Universidad de Granada

decsai.ugr.es

Fundamentos de Bases de Datos

Grado en Ingeniería Informática

Seminario: SQL

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

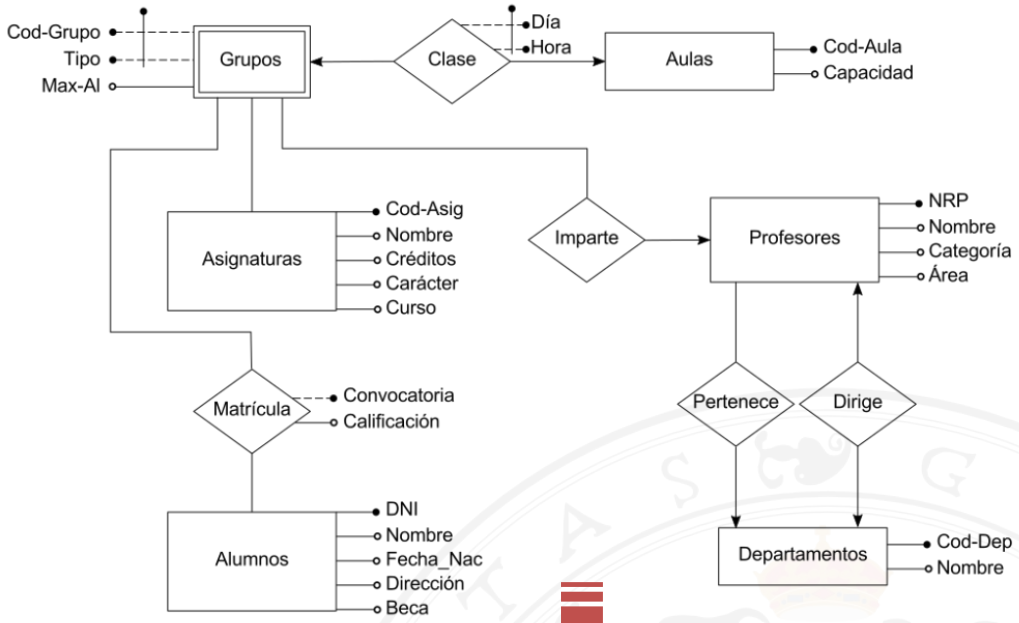
Datos generales sobre una organización concreta

Datos operativos que se manejan en la organización

Esquema conceptual de la base de datos

Modelo lógico de la base de datos

Implementación de la base de datos en un DBMS



Atributo 1	Atributo 2	Atributo 3	...	Atributo n
...	...	...	...	...

Atributo 1	Atributo 2		...	Atributo n
...	...		...	...

La sentencia CREATE TABLE

```
CREATE TABLE [usuario.]nombre_tabla
   ({datos_columna | restricciones de tabla}
   [,{datos_columna | restricciones de tabla}]...)

[PCTFREE n], [PCTUSED n], [INITRANS n ], [MAXTRAN n] [TABLESPACE
nombre],
[STORAGE nombre]
[ CLUSTER nombre_cluster(columna[,columna]...)]
[AS consulta]
```

La parte sombreada corresponde a cuestiones avanzadas:

- Nivel físico
- Control de transacciones
- Creación de tablas derivadas

La sentencia CREATE TABLE: datos_columna

CREATE TABLE [usuario.]nombre_tabla
  ({datos_columna | restricciones de tabla}
  [,{datos_columna | restricciones de tabla}]...)

datos_columna debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

El nombre de columna es el del atributo
El tipo de dato se da entre los permitidos
Se pueden dar valores por defecto
Se pueden añadir reglas de integridad (restricciones de columna)

La sentencia CREATE TABLE: Tipos de datos

Numéricos:

INT, INTEGER, NUMERIC: Enteros sin signo. El rango depende del sistema.

FLOAT: Reales. El rango depende del sistema

NUMBER(p,s): Numero con precisión p (max 38, min 1) y escala s (max 127, min -84)

precisión: número total de dígitos

escala: número de cifras decimales

Cadenas de caracteres:

CHAR(n): Cadena de longitud fija de n caracteres (n<= 4000)

VARCHAR(n): Cadena de longitud variable de hasta n caracteres

VARCHAR2(n): Cadena de longitud variable de hasta n caracteres. Implementación propia de Oracle (más eficiente)

LONG: Cadena larga (hasta 2 GB)

La sentencia CREATE TABLE: Tipos de datos

Datos binarios:

RAW(n): Datos en binario (n<=2000).

LONG RAW: Datos en binario hasta 2 GB

Tipos de datos de fecha y tiempo:

DATE: Fecha.

TIME: Hora.

DATETIME: Fecha y hora.

TIMESTAMP: Instante de tiempo.

Es necesario utilizar las funciones de transformación de cadenas a fecha para usarlo en consultas. Las más relevantes:

SYSDATE: Macro de Oracle que devuelve la fecha y hora actual

TO_DATE(s): Función que devuelve una fecha y hora dada en una cadena.

La sentencia CREATE TABLE: restricciones de columnas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

Debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

Restricciones asociadas a las columnas:

[[CONSTRAINT nombre] NOT NULL]
[[CONSTRAINT nombre] {UNIQUE | PRIMARY KEY}]
[[CONSTRAINT nombre] REFERENCES [usuario.]nombre_tabla [(columna)]]
[[CONSTRAINT nombre] CHECK (condicion)]

Tienen el mismo sentido anterior

NOT NULL: el campo no admite valores nulos.

La sentencia CREATE TABLE: restricciones de columnas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

Debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

Restricciones asociadas a las columnas:

La columna no puede tomar valor nulo

[[CONSTRAINT nombre] NOT NULL]
[[CONSTRAINT nombre] {UNIQUE | PRIMARY KEY}]
[[CONSTRAINT nombre] REFERENCES [usuario.]nombre_tabla [(columna)]]
[[CONSTRAINT nombre] CHECK (condicion)]

Tienen el mismo sentido anterior
NOT NULL: el campo no admite valores nulos.

La sentencia CREATE TABLE: restricciones de columnas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

Debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

Restricciones asociadas a las columnas:

Un valor no puede estar repetido, porque la columna es CC

[[CONSTRAINT
[[CONSTRAINT nombre] {UNIQUE | PRIMARY KEY}]
[[CONSTRAINT nombre] REFERENCES [usuario.]nombre_tabla [(columna)]]
[[CONSTRAINT nombre] CHECK (condicion)]

Tienen el mismo sentido anterior

NOT NULL: el campo no admite valores nulos.

La sentencia CREATE TABLE: restricciones de columnas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

Debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

Restricciones asociadas a las columnas:

La columna es una clave primaria

[[CONSTRAINT nombre] NOT NULL]
[[CONSTRAINT nombre] {UNIQUE | PRIMARY KEY}]
[[CONSTRAINT nombre] REFERENCES [usuario.]nombre_tabla [(columna)]]
[[CONSTRAINT nombre] CHECK (condicion)]

Tienen el mismo sentido anterior

NOT NULL: el campo no admite valores nulos.

La sentencia CREATE TABLE: restricciones de columnas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

Debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

Restricciones asociadas a las columnas:

[[CONSTRAINT
[[CONSTRAINT

La columna es una clave externa al atributo columna de la tabla *nombre_tabla*

[[CONSTRAINT nombre] REFERENCES [usuario.]nombre_tabla [(columna)]]
[[CONSTRAINT nombre] CHECK (condicion)]

Tienen el mismo sentido anterior

NOT NULL: el campo no admite valores nulos.

La sentencia CREATE TABLE: restricciones de columnas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

Debe tener el formato:
nombre tipo_de dato [DEFAULT expresion] [restriccion_de _columna]

Restricciones asociadas a las columnas:

[[CONSTRAINT nombre] NOT NULL]
[[CONSTRAINT nombre] {UNIQUE | PRIMARY KEY}]
[[CONSTRAINT nombre] REFERENCES tabla_referenciada [(columna)]]
[[CONSTRAINT nombre] CHECK (condicion)]

Todos los valores de la columna deben cumplir la condición

Tienen el mismo sentido anterior
NOT NULL: el campo no admite valores nulos.

La sentencia CREATE TABLE: Operadores en Oracle SQL

Oracle dispone de varios operadores que pueden utilizarse tanto en consultas como para definir condiciones o restricciones de tabla o de columna:

+,-,||	Sumar ,restar, concatenar
*,/	Multiplicar,dividir
=,!=,<,>,<=,>=	Comparadores clasicos
Is null, between, in,like	Comparadores especiales
Not,and, or	Operadores logicos clasicos

La sentencia CREATE TABLE: Operadores en Oracle SQL

Oracle dispone de varios operadores que pueden utilizarse tanto en consultas como para definir condiciones o restricciones de tabla o de columna:

Comparador **IS NULL:** Detecta valores nulos (devuelve verdadero/falso)

(A=10), A is null ⇒ false, A is not null⇒true

Comparador **BETWEEN**: Detecta valores entre dos constantes (devuelve verdadero/falso)

between x and y ⇔ >= x and <=y

Comparador **IN**: Detecta pertenencia a conjunto (devuelve verdadero/falso)

a in (1,2,3) ⇒ true si a=1 o a=2 o a=3

Comparador **LIKE:** Sirve para utilizar "mascaras" en cadenas de caracteres

"-" sustituye cualquier carácter

"%" sustituye cualquier cadena

Ejemplos:

x LIKE '-A--' ⇒ true si x tiene 4 caracteres y 'A' en la segunda posición. Ejemplo: true si x='1A23', false si x='1A234')

x LIKE '%A%' ⇒ true si x tiene un número indefinido de caracteres antes de una A, y un número indefinido de caracteres después de la misma. Ejemplo: true si X='ABLA'

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores ‘si’ o ‘no’ (por defecto, valor ‘no’), y que el sexo tenga valores exclusivamente ‘v’ o ‘m’.

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombre varchar(10) not null ,
                       edad number(3)
                      constraint al2 check   (edad between 17 and 90) ,
                       provincia varchar(10),
                       beca char(2) DEFAULT ‘no’ check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores 'si' o 'no' (por defecto, valor 'no'), y que el sexo tenga valores exclusiva

Darle un nombre a la restricción es opcional

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombre varchar(10) not null ,
                       edad number(3)
                      constraint al2 check   (edad between 17 and 90) ,
                       provincia varchar(10),
                       beca char(2) DEFAULT 'no' check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligato
tenga exclusivamente
tenga valores exclusiva

El atributo DNI es la llave primaria. **Sólo un atributo puede ser llave primaria con restricciones de columna.** En el caso de que la llave primaria sea compuesta, se debe hacer como restricción de tabla.

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombre varchar(10) not null ,
                       edad number(3)
                      constraint al2 check   (edad between 17 and 90) ,
                       provincia varchar(10),
                       beca char(2) DEFAULT 'no' check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores 'si' o 'no' (por defecto, valor 'no'), y que el sexo tenga valores exc

El nombre y los apellidos son obligatorios: No pueden tener valor "nulo" o desconocido

```
create table alu
        ape1 varchar(10) not null,
        ape2 varchar(10) not null ,
        nombre varchar(10) not null ,
         edad number(3)
        constraint al2 check   (edad between 17 and 90) ,
         provincia varchar(10),
         beca char(2) DEFAULT 'no' check (beca in ('si','no')),
         sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores 'si' o 'no' (por defecto, valor 'no'), y que el sexo tenga valores exclusivamente 'v' o 'm'.

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombr
                       edad number(3)
                      constraint al2 check   (edad between 17 and 90) ,
                       provincia varchar(10),
                       beca char(2) DEFAULT 'no' check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

La edad debe estar entre 17 y 90

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores ‘si’ o ‘no’ (por defecto, valor ‘no’), y que el sexo tenga valores exclusivamente ‘v’ o ‘m’.

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombre varchar(10) not null ,
                       edad number(3)
                      constraint
                       provincia varchar(10),
                       beca char(2) DEFAULT ‘no’ check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

El valor por defecto para la beca es ‘no’

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores ‘si’ o ‘no’ (por defecto, valor ‘no’), y que el sexo tenga valores exclusivamente ‘v’ o ‘m’.

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombre varchar(10) not null ,
                       edad number(3)
                      constraint
                       provincia varchar(10),
                       beca char(2) DEFAULT ‘no’ check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

Los valores de beca deben estar en el conjunto (‘si’, ‘no’)

La sentencia CREATE TABLE: Ejemplo de tabla con restricciones de columnas

Crear una tabla alumnos con DNI, nombre, apellidos, edad, provincia, beca y sexo. Se debe cumplir que DNI sea llave primaria, que el nombre y los apellidos sean obligatorios, que la edad esté entre 17 y 90 años, que la beca tenga exclusivamente valores ‘si’ o ‘no’ (por defecto, valor ‘no’), y que el sexo tenga valores exclusivamente ‘v’ o ‘m’.

```
create table alumnos (dni varchar(8) constraint al1 primary key,
                      ape1 varchar(10) not null,
                      ape2 varchar(10) not null ,
                      nombre varchar(10) not null ,
                       edad number(3)
                      constraint al2 check  (edad between 17 and 90)
                       provincia
                       beca char(2) constraint al3 check (beca in ('si','no')),
                       sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

Los valores de sexo deben estar en el conjunto (‘v’, ‘m’)

La sentencia CREATE TABLE: Restricciones de tablas

CREATE TABLE [usuario.]nombre_tabla
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)

restricciones de tabla debe tener el formato:

UNIQUE(atrib., atrib…): no se repiten valores en tuplas distintas
PRIMARY KEY (atrib., atrib…): las columnas implicadas forman la llave primaria (UNIQUE +NOT NULL)
FOREIGN KEY: las columnas implicadas forman llave externa a la llave primaria de otra tabla, indicada con REFERENCES.
CHECK permite condiciones lógicas entre varias columnas

Las condiciones se almacenan en el catálogo, para reconocerlas fácilmente es bueno darles nombre.

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser ‘ba’, ‘ra’, ‘es’, ‘op’. El tipo debe ser ‘semestral’ o ‘anual’ (por defecto semestral). Si la asignatura es semestral, los créditos totales deben estar entre 4.5 y 9. Si es anual, entre 6 y 12.

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) not null ,
            credt number(4,1) not null ,
        credpr number(4,1) not null,
        caracter char(2) check (caracter in (‘ba','ra','es','op')),
        tipo varchar(9) DEFAULT ‘semestral’ check (tipo in
(‘semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser ‘ba’, ‘ra’, ‘es’, ‘op’. El tipo debe ser ‘semestral’ o ‘anual’ (por defecto sem[illegible] créditos totales deben estar entre 4

Código de asignatura: Llave primaria

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) not null ,
            credt number(4,1) not null ,
        credpr number(4,1) not null,
        caracter char(2) check (caracter in (‘ba','ra','es','op')),
        tipo varchar(9) DEFAULT ‘semestral’ check (tipo in
(‘semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser ‘ba’, ‘ra’, ‘es’, ‘op’. El tipo debe ser ‘semestral’ o ‘anual’ (por defecto semestral). Si la asignatura es semestral, los créditos totales deben estar entre

Nombre, grado y créditos no nulos (obligatorios)

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) not null ,
            credt number(4,1) not null ,
        credpr number(4,1) not null,
        caracter char(2) check (caracter in (‘ba','ra','es','op')),
        tipo varchar(9) DEFAULT ‘semestral’ check (tipo in
(‘semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser ‘ba’, ‘ra’, ‘es’, ‘op’. El tipo debe ser ‘semestral’ o ‘anual’ (por defecto semestral). Si la asignatura es semestral, los créditos totales deben estar entre 4.5 y 9. Si es anual, entre 6 y 12.

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) n
            credt number(4,1) n
        credpr number(4,1) not null,
        caracter char(2) check (caracter in (‘ba',‘ra',‘es',‘op')),
        tipo varchar(9) DEFAULT ‘semestral’ check (tipo in
(‘semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

El carácter de la asignatura sólo puede ser básica, de rama, de especialidad u optativa

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser ‘ba’, ‘ra’, ‘es’, ‘op’. El tipo debe ser ‘semestral’ o ‘anual’ (por defecto semestral). Si la asignatura es semestral, los créditos totales deben estar entre 4.5 y 9. Si es anual, entre 6 y 12.

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) not null ,
            credt num
        credpr numbe
        caracter char(2) check (caracter in (‘ba',‘ra',‘es',‘op')),
        tipo varchar(9) DEFAULT ‘semestral’ check (tipo in
(‘semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

El tipo de asignatura es por defecto semestral

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser ‘ba’, ‘ra’, ‘es’, ‘op’. El tipo debe ser ‘semestral’ o ‘anual’ (por defecto semestral). Si la asignatura es semestral, los créditos totales deben estar entre 4.5 y 9. Si es anual, entre 6 y 12.

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) not null ,
            credt number(4,1)
        credpr number(4,1) no
        caracter char(2) check (caracter in (‘ba',‘ra',‘es',‘op')),
        tipo varchar(9) DEFAULT ‘semestral’ check (tipo in
(‘semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

El tipo de asignatura sólo puede ser ‘semestral’ o ‘anual’

La sentencia CREATE TABLE: Ejemplo de tabla

Crear una tabla asignatura con código de asignatura, nombre, grado, créditos teóricos y prácticos, carácter, y tipo. La llave primaria es el código de asignatura. Son obligatorios el nombre, grado y los créditos. El carácter sólo puede ser 'ba', 'ra', 'es', 'op'. El tipo debe ser 'semestral' o 'anual' (por defecto semestral). Si la asignatura es semestral, los créditos totales deben estar entre 4.5 y 9. Si es anual, entre 6 y 12.

```
create table asigna (c_asi varchar(4) primary key ,
            nombreas varchar(30) not null,
            grado varchar(20) not null ,
            credt number(4,1) not null ,
        credpr n
        caracter
        tipo varc
('semestral','anual')),
            check ((tipo = semestral ' and credt+credpr between 4.5 and 9)
                or (tipo ='anual' and credt+credpr between 6 and 12)))
```

Restricción de tabla: Implica varios atributos.
Los créditos deben estar entre 4.5 y 9 para asignaturas semestrales, o entre 6 y 12 para asignaturas anuales

La sentencia CREATE TABLE: Ejemplo de tabla con llaves externas

Crear una tabla matricula, que indique que un alumno está matriculado en una asignatura en un curso académico concreto (relación muchos-muchos con atributo discriminador entre alumno y asignatura).

```
create table matricula (
        c_asi varchar(4)  not null REFERENCES asignatura.c_asi,
        dni varchar(8) not null,
    FOREIGN KEY(dni) REFERENCES alumno.dni
        curso VARCHAR(5) not null,
        PRIMARY KEY(c_asi, dni, curso)
)
```

La sentencia CREATE TABLE: Ejemplo de tabla con llaves externas

Crear una tabla matricula, que indique que un alumno está matriculado en una asignatura en un curso académico concreto (relación muchos-muchos con atributo discriminador entre alumno y asignatura).

Restricción de columna: Implica sólo el atributo sobre el que se aplica.
Llave externa al atributo c_asi de asignatura

```
create table matricula(
        c_asi varchar(4)  not null REFERENCES asigna.c_asi
        dni varchar(8) not null,
    FOREIGN KEY(dni) REFERENCES alumno.dni
        curso VARCHAR(5) not null,
        PRIMARY KEY(c_asi, dni, curso)
)
```

La sentencia CREATE TABLE: Ejemplo de tabla con llaves externas

Crear una tabla matricula, que indique que un alumno está matriculado en una asignatura en un curso académico concreto (relación muchos-muchos con atributo discriminador entre alumno y asignatura).

Restricción de tabla: Implica uno o varios atributos.
DNI es llave externa al atributo dni de alumno

```
create ta                                              .c_asi,
        dni varchar(8) not null,
FOREIGN KEY(dni) REFERENCES alumno.dni
        curso VARCHAR(5) not null,
        PRIMARY KEY(c_asi, dni, curso)
)
```

La sentencia CREATE TABLE: Ejemplo de tabla con llaves externas

Crear una tabla matricula, que indique que un alumno está matriculado en una asignatura en un curso académico concreto (relación muchos-muchos con atributo discriminador entre alumno y asignatura).

```
create table matricula (
                                                    asigna.c_asi,
                                                    ni
    curso VARCHAR(5) not null,
    PRIMARY KEY(c_asi, dni, curso)
)
```

Restricción de tabla: Implica uno o varios atributos. La llave primaria está formada por c_asi, dni y curso

La sentencia CREATE TABLE: Ejercicio de creación de tablas

Disponemos de la siguiente BD que gestiona los invitados a una boda

- Además de las restricciones de integridad especificadas en el dibujo, deben considerarse las siguientes:
- El nombre, la dirección y la ciudad de los invitados son cadenas de caracteres.
- Siempre debe registrarse la dirección de los invitados.
- La ciudad de procedencia de los invitados está restringida a los valores: Málaga, Granada, Jaén y Almería, siendo Granada el valor por defecto.
- El número de personas asistentes por invitado es, por defecto, 1.
- La fecha de reserva de los regalos debe estar entre el 22 de Mayo y el 22 de Junio de 2017.

La sentencia DROP TABLE

El formato de la sentencia es:

DROP TABLE *nombretabla*

Elimina los datos de una tabla y su esquema de la BD.
No se podrá eliminar una tabla si existen otras que hacen referencia a la misma, mediante llaves externas.

En este caso, se deberán eliminar las restricciones de llaves externas de las otras tablas, o eliminar dichas tablas con DROP TABLE.

Ejemplo:

DROP TABLE asigna Error. Matricula hace referencia
DROP TABLE Matricula ok
DROP TABLE asigna ok

La sentencia ALTER TABLE

El formato de la sentencia es:

```
ALTER TABLE [usuario].table
[ADD ({datos_columna | restricciones de tabla}
[{datos_columna | restricciones de tabla} ...) ]
[MODIFY (datos_columna [,datos_columna]  ...)]
[DROP CONSTRAINT restriccion]
[PCTFREE n], [PCTUSED n], [INITRANS n ], [MAXTRAN n]
[TABLESPACE nombre],  [STORAGE nombre]
  [BACKUP]
```

Altera el esquema de una tabla, añadiendo/eliminando/modificando atributos o restricciones.

Cuando se modifica una columna solo se puede alterar la restricción de no null

Para alterar otras restricciones hay que borrarlas y volverlas a añadir

La sentencia ALTER TABLE: Ejemplo

```
alter table alumnos add (origen char(2)
        check (origen in ('cu','lo','fp','es','ot'),
        media number(2,2))

alter table alumnos modify (nombre null)

alter table alumnos drop constraint al3

alter table alumnos add ( constraint al3
         check (edad between 18 and 80))
```

La sentencia ALTER TABLE: Ejemplo

Inserta un nuevo atributo “origen” en la tabla alumnos, con la restricción de que sólo pueda tener valores ‘cu’, ‘lo’, ‘fp’, ‘es’, ‘ot’; y otro atributo media de tipo numérico

```
alter table alumnos add (origen char(2)
        check (origen in (‘cu’,’lo’,’fp’,’es’,’ot’),
        media number(2,2))

alter table alumnos modify (nombre null)

alter table alumnos drop constraint al2

alter table alumnos add ( constraint al2
        check (edad between 18 and 80))
```

La sentencia ALTER TABLE: Ejemplo

Modifica el atributo nombre de la tabla alumnos, permitiendo que pueda tener valores nulos

```
alter table alumnos modify (nombre null)
```

```
alter table alumnos drop constraint al2
```

```
alter table alumnos add ( constraint al2
        check (edad between 18 and 80))
```

La sentencia ALTER TABLE: Ejemplo

```
alter table alumnos add (origen char(2)
        check (origen in ('cu','lo','fp','es','ot'),
        media number(2,2))
```

Elimina la restricción de alias al2 que habíamos creado en la tabla alumnos

```
alter table alumnos drop constraint al2
```

```
alter table alumnos add ( constraint al2
        check (edad between 18 and 80))
```

La sentencia ALTER TABLE: Ejemplo

```
alter table alumnos add (origen char(2)
        check (origen in ('cu','lo','fp','es','ot'),
        media number(2,2))

alter table alumnos modify (nombre null)
```

Inserta una nueva restricción en la tabla alumnos

t al2

```
alter table alumnos add ( constraint al2
        check (edad between 18 and 80))
```

La sentencia INSERT

El formato de la sentencia es:

Insert into *tabla* [(*columna*, ...)]{values (*valor*,...) }| *consulta*]

Inserta tuplas (todas las columnas o en una selección de las columnas de una tabla), con valores literales o procedentes de una consulta

Ejemplo de inserción desde consulta:

```
Insert into alumnos_buenos (ape1,ape1,nombre,nota)
select ape1,ape2,nombre,media from alumnos where
media >=7.5;
```

La sentencia INSERT: Ejemplo

Ejemplo de inserción en asignatura:

INSERT INTO asigna VALUES('FBD', 'Fundamentos de Bases de Datos', 'GII', 1.5, 4.5, 'ra', 'semestral')

Como el tipo es por defecto semestral, podríamos también haber escrito:

INSERT INTO asigna(c_asig, nombreas, grado, credt, credp, caracter) VALUES('FBD', 'Fundamentos de Bases de Datos', 'GII', 1.5, 4.5, 'ra')

La sentencia INSERT: Ejemplo

Ejemplo de inserción en Regalo_Reservado:

INSERT INTO Regalo_Reservado VALUES(‘1234AAA', ‘Pepito‘, TO_DATE(‘12/12/2017’))

Inserción utilizando la fecha actual:

INSERT INTO Regalo_Reservado VALUES(‘1234AAA', ‘Pepito‘, SYSDATE)

La sentencia DELETE

Delete tabla [where *condicion*]

- Si se omite la condición borra todas la tuplas de la tabla
- Si se pone una condición de llave candidata borra una tupla concreta
- La condición puede incluir comparadores de conjunto y ser tan compleja como se quiera

Ejemplo:

```
delete asigna where not exists(select * from matricula
where
Matricula.c_asi=asigna.c_asi);
```

Elimina aquellas asignaturas que no tienen alumnos matriculados.

Ejemplo:

```
delete asigna where nombreas LIKE '%A%';
```

Elimina aquellas asignaturas cuyo nombre contenga una A.

La sentencia UPDATE

El formato de la sentencia es:

Update *tabla* set *columna=expr.* [*columna=exp.* ...] [where *condicion*]
o alternativamente
Update *tabla* set (*columna*[,*columna,* ...]) =(*consulta*)
[(*columna*[,*columna*, ...])=(*consulta*)]... [where *condicion*]

- Actualiza las tuplas que verifican la condición expresada con la misma filosofía que el borrado
- Permite sustituir valores bien con expresiones bien con valores resultantes de consultas, estas pueden ser de cualquier tipo.

Ejemplo:

Update asigna set credt=3 where tipo='semestral'

La sentencia UPDATE: Ejemplo avanzado

Actualiza los créditos teóricos y prácticos de todas las asignaturas optativas de 4º curso al valor máximo de dichos campos para todas las asignaturas optativas , independientemente del curso que sean.

```
update asigna asig set (asig.credt,asig.credpr)=(select
max(credt),max(credpr) from asigna where caracter='op')
where asig.carácter='op' and asig.curso='4'
```

La sentencia SELECT (I)

Sirve para obtener información (tuplas, cálculos con valores de tuplas) de las tablas existentes en la Base de Datos.

Formato general:

```
SELECT [ALL|DISTINCT] { * | table.* | expre  [c_alias]}
    [, {table.*| expre [c_alias]}]... FROM [usuario].tabla [t_alias]
    [,[usuario].tabla [t_alias]]... WHERE condicion
    [CONNECT BY condicion [START BY condicion]]
     GROUP BY expre, [expre]... [HAVING condicion]]
    {UNION|INTERSEC|MINUS} SELECT ...
    [ORDER BY {expre|posicion}  [ASC|DESC] [,
    {expre|posicion}  [ASC|DESC]
    FOR UPDATE OF columna [,columna]...[NOWAIT]
```

Opciones que no veremos

Opciones avanzadas

La sentencia SELECT (II)

[**ALL|DISTINCT**] Permite generar tuplas repetidos o no. Por defecto ALL

{ * | table.* | expre [c_alias]} [, {table.*| expre [c_alias]}]... Es el objetivo de la consulta.

El ***** significa obtener todos los campos de la(s) tabla(s)

"expre" es una expresión con campos de la(s) tabla(s). Puede ser un nombre de columna (**realiza la proyección**), o una combinación o función de estas.

c_alias permite dar un nombre a la columna de salida

[ORDER BY {expre|posicion} [ASC|DESC] [, {expre|posicion} [ASC|DESC] Permite ordenar la consulta

La sentencia SELECT (III)

[FROM [usuario].tabla [t_alias [,[usuario].tabla [t_alias]]... indica que tabla(s) se va(n) a utilizar.

Si aparece más de una tabla se establece el **producto cartesiano** de las tablas implicadas.

Los **alias** permiten nombrar una tabla de forma distinta, para evitar conflictos de nombres (realizar el producto de una tabla consigo misma o referenciar distintas tuplas de una misma tabla).

WHERE condicion es una condición que se impone a la consulta, y que deben cumplir todas las tuplas resultantes de la misma. La condición puede ser tan compleja como se quiera, pudiendo incluso llegar a realizar “consultas anidadas”

La sentencia SELECT (IV): Ejemplo visual

SELECT * FROM PROFESORES WHERE CATEGORIA='AS'

NRP	NOM_PROF	CATEGORIA	AREA	COD_DEP
2428456	Juan Sánchez Pérez	AS	COMPUT	CCIA
24283256	Antonia Pérez Rodríguez	CU	COMPUT	CCIA
242256	Luis Pérez Pérez	TE	LENGUA	LSI
84256	Carmen Pérez Sánchez	TU	LENGUA	LSI
324256	David Pérez Jiménez	CU	ARQUIT	ATC
24256	María López Ruiz	TU	ARQUIT	ATC
2842560	José Álvarez Pérez	CE	ELECTR	ELEC
842560	Adela Pérez Sánchez	AS	ELECTR	ELEC
84560	Luis Martínez Pérez	AS	TSEÑAL	TESE
242560	María Gómez Sánchez	CU	TSEÑAL	TESE

Salida (resultado):

NRP	NOM_PROF	CATEGORIA	AREA	COD_DEP
2428456	Juan Sánchez Pérez	AS	COMPUT	CCIA
842560	Adela Pérez Sánchez	AS	ELECTR	ELEC
84560	Luis Martínez Pérez	AS	TSEÑAL	TESE

La sentencia SELECT. Ejemplos

Ejemplo 1: Selecciona todos los elementos de la tabla alumnos y todos sus atributos, y los devuelve ordenados por apellidos y nombre

```
select *  from alumnos order by ape1,ape2,nombre ;
```

Ejemplo 2: Selecciona el nombre y los apellidos de los alumnos menores de 25 años y los devuelve ordenados por apellidos y nombre.

```
select nombre,ape1,ape2 from alumnos  where
   edad <=25 order by edad desc, ape1,ape2,nombre;
```

Ejemplo 3: Selecciona el nombre y los apellidos de aquellos alumnos entre 20 y 30 años que son de Andalucia Oriental. Los devuelve sin ordenar.

```
select dni,nombre,ape1,ape2 from alumnos
    where    (edad between 20 and 30) and
provincia in ('Jaen','Granada','Almeria'))
```

La sentencia SELECT. El producto cartesiano

El producto cartesiano de 2 o más tablas nos permite crear una nueva tabla auxiliar/intermedia que contiene la unión disjunta de todos los atributos de las tablas. Las tuplas del producto cartesiano se obtienen como la combinación de todas las tuplas de las tablas involucradas con todas.

Ejemplo: SELECT * from T1, T2

Tabla T1

A	B
a_1	b_1
a_2	b_2
a_3	b_3

×

Tabla T2

D
d_1
d_2

=

A	B	D
a_1	b_1	d_1
a_1	b_1	d_2
a_2	b_2	d_1
a_2	b_2	d_2
a_3	b_3	d_1
a_3	b_3	d_2

La sentencia SELECT. El producto cartesiano

Ejemplo: SELECT * from profesores, departamentos

NRP	NOM_PROF	CATG.	AREA.	COD_DEP	COD_DEP	NOM_DEP	DIRECTOR
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	CCIA	Ciencias de la Computacion	24283256
24283256	Antonia Perez Rodriguez	CU	COMPUT	CCIA	CCIA	Ciencias de la Computacion	24283256
242256	Luis Perez Perez	TE	LENGUA	LSI	CCIA	Ciencias de la Computacion	24283256
84256	Carmen Perez Sanchez	TU	LENGUA	LSI	CCIA	Ciencias de la Computacion	24283256
324256	David Perez Jimenez	CU	ARQUIT	ATC	CCIA	Ciencias de la Computacion	24283256
24256	Maria Lopez Ruiz	TU	ARQUIT	ATC	CCIA	Ciencias de la Computacion	24283256
2842560	Jose Alvarez Perez	CE	ELECTR	ELEC	CCIA	Ciencias de la Computacion	24283256
842560	Adela Perez Sanchez	AS	ELECTR	ELEC	CCIA	Ciencias de la Computacion	24283256
84560	Luis Martinez Perez	AS	TSECAL	TESE	CCIA	Ciencias de la Computacion	24283256
242560	Maria Gomez Sanchez	CU	TSECAL	TESE	CCIA	Ciencias de la Computacion	24283256
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	LSI	Lenguajes y Sistemas	84256
...	...	...	...	...	...	...	...
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	ATC	Arquitectura de Computadores	324256
...	...	...	...	...	...	...	...
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	ELEC	Electronica	2842560
...	...	...	...	...	...	...	...
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	TESE	Teoria de la Secal	84560
24283256	Antonia Perez Rodriguez	CU	COMPUT	CCIA	TESE	Teoria de la Secal	84560
242256	Luis Perez Perez	TE	LENGUA	LSI	TESE	Teoria de la Secal	84560
84256	Carmen Perez Sanchez	TU	LENGUA	LSI	TESE	Teoria de la Secal	84560
324256	David Perez Jimenez	CU	ARQUIT	ATC	TESE	Teoria de la Secal	84560
24256	Maria Lopez Ruiz	TU	ARQUIT	ATC	TESE	Teoria de la Secal	84560
2842560	Jose Alvarez Perez	CE	ELECTR	ELEC	TESE	Teoria de la Secal	84560
842560	Adela Perez Sanchez	AS	ELECTR	ELEC	TESE	Teoria de la Secal	84560
84560	Luis Martinez Perez	AS	TSECAL	TESE	TESE	Teoria de la Secal	84560
242560	Maria Gomez Sanchez	CU	TSECAL	TESE	TESE	Teoria de la Secal	84560

La sentencia SELECT. El producto cartesiano

Nos interesa juntar 2 tablas con un producto cartesiano, pero limitando las tuplas resultantes a las que nos interesen. **Ejemplo:** Seleccionar el nombre del departamento dirigido por “Perico el de los Palotes”

SELECT departamento.nombre from departamento, profesores where profesores.nombre=‘Perico el de los Palotes’ MAL

El producto cartesiano hace mezcla de todos con todos. Saldrían los nombres de todos los departamentos

SELECT departamento.nombre from departamento, profesores where profesores.nombre=‘Perico el de los Palotes’ AND departamento.director=profesores.NRP BIEN

Así sólo nos quedamos con las tuplas donde el director del departamento coincide con el dni del profesor.

La sentencia SELECT. El producto cartesiano

NRP	NOM_PROF	CATG.	AREA.	COD_DEP	COD_DEP	NOM_DEP	DIRECTOR
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	CCIA	Ciencias de la Computacion	24283256
24283256	Antonia Perez Rodriguez	CU	COMPUT	CCIA	CCIA	Ciencias de la Computacion	24283256
242256	Luis Perez Perez	TE	LENGUA	LSI	CCIA	Ciencias de la Computacion	24283256
84256	Carmen Perez Sanchez	TU	LENGUA	LSI	CCIA	Ciencias de la Computacion	24283256
324256	David Perez Jimenez	CU	ARQUIT	ATC	CCIA	Ciencias de la Computacion	24283256
24256	Maria Lopez Ruiz	TU	ARQUIT	ATC	CCIA	Ciencias de la Computacion	24283256
2842560	Jose Alvarez Perez	CE	ELECTR	ELEC	CCIA	Ciencias de la Computacion	24283256
842560	Adela Perez Sanchez	AS	ELECTR	ELEC	CCIA	Ciencias de la Computacion	24283256
84560	Luis Martinez Perez	AS	TSECAL	TESE	CCIA	Ciencias de la Computacion	24283256
242560	Maria Gomez Sanchez	CU	TSECAL	TESE	CCIA	Ciencias de la Computacion	24283256
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	LSI	Lenguajes y Sistemas	84256
...	...	...	...	...	...	...	...
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	ATC	Arquitectura de Computadores	324256
...	...	...	...	...	...	...	...
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	ELEC	Electronica	2842560
...	...	...	...	...	...	...	...
2428456	Juan Sanchez Perez	AS	COMPUT	CCIA	TESE	Teoria de la Secal	84560
24283256	Antonia Perez Rodriguez	CU	COMPUT	CCIA	TESE	Teoria de la Secal	84560
242256	Luis Perez Perez	TE	LENGUA	LSI	TESE	Teoria de la Secal	84560
84256	Carmen Perez Sanchez	TU	LENGUA	LSI	TESE	Teoria de la Secal	84560
324256	David Perez Jimenez	CU	ARQUIT	ATC	TESE	Teoria de la Secal	84560
24256	Maria Lopez Ruiz	TU	ARQUIT	ATC	TESE	Teoria de la Secal	84560
2842560	Jose Alvarez Perez	CE	ELECTR	ELEC	TESE	Teoria de la Secal	84560
842560	Adela Perez Sanchez	AS	ELECTR	ELEC	TESE	Teoria de la Secal	84560
84560	Luis Martinez Perez	AS	TSECAL	TESE	TESE	Teoria de la Secal	84560
242560	Maria Gomez Sanchez	CU	TSECAL	TESE	TESE	Teoria de la Secal	84560

La sentencia SELECT. Ejemplos

Ejemplo 4: Selecciona la lista de todos los grados existentes, ordenados descendentemente.

```
select distinct grado from asigna order by grado desc ;
```

Ejemplo 5: Selecciona el nombre y los apellidos de los alumnos menores de 25 años matriculados de la asignatura 'bd1s'.

```
select nombre,ape1,ape2 from alumnos, matricula where
(edad >25) and ( alumnos.dni=matricula.dni and
matricula.c_asi='bd1s') order by ape1,ape2,nombre;
```

Ejemplo 6: Selecciona los nombre de asignaturas optativas de 4.5 o más créditos de las que está matriculado 'Jose Lopez Perez'

```
select nombreas from alumnos,asigna,matricula
   where (carácter='op' and credt+credpt>=4.5 and
ape1='Jose' and ape2='Lopez' and nombre='Perez' and
alumnos.dni=matricula.dni and
matricula.c_asi=asigna.c_asi)
```

La sentencia SELECT. Ejemplos

Ejemplo 4: Selecciona la lista de todos los grados existentes, ordenados descendentemente.

```
select distinct grado from asigna order by grado desc ;
```

Ejemplo 5: Selecciona el nombre [illegible] nos menores de 25 años matriculados de la asignatura bd1s.

Producto cartesiano de 2 tablas

```
select nombre,ape1,ape2 from alumnos, matricula where
(edad >25) and ( alumnos.dni=matricula.dni and
matricula.c_asi='bd1s') order by ape1,ape2,nombre;
```

Ejemplo 6: Selecciona los [illegible] tativas de 4.5 o más créditos de las que está matriculado 'Jose' 'Lopez Perez'

Producto cartesiano de 3 tablas

```
select nombreas from alumnos,asigna,matricula
   where (carácter='op' and credt+credpt>=4.5 and
ape1='Jose' and ape2='Lopez' and nombre='Perez' and
alumnos.dni=matricula.dni and
matricula.c_asi=asigna.c_asi)
```

La sentencia SELECT. Ejemplos

Ejemplo 4: Selecciona la lista de todos los grados existentes, ordenados descendentemente.

```
select distinct grado from asigna order by grado desc ;
```

Ejemplo 5: Selecciona el nombre y los apellidos de los alumnos menores de 25 a

```
select no...a where
(edad >25) and ( alumnos.dni=matricula.dni and
matricula.c_asi='bd1s') order by ape1,ape2,nombre;
```

Condición para coger sólo las tuplas del producto cartesiano que son del mismo alumno y matrícula

Ejemplo 6: Selecciona los nombre de asignaturas optativas de 4.5 o más créditos de las que está matriculado 'Jose Lopez Perez'

```
select nombreas from alumnos,asigna,matricula
...t>=4.5 and
...bre='Perez' and
alumnos.dni=matricula.dni an...
matricula.c_asi=asigna.c_asi)...
```

Condición para coger sólo las tuplas de matrícula (producto cartesiano) que son del mismo estudiante.

Condición para coger sólo las tuplas del productor cartesiano donde la matrícula es de su asignatura

Funciones de agregación

Idea básica: Utilizar funciones cuyo resultado sea un “resumen” de los datos de una columna de una tabla.

Forma general de uso: funcion(*expresion)*

Funciones existentes:

- **AVG(expr)** calcula la media de la expresion dada,
- **STDDEV(expr)** calcula la desviación típica,
- **VARIANCE(expr)** calcula la varianza. Ignoran valores nulos:

select avg(edad), stddev(edad) from alumnos where sexo='v'

- **MIN(expr)** calcula el minimo de la expresion dada,
- **MAX(expr)** calcula el maximo

select min(credt+credpr),max(credt+credpr) from asigna

Operadores booleanos adicionales (I)

Forma general de los operadores booleanos adicionales SQL:

[*expresion*] [not] *operador* (*conjunto*)

La **expresion** puede ser una sucesion de expresiones o nombres de columnas..

Conjunto puede ser un conjunto literal o una consulta

Los operadores pueden ser:

IN (ya conocido)

{= | ¡= | < | > | <= | >=} ANY compara con todos los elementos del conjunto citado y es cierta si se cumple la la condición **se cumple para alguno**.

{= | ¡= | < | > | <= | >=} ALL compara con todos los elementos del conjunto citado y es cierta si la condicion **se cumple para todos**.

Operadores booleanos adicionales (II)

Los operadores pueden ser (continuación):
EXISTS detecta si el conjunto está o no vacío.

Los conjuntos asociados pueden ser descritos mediante una sentencia **SELECT** con lo que se obtienen "selects anidados".

Ejemplos:

```
select alumnos.dni,ape1,ape2,nombre from
matricula,alumnos where
    c_asi in (select asigna.c_asi from asigna where
caracter='op') and alumno.dni=matricula.dni order by
ape2,nombre,ape1;
```

Selecciona los alumnos matriculados de alguna asignatura optativa.

Operadores booleanos adicionales (III)

Ejemplos:

select dni from matricula where c_asi='fbd' and curso='2017-2018' and calificacion >= all (select calificacion from matricula);

Selecciona aquellos alumnos que han obtenido la máxima calificación en fbd en el curso 2017-2018

select c_asi,nombreas from asigna where curso>= all(select curso from asigna)

Selecciona asignaturas de último curso

Operadores booleanos adicionales (IV)

Ejemplos:

```
select distinct dni from matricula where
   c_asi in (select asigna.c_asi from asigna where
   curso <=all(select asAlias.curso from asigna asAlias))
```

Selecciona alumnos matriculados de asignaturas del curso más inferior

```
select dni,ape1,ape2,nombre from alumnos
   where exists (select * from matricula where
alumnos.dni=matricula.dni and c_asi='FBD');
```

Selecciona los alumnos matriculados de FBD

Operadores booleanos adicionales (IV)

Ejemplos:

select c_asi,nombreas from asigna where not exists (select * from matricula where asignatura.c_asi=matricula.c_asi);

Selecciona asignaturas de las que no está matriculado ningún alumno

select c_asi,nombreas from asigna where curso >=all(select asAlias.curso from asigna asAlias) and not exists (select * from matricula where asignatura.c_asi=matricula.c_asi);

Selecciona asignaturas de último curso de las que no está matriculado ningún alumno

Operadores conjuntistas

Forma general:

(consulta select ……)
[UNION, INTERSECT, MINUS]
(otra consulta select….)

Ejemplos:

```
(select dni from alumnos)
      minus
   (select alumnos.dni from alumnos, alumnos al
    where (al. edad < alumnos.edad) )
```

Selecciona los alumnos más jóvenes: Coge todos los alumnos y les quita aquellos para los que hay otro alumno con edad menor.

Operadores conjuntistas

Los operadores conjuntistas se aplican sobre las tuplas. Ejemplo con el operador de UNION:

r

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_1	c_1
a_4	b_1	c_1
a_4	b_2	c_2

s

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_2	c_2
a_4	b_2	c_2
a_1	b_2	c_2

r ∪ s

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_1	c_1
a_4	b_1	c_1
a_4	b_2	c_2
a_3	b_2	c_2
a_1	b_2	c_2

Operadores conjuntistas

Los operadores conjuntistas se aplican sobre las tuplas. Ejemplo con el operador de diferencia (MINUS):

r

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_1	c_1
a_4	b_1	c_1
a_4	b_2	c_2

s

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_2	c_2
a_4	b_2	c_2
a_1	b_2	c_2

r – s

A	B	C
a_3	b_1	c_1
a_4	b_1	c_1

Operadores conjuntistas

Ejemplos:

```
(select c_asi from asigna where credt+credpr >6 )
        intersect
 (select codas from matricula where curso='2017-2018')
```

Selecciona aquellas asignaturas de más de seis créditos vigentes en el curso 2017-2018: Coge todas las asignaturas de más de 6 créditos por una parte, las del curso 17/18 por otra, y hace la intersección.

División

Consultas relacionadas con la conexión de un elemento de un conjunto con "**todos**" los elementos de otro.

Algunos ejemplos:

- Encontrar los alumnos que están matriculados de **todas** las asignaturas de primer curso.
- Encontrar las asignaturas en las que dan clase **todos** los profesores del área 'COMPUT' que sean de categoría 'CU'.
- Encontrar los profesores que dan clase a **todos** los grupos de la asignatura de código 'BDI'.
- Encontrar las aulas que están ocupadas **todos** los días de la semana.

División

– Ejemplo de división: Encontrar elementos de una tabla donde uno o varios atributos tienen el valor de todos los elementos de otra tabla:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_1	c_3	d_3
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_3	d_1
a_3	b_3	c_3	d_2

÷

D
d_1
d_2

=

A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

División: Consulta SQL según álgebra relacional

Idea Básica:

Sean $D=R \div S$ y r,s,d instancias de D R y S

$\forall a \in D$; $s \subseteq d(a) = \{b \in S / (a,b) \in R\}$

para detectar la inclusion:

$s \subseteq d(a) \Rightarrow s\text{-}d(a) = \varnothing \Rightarrow$ not exists(s-d(a))

Ejemplos:

```
select c_asi,nombreas from asigna where not exists
((select dni from alumnos where provincia='Almeria')
minus (select matricula.dni from matricula where
matricula.c_asi=asigna.c_asi))
```

Asignaturas en que estan matriculados todos los alumnos de Almeria: ***Selección de asignaturas donde el conjunto de todos los alumnos de Almería, al quitarle todos los alumnos de la asignatura, no tenga algún alumno.***

División: Consulta SQL según álgebra relacional

Ejemplos:

```
select ape1,ape2,nombre from alumnos where not exists
((select c_asi from asigna where caracter='op') minus
(select c_asi from matricula where
matricula.dni=alumno.dni))
```

Alumnos matriculados de todas las asignaturas optativas:
Selección de los alumnos donde el conjunto de las asignaturas optativas, al quitarle las asignaturas de las que está matriculado el alumno, esté vacío.

División: Consulta SQL según álgebra relacional

Ejemplos:

```
select ape1,ape2,nombre from alumnos where not exists
 ((select c_asi from asigna where curso=2 and
 grado='GII')  minus
 (select c_asi from matricula where
matricula.dni=alumno.dni
 and calificacion in ('ap','no','sb','mh')))
```

Alumnos que han aprobado todas las asignaturas de segundo de GII. ***Selección de los alumnos para que, al coger todas las asignaturas de 2º de GII y quitarle las asignaturas de las que estaba matriculado y calificado como aprobado, el conjunto queda vacío.***

División: Consulta SQL según álgebra relacional

Ejemplos:

```
select ape1,ape2,nombre from alumnos
 where beca='si' and
not exists((select c_asi from asigna where
credt+credpr>6)
minus (select c_asi from matricula where
matricula.dni=alumnos.dni))
```

Alumnos becarios matriculados de todas las asignaturas de más de seis créditos. ***Selección de los alumnos que tienen beca y para los que, al coger todas las asignaturas de 6 créditos y quitarle aquellas de las que están matriculados, el conjunto que queda es vacío.***

División: Consulta SQL según álgebra relacional

Ejemplos:

```
Select c_asi,nombreas from asigna where not exists (
(select matricula.curso from matricula)
minus
(select ma.curso from matricula ma where
asigna.c_asi=ma.c_asi))
```

Asignaturas que se han impartido todos los años. ***Selección de asignaturas para las que, al coger el curso académico de todas las asignaturas posibles y quitarle los cursos donde la asignatura ha tenido matrículas, el conjunto que queda es vacío.***

División: Consulta SQL según cálculo relacional

Idea básica:

Que todos los elementos cumplan una propiedad es equivalente a que el conjunto de elementos que no la cumplan esté vacio. ***Es decir, no existe un elemento que no cumple la propiedad***.

Ejemplos:

```
Select ape1,ape2,nombre from alumnos where
not exists (select c_asi from asigna where carácter='op'
and not exists(select * from matricula where
matricula.dni=alumno.dni and
matricula.c_asi=asigna.c_asi))
```

Alumnos matriculados de todas las asignaturas optativas.
Selección de alumnos para los que no existe una asignatura optativa para la que no exista una matrícula del alumno.

División: Consulta SQL según cálculo relacional

Ejemplos:

```
Select c_asi,nombreas from asigna where
not exists(select dni from alumnos where
provincia='Almeria' and not exists (select * from
matricula where matricula.c_asi=asigna.c_asi and
matricula.dni=alumno.dni));
```

Asignaturas en las que están matriculados todos los alumnos de Almería. ***Selección de asignaturas para las que no existe un alumno de Almería para el que no exista una matrícula del alumno en dicha asignatura.***

División: Consulta SQL según cálculo relacional

Ejemplos:

```
Select ape1,ape2,nombre from alumnos where
not exists (select asi# from asigna where curso=2 and
grado ='GII' and not exists (select * from matricula where
alumnos.dni=matricula.dni and matricula.c_asi=asigna.c_asi
and calificacion in('np','no',sb','mh')));
```

Alumnos que han aprobado todas las asignaturas de 2º de GII. ***Selección de alumnos para los que no existe una asignatura de 2º de GII para la que no exista una matrícula del alumno en dicha asignatura con calificación de aprobado.***

División: Consulta SQL según cálculo relacional

Ejemplos:

```
select ape1,ape2,nombre from alumnos where beca='si'
and not exists (select c_asi from asigna where
credt+credpr>6 and not exists (select * from matricula
where matricula.dni=alumnos.dni and
matricula.c_asi=asigna.c_asi))
```

Alumnos becarios matriculados de todas las asignaturas de más de seis créditos. ***Selección de alumnos becarios para los que no existe una asignatura de más de 6 créditos para la que no haya una matrícula de dicho alumno.***

División: Consulta SQL según cálculo relacional

Ejemplos:

```
Select c_asi,nombreas from asigna where not exists
(select matricula.curso from matricula where not exists
(select * from matricula ma where asigna.c_asi=ma.c_asi
and matricula.curso=ma.curso));
```

Asignaturas que se han impartido todos los años. ***Selección de asignaturas para las que no exista un curso académico de matrículas en los que no haya una matrícula de esa asignatura.***

La cláusula GROUP BY

Idea básica:

Obtener “tablas resumen” donde cada fila corresponda al valor de uno o varios atributos y las columnas sean funciones de agregación que resuman dicho atributos.

Ejemplos de dichas tablas:

sexo	avg(edad)
v	----
m	----
.	.

carácter	curso	count(asi#)
ba	1	----
ba	2	----
ba	3	----
.	.	.
:	:	:
ra	1	----
ra	2	----
:	:	:

Nótese que en las columnas sólo aparecen los atributos que “resumen” (agrupan) y funciones de agregación

La cláusula GROUP BY

Forma general:

```
SELECT expre,[expre]...from tabla,tabla...
WHERE condicion .....
 GROUP BY expre, [expre]... [HAVING condicion]]
```

Las expresiones detrás de “select” describen el esquema de la tabla resumen (*solo atributos que agrupan y funciones de agregación*)
La condición detrás de “where” restringe las tablas “de entrada” (*involucra atributos de las tablas originales*)
Las expresiones detrás de “group by” definmen los atributos que agrupan (*deben coincidir con los del “select”*)
La condición detrás de “having” restringe la tabla “de salida” (*involucra columnas que aparecen detrás del select*)

La cláusula GROUP BY

Ejemplos:

select sexo, avg(edad) from alumnos group by sexo order by sexo

Para cada sexo distinto, muestra la media de su edad, ordenada por sexo.

select caracter,curso,count(c_asi) from asigna group by carácter, curso order by caracter,curso

Para cada carácter y curso de asignatura distintos, muestra el número de asignaturas que hay de ese tipo..

La cláusula GROUP BY

Ejemplos:

select curso,c_asi,count(*) from matricula group by curso,c_asi order by curso.codas

Para cada curso y asignatura, cuenta el número de matrículas existentes.

select dni,count(*) from matricula where calificacion in ('ap','no','sb','mh') group by dni order by dni

Entre las matrículas de alumnos que han aprobado, para cada alumno muestra el número de matrículas que cumplen esta condición.

La cláusula GROUP BY

Ejemplos:

select sexo, avg(edad) from alumnos group by sexo order by sexo having sexo ='v';

Para cada sexo distinto de alumnos, muestra la media de edad sólo cuando sexo es 'v'

select curso,codas,count(*) from matricula group by curso,codas order by curso.codas having count(*)>=10

Obtiene el numero de alumnos matriculados en cada curso en cada asignatura, pero sólo en los casos en los que haya más de 10 alumnos.

La sentencia CREATE INDEX

Sintaxis:

create [unique] index *indice* on {*tabla* (*columna*[asc|desc], [*columna*[asc|desc]] ...) | *cluster*)
[initrans n] [maxtrans n] [tablespace tablespace] [storage storage] [pctfree n] [nosort]

- **unique** significa que el valor de la clave verifica una condición de unicidad
- **nosort** significa que no hay que ordenar las filas cuando se crea el indice
- Se pueden crear varios índices por tabla
- Permiten mejorar las consultas cuando se accede a la tabla ordenada según el campo clave del índice y cuando consulta según dicho campo
- Permite crear índices compuestos de hasta 16 componentes
- Por defecto el orden es ascendente
- Los índice pueden ralentizar la actualización de las tablas

La sentencia CREATE INDEX

Indices de clave invertida:

Invierten el orden de los bytes de la clave. Optimizan el rendimiento del acceso secuencial en configuraciones paralelas de Oracle.

Sintaxis:

create [unique] index *indice* on {*tabla* (*columna*[asc|desc],[*columna*[asc|desc]] ...) | *cluster*} **reverse**

La sentencia CREATE INDEX

Indices de mapa de bits (Bitmap).

Solo funcionan bien en atributos categóricos y son especialmente útiles cuando el dominio es pequeño

Sintaxis

create **bitmap** index *indice* on {*tabla* (*columna*[asc|desc], [*columna*[asc|desc]] ...) | *cluster*}

La sentencia CREATE INDEX

Tablas organizadas por índices

Son tablas que están organizadas como arboles B de forma que las hojas de los arboles son la tuplas.
Esta forma de la tabla se debe indicar como una clausula adicional en la sentencia CREATE TABLE

Sintaxis

CREATE TABLE *[usuario.]nombre_tabla*
({datos_columna | restricciones de tabla}
[,{datos_columna | restricciones de tabla}]...)
ORGANIZATION INDEX
La tabla debe tener especificada una llave primaria

En el cuaderno de prácticas pueden encontrarse ejemplos de uso de todos estos tipo de índices

La sentencia CREATE VIEW

Sintaxis:

create view *vista* [(*alias* [,*alias*] ...)] as *consulta* [with check option [constraint resticcion]]

- Los **alias** nos permiten renombrar todas las columnas de la vista
- La consulta nos permite construir una visión de usuario tan compleja como queramos. Solo se impide la clausula "order by"
- "with check option" proporciona restricciones adicionales para la actualización mediante vistas

La sentencia CREATE VIEW

- Una vista puede aparecer en cualquier sentencia “select”.
- Una vista puede ser objetivo en una sentencia de actualización; pero hay que tener en cuenta los problemas que la actualización mediante vistas de usuario puede generar.

En el cuaderno de prácticas pueden encontrarse ejemplos de uso de todos estos tipo de índices

La sentencia CREATE CLUSTER

Concepto de Cluster:

- Un “cluster” es una forma de almacenamiento en la que se almacenan juntas la tuplas de distintas tablas que comparten uno o varios campos comunes y se consultan juntas.
- Los cluster se pueden indexar o crear mediante tablas hash

Ejemplo

- Si se van a consultar siempre conjuntamente (sacar listas de alumnos), la tabla asignaturas y la tabla matrícula se pueden almacenar juntas a través del campo código de asignatura.

  Cada ocurrencia de asignatura se almacenaría conjuntamente con las ocurrencias de la tabla matrícula que le corresponden

La sentencia CREATE CLUSTER

Sintaxis :

```
CREATE CLUSTER [usuario].cluster
(columna tipo_de_dato [, columna tipo_de_dato ]…)
[PCTFREE n], [PCTUSED n], [INITRANS n ],
[MAXTRAN n] [TABLESPACE nombre], [STORAGE
nombre],[SIZE n]
```

La sentencia CREATE CLUSTER: Clusters indexados

Ejemplo: Creación del cluster e inserción de las tablas asigna y matricula

```
create cluster listas(asi# varchar(4))
create table asigna (asi# varchar(4) primary key ,
      …)
      cluster listas(asi#)

      create table matricula(asi# varchar(4) references
      asigna,
            …)
            cluster listas(asi#)
```

Creación del índice asociado:

```
create index idx_listas on cluster listas
```

La sentencia CREATE CLUSTER: Clusters Hash

Sintaxis :

CREATE CLUSTER *[usuario].cluster*
(columna tipo_dato) [HASH is columna] SIZE <tamaño>
HASHKEY <cantidad_valores_distintos_de _la _clave>

La clausula **HASH** se usa cuando la clave de cluster es un valor entero uniformemente distribuido. En caso contrario ORACLE aplica su algoritmo de direccionamiento

SIZE mide el tamaño en bytes del espacio que van a ocupar las tuplas con del mismo valor de clave:

- Hay que tener en cuenta las tuplas de las dos tablas
- Hay que prever si van a haber colisiones
- Se debe estimar un 15% adidional.

HASH KEY estima cuantos valores distintos va a tomar la clave del cluster