LENGUAJE SQL

AUTOR: ORLANDO ESPINOZA BARBOZA

NOVIEMBRE: 2020

San Marcos

Introducción

Principios de la década de 1990. El lenguaje SQL se diseñó fundamentalmente para las aplicaciones de ayuda a la toma de decisiones, que son intensivas en consultas, mientras que el objetivo principal de las bases de datos. Existen diferencias entre la clasificacion de las estructuras de este lengua, estas son: lenguaje de definición de datos (DDL) que se usan para especificar la estructura de los datos y lenguaje de manipulación de datos (DML) que permite a los usuarios acceder o manipular los datos

Fundamentos y administración de BD

Conforme fue creciendo el uso de los sistemas de informacion evolucionando la tecnologias, se han usado desde archivos ficheros hasta los sistemas de almacenamiento de bases de datos. Estos datos son almacenados en estructuras complejas e integrales. Se tienen sistemas de gestión de bases de datos especializados para la administración y almacenamiento de estas estructuras.

Es importante, para efectos de practices tener instalado un Sistema de gestion de bases de datos, por su practicidad y facilidad de obtenerlo, Código abierto y facil instalacion se recomienda el uso de PostgreSQL. Que se baja de la siguiente direccion WEB: https://www.postgresql.org/download/ solo seguir los pasos de instalacion y se consigue mucha informacion para aprender a usarlo, incluso manuals en línea gratuitos.

Tabla de contenido

LENGUAJE SQL

El lenguaje SQL tiene varios componentes:

• Lenguaje de definición de datos (LDD). El LDD de SQL proporciona órdenes para la definición de esquemas de relación, borrado de relaciones, creación de índices y modificación de esquemas de relación.

• Lenguaje interactivo de manipulación de datos (LMD). El LMD de SQL incluye un lenguaje de consultas, basado tanto en el álgebra relacional como en el cálculo relacional de tuplas. Incluye también órdenes para insertar, borrar y modificar tuplas de la base de datos.

• Definición de vistas. El LDD de SQL incluye órdenes para la definición de vistas.

• Control de transacciones. SQL incluye órdenes para la especificación del comienzo y final de transacciones.

• SQL incorporado y SQL dinámico. SQL dinámico e incorporado define cómo se pueden incorporar las instrucciones SQL en lenguajes de programación de propósito general, tales como C, C++, Java, PL/I, Cobol, Pascal y Fortran.

• Integridad. El LDD de SQL incluye órdenes para la especificación de las restricciones de integridad que deben satisfacer los datos almacenados en la base de datos. Las actualizaciones que violen las restricciones de integridad se rechazan.

• Autorización. El LDD de SQL incluye órdenes para especificar derechos de acceso para las relaciones y vistas.

Estructura Básica

Una base de datos relacional consiste en un conjunto de relaciones, a cada una de las cuales se le asigna un nombre único. SQL permite el uso de valores nulos para indicar que el valor o bien es desconocido, o no existe. Se fijan criterios que permiten al usuario especificar a qué atributos no se puede asignar valor nulo. La estructura básica de una expresión SQL consiste en tres cláusulas: select, from y where.

• La cláusula select corresponde a la operación proyección del álgebra relacional. Se usa para listar los atributos deseados del resultado de una consulta.

• La cláusula from corresponde a la operación producto cartesiano del álgebra relacional. Lista las relaciones que deben ser analizadas en la evaluación de la expresión.

• La cláusula where corresponde al predicado selección del álgebra relacional. Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula from.

Un hecho histórico desafortunado es que el término select tiene un significado diferente en SQL que en el álgebra relacional. A continuación, se resaltan las diferentes interpretaciones, a fin de minimizar la posible confusión.

Una consulta típica en SQL tiene la forma

select A1, A2,…, An (listado de campos)

from t1, t2,…, tn (listado de tablas a consultar)

where P (predicados o condiciones a cumplir)

Cada Ai representa un atributo, y cada ri una relación. P es un predicado. La consulta es equivalente a la expresión del álgebra relacional Si se omite la cláusula where, el predicado P es cierto. Sin embargo, con diferencia a la expresión del álgebra relacional, el resultado de la consulta SQL puede contener varias copias de algunas tuplas. SQL forma el producto cartesiano de las relaciones incluidas en la cláusula from, lleva a cabo la selección del álgebra relacional usando el predicado de la cláusula where y entonces proyecta el resultado sobre los atributos de la cláusula select. En la práctica, SQL puede convertir la expresión en una forma equivalente que puede ser procesada más eficientemente.

Para el SELECT se debe tomar en cuenta los renombramientos de campos o tablas con la palabra “as”, las operaciones sobre cadenas con el operador like, el orden en la presentación de las tuplas y las tuplas duplicadas, la operación “unión”, la operación “except”, la operación “interseccion”, los valores nulos, las consultas anidadas, las vistas y las funciones de agregación.

Mediante una sentencia SQL de tipo DML o LDM

- Una insercion se utiliza para incluir datos en una base de datos. La expresión es la siguiente

Insert into t (A1, A2, …, An) (Ai son campos de la tabla t)

Values (v1, v2, …, vn) (los valores que se incluyen, deben llevar el tipo del orden)

- Un borrado se utiliza para eliminar tuplas enteras dependiendo de la condición que se ponga. La espresion SQL es la siguiente:

delete from t (tabla donde se ubica los datos)

where P (predicados o condiciones a cumplir)

- Una modificación o actualización: es para modificar los datos indicados en la condición

update t (tabla con los tados a modificar)

set Ai=v (campo Ai se modifica con el valor v, puede ser un listado separados por “,”)

where P (predicados o condiciones a cumplir)

En todos los casos, los valores “v” deben respetar los tipos de datos definidos para los campos A respectivos. Tambien los valores que se usan para las condiciones del where.

Una transacción es una secuencia de instrucciones de consulta y actualizaciones. La norma SQL especifica que una transacción comienza implícitamente cuando se ejecuta una instrucción SQL y una de las siguientes instrucciones SQL debe finalizar la transacción con commit o rollback:

• **Commit** hace que los cambios realizados por la transacción sean permanentes en la base de datos. Después inicia una nueva transacción automáticamente.

• **Rollback** deshace todas las actualizaciones realizadas por las instrucciones SQL de la transacción; así, el estado de la base de datos se restaura al que existía previo a la ejecución de la transacción. Es aconsejable Revisar en el libro o en la WEB cada uno de estos conceptos

Con una sentencia SQL de tipo DML, de la siguiente forma:

SELECT nombre_columnas_a_seleccionar

FROM tabla1 as t1, tabla2 as t2

[WHERE condiciones1 and condiciones2];

O bien utilizar la estructura INNER JOIN

SELECT nombre_columnas_a_seleccionar

FROM tabla1 as t1 INNER JOIN tabla2 as t2

{ON condiciones2}

[WHERE condiciones1];

Se puede usar también las indicaciones **LEFT OUTER JOIN y RIGTH OUTER JOIN**

Sentencia JOIN

La sentencia JOIN, que significa unir, permite en el lenguaje SQL combinar la información de dos o más tablas de una base de datos relacional, con base en un campo que es común entre ellas.

Existen diferentes tipos de JOIN en SQL:

INNER JOIN
Es el tipo de combinación más común, en este se realizan combinaciones internas y permite recuperar los campos cuyo valor es idéntico en todas las tablas contempladas en el JOIN, su estructura es la siguiente:
SELECT * FROM TablaA INNER JOIN
TablaB ON (Condiciones)

FULL OUTER JOIN
Se trata de combinaciones externas que permiten recuperar los registros de todas las tablas relacionadas en el JOIN o todos los registros que no son comunes a las tablas relacionadas, su estructura es la siguiente:
SELECT * FROM TablaA FULL OUTER
JOIN TablaB ON (Condiciones)
SELECT * FROM TablaA FULL OUTER
JOIN TablaB ON A.id = B.id WHERE
(Condiciones)

LEFT JOIN
Permite recuperar la totalidad de las filas de la tabla a la izquierda del JOIN, tengan o no tengan correspondencia con la tabla de la derecha, su estructura es la siguiente:

```
SELECT * FROM TablaA LEFT JOIN
TablaB ON (Condiciones)
```

RIGHT JOIN
Permite recuperar la totalidad de las filas de la tabla a la derecha del JOIN, tengan o no tengan correspondencia con la tabla de la izquierda, su estructura es la siguiente:

```
SELECT * FROM TablaA RIGHT JOIN
TablaB ON (Condiciones)
```

Mediante una instrucción SQL de tipo DDL, con el siguiente formato:

> **create table t (c1 tipo1 <null | not null>, c2 tipo2 <null | not null>,..., cn tn <null | not null> <, restricción-integridad>)**
> donde ci son los campos de la tabla t que son de tipoi.

Los tipos mas comunes son char (n), varchar (n), int, integer o smallint, numeric (p,d), real, float (n), date, time, datetime y timestamp.

Las restricciones de integridad válidas incluyen: primary key (cj1, cj2,...,cjm) que conforman la llave primaria y única, Los campos de la llave primaria deben ser no nulos y únicos. Aunque la especificación de clave primaria es opcional, es buena idea especificar una clave primaria para cada tabla.

Por ejemplo,

```
create table estudiantes
(id-estudiante char (10) not null,
nombre char (50) not null,
residencia char (50) null,
fecha-nacimiento date not null,
activo char (1) not null,'
primary key (id-estudiante))

create table cursos
(id-curso int not null,
Nombre-curso char (50) not null,
horario char (50) null,
primary key (id-curso))
```

```
create table cursos_matriculados
(id-estudiante char (10) not null,
id-curso int not null,
fecha-matricula date not null,
primary key (id-estudiante, id-curso))
```

Insercion de datos

De la siguiente forma:

```
Insert into t (A1, A2, …, An) (Ai son campos de la tabla t)
Values (v1, v2, …, vn) (los valores que se incluyen, deben llevar el tipo
del orden)
```

Por ejemplo

```
insert into estudiantes ('01-0123-0123', 'Juan Perez López', 'Puriscal',
'01/02/2000', 'S');
insert into estudiantes ('01-0124-0124', 'Ana Flores Rojas', 'San José',
'15/03/2003', 'S');
insert into estudiantes ('01-0125-0125', 'Zacarías Piedras deRio', 'Perez
Zeledón', '11/03/1998', 'N');
insert into estudiantes ('01-0123-0123', 'Juan Perez López', 'Puriscal',
'01/02/2000', 'S');
```

¿Que pasa en el ultimo insert?

```
insert into cursos (100, 'Cálculo básico', 'Diurno');
insert into cursos (101, 'Programacion 1', 'Diurno');
insert into cursos (102, 'Bases de datos 1', 'Diurno');
insert into cursos (103, 'Lógica', 'Diurno');
```

Y la relación con las dos tablas:

```
insert into cursos_matriculados ('01-0123-0123', 100, '01/12/2020');
insert into cursos_matriculados ('01-0124-0124', 100, '01/12/2020');
insert into cursos_matriculados ('01-0125-0125', 100, '01/12/2020');
insert into cursos_matriculados ('01-0123-0123', 101, '01/12/2020');
insert into cursos_matriculados ('01-0124-0124', 101, '01/12/2020');
insert into cursos_matriculados ('01-0123-0123', 102, '01/12/2020');
insert into cursos_matriculados ('01-0123-0123', 103, '01/12/2020');
insert into cursos_matriculados ('01-0124-0124', 103, '01/12/2020');
insert into cursos_matriculados ('01-0125-0125', 103, '01/12/2020');
```

Borrado de datos

Con la instrucción SQL de tipo DML

```
delete from t (tabla donde se ubica los datos)
where P (predicados o condiciones a cumplir)
```

Por ejemplo, borrar todos los cursos del estudiante '01-0125-0125'

```
delete from estudiantes where id-curso= '01-0125-0125'
```

Modificacion de datos

```
update t (tabla con los tados a modificar)
set Ai=v (campo Ai se modifica con el valor v, puede ser un listado
separados por “,”)
where P (predicados o condiciones a cumplir)
```

Por ejemplo

Cambiar el estado a activo los que estén inactivos

```
update estudiantes set activo=’S’ where activo=’N’
```

Consultas

Una consulta de varias tablas

Todos los estudiantes activos que estén matriculados en el curso 103

```
Select * from estudiantes e, cursos c, cursos_matriculados m, cursos c
where e.activo='S'
and c.id-curso=103
and e.id-estudiante =m.id-estudiante
and c.id-curso= m.id-curso
```

Creacion de claves o llaves primarias

Las claves primarias pueden especificarse como parte de la instrucción create table de SQL usando la cláusula primary key (campos llaves) y las claves foráneas con la cláusula foreign key (campos claves) references tabla (campos llave primaria).

```
create table cursos_matriculados
(id-estudiante char (10) not null,
id-curso int not null,
fecha-matricula date not null,
primary key (id-estudiante, id-curso,
foreign key (id-estudiante) references estudiantes(id-estudiante),
foreign key (id-curso) references cursos (id-curso))
```

Modificar la estructura de una tabla

Con la instrucción alter table se utiliza para añadir o eliminar campos a una tabla existente. La sintaxis de la instrucción es la siguiente:

```
alter table t add c1 tipo1 <null | not null>,…, cn tipo1 <null | not null>
```

donde t es el nombre de una tabla existente, c es el nombre del campo que se desea añadir y tipo es el tipo de dato del campo ci. Se pueden eliminar atributos de una tabla utilizando la instrucción alter table t drop c

donde t es el nombre de una tabla existente y c es el nombre de un campo de la tabla.

Por ejemplo:

```
alter table estudiantes add edad integer null;
alter table estudiantes drop edad;
```

Procedimiento almacenado

Con la instrucción create procedure nombre_procedimiento

Por ejemplo, si deseo obtener la cantidad de cursos matriculados de un determinado estudiante debo programar:

```
create procedure proc-cantidad_cursos_matriculados
  (in v_estudiante char(10), out v_cantidad_cursos integer)
begin
  Select count(*) into v_cantidad_cursos
  from estudiantes e, cursos_matriculados m,
  where e.id-estudiante =v_estudiante
  and e.id-estudiante =m.id-estudiante
end procedure
```

Disparador o trigger

Un disparador o trigguer se ejecuta para agregar desencadenantes al esquema de la base de datos. Los disparadores son operaciones de la base de datos que se realizan automáticamente cuando ocurre un evento activador de base de datos. Cada trigger debe especificar que se ejecutara para una de las siguientes operaciones: DELETE, INSERT, UPDATE. El trigger se activa una vez por cada fila que se elimina, inserta o actualiza.

Se usa la sentencia SQL de tipo DDL: `create trigger`

Es aconsejable revisar en el libro o en la WEB cada uno de estos conceptos

Conclusiones y recomendaciones

Que el estudiante desarrolle las capacidades para aplicar una transacción de adición, modificación, borrado y consultas cuando sea necesario dentro de las bases de datos y crear bases de datos a través de sentencias de lenguaje SQL

Es importante recomendar el reforzarmiento de estos conocimientos con investigacion propia de ejemplos y explicaciones que se encuentran en la web.

Refrerencias bibliográficas

1. Silberschatz, Korth, Sudarshan. (2002). FUNDAMENTOS DE BASES DE DATOS. Cuarta edición. Madrid, España: McGRAW-HILL/INTERAMERICANA DE ESPAÑA.
2. Rafael Camps Paré, Luis Alberto Casillas Santillán, Dolors Costal Costa, Marc Gibert Ginestà. (2005). Bases de datos. Barcelona, España, ISBN: 84-9788-269-5

San Marcos

www.usanmarcos.ac.cr

San José, Costa Rica