ENTRENAMIENTO
EXPERIENCIA

GALAXY
TRAINING

ORACLE DATABASE 12c ONWARDS

MODULO III

Ing. Cesar Hijar
Instructor

AGENDA

a) Data Manipulation Language (DML)
b) Inserción, Actualización y eliminación de registros o filas.
c) Data Definition Language (DDL).
d) Objetos de base de datos Oracle.
e) Reglas para trabajar con objetos y DDL.

AGENDA

a) Creación, modificación y eliminación de tablas.
b) Tipos de datos.
c) Comentarios de tabla y columna.
d) Truncate table vs Delete .
e) Rename table.

Sentencias SQL

- **Data manipulation language (DML)**
- SELECT
- INSERT
- UPDATE
- DELETE
- MERGE
- **Data Control Language (DCL)**
- GRANT
- REVOKE

- **Data definition language (DDL)**
- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE
- COMMENT
- **Transaction control**
- COMMIT
- ROLLBACK
- SAVEPOINT

Sentencia SELECT

Basic SELECT Statement

```
SELECT  *|{[DISTINCT] column [alias],...}
FROM    table;
```

- SELECT identifies the columns to be displayed.
- FROM identifies the table containing those columns.

In its simplest form, a SELECT statement must include the following:

- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause

In the syntax:

SELECT	Is a list of one or more columns
*	Selects all columns
DISTINCT	Suppresses duplicates
column|expression	Selects the named column or the expression
alias	Gives different headings to the selected columns
FROM *table*	Specifies the table containing the columns

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)

...

12	Zlotkey	25200
13	Abel	39600
14	Taylor	20640
15	Grant	12600

...

17	Hartstein	(null)
18	Fay	(null)
19	Higgins	(null)
20	Gietz	(null)

If any column value in an arithmetic expression is null, the result is null. For example, if you attempt to perform division by zero, you get an error. However, if you divide a number by null, the result is a null or unknown.

In the example in the slide, employee King does not get any commission. Because the `COMMISSION_PCT` column in the arithmetic expression is null, the result is null.

Sentencia SELECT

Using Literal Character Strings

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG
11	King is a AD_PRES

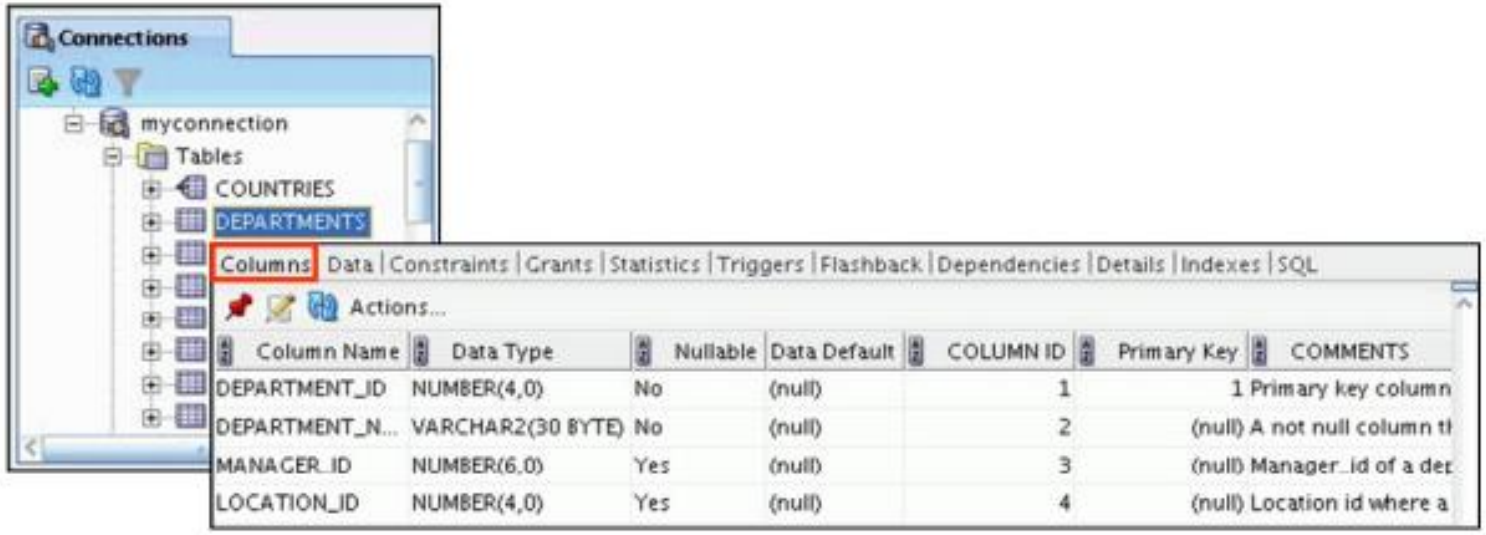
...

Sentencia SELECT

Displaying Table Structure

- Use the `DESCRIBE` command to display the structure of a table.
- Or, select the table in the Connections tree and use the Columns tab to view the table structure.

```
DESC[RIBE] tablename
```



Sentencia SELECT

Using the DESCRIBE Command

```
DESCRIBE employees
```

```
DESCRIBE Employees
Name           Null     Type
-------------- -------- ------------
EMPLOYEE_ID    NOT NULL NUMBER(6)
FIRST_NAME              VARCHAR2(20)
LAST_NAME      NOT NULL VARCHAR2(25)
EMAIL          NOT NULL VARCHAR2(25)
PHONE_NUMBER            VARCHAR2(20)
HIRE_DATE      NOT NULL DATE
JOB_ID         NOT NULL VARCHAR2(10)
SALARY                  NUMBER(8,2)
COMMISSION_PCT          NUMBER(2,2)
MANAGER_ID              NUMBER(6)
DEPARTMENT_ID           NUMBER(4)
```

The example in the slide displays information about the structure of the `EMPLOYEES` table using the `DESCRIBE` command.

In the resulting display, *Null* indicates that the values for this column may be unknown. `NOT NULL` indicates that a column must contain data. *Type* displays the data type for a column.

The data types are described in the following table:

Data Type	Description
`NUMBER(p,s)`	Number value having a maximum number of digits *p*, with *s* digits to the right of the decimal point
`VARCHAR2(s)`	Variable-length character value of maximum size *s*
`DATE`	Date and time value between January 1, 4712 B.C. and December 31, A.D. 9999

Sentencia SELECT

Quiz

Identify the SELECT statements that execute successfully.

a.
```
SELECT first_name, last_name, job_id, salary*12
 AS Yearly Sal
FROM   employees;
```

b.
```
SELECT first_name, last_name, job_id, salary*12
 "yearly sal"
FROM   employees;
```

c.
```
SELECT first_name, last_name, job_id, salary AS
 "yearly sal"
FROM   employees;
```

d.
```
SELECT first_name+last_name AS name, job_Id,
 salary*12 yearly sal
FROM   employees;
```

ENTRENAMIENTO
EXPERIENCIA

Character strings and dates in the WHERE clause must be enclosed within single quotation marks (' '). Number constants, however, need not be enclosed within single quotation marks.

All character searches are case-sensitive. In the following example, no rows are returned because the EMPLOYEES table stores all the last names in mixed case:

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'WHALEN';
```

Character Strings and Dates

- Character strings and date values are enclosed within single quotation marks.
- Character values are case-sensitive and date values are format-sensitive.
- The default date display format is DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	Whalen	AD_ASST	10

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-OCT-03' ;
```

	LAST_NAME
1	Rajs

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Range Conditions Using the BETWEEN Operator

Use the BETWEEN operator to display rows based on a range of values:

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit Upper limit

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

Using Comparison Operators

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

You can also use the BETWEEN operator on character values:

```
SELECT last_name FROM   employees
WHERE  last_name BETWEEN 'King' AND 'Whalen
```

Using the IN Operator

Use the IN operator to test for values in a list:

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12008	101
8	202	Fay	6000	201

The IN operator can be used with any data type. The following example returns a row from the EMPLOYEES table, for any employee whose last name is included in the list of names in the WHERE clause:

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  last_name IN ('Hartstein', 'Vargas');
```

If characters or dates are used in a list, they must be enclosed within single quotation marks (' ').

Pattern Matching Using the LIKE Operator

- Use the LIKE operator to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
  - % denotes zero or more characters.
  - _ denotes one character.

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%' ;
```

	FIRST_NAME
1	Shelley
2	Steven

Combining Wildcard Characters

- You can combine the two wildcard characters (%, _) with literal characters for pattern matching:

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- You can use the ESCAPE identifier to search for the actual % and _ symbols.

Operadores y Wildcards en la cláusula WHERE

- LIKE "_va" busca todos los nombres de tres letras que terminan en "va". (Ej.: Eva, Iva, Ava).

- LIKE "[CM]arlo[ns]" busca todos los nombres: Carlon, Marlon, Carlos y Marlos.

- LIKE "[B-D]elia" busca todos los nombres que terminan en "elia" y que comiencen con las letras de la B a la D. (Ej.: Delia, Celia).

Operadores y Wildcards en la cláusula WHERE

- LIKE “M[!a]%” busca todos los nombres que comiencen con M y cuya segunda letra no es “a”.(Mónica).
- LIKE “_ _ _” busca todas las cadenas de exactamente 3 caracteres.
- LIKE “_ _ _ %” busca las cadenas de al menos 3 caracteres.

Operadores y Wildcards en la cláusula WHERE

Para que los patrones puedan contener los caracteres especiales patrón (Wildcard), se especifica un caracter de escape.

- LIKE “ab\%cd%” escape “\” busca todas las cadenas que empiecen con “ab%cd”.

- LIKE “ab*cd%” escape “\” busca las cadenas que empiecen con “ab*cd”.

Operadores y Wildcards en la cláusula WHERE

....WHERE Nombre

- LIKE “Ma%” busca todos los nombres que comiencen con “Ma”

  (Ej.: María, Mariana, Manuel, Martín)

- LIKE “%ía” busca todos los nombres que terminen con “ía”.

  (Ej.: Sofía, María, Estefanía).

- LIKE “%ar%” busca todos los nombres que tengan las letras “ar”.

  (Ej.:Carlos, Arturo, Eleazar).

Using NULL Conditions

Test for nulls with the IS NULL operator.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

The NULL conditions include the IS NULL condition and the IS NOT NULL condition.

The IS NULL condition tests for nulls. A null value means that the value is unavailable, unassigned, unknown, or inapplicable. Therefore, you cannot test with =, because a null cannot be equal or unequal to any value. The example in the slide retrieves the last_name and manager_id of all employees who do not have a manager.

Here is another example: To display the last name, job ID, and commission for all employees who are *not* entitled to receive a commission, use the following SQL statement:

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

Sorting

- Sorting in descending order:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY department_id DESC ;
```

1

- Sorting by column alias:

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal ;
```

2

Sorting

- Sorting by using the column's numeric position:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY 3;
```

3

- Sorting by multiple columns:

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

4

Tipos de Joins

- Los Joins que son compatibles con SQL:1999 incluyen lo siguiente:
  - Natural join con la cláusula NATURAL JOIN
  - Join con la cláusula USING
  - Join con la cláusula ON
  - OUTER joins:
    - LEFT OUTER JOIN
    - RIGHT OUTER JOIN
    - FULL OUTER JOIN
  - Cross joins

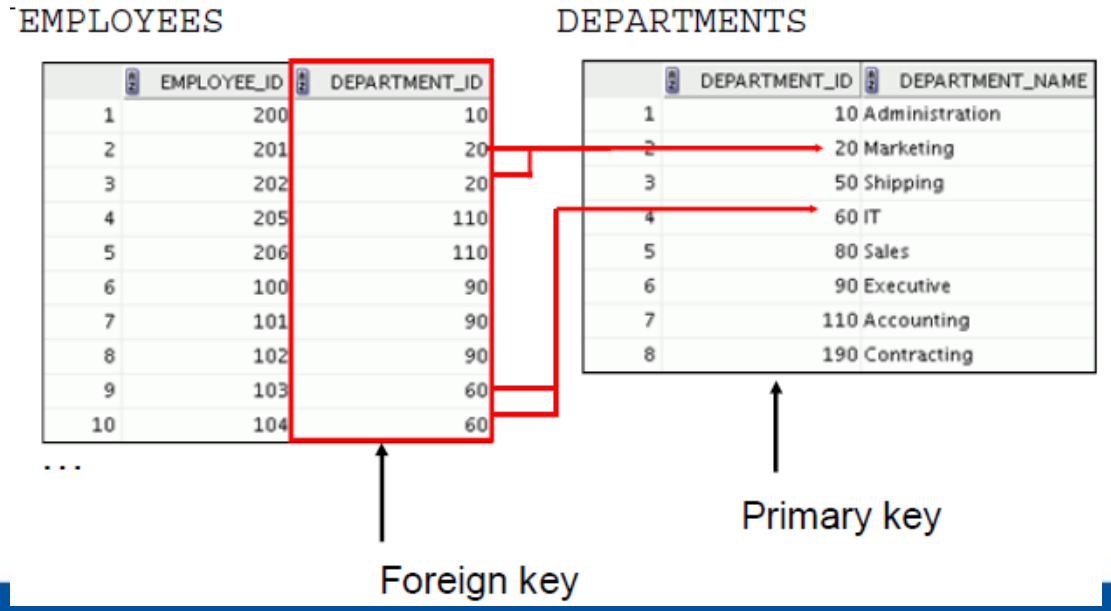
Natural Join

- Se basa en todas las columnas que tienen el mismo nombre en ambas tablas.
- Selecciona todas las filas que tiene coincidencia en todas las columnas iguales
- Si las columnas con el mismo nombre tienen tipos de datos diferentes, se retorna un error.

```
SELECT * FROM table1 NATURAL JOIN table2;
```

Joins utilizando la cláusula USING

- Si varias columnas tienen los mismos nombres, pero los tipos de datos coinciden, utilice la cláusula USING para especificar las columnas para la combinación de igualdad.



Creando Joins con la cláusula ON

- La condición de unión para la reunión natural es básicamente una combinación de igualdad de todas las columnas con el mismo nombre.
- Utilice la cláusula ON para especificar las condiciones arbitrarias o especificar columnas a unirse.
- La condición de unión se separa de otras condiciones de búsqueda.
- La cláusula ON hace que el código fácil de entender.

Creando Joins con la cláusula ON


```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400
11	103	Hunold	60	60	1400

Retornando filas que no tienen coincidencia directa usando OUTER JOIN

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

Equijoin with EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst
. . .		
18	80	Abel
19	80	Taylor

There are no employees in department 190.

Employee “Grant” has not been assigned a department ID.

Producto cartesiano

- El producto cartesiano es una combinación de todas las filas de una tabla a cada fila de la otra tabla.
- Un producto cartesiano genera un gran número de filas y el resultado no suele ser útil.

EMPLOYEES (20 rows)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
. . .			
19	176	Taylor	80
20	178	Grant	(null)

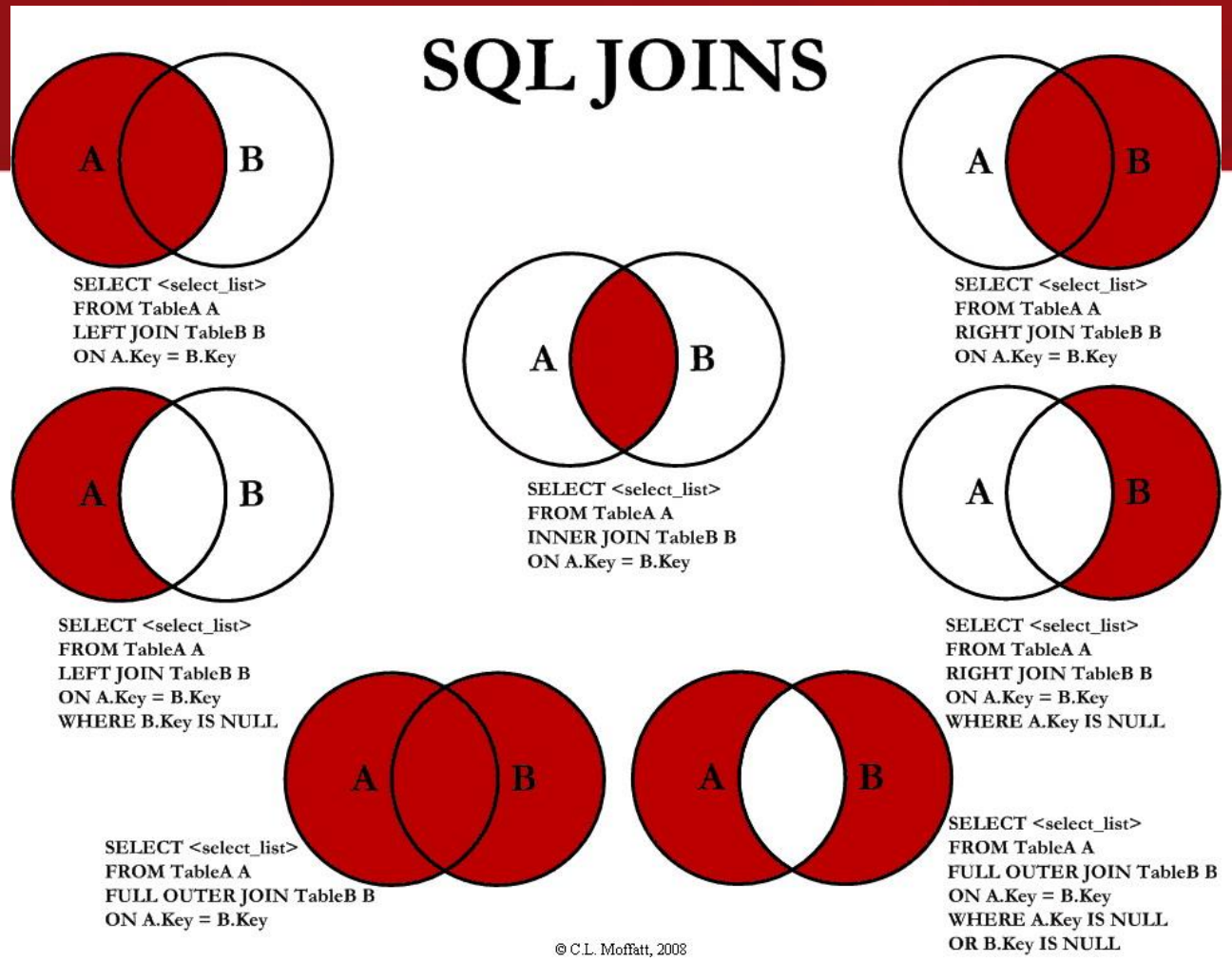
DEPARTMENTS (8 rows)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

Cartesian product:
20 x 8 = 160 rows

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10	1700
2	201	20	1700
. . .			
21	200	10	1800
22	201	20	1800
. . .			
159	176	80	1700
160	178	(null)	1700

Joins


SQL JOINS
A
B
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
© C.L. Moffatt, 2008

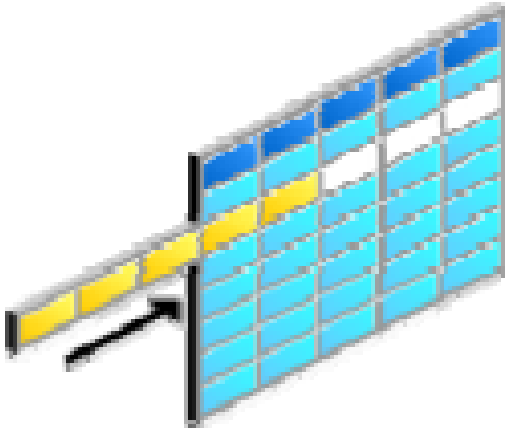
DML – DATA MANIPULATION LANGUAGE

DML

Data Manipulation Language

- DML statements query or manipulate data in the existing schema objects.
- A DML statement is executed when:
  - New rows are added to a table by using the `INSERT` statement
  - Existing rows in a table are modified using the `UPDATE` statement
  - Existing rows are deleted from a table by using the `DELETE` statement
- A *transaction* consists of a collection of DML statements that form a logical unit of work.

INSERT



Sentencia INSERT

INSERT Statement Syntax

- Add new rows to a table by using the INSERT statement:

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- With this syntax, only one row is inserted at a time.

Sentencia INSERT

Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally, list the columns in the `INSERT` clause.

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 rows inserted
```

- Enclose character and date values within single quotation marks.

Sentencia INSERT

Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
INSERT INTO   departments (department_id,
                           department_name)
VALUES        (30, 'Purchasing');
1 rows inserted
```

- Explicit method: Specify the NULL keyword in the VALUES clause.

```
INSERT INTO   departments
VALUES        (100, 'Finance', NULL, NULL);
1 rows inserted
```

Sentencia INSERT

Inserting Special Values

The SYSDATE function records the current date and time.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES                 (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       CURRENT_DATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 110);
```

```
1 rows inserted
```

Sentencia INSERT

Inserting Specific Date and Time Values

- Add a new employee.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 2003', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
1 rows inserted
```

- Verify your addition.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-03	SA_REP	11000	0.2	100

Sentencia INSERT

Copying Rows from Another Table

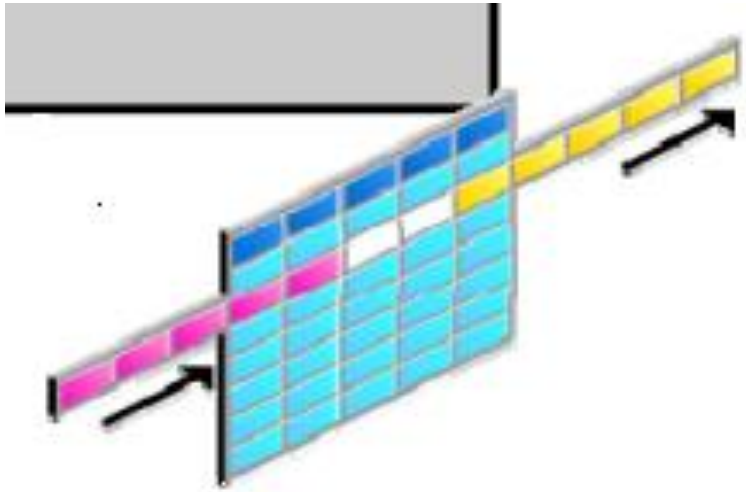
- Write your INSERT statement with a subquery:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

5 rows inserted.
```

- Do not use the VALUES clause.
- Match the number of columns in the INSERT clause to those in the subquery.
- Inserts all the rows returned by the subquery in the table, sales_reps.

UPDATE



Sentencia UPDATE

UPDATE Statement Syntax

- Modify existing values in a table with the UPDATE statement:

```
UPDATE        table
SET           column = value [, column = value, ...]
[WHERE        condition];
```

- Update more than one row at a time (if required).

Sentencia UPDATE

Updating Rows in a Table

- Values for a specific row or rows are modified if you specify the WHERE clause:

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
1 rows updated
```

- Values for all the rows in the table are modified if you omit the WHERE clause:

```
UPDATE   copy_emp
SET      department_id = 110;
22 rows updated
```

- Specify SET *column_name*= NULL to update a column value to NULL.

Sentencia UPDATE

Updating Two Columns with a Subquery

Update employee 103's job and salary to match those of employee 205.

```
UPDATE    employees
SET       (job_id,salary)   = (SELECT  job_id,salary
                               FROM    employees
                               WHERE   employee_id = 205)
          WHERE     employee_id     =  103;
```

```
1 rows updated
```

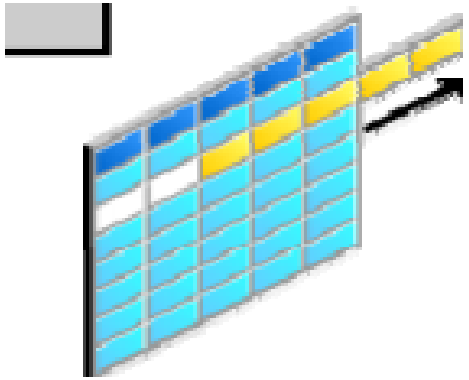
Sentencia UPDATE

Updating Rows Based on Another Table

Use the subqueries in the UPDATE statements to update row values in a table based on values from another table:

```
UPDATE   copy_emp
SET      department_id   =   (SELECT department_id
                              FROM employees
                              WHERE employee_id = 100)
WHERE    job_id          =   (SELECT job_id
                              FROM employees
                              WHERE employee_id = 200);
1 rows updated
```

DELETE



Sentencia DELETE

DELETE Statement

You can remove existing rows from a table by using the DELETE statement:

```
DELETE [FROM]   table
[WHERE          condition];
```

Sentencia DELETE

Deleting Rows from a Table

- Specific rows are deleted if you specify the WHERE clause:

```
DELETE FROM departments
WHERE  department_name = 'Finance';
1 rows deleted
```

- All rows in the table are deleted if you omit the WHERE clause:

```
DELETE FROM  copy_emp;
22 rows deleted
```

Sentencia DELETE

Deleting Rows Based on Another Table

Use the subqueries in the DELETE statements to remove rows from a table based on values from another table:

```
DELETE FROM employees
WHERE  department_id IN
                (SELECT department_id
                 FROM   departments
                 WHERE  department_name
                        LIKE '%Public%');
1 rows deleted
```

DATA DEFINITION LANGUAGE

Database Objects

Object	Description
Table	Is the basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative name to an object

Creating a Table Using a Subquery

- Create a table and insert rows by combining the `CREATE TABLE` statement and the `AS subquery` option.

```
CREATE TABLE table
            [(column, column...)]
AS subquery;
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

Creating a Table Using a Subquery

```
CREATE TABLE dept80
  AS
    SELECT  employee_id, last_name,
            salary*12 ANNSAL,
            hire_date
    FROM    employees
    WHERE   department id = 80;
```

```
table DEPT80 created.
```

```
DESCRIBE dept80
```

```
Name                          Null     Type
----------------------------- -------- -------------
EMPLOYEE_ID                            NUMBER(6)
LAST_NAME                     NOT NULL VARCHAR2(25)
ANNSAL                                 NUMBER
HIRE_DATE                     NOT NULL DATE
```

However, be sure to provide a column alias when selecting an expression. The expression `SALARY*12` is given the alias `ANNSAL`. Without the alias, the following error is generated:

```
Error starting at line 1 in command:
CREATE TABLE  dept80
  AS
    SELECT  employee_id, last_name,
            salary*12,
            hire_date
    FROM    employees
    WHERE   department_id = 80
Error at Command Line:4 Column:18
Error report:
SQL Error: ORA-00998: must name this expression with a column alias
00998. 00000 -  "must name this expression with a column alias"
*Cause:
*Action:
```

GESTIÓN DE TABLAS

Nombramiento de objetos

- La longitud de los nombres deben ser desde 1 a 30 bytes, con estas excepciones:
  - Nombres de base de datos están limitados a 8 bytes.
  - Nombres de database links pueden tener como longitud hasta 128 bytes.
- Nombres sin comillas pueden no ser palabras reservadas de Oracle.
- Nombres sin comillas deben comenzar con un carácter alfabético del juego de caracteres de la base de datos.
- Nombre con comillas no son recomendados.

Nombramiento de objetos

- Nombres sin comillas pueden contener solo:
  - Caracteres alfanuméricos del conjunto de caracteres de tu base de datos
  - Raya abajo (_)
  - Signo dólar (S)
  - Signo michi (#)
- Dos objetos no pueden tener el mismo nombre dentro del mismo esquema (espacio de nombre)

Tipos de datos en las tablas

- Tipo de datos comunes:
  - **CHAR(size [BYTE|CHAR])** : Fixed-length character data of size bytes or characters (max 255)
  - **VARCHAR2 (size [BYTE|CHAR])**: Variable-length character string having a maximum length of size bytes or characters (max 32,767)
  - **DATE**: Valid date range from January 1, 4712 B.C. through A.D. December 31, 9999
  - **NUMBER (p, s)**: Number with precision p and scale s (38 dígitos)

Otros tipos de datos

- BFILE
- BINARY_DOUBLE
- BINARY_FLOAT
- BLOB
- CLOB
- FLOAT
- INTERVAL
- LONG
- LONG RAW
- NCHAR
- NCLOB
- NVARCHAR2
- RAW
- ROWID
- TIMESTAMP
- UROWID

Creando una tabla


```
CREATE TABLE table_name(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
) TABLESPACE tablespace_name;
```

- **Nota**: la clausula **`TABLESPACE`** es opcional y se utiliza para crear la tabla en un tablespace diferente al asignado por defecto al usuario.

Ejemplo de creación de una tabla

```
• CREATE TABLE hr.employees2(
•     employee_id          NUMBER(6),
•     first_name           VARCHAR2(20),
•     last_name            VARCHAR2(25),
•     email                VARCHAR2(25),
•     phone_number         VARCHAR2(20),
•     hire_date            DATE DEFAULT SYSDATE,
•     job_id               VARCHAR2(10),
•     salary               NUMBER(8,2),
•     commission_pct       NUMBER (2,2),
•     manager_id           NUMBER(6),
•     department_id  NUMBER(4)
• );
```

¿Donde se guarda la data de la tabla?

```
SQL> SELECT file_id, block_id, blocks
  2  FROM dba_extents
  3  WHERE owner='HR'
  4  AND segment_name= 'EMPLOYEES2'
  5  AND segment_type='TABLE';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         5        200          8
```

Eliminando una tabla

- Cuando borramos una tabla eliminamos:
  - Datos
  - Estructura de tabla
  - Triggers de base de datos
  - Índices correspondientes
  - Privilegios asociados al objeto
- Clausulas opcionales para la sentencia **`DROP TABLE`**:
  - **`CASCADE CONSTRAINTS`**: restricciones dependientes de integridad referencial **`DROP TABLE hr.employees2 PURGE;`**
  - **`PURGE`**: no es posible flashback (no pasa por la papelera de reciclaje)

Truncando una tabla

- Truncando una tabla hace que sus filas de datos no estén disponibles, y opcionalmente libera el espacio usado.
- No se generan datos UNDO y el comando ejecuta un **COMMIT** implícito por ser un comando DDL
- Los índices correspondientes son truncados.
- Las tablas que son referenciadas por una llave foránea no pueden ser truncadas

```
TRUNCATE TABLE hr.employees2;
```

ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change table to read-only status

After you create a table, you may need to change the table structure for any of the following reasons:

- You omitted a column.
- Your column definition or its name needs to be changed.
- You need to remove columns.
- You want to put the table into read-only mode

You can do this by using the ALTER TABLE statement.

ALTER TABLE Statement

Use the ALTER TABLE statement to add, modify, or drop columns:

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
DROP (column [, column] …);
```

Adding a Column

- You use the ADD clause to add columns:

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
```

```
table DEPT80 altered.
```

- The new column becomes the last column:

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	149	Zlotkey	10500	29-JAN-08	(null)
2	174	Abel	11000	11-MAY-04	(null)
3	176	Taylor	8600	24-MAR-06	(null)

Modifying a Column

- You can change a column's data type, size, and default value.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
```

```
table DEPT80 altered.
```

- A change to the default value affects only subsequent insertions to the table.

GALAXY TRAINING

Dropping a Column

Use the DROP COLUMN clause to drop columns that you no longer need from the table:

```
ALTER TABLE  dept80
DROP (job_id);
```

```
table DEPT80 altered.
```

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	10500	29-JAN-08
2	174	Abel	11000	11-MAY-04
3	176	Taylor	8600	24-MAR-06

You can drop a column from a table by using the ALTER TABLE statement with the DROP COLUMN clause.

Guidelines

- The column may or may not contain data.
- Using the ALTER TABLE DROP COLUMN statement, only one column can be dropped at a time.
- The table must have at least one column remaining in it after it is altered.
- After a column is dropped, it cannot be recovered.
- A primary key that is referenced by another column cannot be dropped, unless the cascade option is added.
- Dropping a column can take a while if the column has a large number of values. In this case, it may be better to set it to be unused and drop it when there are fewer users on the system to avoid extended locks.

Note: Certain columns can never be dropped, such as columns that form part of the partitioning key of a partitioned table or columns that form part of the PRIMARY KEY of an index-organized table. For more information about index-organized tables and partitioned tables, refer to *Oracle Database Concepts* and *Oracle Database Administrator's Guide*.

Read-Only Tables

You can use the `ALTER TABLE` syntax to:

- Put a table in read-only mode, which prevents DDL or DML changes during table maintenance
- Put the table back into read/write mode

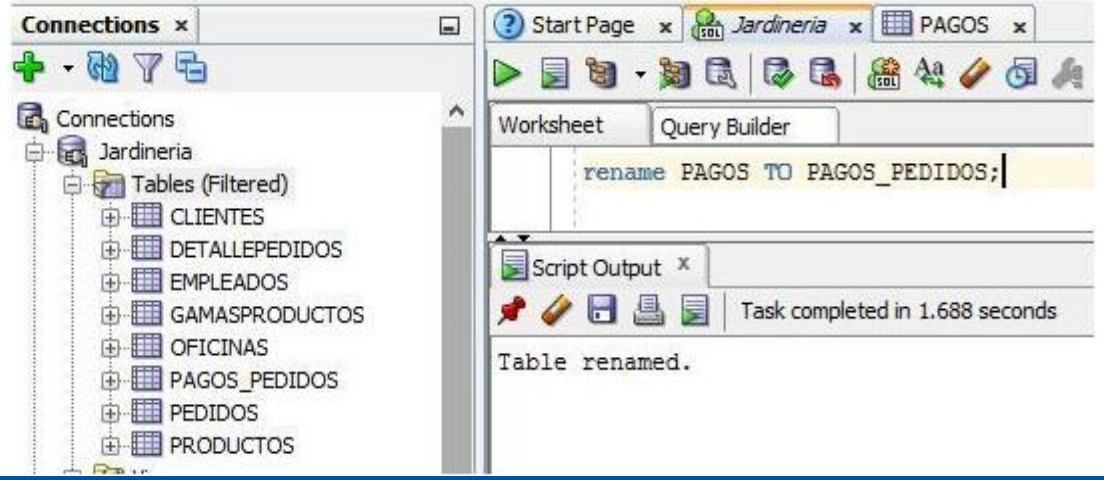
```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```

Renombrando una tabla


```
RENAME PAGOS TO PAGOS_PEDIDOS;
```



Obteniendo información de tablas

- Información acerca de las tablas se puede encontrar consultando:
  - DBA_TABLES
  - DBA_OBJECTS