, numbers, characters
- A column alias declared in the SELECT list can be used in the ORDER BY clause

Conditional logic

```
SELECT columnlist [, CaseExpression ]
FROM tablelist
WHERE condition [, CaseExpression ]
GROUP BY columnlist [, CaseExpression ]
HAVING condition [, CaseExpression ]
ORDER BY columnlist [, CaseExpression ]
```

Conditional logic is applied using the CASE expression, and can appear in any of:

- SELECT
- WHERE
- GROUP BY
- HAVING, or
- ORDER BY

CASE expression

```
CASE ColumnOrExpression
     WHEN value1 THEN result1
    (repeat WHEN-THEN any number of times)
[ELSE DefaultResult]
END
```

- In the example above, the column is stated in the CASE clause.
- Below, the CASE is generic, and any column expressions can be used in WHEN / ELSE resulting in a more powerful logic.

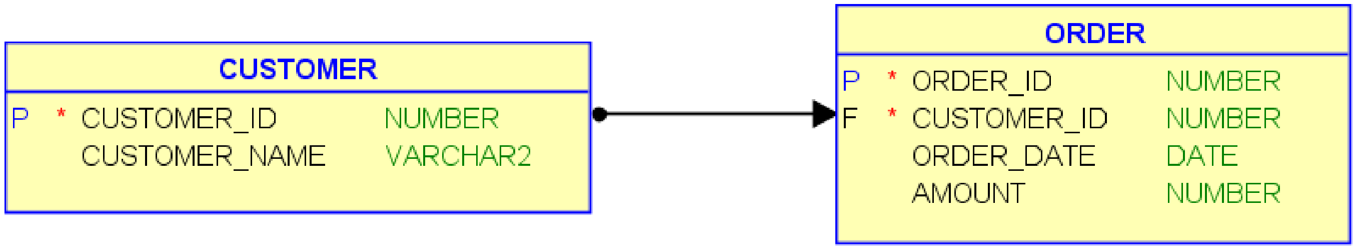
```
CASE
     WHEN expression1 THEN result1
    (repeat WHEN-THEN any number of times)
[ELSE DefaultResult]
END
```

Joins and Relationships

Joins are the most powerful tool in the relational technology. Without them, data could not be assembled together from the multitude of tables.

The joins are based on shared keys, which form a “relationship”.

In the image below, the CUSTOMER_ID is a shared key, which allows to associate an order with the customer that placed it. The origin of the key, the CUSTOMER table is the “Parent” table. The ORDER table depends on the CUSTOMER table to provide the CUSTOMER information, therefore it is a “Child” table.



CUSTOMER

P	*	CUSTOMER_ID	NUMBER
		CUSTOMER_NAME	VARCHAR2

ORDER

P	*	ORDER_ID	NUMBER
F	*	CUSTOMER_ID	NUMBER
		ORDER_DATE	DATE
		AMOUNT	NUMBER

Joins

There are a number of types of joins:

- INNER JOIN
- OUTER JOIN: LEFT, RIGHT, FULL
- SELF JOIN – Relationship
  - Hierarchical Query (Oracle specialty)
- CROSS JOIN (Cartesian product) – don't do it!

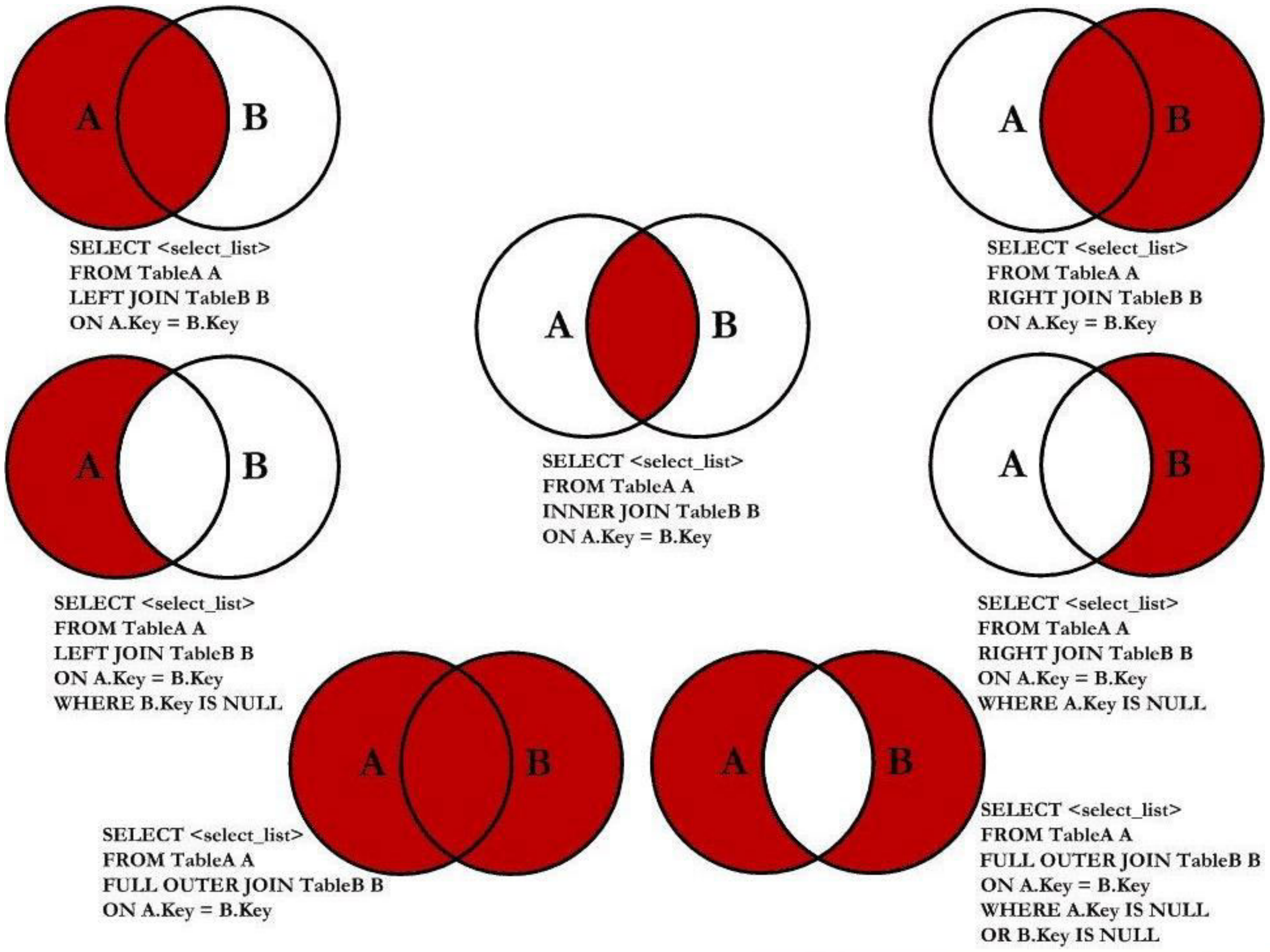
Other classifications:

Equi-join	uses the = operator
Natural join	joined columns have the same name. Dangerous
Non-key join	joined columns are not PK or FK
Semi-join	subquery with IN or EXISTS
Anti-join	subquery with NOT IN or NOT EXISTS

Some rules:

- NULLs never match in any JOIN, not even other NULLs

Joins



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Order of tables in Joins

- For INNER JOINs, order doesn’t matter
- For complex OUTER JOINs of multiple tables, order is very important
- Use () to control the sequence of joins
- Understand what each individual join achieves

Subqueries

```
SELECT columnlist
FROM tablelist
WHERE condition
GROUP BY columnlist
HAVING condition
ORDER BY columnlist
```

A subquery can be used in the *columnlist*, *tablelist*, or *condition* position.

- When a subquery is part of a *tablelist*, it specifies a data source.
- When a subquery is part of a *condition*, it becomes part of the selection criteria.
- When a subquery is part of a *columnlist*, it creates a single calculated column.

Subquery as data source

```
SELECT columnlist
FROM tablelist
WHERE condition
```

- When a subquery is part of a *tablelist*, it specifies a data source.

Subquery as selection

```
SELECT columnlist
FROM tablelist
WHERE condition
HAVING condition
```

When a subquery is part of a *condition*, it becomes part of the selection criteria.

Subquery as calculated

```
SELECT columnlist
FROM tablelist
WHERE condition
GROUP BY columnlist
ORDER BY columnlist
```

- When a subquery is part of a *columnlist*, it creates a single calculated column.

Correlated Subqueries

```
SELECT columnlist
FROM table1 outer
WHERE column1 operator
                    (SELECT columnlist
                     FROM table2
                     WHERE expr1 = outer.expr2);
```

In a correlated subquery, the inner query references the outer query, and cannot execute on its own. They can be inefficient because the subquery executes for each row of the driving query.

Many times, correlated subqueries can be replaced by equivalent straight joins.

Common table expressions

```
WITH alias AS (SELECT…)
SELECT columnlist
FROM tablelist JOIN alias ON …
```

WITH keyword introduces a subquery, which is then used as a regular *tablelist*.

It is an alternative notation, typically used in uncorrelated subqueries.

Aggregate functions

- COUNT [DISTINCT], SUM, AVG, MIN, MAX
- NULL values are excluded in computation

Aggregation with GROUP BY

```
SELECT columnlist, aggregate function
FROM tablelist
WHERE condition
GROUP BY columnlist
HAVING condition
```

All columns in the SELECT *columnlist* must be listed as columns in the GROUP BY clause.

WHERE applies *condition* to selecting individual rows (before GROUP BY), while HAVING applies *condition* to the resulting group values (after GROUP BY).

Rollup

```
SELECT columnlist, function
FROM tablelist
GROUP BY ROLLUP (columnlist)
```

ROLLUP creates subtotals at each level in hierarchical structure.

Country	Warehouse	Quantity
Canada	Toronto	12969
Canada		12969
United States of America	New Jersey	7252
United States of America	San Francisco	28613
United States of America	Seattle, Washington	14860
United States of America	Southlake, Texas	5483
United States of America		56208
		69177

Grouping function

```
SELECT columnlist, [, CaseExpression ]
GROUPING(column), aggregate function
FROM tablelist
GROUP BY ROLLUP (columnlist)
```

The GROUPING() function returns a 1 if the aggregated number is a subtotal row over the specified column.

Country	Warehouse	Grouping (COUNTRY)	Grouping (WAREHOUSE)	Quantity
Canada	Toronto	0	0	12969
Canada		0	1	12969
United States of America	New Jersey	0	0	7252
United States of America	San Francisco	0	0	28613
United States of America	Seattle, Washington	0	0	14860
United States of America	Southlake, Texas	0	0	5483
United States of America		0	1	56208
		1	1	69177

CUBE

```
SELECT columnlist, aggregate function
FROM tablelist
GROUP BY CUBE (columnlist)
```

The GROUPING() function returns a 1 if the aggregated number is a subtotal row over the specified column.

Customer	Category	Year	Total units
AECOM	CPU	2016	51
AECOM	CPU		51
AECOM	Mother Board	2016	144
AECOM	Mother Board		144
AECOM	Storage	2016	287
AECOM	Storage		287
AECOM		2016	482
AECOM			482

PIVOT

```
SELECT * FROM (query)
PIVOT (aggregation_function(column)
      FOR column_for_column_headers
      IN pivot_column_values)
```

Example: Summarize order quantity by customer/year using PIVOT from OT database.

Customer	Year	CPU	Storage	Video Card
AECOM	2016	51	287	
AbbVie	2015	105	309	95
AbbVie	2016	233	35	
AbbVie	2017	192	483	121
Abbott Laboratories	2016	223	503	105

References

Resource	Location
Oracle SQL Reference	https://docs.oracle.com/database/121/SQLRF/toc.htm
Oracle Database 2 Day Developer's Guide	https://docs.oracle.com/database/121/TDDDG/toc.htm
Oracle Ask Tom	https://asktom.oracle.com

Cognixia™

THANK YOU