

Gestión de Datos para Robótica

T1f - Lenguaje SQL

Álvaro Vázquez Álvarez
Departamento de Electrónica e Computación

✉ alvaro.vazquez@usc.es

📍 Pabellón III - Despacho 4

Curso 2023-2024

Tabla de contenidos

- Introducción
- SQL
 - Tipos de datos
 - SQL como DDL
 - SQL como DML
 - SQL como DCL
- Bibliografía

Introducción

SQL (Structured Query Language, Lenguaje Estructurado de Consultas)

- Que permite expresar operaciones diversas (aritméticas, combinatorias, lógicas, selección y ordenación) con datos almacenados en bases de datos relacionales.
- Originalmente, SQL se llamaba SEQUEL y fue diseñado e implementado por IBM Research como interfaz para un SGBD relacional experimental denominado SYSTEM-R
- Actualmente es un estándar ANSI ISO (SQL:2003), pero:
 - Las compañías comerciales crean sus propios dialectos
 - SQL Oracle \neq SQL MySQL
- Ha sido aceptado como lenguaje de facto.
- Está basado en el álgebra relacional
- Sentencias SQL: permiten enunciar operaciones sobre tablas.
 - **DML:** Sentencias que permiten realizar consultas y actualizaciones de datos (inserción, borrado y modificación de filas):
 - SELECT, INSERT, UPDATE, DELETE.
 - **DDL:** Lenguaje de Definición de Datos (Data Definition Language): Sentencias que permiten definir nuevos objetos (tablas, índices, claves, etc) o destruir los ya existentes:
 - CREATE, DROP, ALTER.
 - **DCL:** Sentencias de Control de Datos (Data Control Language): Sentencias que permiten controlar aspectos

Introducción

En resumen, SQL es un lenguaje utilizado por SGBD relacionales que permite:

- Consultar y actualizar datos (DML)
- Definir y destruir objetos de la base de datos (DDL)
- Conceder y denegar autorizaciones para usar estos objetos (DCL)



Tipos de Datos

- SQL proporciona tipos de datos predefinidos, aunque también permite definir nuevos tipos de datos conocidos como *Tipos de Datos Distintos definidos por el Usuario*.
- En el momento de asignar un tipo de datos a una columna, se está definiendo:
 - **Conjunto** de todos los **valores** posibles que puede tomar la columna.
 - Las **operaciones** que se pueden realizar con los valores de la columna.
- El estándar SQL ANSI/ISO especifica varios tipos de datos que pueden ser almacenados en una base de datos basada en SQL y manipulados por el lenguaje SQL (conjunto mínimo).
 - Casi todos los productos comerciales soportan el conjunto mínimo y además especifica conjuntos de datos más amplios.
- Existen cuatro tipos de datos:
 - Numéricos
 - Texto
 - Fecha
 - Binarios

Tipos de Datos: *Numérico*

- Números enteros:
 - INT: Normalmente 32 bits (-2147483648 a +2147483647)
 - SMALLINT: Normalmente 16bits (-32768 y +32767)
- Números en coma flotante
 - FLOAT (*precision*)
 - REAL
 - DOUBLE PRECISION
- Números con formato:
 - DECIMAL (precis, [,decim])
 - NUMERIC (precis, [,decim])

Tipos de Datos: *Texto*

- CHAR(long): Admite letras, números o caracteres especiales. Internamente, se almacena en formato de longitud fija. Su longitud máxima es de 2000 caracteres. CHAR es equivalente a CHAR(1)
- VARCHAR(long): Admite letras, números o caracteres especiales. Se almacena en un formato de longitud variable. Su longitud máxima es de 4000 caracteres.

Las cadenas se representan entre comillas simples



Tipos de Datos: *Fecha*

- DATE: campo de longitud fija de 7 bytes, que se utiliza para almacenar datos temporales, lo que incluye la fecha (año, mes, día) y hora (hora, minutos y segundos, e incluso fracciones de segundo). Es importante tener esto en cuenta cuando se desea comparar dos fechas.
 - El formato predeterminado de fecha es YYYY-MM-DD, donde
 - YYYY es el año,
 - MM representa el mes
 - DD indica el día del mes.
 - Permite almacenar fechas desde el 1 de enero del 4712 a.C. hasta el 31 de diciembre del 4712 d.C.

Tipos de Datos: *Binarios*

- BLOB: Es un objeto binario de gran tamaño, siendo el tamaño máximo 4 GB (gigabytes). Normalmente un blob se utiliza para almacenar una imagen, datos de voz, o cualquier otro bloque de datos grande no estructurado.

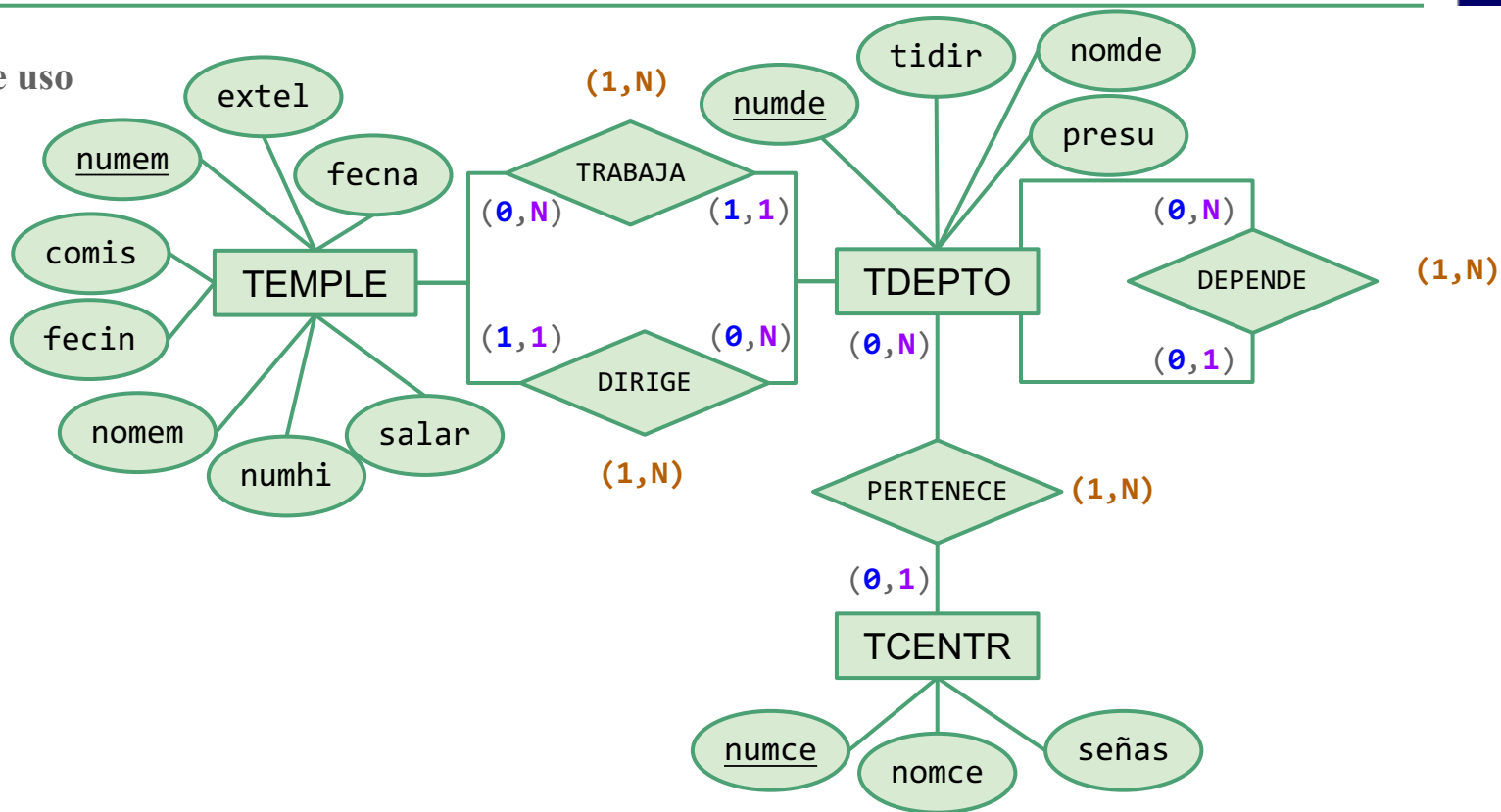
TODOS LOS TIPOS DE DATOS INCLUYEN EL VALOR NULO!



Tipos de Datos: *Operadores de comparación*

- Los operadores de comparación son: =, <, >, <>, <=, >=
 - Si uno de los valores que se compara es Nulo, el resultado de la comparación también lo es.
 - Sólo se pueden comparar valores homogéneos.
 - Todos los tipos de datos numéricos pueden ser comparados unos con otros. Lo mismo para tipos de datos de texto.
 - Dos cadenas vacías se consideran iguales.

Caso de uso



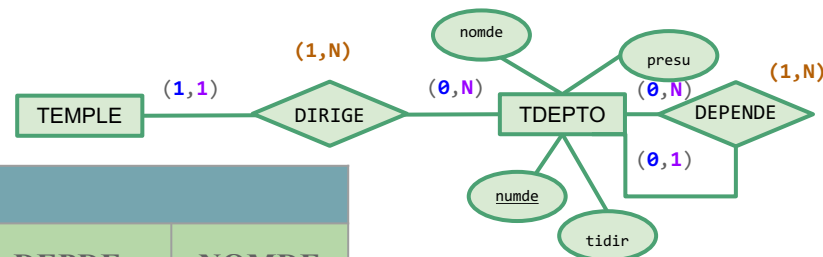
Caso de uso

- Las tablas de la BD son las siguientes:

TDEPTO						
NUMDE	NUMCE	DIREC	TIDIR	PRESU	DEPDE	NOMDE
PK						
	FK.TCENT R	FK.TEMPL E			FK.TDEPT O	

SIGNIFICADO DE LAS COLUMNAS

NUMDE: Número de identificación del departamento (INT)
NUMCE: Número del centro de trabajo donde está el departamento (INT)
DIREC: Número del empleado que es director del departamento (INT)
TIDIR: Tipo de director (Char, P: en propiedad, F: en funciones)
PRESU: Presupuesto anual del departamento (DECIMAL) en miles de euros
DEPDE: Número del departamento del que depende (INT)
NOMDE: Nombre del departamento (VARCHAR)



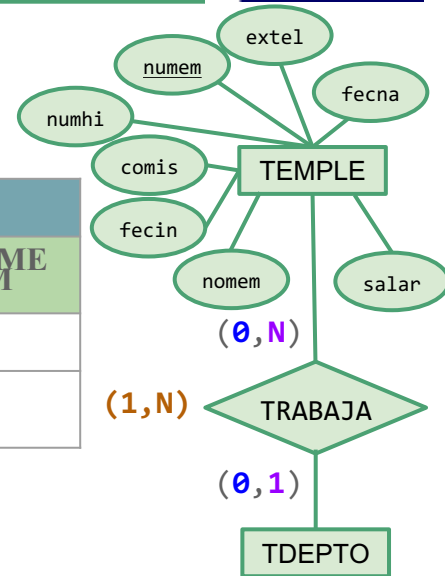
Caso de uso

- Las tablas de la BD son las siguientes:

TEMPLE								
NUMEM	NUMDE	EXTEL	FECNA	FECIN	SALAR	COMIS	NUMHI	NOMEM
PK								
	FK.TDEPT							

SIGNIFICADO DE LAS COLUMNAS

NUMEM: Número identificador del empleado (INT).
NUMDE: Número del departamento al que está asignado (INT)
EXTEL: Extensión telefónica correspondiente al empleado (SMALLINT)
FECNA: Fecha de nacimiento (DATE)
FECIN: Fecha de ingreso (DATE)
SALAR: Salario mensual (DECIMAL), en euros
COMIS: Comisión mensual (DECIMAL), en euros
NUMHI: Número de hijos (SMALLINT)
NOMEM: Nombre del empleado (VARCHAR)



Caso de uso

- Las tablas de la BD son las siguientes:

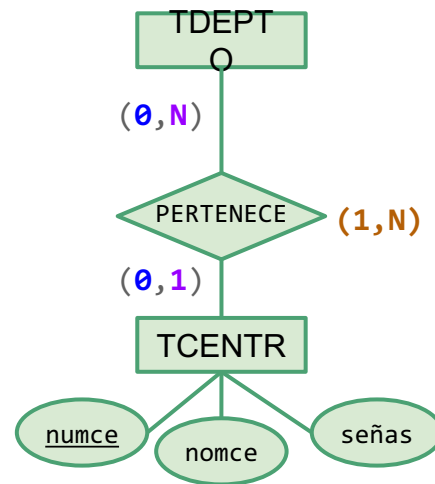
TCENTR		
NUMC E	NOMCE	SEÑAS
PK		

SIGNIFICADO DE LAS COLUMNAS

NUMCE: Número identificador del centro (INT)

NOMCE: Nombre del centro (VARCHAR)

SEÑAS: Dirección del local (VARCHAR)



SQL como DDL

- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - CREATE: Utilizado para crear nuevas tablas, campos e índices (PK y FK).

NOT NULL: impide que el atributo tenga valores NULOS.

AUTO_INCREMENT: autoincrementa el valor del atributo en uno cada vez que se inserta una nueva tupla

DEFAULT: valor por defecto de un atributo.

CHECK: garantiza que el valor insertado para el atributo cumple con una determinada condición

UNIQUE: garantiza que el valor del atributo es único

PRIMARY KEY: establece el atributo como clave primaria

```
CREATE TABLE [IF NOT EXISTS] table_name(
    column_1_definition,
    column_2_definition,
    ...,
    [table_constraints]
) ENGINE=storage_engine;
```

InnoDB: soporte ACID y transacciones.
MyISAM: compresión y velocidad (no ACID).

column_name data_type(length) [NOT NULL] [CHECK(expression)] [UNIQUE]
[DEFAULT value] [AUTO_INCREMENT] [PRIMARY KEY [pk_name]]

[[CONSTRAINT] PRIMARY KEY [pk_name] (attr1, ...)]
[[CONSTRAINT] FOREIGN KEY [fk_name] (attr1, ...)]
REFERENCES parent_table (attr1, ...)]
[ON DELETE CASCADE | RESTRICT | SET NULL]
[ON UPDATE CASCADE | RESTRICT | SET NULL]]

CASCADE: modificaciones de la tabla padre se realizan en la tabla hija.

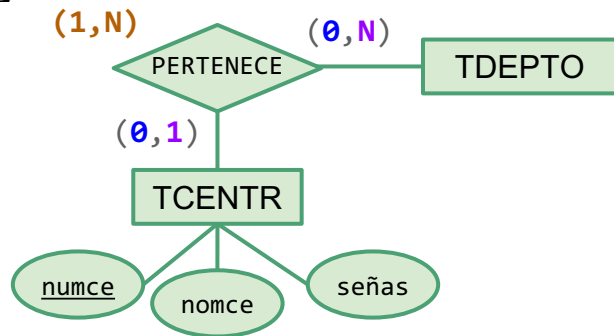
RESTRICT: no se modifica el padre si tiene valores coincidentes en la tabla hija.

SET NULL: modificaciones de la tabla padre implican valor a NULL en los campos coincidentes de la hija.

SQL como DDL

- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - CREATE: Utilizado para crear nuevas tablas, campos e índices (PK y FK).

MERE



MER

TCENTR		
NUMC	NOMCE	SEÑAS
PK		

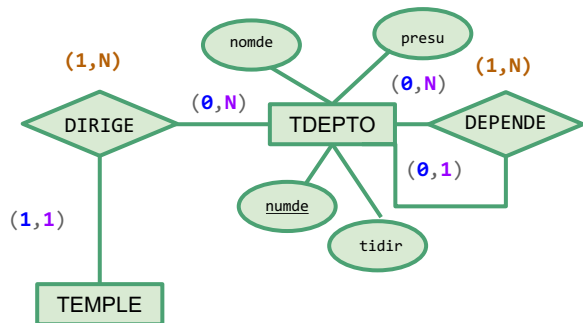
SQL - DDL

```
CREATE TABLE IF NOT EXISTS TCENTR(  
  NUMCE int(11),  
  NOMCE varchar(40),  
  SENAS varchar(40),  
  PRIMARY KEY (NUMCE)  
) ENGINE=InnoDB;
```

SQL como DDL

- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - CREATE: Utilizado para crear nuevas tablas, campos e índices (PK y FK).

MERE



MER

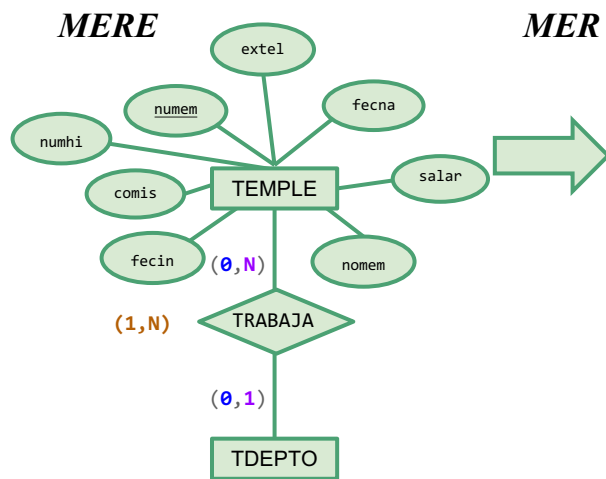
TDEPTO						
NUMDE	NUMCE	DIREC	TIDIR	PRESU	DEPDE	NOMDE
PK						
	FK.TCENTR	FK.TEMPL			FK.TDEPT	

```
CREATE TABLE IF NOT EXISTS TDEPTO (
  NUMDE int(11),
  NUMCE int(11) NOT NULL,
  DIREC int(11),
  TIDIR char(1),
  PRESU decimal(10,0) NOT NULL,
  DEPDE int(11) NOT NULL,
  NOMDE varchar(50) NOT NULL,
  PRIMARY KEY (NUMDE),
  FOREIGN KEY (NUMCE) REFERENCES TCENTR
(NUMCE),
  FOREIGN KEY (DIREC) REFERENCES TEMPLE
(NUMEM)
) ENGINE=InnoDB;
```

No se puede crear una FK que referencia a la propia tabla si no se ha creado previamente

SQL como DDL

- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - CREATE: Utilizado para crear nuevas tablas, campos e índices (PK y FK).



ER

TEMPLE								
NUME	NUMDE	EXTE	FECNA	FECIN	SALAR	COMI	NUMHI	NOME
PK								
	FK.TDEPT							

```

CREATE TABLE IF NOT EXISTS TEMPLE (
    NUMEM int(11),
    NUMDE int(11),
    EXTEL smallint(6) NOT NULL,
    FECNA date NOT NULL,
    FECIN date NOT NULL,
    SALAR decimal(10,0) NOT NULL,
    COMIS decimal(10,0) NOT NULL,
    NUMHI smallint(6),
    NOMEM varchar(40) NOT NULL,
    PRIMARY KEY (NUMEM),
    FOREIGN KEY (NUMDE) REFERENCES TDEPTO (NUMDE)
) ENGINE=InnoDB;
    
```

SQL como DDL

- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - ALTER: Utilizado para modificar las tablas agregando campos o cambiando la definición de los mismos.

```
ALTER TABLE tbl_name
```

```
[alter_option [, alter_option] ...]
```

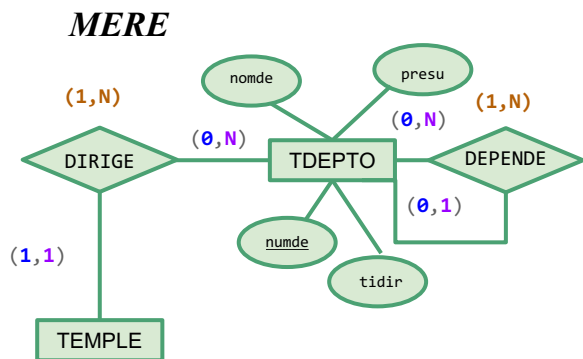
```
[ ADD add_expr ] | [ DROP drop_expr ]
```

```
[ COLUMN col_name col_definition [FIRST | AFTER col_name] ]  
[ [CONSTRAINT constraint_name] CHECK (expr) ]  
[ [CONSTRAINT pk_name] PRIMARY KEY (attr, ...) ]  
[ [CONSTRAINT fk_name] FOREIGN KEY (attr, ...) ]  
  REFERENCES parent_table (attr, ...) ]  
  [ON DELETE CASCADE | RESTRICT | SET NULL]  
  [ON UPDATE CASCADE | RESTRICT | SET NULL] ]
```

```
[ [COLUMN] col_name ]  
[ CHECK symbol ]  
[ PRIMARY KEY ]  
[ FOREIGN KEY fk_name ]
```

SQL como DDL

- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - CREATE: Utilizado para crear nuevas tablas, campos e índices (PK y FK).



MER

TDEPTO						
NUMDE	NUMCE	DIREC	TPI	PRESU	DEPDE	NOMDE
PK						
	FK.TCENT	FK.TEMPL			FK.TDEPT	



```

ALTER TABLE TDEPTO
  ADD FOREIGN KEY (DEPDE) REFERENCES TDEPTO(NUMDE)
    
```



Como la tabla ya está creada, ahora si que se puede definir una FK a la tabla.

SQL como DDL

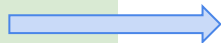
- Permite crear y modificar la estructura de una base de datos.
- Sentencias:
 - DROP: Empleado para eliminar tablas e índices

```
DROP TABLE [IF EXISTS] tbl_name [, tbl_name]
```

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - INSERT: permite añadir registros (tuplas) en una tabla.

```
INSERT INTO tbl_name (col1, col2, ...)  
VALUES  
  (v11, v12, ...)[,  
  (v21, v22, ...),  
  ...  
  (vn1, vn2, ...)];
```



Se pueden insertar múltiples tuplas simultáneamente

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - INSERT: permite añadir registros (tuplas) en una tabla.

MER

TCENTR		
NUMC	NOMCE	SEÑAS
PK		

SQL - DDL

```
CREATE TABLE IF NOT EXISTS TCENTR(
    NUMCE int(11),
    NOMCE varchar(40),
    SENAS varchar(40),
    PRIMARY KEY (NUMCE)
) ENGINE=InnoDB;
```

TABLA

TCENTR		
NUMC	NOMCE	SEÑAS
0001	Informatica	Avd Madrid num 5
0002	Administración	Benigno Ledo num 3
0003	Marketing	Avd Fontiñas num 12

SQL - DML

```
INSERT INTO TCENTR(NUMCE, NOMCE, SENAS) VALUES
    (0001, "Informatica", "Adv Madrid num
    5"),
    (0002, "Administracion", "Benigno Ledo
    num 3"),
    (0003, "Marketing", "Avd Fontiñas num 12");
```

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - UPDATE: se encarga de modificar la información contenida en una tabla.

```
UPDATE [IGNORE] tbl_name
SET
    column_name1 = expr1,
    column_name2 = expr2,
    ...
[WHERE where_condition]
[ORDER BY col [ASC | DESC]]
```

Actualiza las tuplas que cumplen con la expresión.

Indica si se quiere realizar la actualización ordenada por los valores de un determinado atributo

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - INSERT: permite añadir registros (tuplas) en una tabla.

SQL - DML

```
INSERT INTO TCENTR(NUMCE, NOMCE,SENAS) VALUES
    (0001, "Informatica", "Adv Madrid num
5"),
    (0002, "Administracion", "Benigno Ledo
num 3");
    (0003, "Marketing", "Avd Fontiñas num 12");
```



TABLA

TCENTR		
NUMCE	NOMCE	SEÑAS
0001	Informatica	Adv Madrid num 5
0002	Administración	Benigno Ledo num 3
0003	Marketing	Avd Fontiñas num 12

TABLA

TCENTR		
NUMCE	NOMCE	SEÑAS
0001	TIC	Adv Madrid num 5
0002	Administración	Benigno Ledo num 3
0003	Marketing	Avd Fontiñas num 12



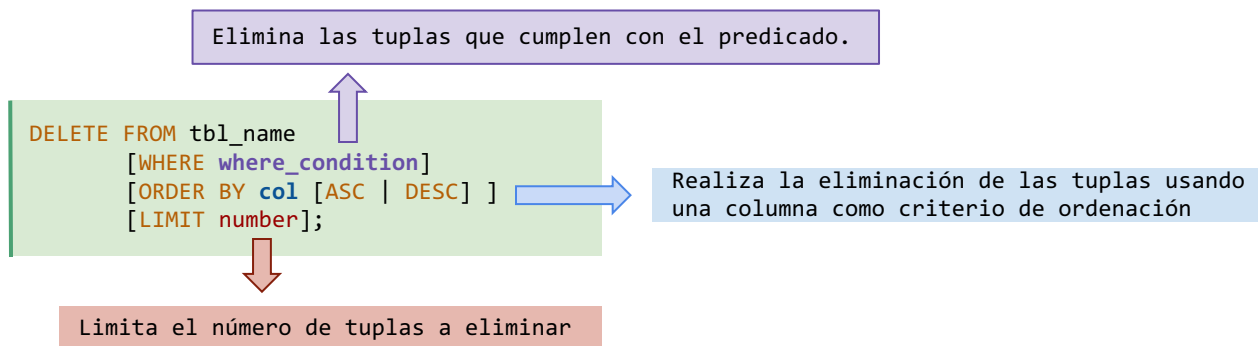
SQL - DML

```
UPDATE TCENTR SET
    NOMCE = "TIC"
WHERE NUMCE = 0001
```



SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - DELETE: permite eliminar los registros de una tabla.



SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - INSERT: permite añadir registros (tuplas) en una tabla.

SQL - DML

```
INSERT INTO TCENTR(NUMCE, NOMCE,SENAS) VALUES
    (0001, "Informatica", "Adv Madrid num
5"),
    (0002, "Administracion", "Benigno Ledo
num 3");
    (0003, "Marketing", "Avd Fontiñas num 12");
```



TABLA

TCENTR		
NUMC	NOMCE	SEÑAS
0001	Informatica	Adv Madrid num 5
0002	Administración	Benigno Ledo num 3
0003	Marketing	Avd Fontiñas num 12



SQL - DML

```
DELETE FROM TCENTR
WHERE NOMCE = "Administración"
```



TABLA

TCENTR		
NUMC	NOMCE	SEÑAS
0001	TIC	Adv Madrid num 5
0003	Marketing	Avd Fontiñas num 12

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

Forma grupos de filas que tengan el mismo valor de columna.

```
SELECT [DISTINCT] * | [expr_col[AS new_name]][,...]  
FROM tbl_name  
[WHERE where_clause]  
[GROUP BY col [,...]] [HAVING having_clause]  
[ORDER BY [col [ASC | DESC]][,...]]
```

Filtra los grupos formados (group by) de acuerdo con alguna condición

Especifica el orden de salida de la información

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

SELECT Básico

Listar toda la información de los centros

```
SELECT * FROM TCENTRO
```

Listar el nombre de los centros

```
SELECT NOMCE FROM TCENTRO o
```

```
SELECT NOMCE AS NOMBRE_CENTRO FROM TCENTRO
```

Listar el número la dirección del centro y cuyo nombre es Informática

```
SELECT NUMCE, SENAS FROM TCENTRO  
WHERE NOMCE = "Informática"
```

TCENTR		
NUMCE	NOMCE	SEÑAS
0001	Informatica	Avd Madrid num 5
0002	Administración	Benigno Ledo num 3
0003	Marketing	Avd Fontiñas num 12

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

CLÁUSULA ORDER BY

Listar los nombres y dirección de los centros ordenados por nombre

```
SELECT NOMCE, SENAS FROM TCENTRO ORDER BY NOMCE
```

o

```
SELECT NOMCE, SENAS FROM TCENTRO ORDER BY 1
```

Listar descendentemente los nombres y dirección de los centros ordenados por nombre

```
SELECT NOMCE, SENAS FROM TCENTR ORDER BY NOMCE DESC
```

o

```
SELECT NOMCE, SENAS FROM TCENTR ORDER BY 1 DESC
```

TCENTR		
NUMCE	NOMCE	SEÑAS
0001	Informatica	Avd Madrid num 5
0002	Administración	Benigno Ledo num 3
0003	Marketing	Avd Fontiñas num 12

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

CLÁUSULA WHERE

Obtener los números de los departamentos de los empleados que cobren más de 1900 euros

```
SELECT NUMDE FROM TEMPLE  
WHERE SALAR > 1900
```



repetidos

NUMDE
101
101
110

```
SELECT DISTINCT NUMDE FROM TEMPLE  
WHERE SALAR > 1900
```



sin repetir

NUMDE
101
110

TEMPLE				
NUME	NUMDE	...	SALAR	...
01	101	...	2000	...
02	110	...	2134	...
03	101	...	2250	...
04	204	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

USO DE EXPRESIONES

Obtener los nombres y sueldos anuales expresados en euros de los empleados del departamento 101. Presentarlos por orden creciente de sueldo

```
SELECT NOMEM, SALAR*12 AS ANUAL FROM TEMPLE  
WHERE NUMDE = 101  
ORDER BY ANUAL ASC
```



NOMEM	ANUAL
Luis	24000
Pedro	27000

TEMPLE					
NUME	NUMDE	NOME	...	SALAR	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

SELECT ANIDADOS

Obtener por orden alfabético los nombres de los empleados cuyos sueldos igualan o superan al de Maria en más de un 20%

```
SELECT NOMEM FROM TEMPLE
WHERE SALAR >= (SELECT SALAR*1.20 FROM TEMPLE
                WHERE NOMEM = "Maria")
ORDER BY NOMEM
```

NOMEM
Ana
Pedro

Obtener por orden alfabético los nombres de los empleados cuyo salario supera al máximo salario de los empleados del departamento 110

```
SELECT NOMEM FROM TEMPLE
WHERE SALAR > ALL (SELECT SALAR FROM TEMPLE
                  WHERE NUMDE = "110")
ORDER BY NOMEM
```

NOMEM
Pedro

TEMPLE					
NUME	NUMDE	NOME	...	SALA	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

ALL = Se tiene que cumplir para todas las tuplas
SOME = Se tiene que cumplir para alguna tupla



SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

USO DE PREDICADOS - BETWEEN

expr1 [NOT] BETWEEN expr2 AND expr3



Permite comparar si un valor está comprendido entre otros dos (inclusive) o no.
expr1, expr2, expr3 deben ser de cualquier tipo comparable

Obtener por orden alfabético los nombres de los empleados cuyo salario está entre 1500 y 1900 euros

```
SELECT NOMEM FROM TEMPLE
WHERE SALAR BETWEEN 1500 AND 1900
ORDER BY NOMEM
```



NOMEM
María

TEMPLE					
NUME	NUMDE	NOME	...	SALAR	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

USO DE PREDICADOS - LIKE

`expr1 [NOT] LIKE pattern`



Comparación de igualdad parcial

expr1 y expr2 deben ser de tipo texto

pattern: permite usar los wildcards:

“_” : sustituye por un caracter cualquiera

“%” : sustituye por 0 o muchos caracteres.

Obtener por orden descendente el número de despacho de los empleados cuyo nombre acaba en “a”

```
SELECT NUMDE FROM TEMPLE
WHERE NOME LIKE "%a"
ORDER BY NUMDE DESC
```



NUMDE
204
110

TEMPLE					
NUMDE	NUMDE	NOME	...	SALA	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

USO DE PREDICADOS - IN

`expr1 [NOT] IN [select_statement | (val1, val2, ...)]`

Determina si el valor de `expr1` pertenece a un conjunto de valores especificados detrás de la cláusula `IN`. El resultado de `select_statement` pueden ser muchas filas pero solo una columna.

Obtener el nombre de los empleados que no trabajan en el mismo despacho que María

```
SELECT NOMEM FROM TEMPLE
WHERE NUMDE NOT IN (SELECT NUMDE FROM TEMPLE
                    WHERE NOMEM ="María")
```

NOMEM
Luis
Ana
Pedro

TEMPLE					
NUME	NUMDE	NOME	...	SALA	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

USO DE PREDICADOS - EXISTS

`expr1 [NOT] EXISTS [select_statement]`



Permite averiguar si la sentencia subordinada tiene una o más filas (Verdadero), o es una tabla vacía (Falso)

Obtener los nombres de los trabajadores, siempre y cuando haya alguno con un salario mayor que 2200€

```
SELECT NOMEM FROM TEMPLE
WHERE EXISTS (SELECT * FROM TEMPLE
              WHERE SALAR >2200)
```



NOMEM
Pedro

TEMPLE					
NUME	NUMDE	NOME	...	SALAR	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

USO DE PREDICADOS - EXISTS

`expr1 [NOT] EXISTS [select_statement]`



Permite averiguar si la sentencia subordinada tiene una o más filas (Verdadero), o es una tabla vacía (Falso)

Obtener los nombres de los trabajadores, siempre y cuando haya alguno con un salario mayor que 2200€

```
SELECT NOMEM FROM TEMPLE
WHERE EXISTS (SELECT * FROM TEMPLE
              WHERE SALAR >2200)
```



NOMEM
Pedro

TEMPLE					
NUME	NUMDE	NOME	...	SALAR	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

FUNCIONES DE AGREGACIÓN

funcAgregación ([DISTINCT] **expr**)



Permite obtener un solo valor como resultado de aplicar una determinada operación (suma, media, máximo, min, contar, ...)

DISTINCT elimina valores repetidos

Hallar el salario máximo para el conjunto de todos los empleados

SELECT MAX(SALAR) AS MAXIMO FROM TEMPLE



MAXIMO

2250

Hallar el salario medio de los empleados del despacho 101

**SELECT AVG(SALAR) AS MEDIA FROM TEMPLE
WHERE NUMDE = 101**



MEDIA

2125

TEMPLE					
NUME	NUMDE	NOME	...	SALAR	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

COUNT: devuelve la cuenta del número de filas obtenidas
MAX, MIN: devuelve el valor máximo/mínimo
AVG: calcula el valor medio a partir de los datos obtenidos
SUM: realiza la suma todos los valores obtenidos

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

CLÁUSULA GROUP BY

Permite formar grupos de filas de acuerdo a un determinado criterio (especificado en la cláusula GROUP BY) para aplicarles luego una función de agregación.

TEMPLE					
NUME	NUMDE	NOME	...	SALAR	...
01	101	Luis	...	2000	...
02	110	Ana	...	2134	...
03	101	Pedro	...	2250	...
04	204	María	...	1700	...

Hallar el número y salario medio de cada departamento

```
SELECT NUMDE, MAX(SALAR) AS MAXIMO FROM TEMPLE
GROUP BY NUMDE
ORDER BY NUMDE
```

NUMDE	MAXIMO
101	2500
110	2134
204	1700

PASOS:

- 1-FROM: Se consulta la tabla
- 2-GROUP BY: Se agrupan las filas con el mismo valor en la expresión de agrupamiento
- 3-SELECT: se evalúan las expresiones del SELECT
- 4-ORDER BY: se ordenan los resultados según el criterio especificado.



SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

OPERADORES DE CONJUNTOS

Permite combinar los resultados obtenidos de múltiples sentencias SELECT. Hay 3 tipos de operadores:

UNION: combina todas las filas distintas de ambas consultas omitiendo los resultados repetidos.

INTERSECT: obtiene las filas comunes a ambas consultas. No está soportado nativamente pero se puede hacer con IN.

MINUS: filas de la primera consulta que no existe en la segunda

```
SELECT [DISTINCT] * | [expr_col[AS new_name]]
FROM tbl_name
[WHERE where_clause]
UNION
SELECT [DISTINCT] * | [expr_col[AS new_name]]
FROM tbl_name
[WHERE where_clause];
```

```
SELECT [DISTINCT] * | [expr_col[AS new_name]]
FROM tbl_name
WHERE where_clause
IN
(SELECT [DISTINCT] * | [expr_col[AS new_name]]
FROM tbl_name
[WHERE where_clause;])
```

```
SELECT [DISTINCT] * | [expr_col[AS new_name]]
FROM tbl_name
WHERE where_clause
NOT IN
(SELECT [DISTINCT] * | [expr_col[AS new_name]]
FROM tbl_name
[WHERE where_clause;])
```


SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

ERRORES COMUNES

1 - En la cláusula SELECT no pueden aparecer combinaciones de valores colectivos (MAX, SUMO) y no colectivos (columnas).

```
SELECT NUMEM, MAX(SALAR) AS MAXIMO FROM TEMPLE
```

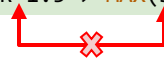


Solución, usar sentencias SELECT

```
SELECT NUMEM, (SELECT MAX(SALAR) FROM TEMPLE) FROM TEMPLE
```

2- En el predicado no pueden aparecer funciones de agregación con valores colectivos

```
SELECT NUMEM FROM TEMPLE  
WHERE SALAR*1.5 > MAX(SALAR)
```



Solución, usar sentencias SELECT subordinadas

```
SELECT NUMEM FROM TEMPLE  
WHERE SALAR*1.5 > (SELECT MAX(SALAR) FROM TEMPLE)
```

SQL como DML


- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

CONSULTAS SOBRE VARIAS TABLAS

En ocasiones NO será suficiente con realizar consultas sobre una única tabla. Es decir es necesario combinar la información de múltiples tablas (JOIN).

Obtener el n° de empleado de cada trabajador y el n° y nombre de su departamento

```
SELECT NUMEM, NOMDE, TEMPLE.NUMDE FROM TEMPLE, TDEPTO
WHERE TEMPLE.NUMDE = TDEPTO.NUMDE
```



NUME	NOMDE	NUMDE
01	RRHH	204
03	RRHH	204
04	Finanzas	101

TEMPLE				
NUME	NUMDE	...	SALA	...
01	101	...	2000	...
02	110	...	2134	...
03	101	...	2250	...
04	204	...	1700	...

TDEPTO		
NUMDE	...	NOMDE
204	...	RRHH
101	...	Finanzas

SQL como DML

- Se utiliza para manipular, y obtener la información contenida en la base de datos.
- Sentencias:
 - SELECT: permite obtener información contenida en una o múltiples tablas.

CONSULTAS SOBRE VARIAS TABLAS

En ocasiones NO será suficiente con realizar consultas sobre una única tabla. Es decir es necesario combinar la información de múltiples tablas (JOIN).

Averiguar los empleados que ganan más de 2200€, así como los nombres de dpto en los que trabajan

```
SELECT NUMEM, D.NOMDE FROM TEMPLE E, TDEPTO D
WHERE SALAR > 2200 AND E.NUMDE = D.NUMDE
```



NUME	D.NOMDE
03	Finanzas

Se usan sinónimos para renombrar las tablas y especificar concretamente el campo de la tabla que se quiere referenciar.

TEMPLE				
NUME	NUMDE	...	SALAR	...
01	101	...	2000	...
02	110	...	2134	...
03	101	...	2250	...
04	204	...	1700	...

TDEPTO		
NUMDE	...	NOMDE
204	...	RRHH
101	...	Finanzas

SQL como DCL

- Se utiliza para otorgar o denegar permisos a uno o más roles para realizar determinadas tareas.
- Sentencias:
 - CREATE USER: permite crear usuarios en la base de datos

```
CREATE USER [IF NOT EXISTS] account_name  
IDENTIFIED BY "account_password"
```



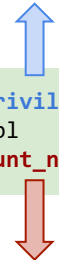
username@[hostname]

username: nombre del usuario
hostname: nombre del host (ubicación) desde la que el usuario se va a conectar. Si se omite se permite la conexión desde cualquier host.

SQL como DCL

- Se utiliza para otorgar o denegar permisos a uno o más roles para realizar determinadas tareas.
- Sentencias:
 - GRANT: permite otorgar permisos

```
[SELECT [(col1,...)] | INSERT [(col1,...)] | UPDATE [(col1,...)], DELETE [(col1,...)] | ALL [(col1,...)] ]
```



```
GRANT privilege [,privilege],..  
ON db.tbl  
TO account_name;
```

Indica la cuenta del usuario a la que se le asignan los privilegios

SQL como DCL

- Se utiliza para otorgar o denegar permisos a uno o más roles para realizar determinadas tareas.
- Sentencias:
 - REVOKE: permite otorgar permisos

```
[SELECT [(col1,...)] | INSERT [(col1,...)] | UPDATE [(col1,...)], DELETE [(col1,...)] | ALL [(col1,...)] ]
```



```
REVOKE privilege [,privilege],..  
ON db.tbl  
FROM user1 [, user2] ...;
```

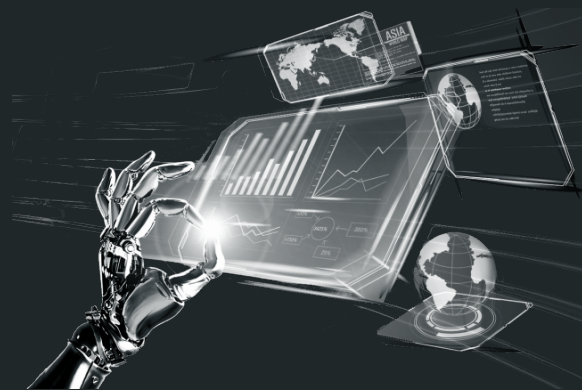
SQL como DCL

- Se utiliza para otorgar o denegar permisos a uno o más roles para realizar determinadas tareas.
- Sentencias:
 - DROP USER: elimina usuarios de la base de datos

```
DROP USER [IF NOT EXISTS] account_name1 [, account_name2]...
```

Bibliografía

RIVERO C. Enrique, et.al. Introducción al SQL para Usuarios y Programadores (2º edic.). Ed Thompson. (Caps 9-11)



Gestión de Datos para Robótica

T1f - Lenguaje SQL

Álvaro Vázquez Álvarez
Departamento de Electrónica e Computación

✉ alvaro.vazquez@usc.es

📍 Pabellón III - Despacho 4

Curso 2023-2024