TEMA 22: LENGUAJES DE MANIPULACIÓN Y DEFINICIÓN DE DATOS.

22.1. Introducción

Un lenguaje de Manipulación de Datos (Data Manipulation Languaje (DML)) es un lenguaje proporcionado por el sistema de gestión de bases de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

El lenguaje de manipulación de datos más popular hoy en día es SQL, usado para recuperar y manipular datos en una base de datos relacional. Otros ejemplos de DML son los usados por bases de datos IMS/DL1, CODASYL u otras.

Se clasifican en dos grandes grupos:

.Lenguajes de consulta procedimentales.

.Lenguajes de consulta no procedimentales

El lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla. Es un lenguaje de cuarta generación (4GL)

22.1.2. Orígenes y Evolución

Los orígenes del SQL están ligados a los de las bases de datos relacionales. En 1970 E.F. Codd propone el modelo relacional y asociado a este un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basándose en estas ideas, los laboratorios de IBM definen el lenguaje SEQUEL (Structured English Query Languaje) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo éste una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL−86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

Sin embargo este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado SQL−92 o SQL 2.

En la actualidad el SQL es el estándar de facto de la inmensa mayoría de los SGBD comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL−92 es general y muy amplio.

El ANSI SQL sufrió varias revisiones y agregados a lo largo del tiempo:

AÑO	NOMBRE	ALIAS	COMENTARIOS
1986	SQL	SQL−87	

			Primera publicación hecha por ANSI. Confirmada por ISO en 1987
1989	SQL−89		Revisión menor
1992	SQL−92	SQL2	Revisión mayor
1999	SQL:1999	SQL 2000	Se agregaron expresiones regulares consultas recursivas (para relaciones jerárquicas), triggers y algunas características orientadas a objetos.
2003	SQL:2003		Introduce algunas características de XML, cambios en las funciones, estandarización del objeto sequence y de las columnas autonúmericas. (Ver Eisenberg et al: SQL:2003 Has Been Publisher)
2006	SQL:2006		ISO/IEC 9075−14:2006 Define las maneras en las cuales el SQL se puede utilizar conjuntamente con XML. Define maneras importar y guardar datos XML en una base de datos SQL, manipulandolos dentro de la base de datos y publicando el XML y los datos SQL convencionales en forma XML. Además, proporciona facilidades que permiten a las aplicaciones integrar dentro de su código SQL el uso de Xquery, lenguaje de consulta XML publicado por el W3C (World Wide Web Consortium) para acceso concurrente a datos ordinarios SQL y documentos XML

22.1.3. Características generales

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o mas programas que utilizasen un lenguaje de bajo nivel orientado a registro.

22.1.4. Optimización

Como ya se dijo arriba, y como suele ser común en los lenguajes de acceso a bases de datos de alto nivel, el SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente una orden de ejecución. El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a

cabo una optimización antes de la ejecución de la misma. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difieren sensiblemente en cada motor de base de datos y depende de muchos factores. Existe una ampliación de SQL conocida como FSQL (Furry SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa. Este lenguaje ha sido implementado a nivel experimental y está evolucionando rápidamente.

22.1.5 Lenguaje de Definición de datos (LDD)

El lenguaje de Definición de datos, en inglés Data Definition Language (DLL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

.CREATE

Este comando crea un objeto dentro de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte.

Ejemplo 1 (creación de una tabla):

CREATE TABLE TABLA_NOMBRE (

cl integer not null

nombre VARCHAR (50)

fecha_nac DATE NOT NULL,

PRIMARY KEY (my_field1, my_field 2)

.ALTER

Este comando permite modificar la estructura de un objeto− Se pueden agregar / quitar campos a una tabla, modificar el tipo de un campo, agregar / quitar índices a una tabla, modificar un trigger, etc.

Ejemplo 1 (agregar columna a una tabla):

ALTER TABLE TABLA NOMBRE (

ADD NUEVO_ CAMPO INT UNSIGNED

)

DROP

Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo 1:

DROP TABLE TABLA_NOMBRE

Ejemplo 2:

ALTER TABLE TABLA_NOMBRE

(

DROP COLUMN CAMPO_NOMBRE 1

)

TRUNCATE

Este comando trunca todo el contenido de una tabla. La ventaja sobre el comando DELETE, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande, la desventaja es que TRUNCATE solo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Manipulación de Datos), es en realidad una DDL, ya que internamente, el comando truncate borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

Ejemplo 1:

TRUNCATE TABLE TABLA_NOMBRE

22.1.6. Lenguaje de Manipulación de datos (LMD)

.INSERT

Una sentencia INSERT de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional.

.Forma básica

INSERT INTO tabla (columna1,[columna2,]) VALUES (valor1, [valor2,])

Las cantidades de columnas y valores deben ser las mismas. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo:

INSERT INTO agenda.telefonica (nombre, número) VALUES (`Roberto Fernández`, `4886850`)

Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada.

INSERT INTO tabla VALUES (valor1,; [valor2,])

Ejemplo (asumiendo que `nombre` y `número` son los únicas columnas de la tabla ` agenda_telefonica`):

INSERT INTO agenda_telefonica VALUES (`Roberto Fernández, `4886850`)

.Formas avanzadas

.Inserciones en múltiples filas

Una característica de SQL (desde SQL−92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

INSERT INTO tabla (columna1, [columna2,]) VALUES (valor1a, [valor16,], (value2a, [value26,]),

Ejemplo (asumiendo ese `nombre` y `número `son las únicas columnas en la tabla `agenda_telefonica`):

INSERT INTO agenda_telefonica VALUES (`Roberto Fernández`, `4886850`), (` Alejandro Sosa`, `4556550`); que podía haber sido realizado por las sentencias.

INSERT INTO agenda_telefonica VALUES (`Roberto Fernández`, `4886850`);

INSERT INTO agenda_telefonica VALUES (`Alejandro Sosa`, `0`);

Notar que las sentencias separadas pueden tener semántica diferente (especialmente con respecto a los triggers), y puede tener diferente performance que la sentencia de inserción múltiple.

22.1.7.Sistemas de gestión de base de datos.

Los sistemas de gestión de base de datos con soporte SQL más utilizados son, por orden alfabético:

.DBZ

.Firebird

.Informix

.Interbase

.MySQL

.Oracle

.Postgre SQL

.SQL Server

.Sysbase ASE

22.2. Definición de Datos

22.2.1. SQL : El lenguaje de definición de datos (DDL)

El lenguaje de definición de datos permite:

.Definir y crear una nueva tabla

.Suprimir una tabla que ya no se necesita

.Cambiar la definición de una tabla existente

.Definir una tabla virtual (o vista) de datos)

.Construir un índice para hacer más rápido el acceso a una tabla

.Controlar el almacenamiento físico de los datos por parte del SGBD

22.2.2. Sentencias sobre Creación de Tablas

22.2.2.1. Sintaxis General de la sentencia CREATE TABLE

CREATE TABLE <nombre de tabla>

(nombre_columna 1 tipo [restricción de columna],

nombre:columnaN tipo [restricción de columna],

[restricción_de_tabla])

La sentencia CREATE TABLE se utiliza para crear una tabla dentro de la cual habrá columnas que contienen datos y restricciones.

22.2.2.2. Más en detalle

CREATE TABLE <nombre tabla

(<nombre columna><tipo de dato>

[NOT NULL][UNIQUE][CONSTRAINT< nombre restrcción>[PRIMARY KEY]

[REFERENCES][DEFAULT][CHECK>]

| [PRIMARY KEY (< lista columnas>)]

| [FOREIGN KEY (<lista columnas>)]

| UNIQUE(<lista columnas>)][CONSTRAINT <nombre restricción>], [2])

| [CHECK (condición de búsqueda)]

22.2.3. Tipos de datos

Tipo de dato Descripción

Char (tamaño) Almacena datos de tipo carácter de longitud fija, con un máximo de 2000 caracteres)

Varchar2 (tamaño) Almacena datos de tipo carácter de longitud variable, con un tamaño máximo de 4000

Varchar Actualmente es igual que char

Long Almacena datos de tipo carácter de longitud variable, hasta 2 gigabytes. Solo se permite un Long por

tabla. Una columna de tipo Long no puede utilizarse como parte de un índice. Una función almacenada no puede devolver un Long. Las cláusulas Where, Group By, Order By, Unique, o Connect By no pueden referenciar a una columna Long.

Blob Es un objeto binario de gran tamaño, siendo el tamaño máximo 4GB (gigabytes). Normalmente un blob se utiliza para almacenar una imagen datos de voz, o cualquier otro bloque de datos grande no estructurado.

Date Almacena fechas desde el 1 de enero del 4712 a.c. hasta el 31 de diciembre del 4712 d.c.

Integer Un número entero que no tiene parte fraccionaria

Normalmente un Integer será un valor de 32 bits con un rango de −2147483648 a + 2147483647

Smallint Representa un número entero que no contiene parte fraccionaria. Su precisión nunca será mayor que la de un Integer. Es un valor de 16bits entre −32768 y +32767

Number (1,d) Almacena datos de tipo numérico, siendo 1 la longitud y d el número de dígitos decimales.

Raw (tamaño) Datos binarios puros con una longitud máxima de 2000 bytes. Sirven para almacenar datos de tipo binario como sonido e imágenes digitalizadas.

22.2.4. Restricciones de columnas

NOT NULL. La columna no permitirá valores nulos.

CONSTRAINT. Permite asociar un nombre a una restricción

DEFAULT valor. La columna tendrá un valor por defecto. El SBGD utiliza este valor cuando no se especifica un valor para dicha columna,

PRIMARY KEY. Permite indicar que esta columna es la clave primaria.

REFERENCES. Es la manera de indicar que este campo, es clave ajena y hace referencia a una clave candidata de otra tabla. Esta foreign Key es sólo de una columna.

UNIQUE. Obliga a que los valores de una columna tomen valores únicos (no puede haber dos filas con igual valor). Se implementa creando un índice para dicha (s) columna(s)

CHECK (condición) Permite indicar una condición que debe de cumplir esa columna.

22.2.5. Restricciones de tablas

PRIMARY KEY (columna1, columna2) Permite indicar las columnas que forman la clave primaria.

FOREIGN KEY (columna1, columna2) REFERENCES NombreTabla

Indica las columnas que son clave ajena referenciando a una clave candidata de otra tabla.

UNIQUE (columna1, columna2) El valor combinado de una o varias columnas es único.

CHECK (condición) Permite indicar una condición que deben cumplir las filas de la tabla.

Puede afectar a varias columnas.

La cláusula Foreign Key tiene unas opciones que se explican a continuación (no soportadas) en su totalidad por Oracle)

−Tratamiento de nulos: Se puede indicar cómo debe tratar el SGBD un valor NULL en una o más columnas de la clave ajena, cuando lo compare con las filas de la tabla padre.

−Modo de borrado: Para determinar la acción que se debe realizar cuando se elimina una fila referenciada, se debe utilizar una regla de supresión opcional para la relación (CASCADE, SET NUL, SET DEFAULT, NO ACTION)

22.2.6. Ejemplos

áreas (codigo, nombre, departamento) (código es la clave primaria)

departamentos (código_dpto, nombre) (código_dpto es la clave primaria)

La tabla áreas tiene una clave ajena

áreas. departamento departamentos

CREATE TABLE áreas

(

código char (3) not null,

nombre char (55) not null

departamento char (3) not null,

Primary Key (código)

Foreign key (departamento) REFERENCES departamentos

ON DELETE SET NULL ON UPDATE CASCADE);

ON DELETE SET NULL Significa que si se borra algún departamento de la tabla.

departamentos el campo departamento de las filas de la tabla areas que le reverenciaban se pone como Null.

ON UPDATE CASCADE Significa que si se modifica el código_dpto de una fila de la tabla.

departamento, también se modificara en las filas de la tabla áreas que le referencian.

CREATE TABLE departamentos

(

código_dpto char (3) not null,

nombre char (40) not null,

Primary Key (código_dpto)

);

22.2.7. Renombrar una tabla

RENAME TABLE< nombre tabla existente> TO< nuevo nombre tabla>

22.2.8. Eliminar una tabla de la base de datos

DROP TABLE<nombretabla>[CASCADE, RESTRICT]

22.2.9. Ejemplos

DROP TABLE DEPARTAMENTOS CASCADE

(La tabla se borra, así como las posibles restricciones relativas a esta tabla)

DROP TABLE DEPARTAMENTOS RESTRICT

(La tabla se borra sólo si no se hace referencia a ella en ninguna restricción, p.e. en la definición de claves ajenas.)

22.2.10. Modificar una tabla

ALTER TABLE<nombre tabla>

La sentencia Alter Table se utiliza para cambiar una tabla existente. Dentro de la tabla podemos Add (añadir) o Drop (borrar) columnas y restricciones (PRIMARY KEY , FORING KEY, UNIQUE, CHECK CONSTRAINT)

{ ADD <nombre columna nueva><tipo de dato>[NOT NULL]

MODIFY < nombre columna> [DAFAULT| DROP DEFAULT] valor

DROP < nombre columna> [CASCADE| RESTRICT]

ADD [PRIMARY KEY(nombre columna)|

FOREIGN KEY (nombre columna) REFERENCES nombre_tabla | UNIQUE (nombre columna)| CHECK (condición)

DROP CONSTRAINT nombre_restricción [CASCADE| RESTRICT]

22.2.11. Ejemplos

Agregar a la tabla áreas el campo Responsable de tipo char (30)

alter table áreas

ADD responsable char (30) not null;

Modificar el campo nombre de la tabla departamentos a char (50)

alter table departamentos

MODIFY nombre char (50);

22.2.12. Sentencias sobre sinónimos

Un sinónimo es un nombre que puede utilizarse como sustituto o alias del nombre real de una tabla. Puede ser útil para simplificar algunas expresiones.

22.2.13. Crear un sinónimo

CREATE SYNONYM<nombre sinónimo>FOR<nombre tabla>

Create synonym are for areas;

Create synonym dep for departamentos;

22.2.14. Borrar un sinónimo

DROP SYNONYM < nombre sinónimo>

Drop synonym are

22.2.15. Gestión de Dominios (SI en SQL − NO en ORACLE)

Los dominios se usan como tipos de datos en la definición de columnas al crear tablas. Permiten definir los valores aceptados en las columnas, así como la definición de valore por defecto.

22.2.16. Creación de Dominios:

CREATE DOMAIN < nombre Dominio>AS <Tipo Datos>

[DEFAULT < valor defecto>]

[

{[CONSTRAINT < nombre restricción>] CHECK (< condición check>)}];

22.2.17 Ejemplos:

CREATE DOMAIN Seisdigitos AS CHAR (6) DEFAULT `000000`;

CREATE DOMAIN Seisdigitos AS CHAR (6) DEFAULT `000000` CHECK (VALUE IS NOT NULL) CHECK (CHAR_LENGTH (TRIM (VALUE))=6 AND VALUE

BETWEEN `000000`AND `500000`);

22.2.18. Borrado de Dominios:

DROP DOMAIN <Nombre Dominio>

22.2.19. Modificado de Dominios:

ALTER DOMAIN < Nombre Dominio> { SET DEFAULT < Valor Defecto | DROP DEFAULT}

ALTER DOMAIN < Nombre Dominio> DROP CONSTRAINT < Nombre Restricción < [RESTRICT | CASCADE]