Curso de SQL

Área de Medios y Tecnologías en Educación

Agenda del curso

- **Día 1:**
  - Reglas del juego
  - Iniciando con SQL
- **Día 2:**
  - SELECT
  - WHERE
- **Día 3:**
  - Funciones de agrupamiento (Single-row)
  - Funciones de agrupamiento (Multi-row)
- **Día 4,5:**
  - JOIN
- **Día 6,7:**
  - Group by
- **Día 8:**
  - Set Operations
  - Sub Queries
- **Día 9:**
  - DML, Insert, Update & Delete
- **Día 10:**
  - DDL
  - Data types
  - Managing tables

Perfil de ingreso

- El taller esta enfocado a toda persona que esta interesada en conocer, practicar e incrementar sus conocimientos en SQL.
- Se requiere contar con conocimientos de Introducción a la computación e Internet con WINDOWS, LINUX o MacOS X, así mismo conocer conceptos de Bases de datos relacionales y tener entendimiento de moleros Entidad relación y Diccionarios de datos.

Objetivos del taller

- Después de concluir el taller, serás capaz de :
  - Exportar e importar bases de datos en PostgreSQL
  - Seleccionar y proyectar información de las tablas de una base datos con sentencias SELECT
  - Crear reportes ordenados y con restricción de datos
  - Aplicar funciones SQL para generar y obtener datos modificados
  - Ejecutar consultas complejas para obtener información de diversas tablas
  - Ejecutar sentencias de manipulación de datos (DML) para modificar datos dentro de la base de datos
  - Ejecutar sentencias de definición de datos para crear esquemas, y objetos en los diferentes esquemas.

MEDIOS Y TECNOLOGÍAS en educación

División de Innovación Educativa

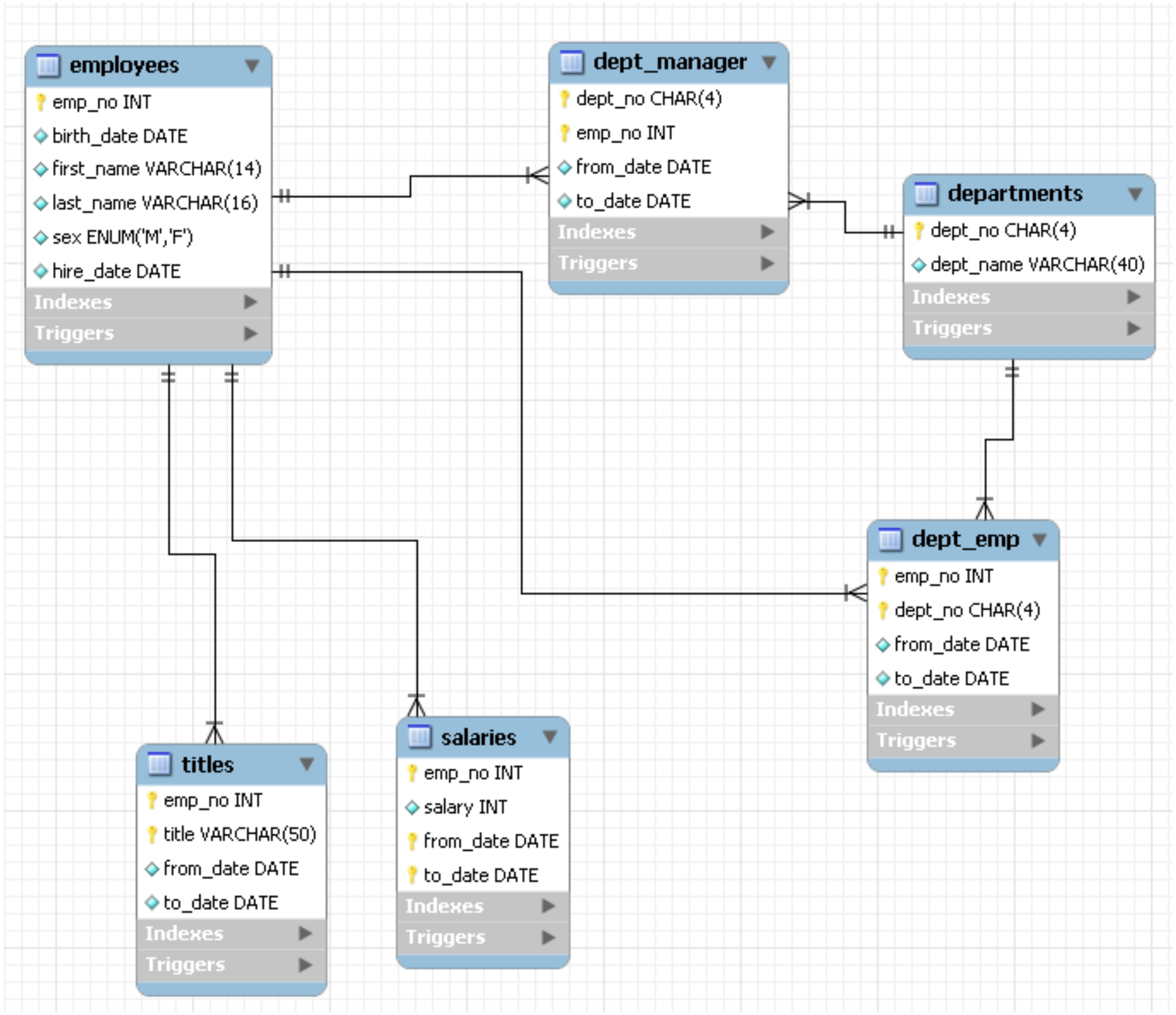
CES Coordinación de Educación en Salud

IMSS

Reglas del juego:

- Las sesiones se consideran de 90 minutos por 10 días (17:00 a 18:30)
- Por favor apagar sus celulares celulares
- Es un taller que tienen como única finalidad para mejorar nuestras habilidades de SQL
- Siempre profesionalismo
- **No hay evaluación**

Schema HR


employees
emp_no INT
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
sex ENUM('M','F')
hire_date DATE
Indexes
Triggers
dept_manager
dept_no CHAR(4)
emp_no INT
from_date DATE
to_date DATE
Indexes
Triggers
departments
dept_no CHAR(4)
dept_name VARCHAR(40)
Indexes
Triggers
dept_emp
emp_no INT
dept_no CHAR(4)
from_date DATE
to_date DATE
Indexes
Triggers
titles
emp_no INT
title VARCHAR(50)
from_date DATE
to_date DATE
Indexes
Triggers
salaries
emp_no INT
salary INT
from_date DATE
to_date DATE
Indexes
Triggers

DML, sentencia SELECT

Projección

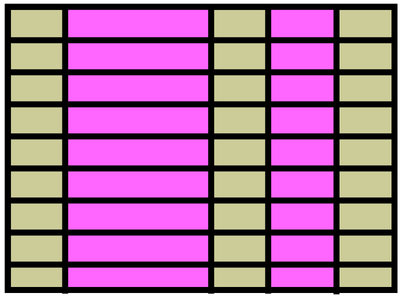


Tabla 1

Selección

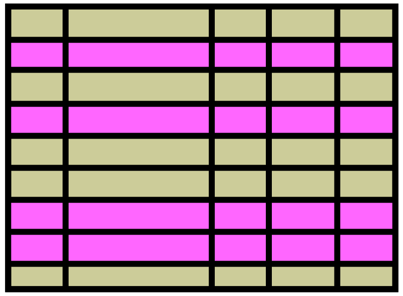


Tabla 1

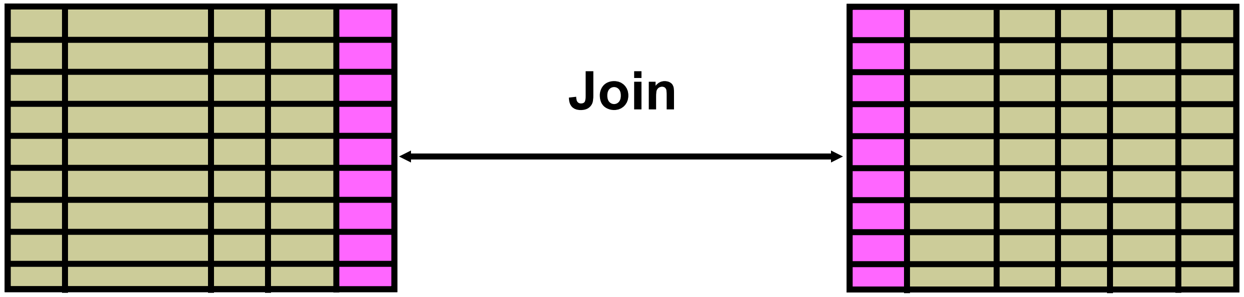


Tabla 1

Tabla 2

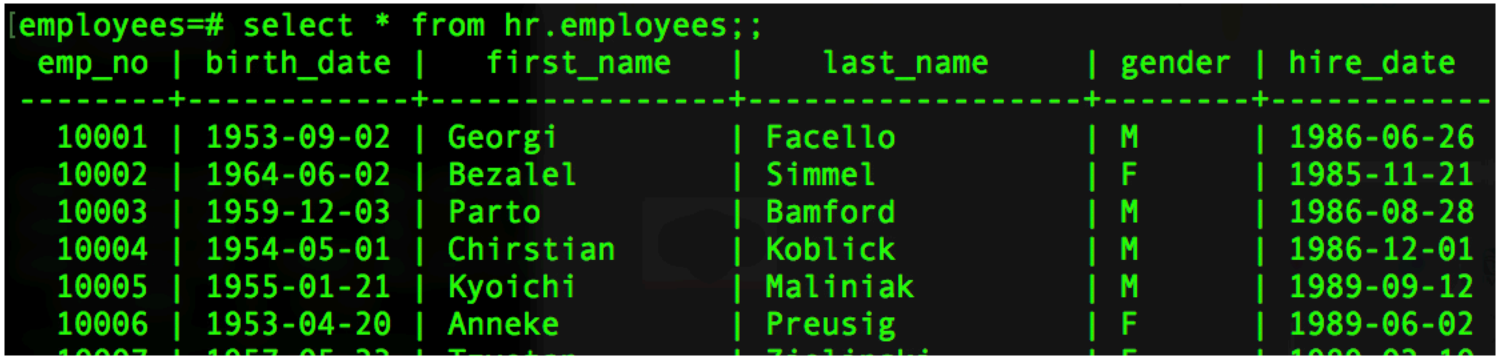
Sentencia SELECT básica

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

- SELECT identifica las columnas que serán mostradas
- FROM identifica la tabla que contiene las preguntas
- DISTINCT …
- ALIAS …

Seleccionando todas las columnas de una tabla

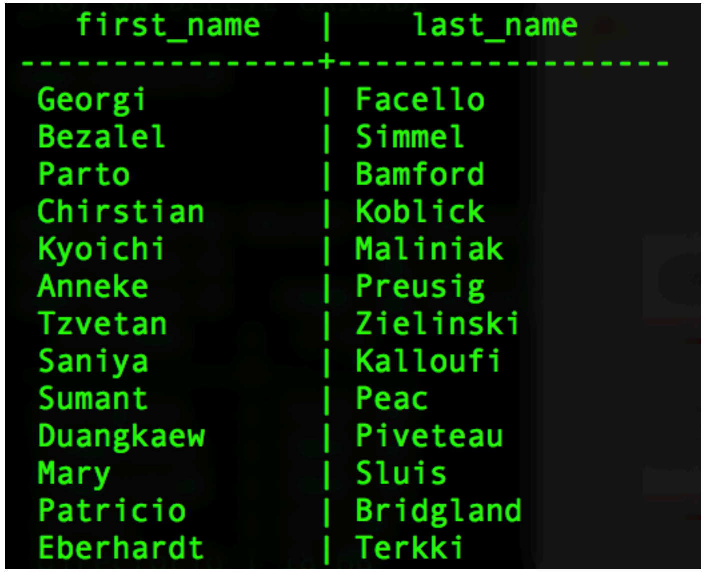
```
SELECT *
FROM   hr.employees;
```



```
[employees=# select * from hr.employees;;
 emp_no | birth_date |   first_name   |    last_name     | gender | hire_date
--------+------------+----------------+------------------+--------+-----------
  10001 | 1953-09-02 | Georgi         | Facello          | M      | 1986-06-26
  10002 | 1964-06-02 | Bezalel        | Simmel           | F      | 1985-11-21
  10003 | 1959-12-03 | Parto          | Bamford          | M      | 1986-08-28
  10004 | 1954-05-01 | Chirstian      | Koblick          | M      | 1986-12-01
  10005 | 1955-01-21 | Kyoichi        | Maliniak         | M      | 1989-09-12
  10006 | 1953-04-20 | Anneke         | Preusig          | F      | 1989-06-02
```

Seleccionando campos específicos

```
SELECT first_name, last_name “apellido paterno”
FROM   hr.employees;
```



```
  first_name    |     last_name
----------------+-----------------
 Georgi         | Facello
 Bezalel        | Simmel
 Parto          | Bamford
 Chirstian      | Koblick
 Kyoichi        | Maliniak
 Anneke         | Preusig
 Tzvetan        | Zielinski
 Saniya         | Kalloufi
 Sumant         | Peac
 Duangkaew      | Piveteau
 Mary           | Sluis
 Patricio       | Bridgland
 Eberhardt      | Terkki
```

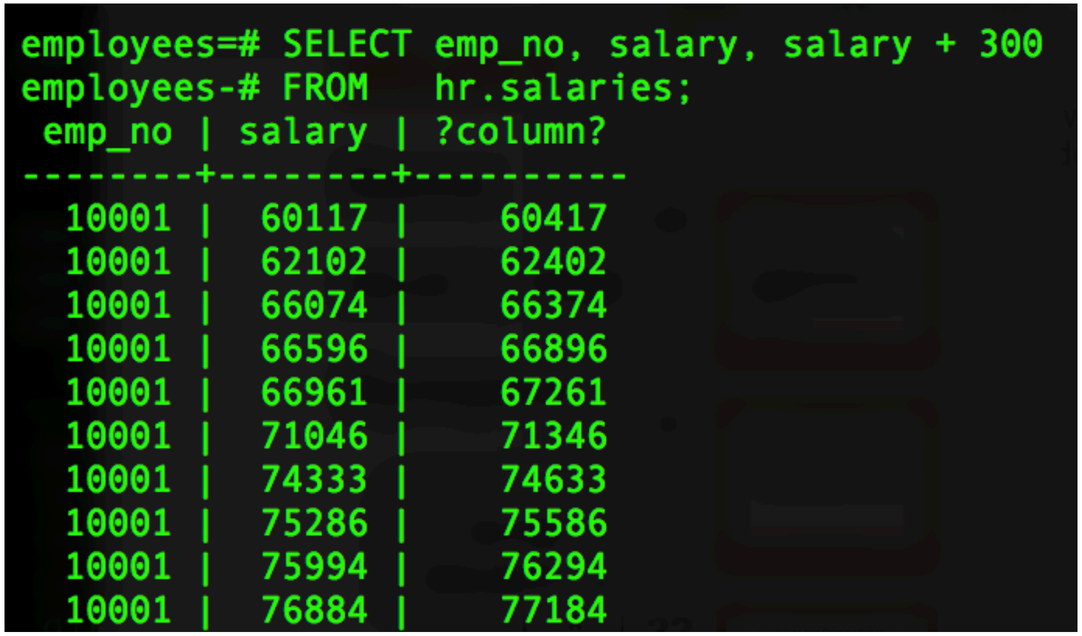
Expresiones aritméticas

- Crear una expresión con números y fechas, nos permite utilizar operadores aritméticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División

Usando Operadores aritméticos

```
SELECT emp_no, salary, salary + 300
FROM   hr.salaries;
```

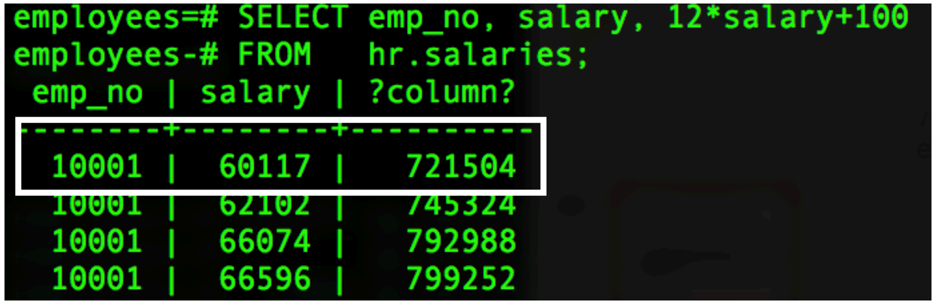


```
employees=# SELECT emp_no, salary, salary + 300
employees-# FROM   hr.salaries;
 emp_no | salary | ?column?
--------+--------+----------
  10001 |  60117 |    60417
  10001 |  62102 |    62402
  10001 |  66074 |    66374
  10001 |  66596 |    66896
  10001 |  66961 |    67261
  10001 |  71046 |    71346
  10001 |  74333 |    74633
  10001 |  75286 |    75586
  10001 |  75994 |    76294
  10001 |  76884 |    77184
```

Prioridad de los operadores

```
SELECT emp_no, salary, 12*salary+100
FROM   hr.salaries;
```

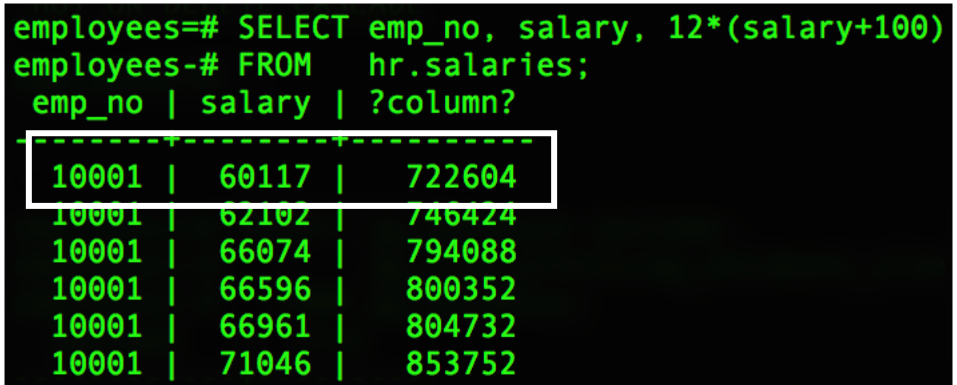
1



```
employees=# SELECT emp_no, salary, 12*salary+100
employees-# FROM   hr.salaries;
 emp_no | salary | ?column?
--------+--------+----------
  10001 |  60117 |   721504
  10001 |  62102 |   745324
  10001 |  66074 |   792988
  10001 |  66596 |   799252
```

```
SELECT emp_no, salary, 12*(salary+100)
FROM   hr.salaries;
```

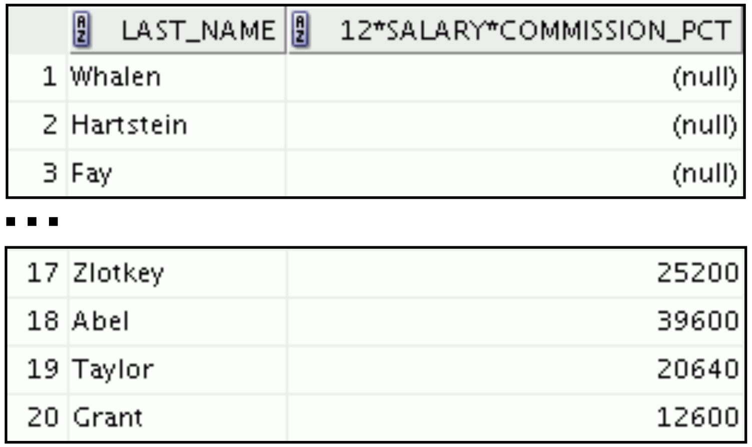
2



```
employees=# SELECT emp_no, salary, 12*(salary+100)
employees-# FROM   hr.salaries;
 emp_no | salary | ?column?
--------+--------+----------
  10001 |  60117 |   722604
  10001 |  62102 |   746424
  10001 |  66074 |   794088
  10001 |  66596 |   800352
  10001 |  66961 |   804732
  10001 |  71046 |   853752
```

Valores nulos con operaciones aritméticas

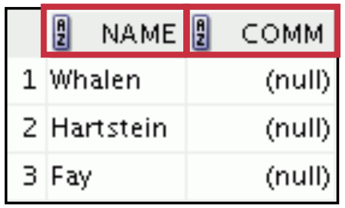
```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```



	LAST_NAME	12*SALARY*COMMISSION_PCT
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)
...		
17	Zlotkey	25200
18	Abel	39600
19	Taylor	20640
20	Grant	12600

Usando alias en columnas

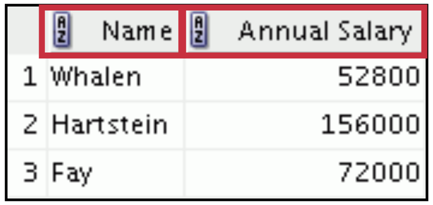
```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```



```
  NAME       COMM
1 Whalen     (null)
2 Hartstein  (null)
3 Fay        (null)
```

...

```
SELECT last_name "Name" , salary*12 "Annual Salary"
FROM   employees;
```



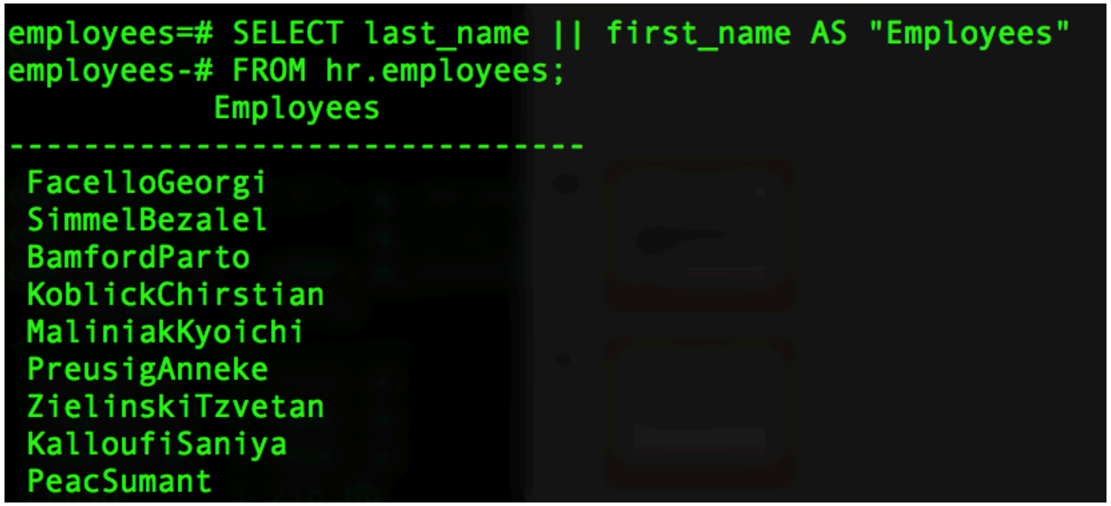
```
  Name       Annual Salary
1 Whalen            52800
2 Hartstein        156000
3 Fay               72000
```

...

Operador de concatenación

- Une el contenido de dos columnas
- Es representado por las barras verticales / pipes(||)

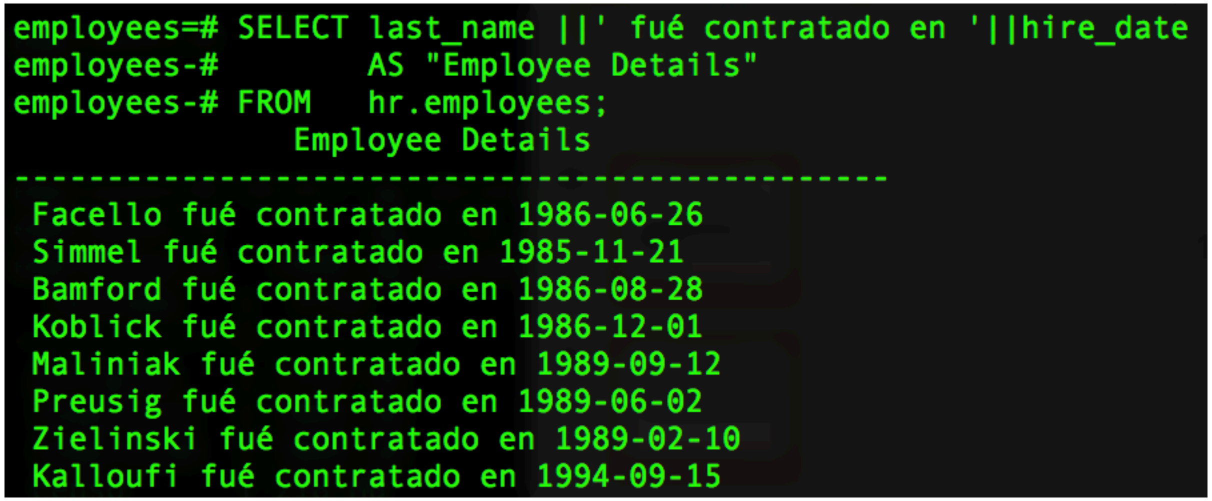
```
SELECT   last_name || first_name AS "Employees"
FROM     hr.employees;
```



```
employees=# SELECT last_name || first_name AS "Employees"
employees-# FROM hr.employees;
           Employees
-------------------------------
 FacelloGeorgi
 SimmelBezalel
 BamfordParto
 KoblickChirstian
 MaliniakKyoichi
 PreusigAnneke
 ZielinskiTzvetan
 KalloufiSaniya
 PeacSumant
```

Caracteres literales

```
SELECT last_name ||' fué contratado en '||hire_date
       AS "Employee Details"
FROM   hr.employees;
```



```
employees=# SELECT last_name ||' fué contratado en '||hire_date
employees-#        AS "Employee Details"
employees-# FROM   hr.employees;
               Employee Details
-----------------------------------------------
 Facello fué contratado en 1986-06-26
 Simmel fué contratado en 1985-11-21
 Bamford fué contratado en 1986-08-28
 Koblick fué contratado en 1986-12-01
 Maliniak fué contratado en 1989-09-12
 Preusig fué contratado en 1989-06-02
 Zielinski fué contratado en 1989-02-10
 Kalloufi fué contratado en 1994-09-15
```

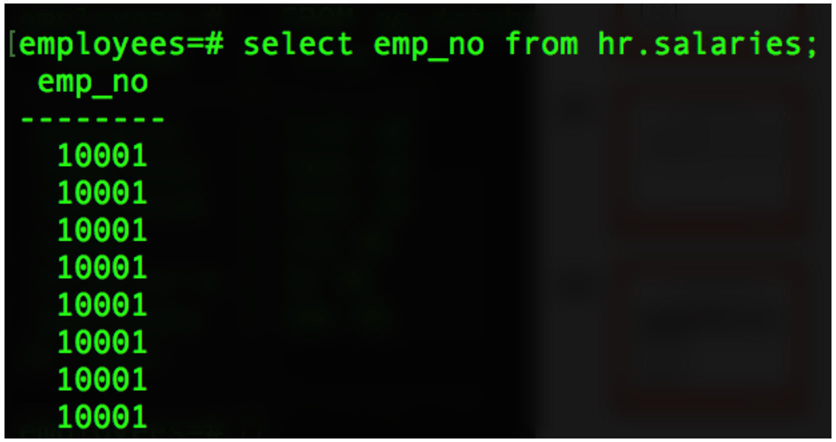
Duplicidad de datos

1

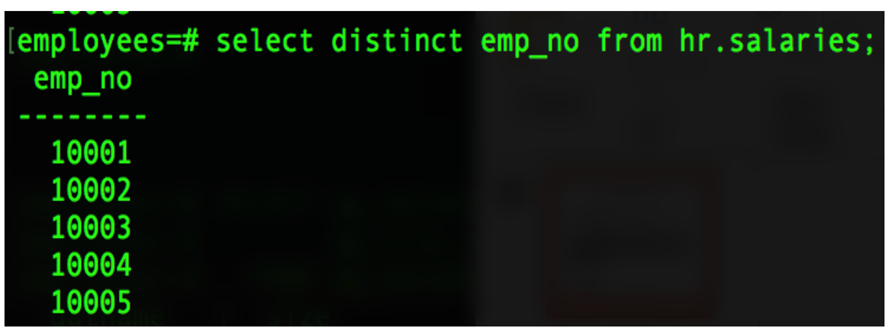
```
SELECT  emp_no
FROM    hr.salaries;
```

2

```
SELECT DISTINCT emp_no
FROM    hr.salaries;
```

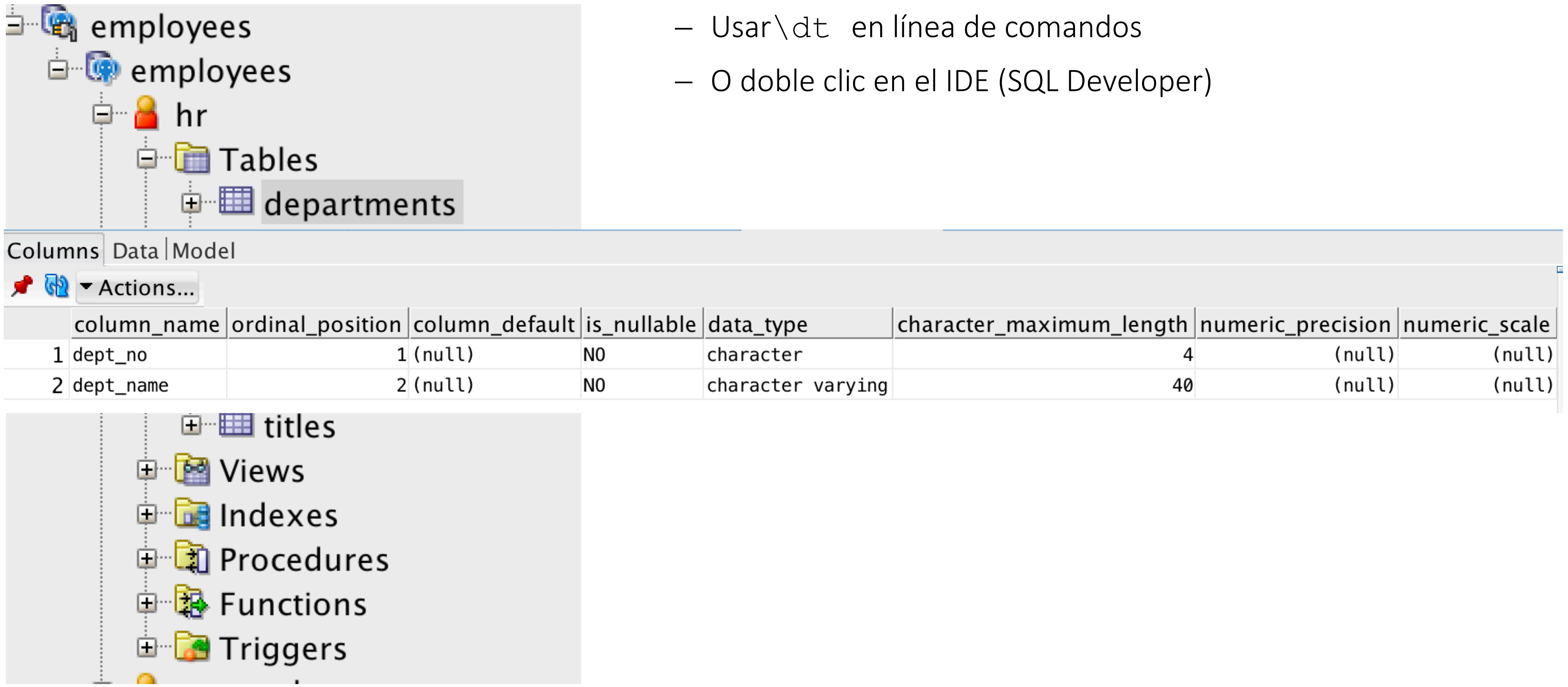


```
[employees=# select emp_no from hr.salaries;
  emp_no
---------
   10001
   10001
   10001
   10001
   10001
   10001
   10001
   10001
```



```
[employees=# select distinct emp_no from hr.salaries;
 emp_no
--------
  10001
  10002
  10003
  10004
  10005
```

Viendo la estructura de una tabla



- Usar `\dt` en línea de comandos
- O doble clic en el IDE (SQL Developer)

	column_name	ordinal_position	column_default	is_nullable	data_type	character_maximum_length	numeric_precision	numeric_scale
1	dept_no	1	(null)	NO	character	4	(null)	(null)
2	dept_name	2	(null)	NO	character varying	40	(null)	(null)

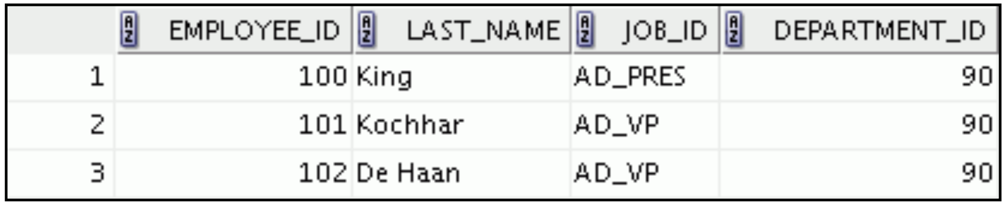
Limitando renglones

– Usando WHERE nos permite restringir los renglones de la selección

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM   table
[WHERE condition(s)];
```

Usando WHERE

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Character Strings and Dates

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

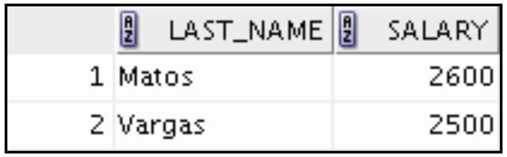
```
SELECT last_name
FROM   employees
WHERE  hire_date = '1985-11-21' ;
```

Operadores de comparación

Operador	Significado
=	Igualdad
>	Mayor que
>=	Mayor igual que
<	Menor que
<=	Menor igual que
<>	Diferente de
`BETWEEN ...AND...`	Rango
`IN(set)`	Coincidencia en un conjunto
`LIKE`	Coincidencia en un patrón
`IS NULL`	Es nulo

Usando operadores de comparación

```
SELECT emp_no salary
FROM   hr.salaries
WHERE  salary <= 3000 ;
```



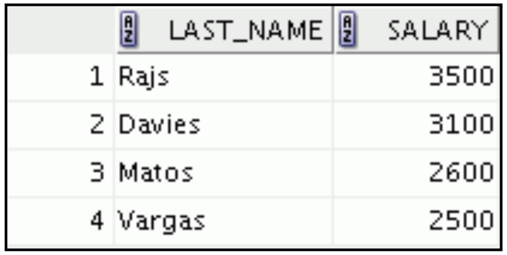
	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

Range Conditions Using the BETWEEN Operator

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

```
SELECT emp_no, salary
FROM   hr.salaries
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit   Upper limit

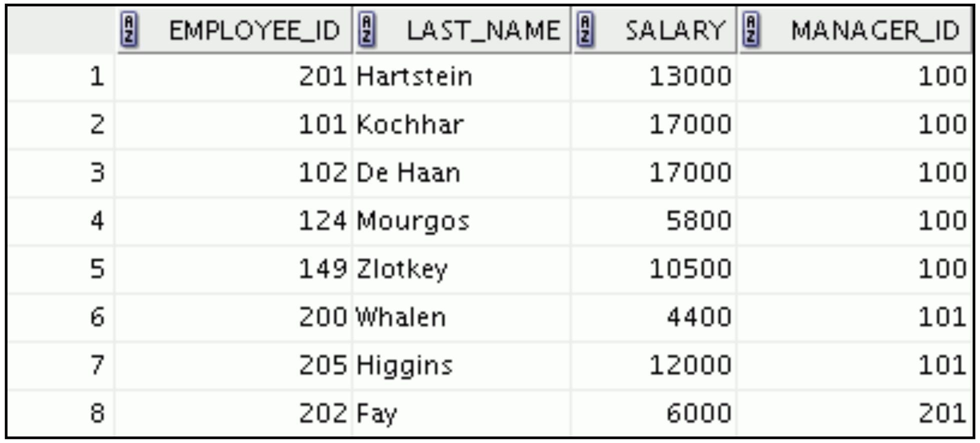


	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

Membership Condition Using the IN Operator

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

```
SELECT em_no, last_name
FROM   hr.employees
WHERE  emp_no IN (100, 101, 201) ;
```



	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	201	Hartstein	13000	100
2	101	Kochhar	17000	100
3	102	De Haan	17000	100
4	124	Mourgos	5800	100
5	149	Zlotkey	10500	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

Pattern Matching Using the `LIKE` Operator

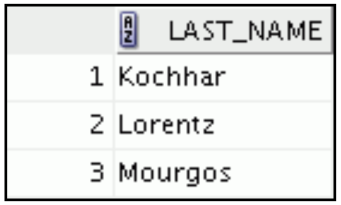
- Use the `LIKE` operator to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
  - `%` denotes zero or many characters.
  - `_` denotes one character.

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

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%' ;
```



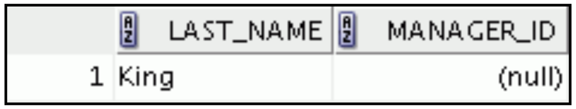
	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

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

Using the 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)

Defining Conditions Using the Logical Operators

Operador	Significado
`AND`	Returns `TRUE` if *both* component conditions are true
`OR`	Returns `TRUE` if *either* component condition is true
`NOT`	Returns `TRUE` if the condition is false

Using the AND Operator

- AND requires both the component conditions to be true:

```
SELECT emp_no, last_name
FROM   employees
WHERE  emp_no >= 10014
AND    last_name LIKE '%cel%'  ;
```

Using the OR Operator

- OR requires either component condition to be true:

```
SELECT emp_no, last_name
FROM   employees
WHERE  emp_no >= 10000
         OR last_name LIKE '%cel%' ;
```

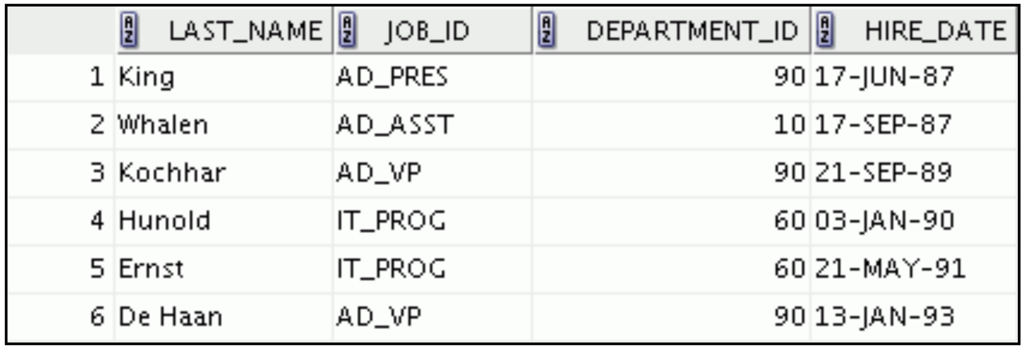
Using the NOT Operator

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

Using the ORDER BY Clause

– Sort the retrieved rows with the ORDER BY clause:
  – ASC: Ascending order, default
  – DESC: Descending order
– The ORDER BY clause comes last in the SELECT statement:

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



	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

...

Sorting

– Sorting in descending order:

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

1

```
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

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

4