

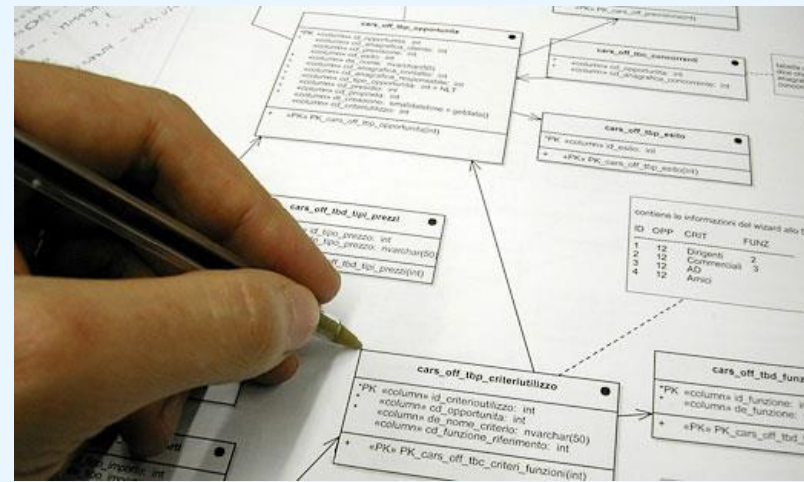
Encuentro 02/03 – Parte 02

Introducción a SQL

[illegible]

Que es SQL?

- Es un Lenguaje de Consulta para el Modelo Relacional
- Es utilizado por (casi) todos los Manejadores de Base de Datos de la actualidad
- No solo permite hacer consultas a los datos sino también modificaciones
- Inclusive se utiliza para crear las estructuras de datos (tablas, claves, índices, vistas, triggers, etc)
- Es un lenguaje “Declarativo” de alto nivel.

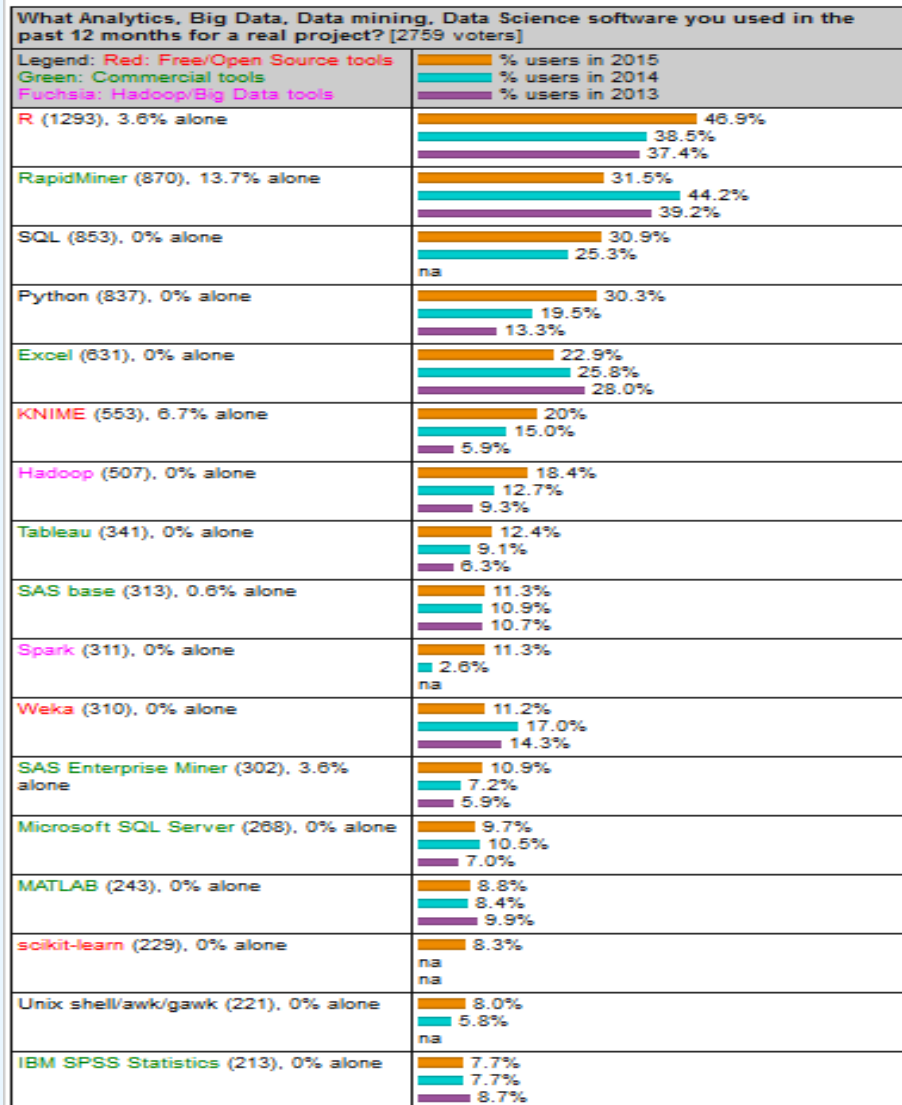


Vamos a trabajar con querys...

- Una Query es una consulta que se hace a la base de datos.
- Las queries se escriben en SQL (usando SELECT)
- En nuestro ejemplo podemos ver 2 usuarios ejecutando 2 queries o consultas a la base de datos.
- SQL también permite Modificar (usando UPDATE, DELETE e INSERT)



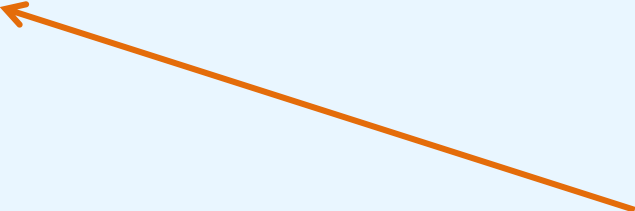
Porque SQL y no SPSS?



- Decrece el uso de herramientas licenciadas.
- Tendencia en el mercado a inclinarse por SQL + herramientas para la capa de presentación + herramientas para la generación de modelos/conocimiento.

En que consiste SQL?

- El Lenguaje SQL se divide en 2 grandes grupos
- DDL (Data Definition Language)
 - CREATE / DROP / ALTER
- DML (Data Manipulation Language)
 - SELECT / INSERT / UPDATE / DELETE



**Nos vamos a centrar en la
instrucción SELECT**

Veamos como podemos crear la tabla PROFESOR mediante SQL

```
CREATE TABLE PROFESOR (  
    LEGAJO INTEGER,  
    NOMBRE CHAR(30),  
    FEC_NAC DATE,  
    DEPTO CHAR(10),  
    PRIMARY KEY (LEGAJO)  
)
```

PROFESOR

Legajo	Nombre	Fec_Nac	Depto

DDL (II)

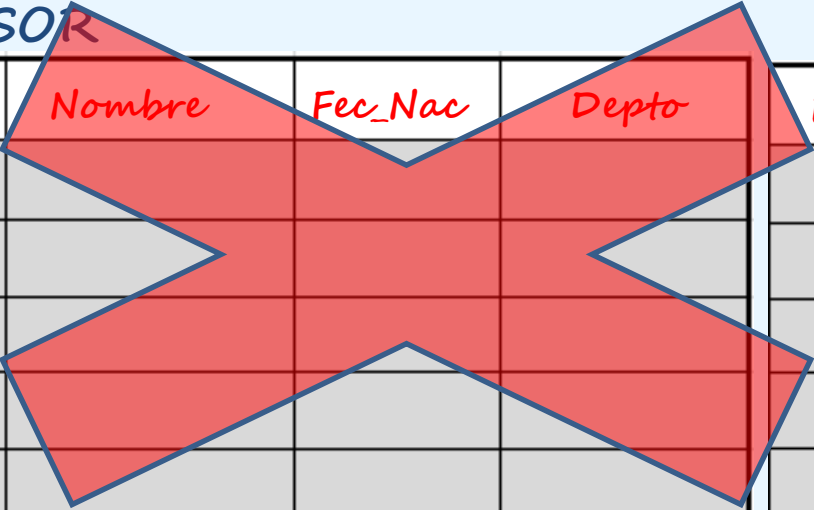
También disponemos de las instrucciones
DROP TABLE y **ALTER TABLE** para borrar y modificar
una tabla

ALTER TABLE PROFESOR ADD COLUMN DNI Integer;

DROP TABLE PROFESOR;

PROFESOR

<i>Legajo</i>	<i>Nombre</i>	<i>Fec_Nac</i>	<i>Depto</i>	<i>DNI</i>



- Ahora que sabemos crear una tabla podemos volver al DML para ver como manipular datos
- DML (Data Manipulation Language)
 - INSERT : Crea 1 (o varias) fila(s) en una tabla
 - UPDATE : Modifica los datos de 1 (o varias) fila(s)
 - DELETE : Borra 1 (o varias) filas(s)
 - SELECT : Leer datos de una tabla (o más tablas)

INSERT -SQL-

- INSERT : Crea una fila en una tabla

INSERT INTO PROFESOR

VALUES (14567, 'Juan Pérez', '04/04/1973','Química')

PROFESOR

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química

MATERIA

Código	Nombre	Inscrip

UPDATE & DELETE

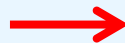
UPDATE : Modifica Filas

DELETE : Borra Filas

Ambas utilizan un WHERE para identificar las Filas que van a ser afectadas.

PROFESOR

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química
12447	María López	6/3/1971	Física
13987	José Cito	NULL	Química



**DELETE FROM PROFESOR
WHERE Depto = 'Física'**



**UPDATE FROM PROFESOR
WHERE Legajo = 14567 SET
Nombre = 'Juan M. Perez'**

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química
13987	José Cito	NULL	Química

INSTRUCCIÓN SELECT



Nuestro interés reside en la instrucción **SELECT** ya que es la que nos va a permitir manipular/recuperar la información.

SELECT -SQL-

- **SELECT:** Lee y devuelve filas de una tabla (o más tablas)

```
SELECT columna1, columna2, columna3  
FROM tabla;
```

Los nombres de las columnas y la tabla deben ser las del modelo de datos.

Si quiero seleccionar todas las columnas uso el “*” en vez de detallar las columnas.

SELECT -SQL-

```
SELECT Legajo, Nombre  
FROM PROFESOR;
```



<i>Legajo</i>	<i>Nombre</i>
14567	Juan Perez
12447	María López
13987	José Cito

PROFESOR

<i>Legajo</i>	<i>Nombre</i>	<i>Fec_Nac</i>	<i>Depto</i>
14567	Juan Perez	4/4/1971	Química
12447	María López	6/3/1971	Física
13987	José Cito	NULL	Química

MATERIA

<i>Código</i>	<i>Nombre</i>	<i>Inscrip</i>
ANA2	Análisis II	71
QUI1	Química I	20

SELECT -SQL-

```
SELECT *  
FROM MATERIA;
```

MATERIA

Código	Nombre	Inscrip
ANA2	Análisis II	71
QUI1	Química I	20



PROFESOR

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química
12447	María López	6/3/1971	Física
13987	José Cito	NULL	Química

MATERIA

Código	Nombre	Inscrip
ANA2	Análisis II	71
QUI1	Química I	20

- En general, vamos a necesitar establecer “filtros” (condiciones) en los datos que queremos obtener.
- Esto lo hacemos con la palabra WHERE de la siguiente manera:

```
SELECT    columna1, columna2  
WHERE condición;
```

Una condición es una restricción que deben cumplir los datos para ser parte del resultado.

Por ejemplo: Estudiantes entre 18-24 años de San Miguel.

WHERE - EJEMPLO

- WHERE : Filtra algunas de las filas de la tabla

```
SELECT Nombre  
FROM MATERIA  
WHERE inscrip > 20;
```



Nombre
Análisis II

PROFESOR

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química
12447	María López	6/3/1971	Física
13987	José Cito	NULL	Química

MATERIA

Código	Nombre	Inscrip
ANA2	Análisis II	71
QUI1	Química I	20

En las condiciones nos vamos a ayudar con los:

- Operadores lógicos:
 - **AND** -y-,
 - **OR** -o-,
 - **NOT** -no-.
- Operadores relacionales (=, <>, <, <=, >=).

OPERADORES LÓGICOS



Los operadores lógicos nos sirven para encadenar condiciones:

COND1 AND COND2

(se deben cumplir ambas condiciones para que la instancia sea devuelta como resultado)

SELECT apellido_nombre, correo_electronico

FROM estudiantes

WHERE anio_ingreso = 2019 **AND** sede = 'C.R. SAN MIGUEL';

(devuelve el nombre y correo de los ingresantes 2019 de San Miguel)

OPERADORES LÓGICOS



Los operadores lógicos nos sirven para encadenar condiciones:

COND1 OR COND2

(Alcanza con que se cumpla una de las condiciones para que la instancia sea devuelta como resultado)

SELECT apellido_nombre, correo_electronico

FROM estudiantes

WHERE anio_ingreso = 2019 **OR** anio_ingreso = 2018;

(devuelve el nombre y correo de los ingresantes 2018 y 2019)

OPERADORES LÓGICOS



Los operadores lógicos nos sirven para encadenar condiciones:

NOT(COND)

(La condición no debe cumplirse para ser parte del resultado de la consulta)

```
SELECT apellido_nombre, correo_electronico
```

```
FROM estudiantes
```

```
WHERE NOT(anio_ingreso = 2019);
```

(devuelve el nombre y correo de todos los estudiantes excepto los de año de ingreso 2019)

OPERADORES RELACIONALES



Los operadores relacionales nos sirven para armar condiciones determinando relaciones entre el atributo y otro valor:

Columna1 op VALOR

```
SELECT apellido_nombre, correo_electronico  
FROM estudiantes  
WHERE (anio_ingreso > 2010);
```

(devuelve el nombre y correo de todos los estudiantes con año de ingreso mayor a 2010)

Qué devuelve cambiando de operador? (<, >, <>, etc)

INSTRUCCIÓN LIKE

- Se utiliza como condición en el WHERE para verificar si un texto es «parecido» al texto por el que se está preguntando.

```
1 SELECT *
2 FROM estudiantes e
3 WHERE apellido_nombre LIKE 'FERNAND%';
```

El % indica que se ignora lo que el texto lo precede/antecede.

	legajo integer	apellido... text	sexo text	fecha_n... date	tipo_doc... text	numero_... integer	nacionali... text	estado_... text	carrera text	plan_est... integer	se te
<input type="checkbox"/>	87	FERNAND...	F	1959-07-...	DOC. NA...	12881718	ARG. NAT....	SOLTERO/A	INGENIE...	105	SE
<input type="checkbox"/>	232	FERNAND...	F	1963-06-...	DOC. NA...	16983005	ARG. NAT....	SOLTERO/A	INGENIE...	105	SE
<input type="checkbox"/>	752	FERNAND...	M	1955-12-...	DOC. NA...	12078689	ARG. NAT....	SOLTERO/A	LICENCIA...	308	SE
<input type="checkbox"/>	753	FERNAND...	F	1960-05-...	DOC. NA...	13982716	ARG. NAT....	SOLTERO/A	LICENCIA...	308	SE
<input type="checkbox"/>	754	FERNAND	M	1961-11-	DOC. NA	14490481	ARG. NAT	SOLTERO/A	LICENCIA	308	SE

Por ahora, todas nuestras consultas fueron sobre una sola tabla, pero como hacemos para recuperar las relaciones?

JOINS: INNER, LEFT, RIGHT

INNER JOIN



- Los problemas de la realidad requieren datos de diversas relaciones.
- Es una de las instrucciones mas complejas y a su vez, utilizadas.

Se entrecruzan todas las instancias de ambas tablas de acuerdo a una clave.

INNER JOIN

Se escribe de la siguiente manera:

```
SELECT    tabla1.columna1,  
            tabla2.columna1
```

```
FROM tabla1
```

```
INNER JOIN tabla2 ON tabla1.foranea=tabla2.clave;
```

Indicamos el atributo a partir del cual se relacionan (puede ser mas de 1)

Va a relacionar todas las instancias de la tabla1 con las de la tabla2 en que coincidan las claves.

INNER JOIN (II)



Selección de los correos de todos los estudiantes que rindieron examen en Marzo-97:

#1 SELECT legajo, correo_electronico FROM estudiantes;

#2 SELECT legajo, fecha_examen FROM finales

WHERE fecha_examen >= '01-03-1997' and fecha_examen < '01-04-1997';

```
1  SELECT  correo_electronico,
2          fecha_examen
3  FROM  estudiantes e
4  INNER JOIN finales f ON e.legajo = f.legajo
5  WHERE fecha_examen >= '01-01-1997' and fecha_examen <= '01-03-1997';
```

INNER JOIN (II)



SELECT legajo, correo_electronico
FROM estudiantes;

Legajo	Correo_electronico
60925	rosanar@unlu.edu.ar
840	moloriz@unlu.edu.ar
89937	jmfernandez@unlu.edu.ar

SELECT legajo, fecha_examen
FROM finales
WHERE fecha_examen >= '01-03-1997' and ...;

Legajo	Fecha_examen
60925	05-03-1997
60925	12-03-1997
840	14-03-1997

RESULTADO:

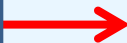
Correo_electronico	Fecha_examen
rosanar@unlu.edu.ar	05-03-1997
rosanar@unlu.edu.ar	12-03-1997
moloriz@unlu.edu.ar	14-03-1997

```
1 SELECT correo_electronico,  
2     fecha_examen  
3 FROM estudiantes e  
4 INNER JOIN finales f ON e.legajo = f.legajo  
5 WHERE fecha_examen >= '01-01-1997' and fecha_examen <= '01-03-1997';
```

ORDER BY

- Permite ordenar las filas que retorna el SELECT con algun criterio

```
SELECT Legajo, Nombre, Depto  
FROM PROFESOR  
ORDER BY Legajo;
```



Legajo	Nombre	Depto
12447	María López	Física
13987	José Cito	Química
14567	Juan Perez	Química

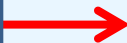
PROFESOR

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química
12447	María López	6/3/1971	Física
13987	José Cito	NULL	Química

LIMIT

- LIMIT X: Devuelve las X primeras filas que cumplen con la consulta.

```
SELECT Legajo, Nombre, Depto  
FROM PROFESOR  
ORDER BY Legajo  
LIMIT 2;
```



Legajo	Nombre	Depto
12447	María López	Física
13987	José Cito	Química

PROFESOR

Legajo	Nombre	Fec_Nac	Depto
14567	Juan Perez	4/4/1971	Química
12447	María López	6/3/1971	Física
13987	José Cito	NULL	Química

SQL



FIN

