Universidad de Granada

decsai.ugr.es

Fundamentos de Bases de Datos

Grado en Ingeniería Informática

Introduccion al SQL: Creacion y manipulacion de tablas

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

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

- **INT, INTEGER ó NUMERIC:**
  - Enteros con signo (ep rango depende del sistema)
- **FLOAT**
  - Datos numèros en como flotante

Tipos de datos en SQL

- **VARCHAR2(size): (varchar)**
  - Cadena de caracteres de longitud variable (max 4000, min 1). Hay que especificar el tamaño
- **CHAR(size)**
- **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**

Tipos de datos en SQL

- **LONG**
  - Cadena de caracteres de longitud variable de hasta 2 gigabytes
- **LONG RAW(size)**
  - Cadena de datos binarios de longitud variable hasta 2 gigabytes. Hay que especificar el tamaño
- **DATE oTIME o TIMESTAMP**
  - Tipo fecha se estudiara con detalle mas adelante
- **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--’ $\Rightarrow$ true si x=‘1A23’
      $\Rightarrow$false si x=‘1A234’
    - x LIKE ‘%A%’ $\Rightarrow$ true si x=‘1AX”%’
      $\Rightarrow$true si X=‘ABLA’

El tipo fecha

- Internamente es un entero expresado en Juliano. Realmente almacena los segundos
- Tiene una constante SYSDATE que es la fecha del dia. Permite sumar y restar fechas.
- SYSDATE-1 es ayer

El tipo fecha. Funciones

- Para introducir:

  TO_DATE(cadena,formato,null)

- Para visualizar

  TO_CHAR(dato-tipo-fecha,formato)

  Los posibles formatos están en el cuaderno

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 no sombreada 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.

Sobre la claúsula REFERENCES

- Esté asociada a columna:

  REFERENCES [usuario.]nombre_tabla [(columna)]

- O a tabla:

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

Sobre la claúsula REFERENCES

- Se le puede añadir:

  [ON DELETE CASCADE | NULLIFIES]

  [ON UPDATE CASCADE | NULLIFIES]

Permite decidir en su caso, si se hace nula una llave externa cuando se actualiza o borra la llave a la que referencia

La sentencia Create Table

- Ejemplo 1:

```
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) constraint al3 check (beca in ('si','no')),
        sexo char(1) constraint al4 check  (sexo in ('v','m')))
```

Crea una tabla de alumnos con restricciones asociadas a columnas, describe llave primaria, condiciones de no nulo y restricciones de dominio.

La sentencia Create Table

- Ejemplo 2:

```
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

La sentencia Create Table

- Ejemplo 3:

```
create table matricula(codasi# varchar(4) references asigna,
DNI varchar(8) references alumnos,
curso_academico varchar(9) not null,
calificacion char(2) (check calificacion in
    ('np','su','ap','no','sb','mh')),
notas number(5,2) (check notas between o and 10
primary key (codasi#,dni,curso_academico))
```

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'),
        nota number(5,2))
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**

Algunas sentencias DML
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 DML
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 condicion 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 creditos teoricos y practicos de todas las asignaturas optativas de 5 curso al valor maximo de dichos campos para todas las asignaturas optativas

Algunas sentencias DML
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.