juanmanuel_ug
www.wuolah.com/student/juanmanuel_ug

ResumenSQL.pdf

Resumen Sesiones Prácticas SQL

2° Fundamentos de Bases de Datos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

FBD SQL

Sesión 1

Creación de tablas. CREATE TABLE nombretable(*aquí dentro los datos de la tabla*)

Ej:

```
CREATE TABLE alumnos (
DNI CHAR(8),
nom_alum VARCHAR2(35),
edad NUMBER,
direccion VARCHAR2(30),
provincia VARCHAR2(20),
beca VARCHAR2(2)
);
```

Tipos de datos.

INT o INTEGER o NUMERIC	Enteros con signo (su rango depende de la implementación del sistema).
REAL o FLOAT	Datos numéricos en coma flotante.
CHAR(n)	Cadena de longitud fija n.
VARCHAR(n)	Cadena de longitud variable de hasta n caracteres.
VARCHAR2(n)	De mínimo 1 carácter y máximo 4000. Otra implementación de cadena más eficiente (específico de Oracle).
NUMBER(p,s)	Número con precisión p y escala s, donde precisión indica el número de dígitos, y escala el número de cifras decimales.
LONG	Cadena de caracteres de longitud variable de hasta 2 gigabytes (específico de Oracle).
LONG RAW(size)	Cadena de datos binarios de longitud variable de hasta 2 gigabytes (específico de Oracle).
DATE o TIME o TIMESTAMP	Fecha.

Clave Primaria.

Ej.

```
CREATE TABLE alumnos (
DNI CHAR(8) PRIMARY KEY,
nom_alum VARCHAR2(35),
edad NUMBER,
direccion VARCHAR2(30),
provincia VARCHAR2(20),
beca VARCHAR2(2)
);
```

```
CREATE TABLE alumnos (
DNI CHAR(8),
nom_alum VARCHAR2(35),
edad NUMBER,
direccion VARCHAR2(30),
provincia VARCHAR2(20),
beca VARCHAR2(2),
PRIMARY KEY(DNI)
);
```

Más de 1 clave primaria.

```
CREATE TABLE matriculas(
cod_asig CHAR(4),
cod_grup CHAR(4),
tipo CHAR(1),
DNI CHAR(8),
convocatoria NUMBER,
calificacion NUMBER,
PRIMARY KEY (cod_asig, cod_grup, tipo, DNI, convocatoria)
);
```

Unicidad de los valores.

UNIQUE -> Restricción para que el valor de un atributo sea único

Control de valores nulos.

NOT NULL -> Restricción para que no se pueda tener valor nulo en un atributo

Rango de valores permitidos.

CHECK(condición) -> Restricción para indicar el rango de valores que se pueden introducir en la tupla.

Dentro del CHECK podemos usar IN -> para controlar la pertenencia o no de un elemento a un conjunto especificado mediante una lista de valores separados por comas.

Ej.

```
CREATE TABLE asignaturas (
cod_asig CHAR(4) PRIMARY KEY,
nom_asig VARCHAR2(30) UNIQUE,
creditos NUMBER(4,1),
caracter CHAR(2) CHECK(caracter IN ('tr','ob','op','lc')),
curso NUMBER CHECK (curso BETWEEN 1 AND 5)
);
```

Operadores Lógicas.

AND, OR Y NOT.

Claves Externas.

Dos maneras de referenciar una clave.

REFERENCES id_tabla(id_columna)

FOREIGN KEY (id_columna,…) REFERENCES id_tabla(id_columna,…)

Ej.

```
CREATE TABLE grupos (
cod_grup CHAR(4),
cod_asig CHAR(4),
tipo CHAR(1) CHECK (Tipo IN ('T','P')),
NRP CHAR(4),

max_al INT CHECK (max_al BETWEEN 10 AND 150),

PRIMARY KEY (cod_asig, cod_grup, tipo),

FOREIGN KEY cod_asig REFERENCES asignaturas(cod_asig),

FOREIGN KEY NRP REFERENCES profesores(NRP)

);
```

Borrado/actualización en cascada

```
CREATE TABLE grupos (
cod_grup CHAR(4),
cod_asig CHAR(4),
tipo CHAR(1) CHECK (Tipo IN ('T','P')),
NRP CHAR(4),
max_al INT CHECK (max_al BETWEEN 10 AND 150),
PRIMARY KEY (cod_asig, cod_grup, tipo),
FOREIGN KEY cod_asig REFERENCES asignaturas(cod_asig)
ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY NRP REFERENCES profesores(NRP)

ON DELETE SET NULL

ON UPDATE CASCADE

);
```

Valores por defecto.

DEFAULT valor-> El sistema poner por defecto el valor que ponemos en valor

Ej.

```
CREATE TABLE matriculas(

cod_asig CHAR(4),

cod_grup CHAR(4),

tipo CHAR(1),

DNI CHAR(8) REFERENCES alumnos(DNI),

convocatoria NUMBER DEFAULT 1,
```

```
calificacion NUMBERrestric-rangocalif
      CHECK ((calificacion>=0) AND (calificacion<=10)),
PRIMARY KEY (cod_asig, cod_grup, tipo, DNI, convocatoria),
FOREIGN KEY (cod_asig,cod_grup,tipo)
      REFERENCES grupos(cod_asig,cod_grup,tipo)
);
```

Eliminación de tabla.

```
DROP TABLE nombre_tabla; -> Elimina la tabla indicada
DROP TABLE alumanos CASCADE CONSTRAINTS; -> Elimina la tabla indicada aunque tenga claves externas apuntando a ella
```

Ampliar tabla.

```
ALTER TABLE nombre_tabla ADD(atributo TIPO restricciones)
```

Ej.

```
ALTER TABLE ventas ADD (fecha date not null default sysdate,
total_pedido number(10,2) not null,
tipo_pago varchar2(20) check (tipo_pago IN ('CONTADO','TRANSFERENCIA','TARJETA CREDITO')));
```

Ejecutar para saber el esquema.

```
Describe nombre_tabla;
```

Insertar filas.

```
INSERT INTO nombre_tabla VALUES(valores);
COMMIT; -> Para guardar los cambios
```

Comprobar el contenido de tablas.

```
SELECT * from nombre_tabla;
```

Sesión 2.

Consultas.

SELECT * FROM nombre_tabla WHERE condición;

WHERE -> impone una condición booleana que deben cumplir las tuplas para ser recuperadas.

Ej. Encontrar aquellos profesores cuya categor ía no es 'AS' y cuya áreade conocimiento es 'COMPUT' o 'ELECTR'.

```
SELECT * FROM profesores WHERE categoria<> 'AS' AND (area='COMPUT' OR area='ELECTR');
```

Comparación de cadenas de caracteres.

LIKE cadena_caracteres -> Sirve para comparar las cadenas de caracteres con la cadena_caracteres

% -> Representa cualquier número de caracteres, 0 o más

_ -> Representa un único carácter

Ej. Mostrar los datos de aquellos alumnos cuyo nombre empieza por H.

SELECT * FROM alumnos WHERE nom_alum LIKE '%H%'; -> Mostrar los datos de aquellos alumnos cuyo nombre contenga una H.

Eliminación de valores duplicados.

SELECT DISTINCT id_columna FROM nombre_tabla;

DISTINCT -> Al usar esta expresión hacemos que al mostramos los valores, si existe alguno repetido, solo se muestre una vez.

Ordenación de los resultados.

ORDER BY id_columna [ASC|DESC] -> Al usar esta expresión permite que ordenemos la columna según unos criterios

Ej. Mostrar la lista de alumnos ordenados por su provincia de procedencia de forma descendente y, dentro de cada provincia, ordenados alfabéticamente por su nombre.

SELECT nom_alum FROM alumnos WHERE beca='SI' ORDER BY nom_alum ASC;

Actualización y borrado de filas.

UPDATE nombre_tabla SET atributo=nuevo_valor WHERE (condición);

Ej. aumentar en 10 la cantidad de piezas tipo 'P4' en todos los pedidos en los que aparezca.

```
UPDATE VENTAS SET CANTIDAD=CANTIDAD+10 WHERE CODPIE='P4';
```

DELETE nombre_tabla WHERE (condición);

Ej.

DELETE VENTAS WHERE COPRO='P5'

Sesión 3.

Consulta sobre varias tablas (Reunión o Producto Cartesiano).

SELECT id_columna,… FROM id_tabla,… WHERE condicion;

Ej. Mostrar el nombre y el NRP de cada profesor junto con el nombredel departamento al que pertenece.

```
SELECT nom_prof, NRP, nom_dep FROM profesores, departamentos WHERE
profesores.cod_dep=departamentos.cod_dep;
```

Sesión 4.

Operadores de conjunto.

UNION, INTERSECT y MINUS -> Para poder utilizarlo debemos usarlo entre dos consultas cuyos resultados sean compatibles desde el punto de vista estructural (mismo n° columnas y concordancia o compatibilidad de tipos).

Ej. Encontrar los alumnos que están matriculados de, al menos, una asignatura de primero una asignatura de segundo.

```
SELECT DNI FROM matriculas, asignaturas WHERE
matriculas.cod_asig=asignaturas.cod_asig AND curso=1
UNION
SELECT DNIFROM matriculas, asignaturas WHERE
matriculas.cod_asig=asignaturas.cod_asigAND curso=2;
```

Subconsultas.

EXISTS -> Nos indica si una operación devuelve al menos una tupla, sin importar el contenido ni el número de campos de esa tupla.

Ej. Mostrar los nombres de las asignaturas que tienen algun alumno matriculado.

```
SELECT nom_asig FROM asignaturas WHERE EXISTS(SELECT * FROM matriculas WHERE
matriculas.cod_asig=asignaturas.cod_asig);
```

Sesión 5.

Comparación de un valor con los elementos de un conjunto.

ANY -> Controla que la comparación se cumpla con al menos un elemento del conjunto.

Ej. Mostrar el DNI de los alumnos que est án matriculados en algunaasignatura de tercero.

```
SELECT DNI FROM matriculas WHERE cod_asig = ANY (SELECT cod_asig FROM
asignaturas WHERE curso=3);
```

ALL -> Impone que la comparación se cumpla con todos los elementos del conjunto.

Ej. Mostrar la lista de alumnos con la nota m ás alta en BD2

```
SELECT DNI FROM matriculas WHERE cod_asig =’BD2’ AND calificion >= ALL
(SELECT calificacion FROM matriculas WHERE cod_asig=’BD2’);
```

Funciones de agregación.

Sum(): Suma de una distribución de valores.

Min(): Valor mínimo de una distribución de valores.

Max(): Valor máximo de una distribución de valores.

Avg(): Media de una distribución de valores.

Stddev(): Desviación de una distribución de valores.

Count(): Cardinal de una distribución de valores.

Ej. Hallar la media de cr éditos de las asignaturas de primero.

```
SELECT avg(creditos) FROM asignaturas WHERE curso=1;
```

Sesión 7.

Grupos de tuplas.

GROUP BY id_columna -> En esta clausula se indica el atributo o conjunto de atributos por cuyos valores que-remos agrupar las tuplas, para poder aplicar funciones de agregacion sobre cada uno de los distintos grupos.

Ej. Mostrar, para cada alumno, el n úmero de asignaturas de las queest á matriculado y la media de las calificaciones.

```
SELECT DNI, avg(calificación) califica_avg, count(*) num_asig FROM matriculas
GROUP BY DNI;
```

HAVING condición -> Se devuelve como resultado de la consulta que identifique al grupo

```
SELECT cod_asig FROM matriculas WHERE calificacion>=5 GROUP BY cod_asig HAVING
count(*)>= ALL (SELECT count(*) FROM matriculas WHERE calificacion >=5 GROUP
BY cod_asig);
```

Sesión 8.

División.

En sql no se dispone de un método directo que aplique la división.

Ej. En nuestra consulta de ejemplo, un alumno formar á parte del resultado de la division si el conjunto de asignaturas obligatorias de segundo esta contenido en el conjunto de asignaturas de las que esta matriculado. Es decir, el conjunto representado por el divisor debe estar contenido en el conjunto de valores con los que esta relacionada en el dividendo la tupla candidata a formar parte del resultado.

```
SELECT cod_asig FROM asignaturas WHERE character='ob' AND curso=2;
```