Bases de datos relacionales y el modelo entidad-relación

¿ Qué es una base de datos relacional ?

El sistema gestor de bases de datos

El modelo entidad-relación

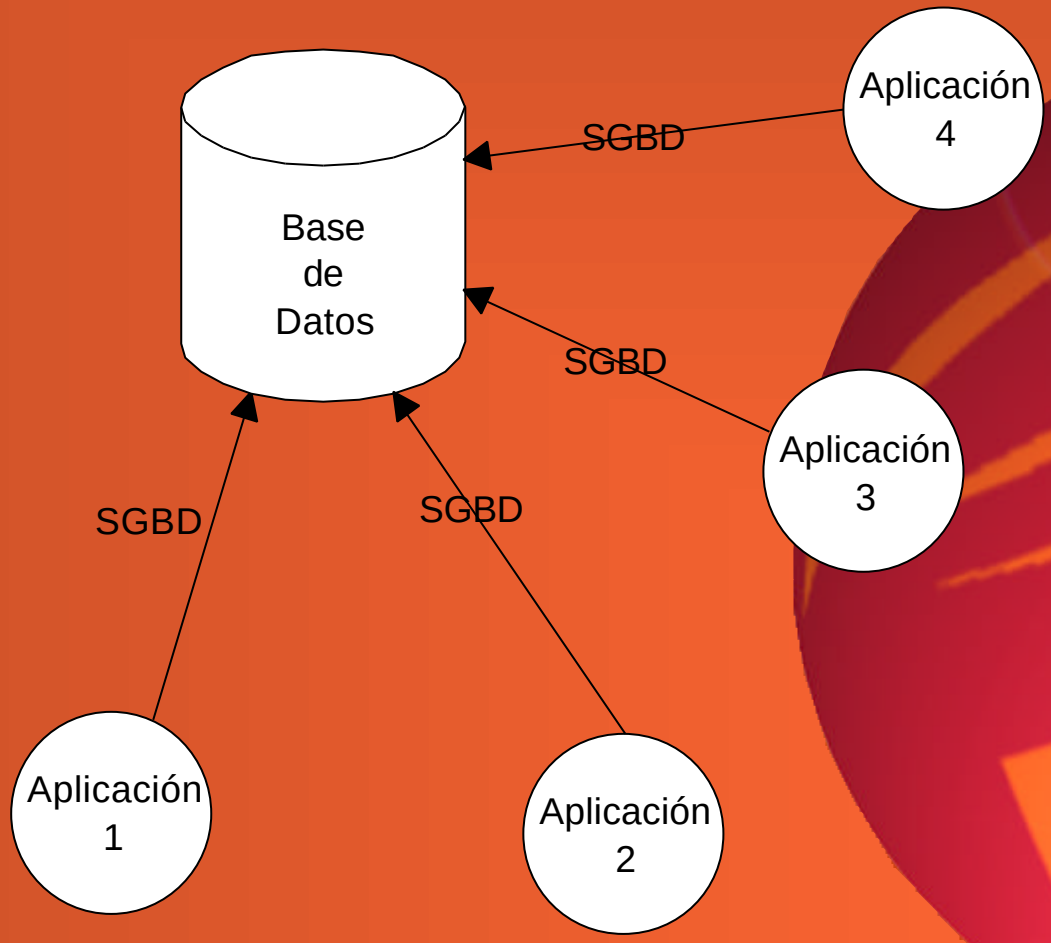
- –entidad, atributos y elementos
  (tablas, columnas y filas)
- –relaciones

¿ Qué es SQL ?

Definición de BD

- Una base de datos (BD) es un conjunto de datos interrelacionados almacenados en conjunto, sin redundancias innecesarias, de forma independiente de los programas que acceden a ellos.

El sistema gestor de BD (I)

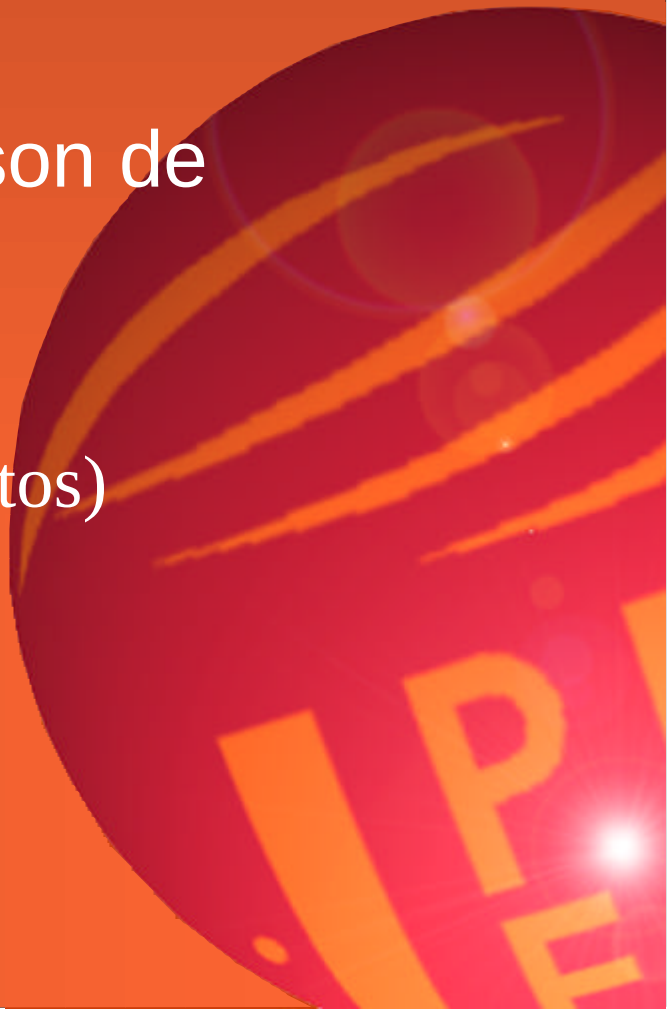


El sistema gestor de BD (II)

- Sirve para:
  - definir y crear datos
  - manipular esos datos
  - seguridad e integridad de los datos
  - recuperar los datos: lenguaje SQL
  - mantenimiento de un diccionario de datos
  - alto rendimiento: se debe asegurar que todas estas funciones se ejecuten lo más rápidamente posible.

El modelo entidad-relación

- Se usa para diseñar la BD
- La mayoría de BD actuales son de tecnología **relacional**
- Conceptos base:
  - entidades (y atributos y elementos)
  - relaciones



Entidades

- Los objetos que aparece en la vida real, es lo que llamamos **entidad**. Por ejemplo, alumnos, empleados, aviones, coches, alojamientos, ...
- Una entidad da lugar a una **tabla** en la BD.

Atributos

- Estas entidades están compuestas por varios **atributos**, que vienen a ser sus propiedades. Por ejemplo, la entidad alumnos, tendrá los atributos nombre, DNI, nacionalidad, fecha de nacimiento, ...
- Los atributos también reciben el nombre de **columnas** en la terminología de BD

Elementos

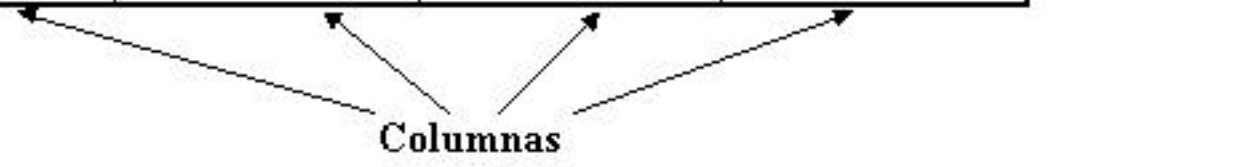
- Cada entidad tendrá un número ilimitado de elementos. Por ejemplo, un elemento de la entidad alumnos será un alumno en sí; así el alumno Pepe será un elemento, José será otro, ...
- Cada uno de esos elementos también recibe el nombre de **fila** en la terminología de BD

Tablas

- Combinando estos tres conceptos tenemos una estructura del tipo tabla, la base de las BD.

Tabla ALUMNOS:

Nombre	DNI	Nacionalidad	Fecha nacim.
Joseph	99.999.999	Irlanda	2-2-1969
Pepe	88.888.888	España	8-8-1968
José	77.777.777	Chile	7-7-1967
...			



Relaciones

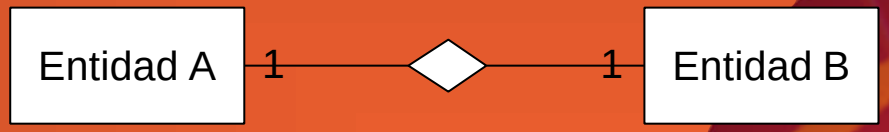
- Las entidades no están aisladas sino que están relacionadas entre sí.
- Estas **relaciones** pueden ser de tres tipos diferentes:
  - 1 a 1
  - 1 a muchos (1 a N)
  - Muchos a muchos (M a N)

Representación del modelo (I)

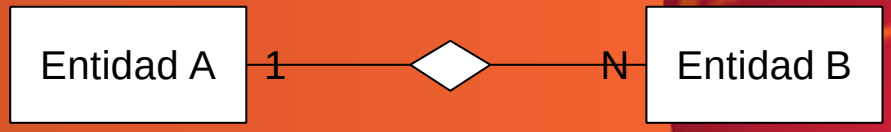
- Representaremos las entidades con recuadros con su nombre en el interior

Representación del modelo (I)

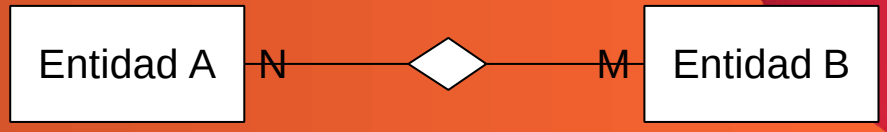
- Representaremos las relaciones así:
  - Relación 1:1



  - Relación 1:N



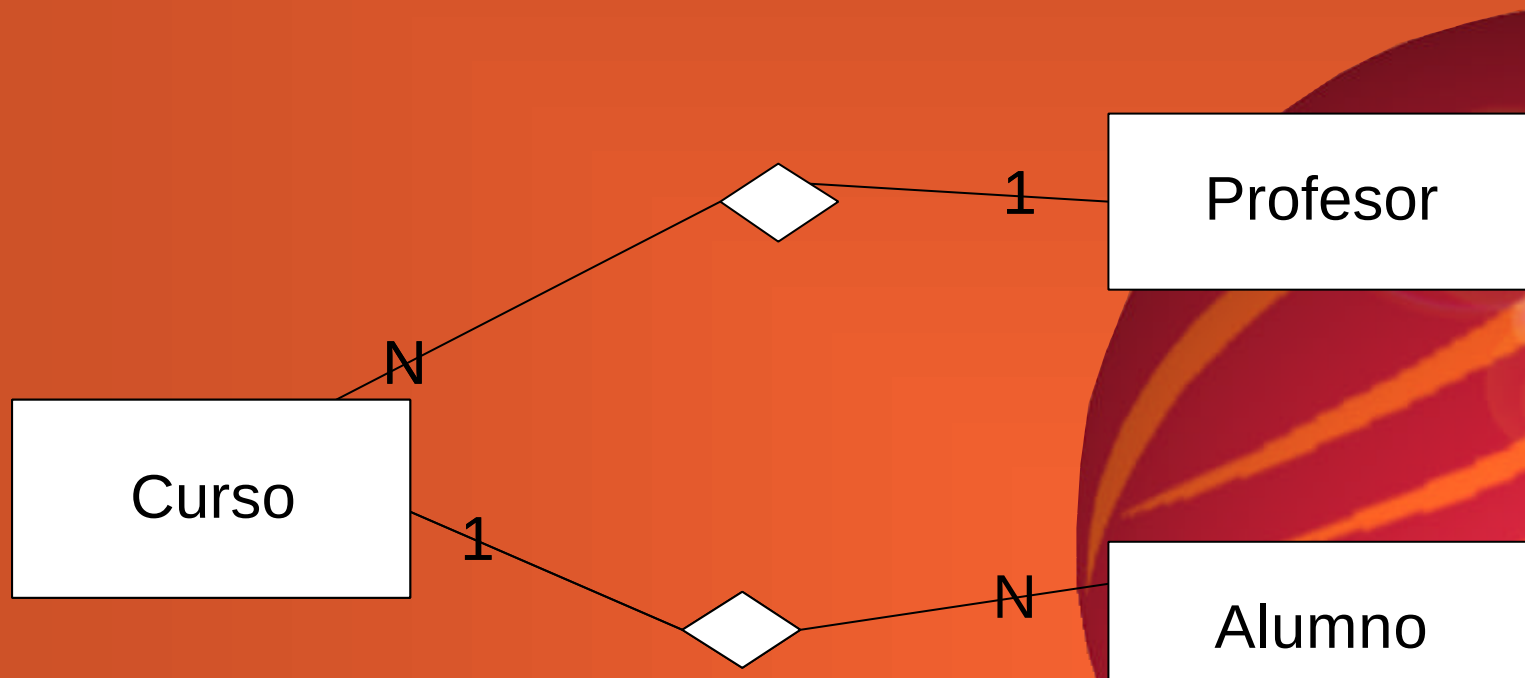
  - Relación M:N



Ejercicio 1

- Hacer un modelo Entidad-Relación para la siguiente situación:
  - tenemos una universidad, en la que hay varios cursos. Cada curso está dirigido por un profesor, el cual puede dirigir varios cursos. Los cursos son subveniados, por lo que sólo se permite que un alumno se matricule de un curso.

Solución (ej. 1)



Solución: las tablas

Profesor:

Código	Nombre	DNI	Fecha nac.
1	Don Pepito	11.111.111	1-1-1950
2	Don José	22.222.222	2-2-1950

Curso:

Código	Nombre	Horas	Código Profesor
1	Word	10	2
2	Photoshop	10	2
3	Director	10	2
4	Internet	30	1

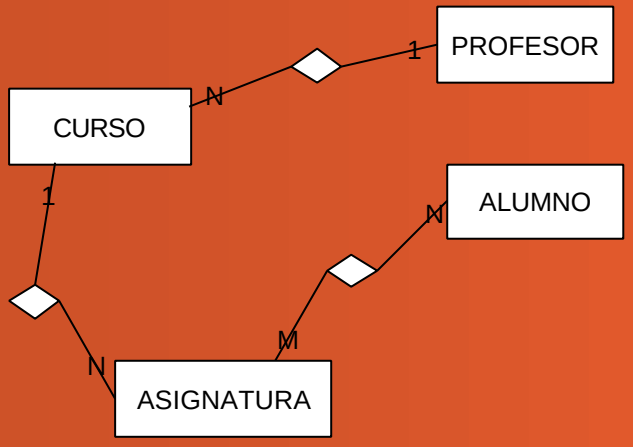
Alumno:

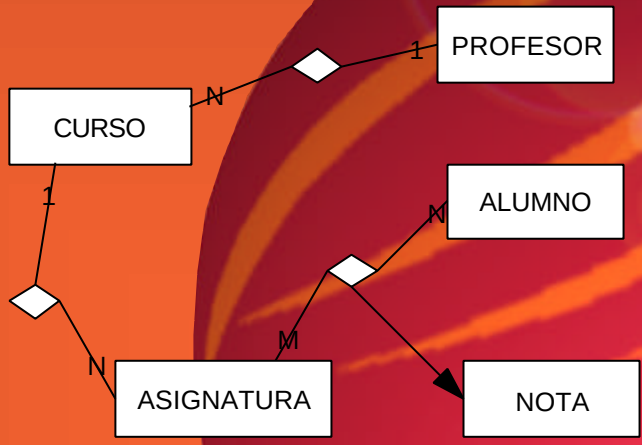
Código	Nombre	DNI	Fecha nacim.	Código Curso
1	Juan	33.333.333	3-3-1970	1
2	Pedro	44.444.444	4-4-1970	1
3	Antonio	55.555.555	5-5-1970	2
...	...	...	...	...

Ejercicio 2

- Compliquemos un poco la situación anterior:
  - ahora supongamos que un curso está compuesto por varias asignaturas. Cada una de ellas tiene un número de créditos. Los alumnos se matriculan de las asignaturas que quieren. Por último el alumno recibe una nota para cada asignatura, al final del curso.

Solución (ej. 2)

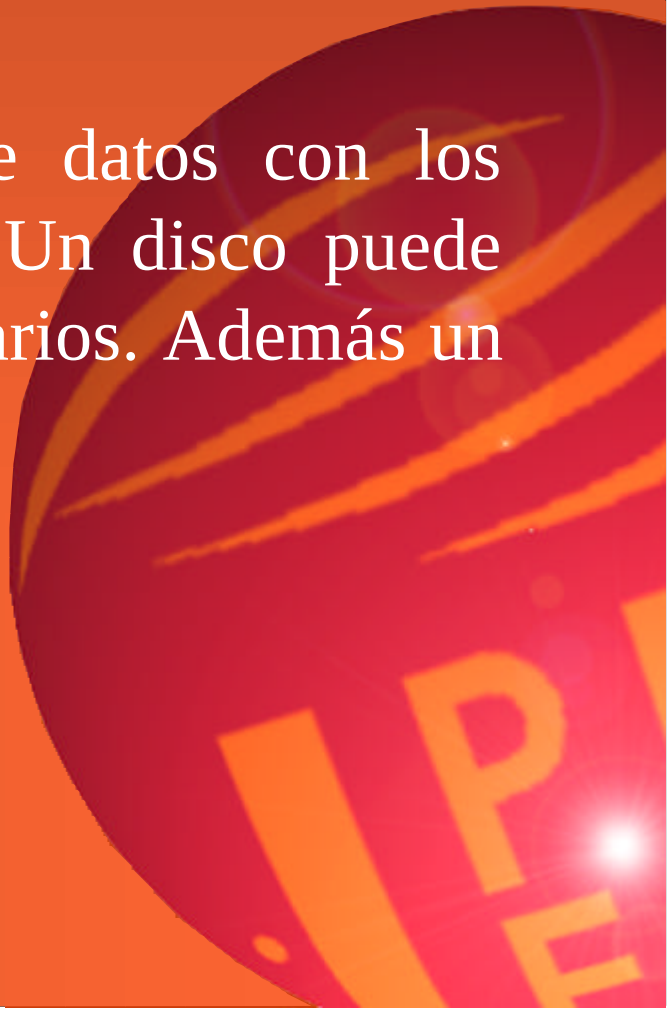




Las relaciones N:M implican la creación de una nueva entidad

Más ejercicios

- Ejercicio 1
  - Queremos hacer una base de datos con los discos que tenemos en casa. Un disco puede tener un cantante o grupo, o varios. Además un disco tiene una discográfica.



Más ejercicios

- Ejercicio 2
  - Vamos a complicar un poco el ejemplo anterior: ahora hemos de tener en cuenta que un disco está compuesto por canciones. Éstas pueden estar escritas por la misma persona que las canta, pero a menudo se trata de personas diferentes.

Más ejercicios

- Ejercicio 3
  - Imaginemos que hemos de hacer un sistema para la Eurocopa2000. 16 selecciones nacionales se han clasificado, y en cada una hay 22 jugadores. Hemos de poder reflejar todos los partidos que se disputan (con los resultados) y qué jugadores lo hacen por cada equipo, así como quien marca los goles.

Normalización (1FN)

Para que un modelo sea correcto debe cumplir tres normas, que conocemos como las tres formas de normalización:

1FN: Tenemos 1FN si y sólo si todas las columnas de todas las tablas contienen sólo valores atómicos, es decir un único valor. Ejemplo, alumnos matriculados de asignaturas y sus calificaciones:

Asignatura	Alumno
Director	Pedro, Pablo
Photoshop	Pedro, Pablo, Antonio

Esta tabla no está en forma normal. Hay que hacer parejas, y el identificador será la combinación de asignatura y alumno.

Asignatura	Alumno	DNI	Nota	Calificación
Director	Pedro	11.111.111	6	Apto
Director	Pablo	22.222.222	7	Apto
Photoshop	Pedro	11.111.111	8	Apto
Photoshop	Pablo	22.222.222	3	No Apto
Photoshop	Antonio	33.333.333	5	Apto
...				

Normalización (2FN)

2FN: todos los atributos que no forman parte de la clave, deben depender de ella en conjunto, y nunca de un subconjunto de la misma.

En el ejemplo anterior, la clave es la combinación de asignatura y alumno. Para saber si está en 2FN, miramos si DNI depende de sólo una parte de la clave. En este caso sí, ya que sólo depende de alumno. En el caso de nota y calificación no ocurre. La solución es separarlo en dos tablas:

Asignatura	Alumno	Nota	Calificac.
Director	Pedro	6	Apto
Director	Pablo	7	Apto
Photoshop	Pedro	8	Apto
Photoshop	Pablo	3	No Apto
Photoshop	Antonio	5	Apto
...			

Alumno	DNI
Pedro	11.111.111
Pablo	22.222.222
Antonio	33.333.333

Normalización (3FN)

3FN: Todos los atributos que no forman parte de la clave primaria deben ser independientes entre sí.

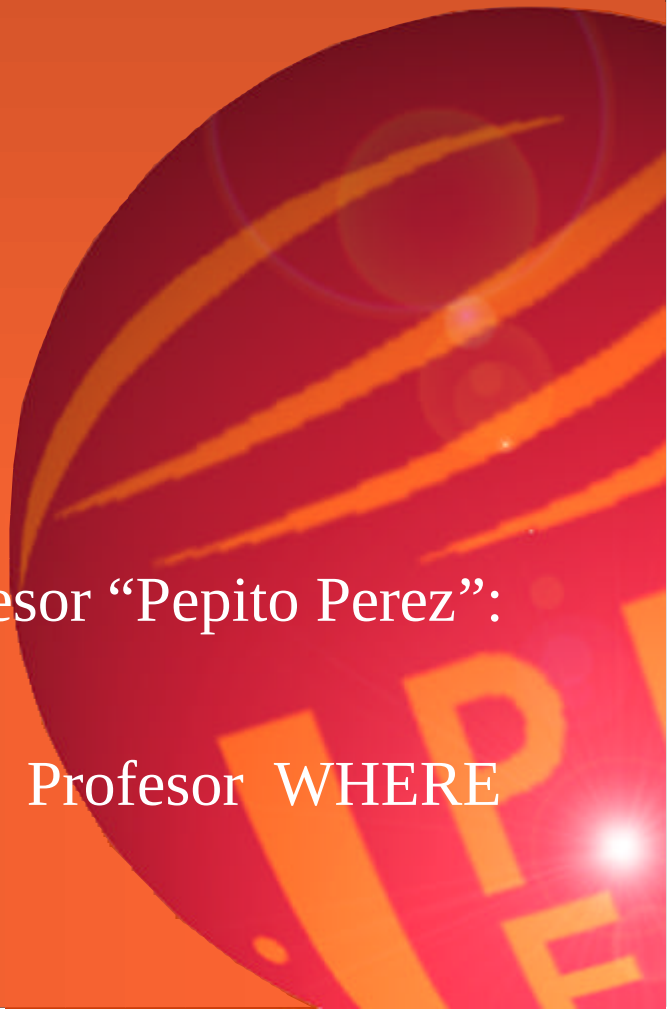
En el ejemplo anterior, en la primera tabla, hemos de mirar si nota depende de califiación o viceversa. En este caso, tenemos que calificación depende de nota, con lo cual, se puede eliminar de la tabla, porque es un dato redundante que se puede calcular fácilmente. Imaginemos que tenemos una tabla con 100.000 filas. Estaríamos utilizando 100.000 veces el espacio de esa columna innecesariamente.

SQL

- SQL es el lenguaje estándar utilizado para consultar las bases de datos relacionales
- Permite (además de opciones más avanzadas) crear, modificar o borrar tablas, así como insertar, eliminar, modificar o consultar los elementos de las tablas
- Lo más común es realizar consultas:
  - sentencia SELECT

SQL: sentencia select

SELECT nombre_de_columna
FROM nombre_de_tabla
WHERE condición;



- Ejemplo:

  Queremos saber el código del profesor “Pepito Perez”:

  SELECT codigo_profesor FROM Profesor WHERE nombre=‘Don Pepito’;

SQL: ejemplos 1 y 2

- Queremos saber el código del alojamiento “Arts”:
  - SELECT alojamiento_id FROM Alojamiento WHERE nombre=‘Arts’;
- Queremos saber los nombres de todos los alojamientos que se construyeron antes de 1980:
  - SELECT nombre FROM Alojamiento WHERE anyo_construccion < 1980;

SQL: ejemplo 3

- Queremos saber los alojamientos de todos los profesores que se construyeron durante los ’80, es decir, que su año de construcción está entre el 1980 y el 1989
  - SELECT nombre FROM Alojamiento WHERE (anyo_construccion >= 1980) and (anyo_construccion <= 1989);
- Nota: los paréntesis no son obligatorios, pero ayudan a la lectura.

SQL: ejemplos 4 y 5

- Queremos saber los nombres de todos los alojamientos que se construyeron tanto durante la década de los ’60 como durante los ’80. Para ello necesitamos utilizar una OR entre las condiciones:
  - SELECT nombre FROM Alojamiento WHERE ((anyo_construccion >= 1960) and (anyo_construccion <= 1969)) or ((anyo_construccion >= 1990) and (anyo_construccion <= 1989));

- Queremos saber qué alojamientos (todos los datos) están en el municipio con código ‘BARC’:
  - SELECT * FROM alojamiento WHERE municipio_id=‘BARC’;

SQL: ejemplo 6

- Queremos saber qué alojamientos (todos los datos) son del tipo ‘Hotel’. Ahora necesitaremos usar más de una tabla, ya que el ‘Hotel’ es el nombre del Tipo de alojamiento y necesitaremos saber cuál es su código
  - SELECT * FROM Alojamiento, Tipo WHERE (Tipo.nombre = ‘Hotel’) and (Alojamiento.tipo_id = Tipo.tipo_id);
- Lo que hemos hecho es primero buscar el código del tipo en cuestión:
  - Tipo.nombre = ‘Hotel’

Ahora, nos devuelve la lista de todos los tipos cuyo nombre es ése (en este caso, sólo uno). Ya sólo nos queda buscar los alojamientos con ese código (que queda almacenado en Tipo.tipo_id):

  - Alojamiento.tipo_id = Tipo.tipo_id

SQL: ejemplo 7 (y último)

- Queremos saber qué alojamientos se encuentran en la provincia de Tarragona (todos los datos). Primero buscaremos los municipios de Tarragona y luego los alojamientos cuyo municipio_id coincida con ésos:
  - SELECT Alojamiento.* FROM Alojamiento, Municipio, Provincia WHERE (Provincia.nombre = 'Tarragona') and (Municipio.provincia_id = Provincia.provincia_id) and (Alojamiento.municipio_id = Municipio.municipio_id);

SQL: más cosas ...

- **Insertar, borrar y modificar los datos de una tabla**

1) Insertar:

INSERT INTO Tipo VALUES (‘AG’,’Agroturismo’,’Centros de agroturismo y turismo rural’);

(Inserta todos los valores de una fila de Tipo)

INSERT INTO Hotel (hotel_id, nombre) VALUES (7,’Arts’);

(Inserta sólo algunos valores de una fila de Hotel)

2) Eliminar:

DELETE FROM Hotel;

(Borrar todas las filas de la tabla avión)

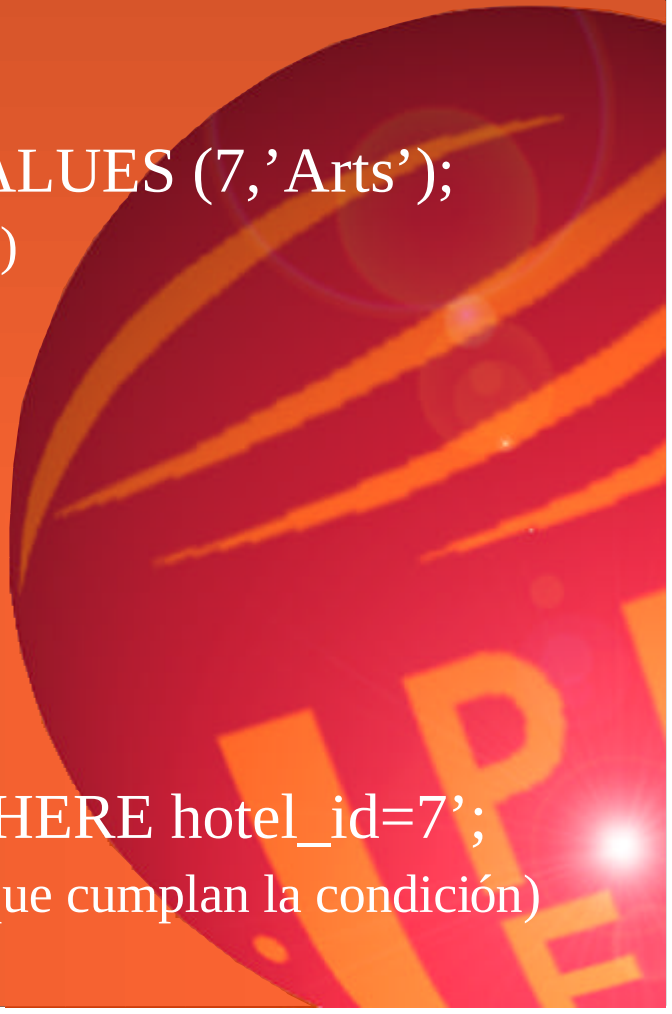
DELETE FROM Hotel WHERE codigo=5;

(Borra sólo las filas que cumplan una condición)

3) Modificar:

UPDATE Hotel SET nombre=‘Les Arts’ WHERE hotel_id=7’;

(Modificar el atributo nombre en todas las filas que cumplan la condición)



SQL: más cosas ...

Cómo crear una tabla

```
CREATE TABLE Municipio
(
municipio_id CHAR(4) PRIMARY KEY,
nombre VARCHAR2(20) NOT NULL,
provincia_id CHAR(2) REFERENCES
   Provincia(provincia_id)
)
```

También hay sentencias para borrar (Drop table) una tabla y también para modificarla (Alter table), pero no las veremos aquí.