BASES DE DATOS. TEMA 7.

SQL. EL LENGUAJE DE CONSULTA.

La creación y manipulación de tablas

- Tipos de datos en SQL
- Operadores y condiciones lógicas
- La sentencia CREATE TABLE
  - Estructura general
  - Uso simplificado
  - Manejo de restricciones de integridad
- La sentencia ALTER TABLE
- La sentencia DROP TABLE

Tipos de datos en SQL

- **VARCHAR2(size):**
  (varchar) Cadena de caracteres de longitud variable (max 4000, min 1). Hay que especificar el tamaño
- **INT, INTEGER o NUMERIC** Enteros con signo
- **NUMBER(p,s)** Numero con precisión p (max 38, min 1) y escala s (max 127, min -84)
  - **precisión: número de dígitos**
  - **escala: número de cifras decimales**
- **REAL, FLOAT** Coma flotante

Tipos de datos en SQL

- **LONG** Cadena de caracteres de longitud variable de hasta 2 gigabytes (oracle)
- **LONG RAW(size)** Cadena de datos binarios de longitud variable hasta 2 gigabytes. Hay que especificar el tamaño. (oracle)
- **DATE, TIME o TIMESTAMP** Tipo fecha
- **RAW(size)** Cadena de datos binarios de longitud variable hasta 2000 bytes. Hay que especificar el tamaño

Operadores en SQL

Resumen de operadores

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

Comparadores Especiales

- Comparador **IS NULL**

  Detecta valores nulos (MAY BE SELECT)

  - (A=10), **A is null** » false, **A is not null** » true
  - A = (<>) **Null** » desconocido

- Comparador BETWEEN

  Detecta valores entre dos constantes

  - **between x and y** » >= x and <=y

- Comparador **IN**

  Detecta pertenencia a conjunto

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

Comparadores Especiales

- Comparador **LIKE**

  Sirve para utilizar “mascaras” en cadenas de caracteres

  - “-” sustituye cualquier carácter
  - “%” sustituye cualquier cadena

  Ejemplos:

  - **x LIKE ‘-A--’** » true si x=‘1A23’
    » false si x=‘1A234’
  - **x LIKE ‘%A%’** » true si x=‘1AX”%’
    » true si X=‘ABLA’

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 gris corresponde cuestiones avanzadas

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

La sentencia Create Table

- datos_columna:

  nombre tipo_de dato [DEFAULT expresion]

  [restriccion_de _columna]

  - El tipo de dato se da entre los permitidos
  - Se pueden dar valores por defecto distintos del nulo
  - Se pueden restringir las columnas individualmente

    (Análisis posterior)
  - El nombre de columna es el del atributo

La sentencia Create Table

- Restricciones asociadas a tablas:

[CONSTRAINT nombre]

[{UNIQUE | PRIMARY KEY} (columna[,columna]...)]

[CONSTRAINT nombre]

[FOREING KEY (columna[,columna]...) REFERENCES [usuario.]nombre_tabla [(columna[,columna]...)]

[CONSTRAINT nombre]

[CHECK (condicion_con_varios_campos)]

La sentencia Create Table

- Restricciones asociadas a tablas:
  - Las condiciones se almacenan en el catálogo, para reconocerlas fácilmente es bueno darles nombre
  - **UNIQUE** : no se repiten valores en tuplas distintas
  - **PRIMARY KEY**: las columnas implicadas forman la llave primaria (UNIQUE +NOT NULL)
  - **FOREING KEY**: las columnas implicadas forman llave exterior a la llave primaria de la tabla **REFERENCES**.
    - Si se especifican campos imponemos una condición de inclusión de dominio
  - **CHECK** permite condiciones lógicas entre campos

La sentencia Create Table

- 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.

Ejemplo 2: Create Table

```
CREATE TABLE asigna (
        asi# VARCHAR(4) PRIMARY KEY ,
        nombreas VARCHAR(30) NOT NULL,
        curriculum VARCHAR(20) NOT NULL ,
        credt NUMBER(4,1) NOT NULL ,
        credpr NUMBER(4,1) NOT NULL,
        caracter CHAR(2) CHECK (caracter IN ('tr','ob','op','lc')),
        temp CHAR(2) CHECK (temp IN ('cu','an')),
        CHECK ((temp='cu' AND credt+credpr BETWEEN 4.5 AND
    9) OR (temp='an' AND credt+credpr BETWEEN 6 AND          12)))
```

Crea una tabla de asignaturas con restricciones asociadas a columnas y a tabla

Ejemplo 3: Create Table

```
CREATE TABLE matricula(codas VARCHAR(4) REFERENCES
    asigna,
        codal VARCHAR(8) REFERENCES alumnos,
        curso VARCHAR(9) NOT NULL,
        calificacion CHAR(2) (CHECK calificacion IN
('np','su','ap','no','sb','mh')),
        PRIMARY KEY (codas,codal,curso))
```

Crea una tabla de asignaturas con restricciones asociadas a columnas y a tabla . Establece llaves exteriores

La sentencia Alter Table

Formato general

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]

La sentencia Alter Table

Ejemplos:

```
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))
```

- 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 DROP TABLE

Formato general

DROP TABLE [usuario.]tabla

Borra la tabla y su contenido. Borra índices asociados a ella. Deja inválidos vista y sinónimos

Solo puede borrar una tabla su propietario o el DBA

La consulta a tablas

- **Introducción a la sentencia SELECT. Consultas simples de selección y proyección.**
- **Consulta con producto cartesiano: reunión**
- **Consulta con unión, diferencia e intersección.**
- **La sentencia SELECT anidada. Operadores booleanos especiales. Alias**
- **La división en SQL**
- **El uso de funciones de agregación. La cláusula GROUP BY**

Introducción al SELECT

Forma general

```
SELECT [ALL|DISTINCT] { * | table.* | expre  [c_alias]}
[, {table.*| expre [c_alias]}]...
FROM [usuario].tabla [t_alias][,[usuario].tabla [t_alias]]...
WHERE condicion
[GROUP BY expre, [expre]... [HAVING condicion] ]
          {UNION|INTERSEC|MINUS} SELECT …
[ORDER BY {expre|posicion}  [ASC|DESC]
   [,{ expre|posicion}  [ASC|DESC]]
]
```

Introducción al SELECT

[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.

* Significa obtener todos los campos de la(s) tabla(s).

expre es una expresion con campos 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.

Introducción al SELECT

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 realizar el producto de una tabla consigo misma o referenciar distintas tuplas de una misma tabla.

WHERE condicion implica **seleccionar** las tuplas de la tabla resultante que verifican la condición. La condición puede ser tan compleja como se quiera lo que conducirá a “consultas anidadas”

Ejemplos simples

- Selecciona todos los elementos de la tabla alumnos y todos sus atributos:

  SELECT * FROM alumnos ORDER BY ape1,ape2,nombre

- Selecciona el nombre y los apellidos de los alumnos menores de 25 años:

  SELECT nombre,ape1,ape2 FROM alumnos WHERE edad <=25 ORDER BY edad desc, ape1,ape2,nombre;

- Selecciona el nombre y los apellidos de aquellos alumnos entre 20 y 30 años que son de Andalucia Oriental:

  SELECT dni,nombre,ape1,ape2 FROM alumnos WHERE (edad BETWEEN 20 AND 30) AND provincia IN ('Jaen','Granada','Almeria')) ORDER BY ape1,ape2,nombre;

Ejemplos simples

- Selecciona la lista de todos los curriculums existentes:

```
SELECT DISTINCT curriculum FROM asigna ORDER BY curriculum;
```

- 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.codal and
  matricula.codas=‘bd1s’) ORDER BY ape1,ape2,nombre;
```

- 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
  nombre=‘Jose’ and ape1=‘Lopez’ AND ap2=‘Perez’ AND dni=codal
  AND asi#=codas) ORDER BY nombreas;
```

Los operadores conjuntistas

Forma general

(select ……)

[UNION, INTERSECT, MINUS

(select….)

Ejemplos:

```
SELECT dni FROM alumnos
    MINUS
  SELECT alumnos.dni FROM alumnos, alumnos al
   WHERE (al. edad < alumnos.edad)
```

Selecciona los alumnos más jóvenes.

```
SELECT asi# FROM asigna WHERE credt+credpr >6
        INTERSECT
   SELECT codas FROM matricula WHERE curso='1998-1999'
```

Selecciona aquellas asignaturas de más de seis créditos vigentes en el curso 1998-1999.

Operad. booleanos adicionales

Forma general: **[*expresion*] [not] *operador* (*conjunto*)**

- La expresion puede ser una sucesion de expresiones o nombres de columnas.
- Los operadores pueden ser:
  - **IN** ya conocido
  - **{= | ¡= | < | > | <= | >=} ANY** compara con cualquier elemento del conjunto citado y es cierta si se cumple la condicion.
  - **{= | ¡= | < | > | <= | >=} ALL** compara con todos los elementos del conjunto citado y es cierta si se cumple la condicion.

Ejemplo: **select codal from matricula where codas='bd1s' and curso='1998-1999' and calificacion >= all (select calificacion from matricula);**

Selecciona aquellos alumnos que han obtenido la máxima calificación en bd1s en el curso 1998-1999.

Operad. booleanos adicionales

EXISTS detecta si el conjunto está no vacío, no hay expresión asociada.

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

Ejemplos:

SELECT codal,ape1,ape2,nombre FROM matricula,alumnos WHERE codas IN (SELECT asig# FROM asigna WHERE caracter=‘op’) AND dni=codal ORDER BY ape2,nombre,ape1;

Selecciona los alumnos matriculados de alguna asignatura optativa.

Operad. booleanos adicionales

Ejemplos

SELECT asi#,nombreas FROM asigna WHERE curso>= ALL(SELECT curso FROM asigna)

Selecciona asignaturas de mayor curso.

SELECT distinct codal FROM matricula WHERE codas IN (select asi# WHERE curso <=ALL(SELECT curso FROM asigna))

Selecciona alumnos matriculados de asignaturas del curso más inferior.

Operad. booleanos adicionales

Ejemplos

SELECT dni,ape1,ape2,nombre FROM alumnos
WHERE EXISTS (SELECT * FROM matricula WHERE dni=codal AND codas='dbs1');

Selecciona los alumnos matriculados de bds1, es una forma adicional de conectar tablas propia del cálculo relacional.

SELECT asi#,nombreas FROM asigna WHERE NOT EXISTS (SELECT * FROM matricula WHERE asi#=codas);

Selecciona asignaturas de las que no esta matriculado ningun alumno, **notese la equivalencia con la diferencia.**

La division en SQL

La división utilizando 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.

Ejemplos:

Si un alumno está matriculado de todas las asignaturas optativas es lo mismo que si no existe ninguna asignatura optativa de la que no esté matriculado.

```
SELECT ape1,ape2,nombre FROM alumnos WHERE
NOT EXISTS (SELECT asi# FROM asigna WHERE carácter='op'
AND NOT EXISTS(SELECT * FROM matricula WHERE codal=dni
  AND codas=asi#))
```

La division en SQL

Ejemplos:

Si en una asignatura están matriculados todos los alumnos de Almería entonces el conjunto de alumnos de Almería que no están matriculados de esta asignatura está vacío.

SELECT asi#,nombreas FROM asigna WHERE

NOT EXISTS(SELECT dni FROM alumnos WHERE

provincia='Almeria' AND NOT EXISTS (SELECT * FROM matricula WHERE codas=asi# AND codal=dni));

La division en SQL

Ejemplos:

Encontrar los alumnos que han aprobado todas las asignaturas de primero de Ingenieria Superior.

No existe ninguna asignatura de primero de Ingeniería Superior tal que no exista una matricula del alumno donde esté dicha asignatura y la calificación sea aprobado o mayor

```
SELECT ape1,ape2,nombre FROM alumnos WHERE
NOT EXISTS (SELECT asi# FROM asigna WHERE curso=1
  AND curriculum ='Informatica Superior' AND NOT EXISTS
  (SELECT * FROM matricula WHERE dni=codal AND
  codas=asi# AND calificacion IN('ap','no',sb','mh')));
```

La division en SQL

Ejemplos:

Encontrar los alumnos becarios matriculados de todas las asignaturas de más de seis créditos

SELECT ape1,ape2,nombre FROM alumnos WHERE beca='si' AND NOT EXISTS (SELECT asi# FROM ASIGNA WHERE credt+credpr>6 AND NOT EXISTS (select * FROM matricula WHERE codal=dni AND asi#=codas))

Encontrar los profesores que dan clase a todos los grupos de la asignatura 'BDI'.

SELECT nrp,nom_prof FROM profesores WHERE NOT EXISTS (SELECT cod_grup g1 FROM grupos WHERE cod_asig='BDI' AND NOT EXISTS (SELECT cod_grup FROM grupos g2 WHERE g1.cod_grp=g2.cod_grup AND nrp=g2.nrp AND cod_asig='BDI'))

La division en SQL

La división utilizando Algebra Relacional

Idea Básica:

Sean D=R□S y r,s,d instancias de D R y S

. a□D ; s□ d(a) ={b□S /(a,b)□R}

para detectar la inclusion:

s□ d(a) □ s-d(a)=□□not exists(s-d(a))

Ejemplos:

Alumnos matriculados de todas las asignaturas optativas

SELECT ape1,ape2,nombre FROM alumnos WHERE NOT EXISTS ((SELECT asi# FROM ASIGNA WHERE caracter='op') MINUS (SELECT codas FROM MATRICULA WHERE codal=dni))

La division en SQL

Ejemplos:

- *Asignaturas en que estan matriculados todos los alumnos de Almeria*

SELECT asi#,nombreas FROM asigna WHERE NOT EXISTS ((SELECT dni FROM alumnos WHERE provincia='Almeria') MINUS (SELECT codal FROM matricula WHERE asi#=codas))

- *Alumnos que han aprobado todas las asignaturas de primero de Ingenieria Superior*

SELECT ape1,ape2,nombre FROM alumnos WHERE NOT EXISTS ((SELECT asi# FROM asigna WHERE curso=1 AND curriculum ='Informatica Superior') MINUS

(SELECT codas FROM matricula WHERE dni=codal

AND calificacion IN ('ap','no','sb','mh')))

La division en SQL

- *Alumnos becarios matriculados de todas las asignaturas de más de seis creditos*

```
SELECT ape1,ape2,nombre FROM alumnos
 WHERE beca='si' AND NOT EXISTS((SELECT asi#
   FROM asigna WHERE credt+credpr>6)
MINUS
(SELECT codas FROM matricula WHERE
   codal=dni))
```

Funciones de Agregación

Idea básica:

Utilizar funciones cuyo resultado sea un "resumen" de los datos de una columna de una tabla.

- Forma general: funcion(*expresion)*

Funciones existentes:

- AVG(.) calcula la media de la expresión dada,
- STDDEV(.) calcula la desviación típica (raiz de la varianza),
- VARIANCE(.) calcula la varianza (la media de las diferencias cuadráticas de *n* puntuaciones con respecto a su media aritmética). Ignoran valores nulos:

SELECT AVG(media), STDDEV(media) FROM alumnos WHERE sexo='v'

- MIN(.) calcula el mínimo de la expresion dada,
- MAX(.) calcula el máximo

SELECT MIN (credt+credpr),MAX(credt+credpr) FROM asigna

Funciones de Agregación

Funciones existentes:

- COUNT(*expresion*) calcula el número de filas donde la expresión es no nula.

SELECT count(calificacion) FROM matricula

- COUNT(*) calcula el número total de filas de una tabla. No tiene en cuenta que las filas sean iguales. Si queremos que lo tenga en cuenta COUNT(DISTINCT calificacion)

SELECT count(*) FROM matricula

Otros usos: Se pueden combinar las funciones de agregación con los operadores ALL y ANY.

Ejemplos

SELECT ape1,ape2,nombre,edad FROM alumnos WHERE edad=ANY(SELECT min(edad) FROM alumnos);

Calcula los alumnos de edad minima

Funciones de Agregación

Otros usos:

SELECT ape1,ape2,nombre,media FROM alumnos WHERE media<=ANY(SELECT AVG(media) FROM alumnos);

Calcula los alumnos con nota menor o igual a la media

SELECT asi#,nombreas,credt+credpr FROM asigna WHERE (credt+credpr)=ANY(SELECT max(credt+credpr) FROM asigna);

Calcula las asignaturas con número máximo de creditos

SELECT ape1,ape2,nombre FROM alumnos WHERE 15<=ANY(SELECT count(*) FROM matricula WHERE dni=codal);

Calcula los alumnos matriculados de más de 15 asignaturas

SELECT asi#,nombreas FROM asigna WHERE 15<=ANY(SELECT count(*) FROM matricula WHERE asi#=codas);

Calcula las asignaturas con más de 15 alumnos

La Clausula 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 dichos atributos.

Ejemplos de dichas tablas:

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

carácter	curso	count(asi#)
tr	1	----
tr	2	----
tr	3	----
.	.	.
.	.	.
ob	1	----
ob	2	----
.	.	.
.	.	.

- Nótese que en las columnas sólo aparecen los atributos que agrupan y funciones de agregación

La Clausula 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 Clausula GROUP BY

Ejemplos:

SELECT sexo, AVG(edad) FROM alumnos GROUP BY sexo ORDER BY sexo

SELECT caracter,curso,COUNT(asi#) FROM asigna GROUP BY carácter, curso ORDER BY caracter,curso

Obtienen las dos tablas que aparecen en los ejemplos iniciales

SELECT curso,codas,COUNT(*) FROM matricula GROUP BY curso,codas ORDER BY cursocodas

Obtiene el numero de alumnos matriculados en cada curso en cada asignatura

SELECT codal,COUNT(*) FROM matricula WHERE calificacion IN ('ap','no','sb','mh') GROUP BY codal ORDER BY codal

Obtiene el numero de asignaturas que tiene aprobadas cada alumnos

La Clausula GROUP BY

Ejemplos:

- **SELECT sexo, AVG(edad) FROM alumnos GROUP BY sexo ORDER BY sexo HAVING sexo ='v';**

Obtiene la edad media de los alumnos varones

- **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 siempre que haya más de 10 alumnos.

Consideraciones adicionales:

El calculo y mantenimiento de resúmenes es una de las actividades a las que se han dedicado más atención en los últimos tiempos y es la base del concepto de "OLAP" (procesamiento analítico en línea) y de "data warehouse" (almacenes de datos).

Algunas sentencias adicionales: Insert

Sintaxis:

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

- Permite la inserción dando valores sólo a algunas columnas. Las columnas no mencionadas se rellenan por defecto
- Permite la inserción tupla a tupla (opción “values)
- Permite la inserción “global” (opción “*consulta*”) insertando de golpe en la tabla el conjunto de tuplas resultado de la consulta
- En cualquier caso las columnas donde se insertan los datos han de ser compatibles con lo que se inserta.

Ejemplo:

Insert into alumnos_buenos (ape1,ape1,nombre,nota)

select ape1,ape2,nombre,media from alumnos where

media >=7.5;

Algunas sentencias adicionales: Delete

Sintaxis:

Delete tabla [where condicion]

- Si se omite la condicion 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 asi#=codas);
```

Elimina aquellas asignaturas que no tienen alumnos matriculados.

Algunas sentencias adicionales
Update

Sintaxis:

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 asig set (asig.credt,asig.credpr)=(select max(credt),max(credpr) from asigna where caracter='op') where asig.carácter='op' and asig.curso='5'

- Actualiza los créditos teóricos y prácticos de todas las asignaturas optativas de 5 curso al valor máximo de dichos campos para todas las asignaturas optativas

Algunas sentencias adicionales: 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 índice
- 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

Algunas sentencias adicionales: Create View

Sintaxis:

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

- Una vista puede aparecer en cualquier sentencia “select”.
- 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. Limita las inserciones, modificaciones y borrados de forma que no se pueda insertar, modificar o borrar algo que no pueda recuperar la propia vista.
- Al borrar una tabla quedan inutilizados todas las vistas que se relacionan a ella.
- Una vista puede ser objetivo en una sentencia de **actualización**. Afecta a la tabla base sobre la que está construida la vista. Solo se puede actualizar las vistas que estén definidas sobre una sola tabla base.