Facultad de Ingeniería

Diseño físico de una base de datos

Tema VI

Semestre 2023-1

UNAM
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Objetivo

El alumno comprenderá y aplicará los elementos necesarios para la implementación física del diseño lógico de la base de datos a través del lenguaje SQL, así como la manipulación y uso de transacciones a través de sentencias.

Introducción

SQL, acrónimo de Structured Query Language, es un lenguaje que permite interactuar con los DBMS para realizar operaciones sobre los datos o sobre la estructura de los datos.

Historia

En los laboratorios de IBM, en los años 70’s, se trabajaba en el desarrollo de un lenguaje que se adaptara a las características del modelo relacional, produciendo un lenguaje llamado sequel.

Historia

En el año de 1986 se convierte en un estándar para la ANSI y en 1987, forma parte de la ISO.

Categorías

- **DDL (Data Definition Language): Sentencias que definen y crean objetos en la BD.**
- **DML (Data Manipulation Language): Sentencias para operar sobre los datos.**
- **DCL (Data Control Language): Sentencias orientadas hacia la administración de la BD.**

DDL - Tablas

Ordenes

Id_orden	Fecha	Id_cliente	Nom_cliente	Estado	Num_art	nom_art	cant	Precio
2301	23/02/11	101	Martin	Caracas	3786	Red	3	35,00
2301	23/02/11	101	Martin	Caracas	4011	Raqueta	6	65,00
2301	23/02/11	101	Martin	Caracas	9132	Paq-3	8	4,75
2302	25/02/11	107	Herman	Coro	5794	Paq-6	4	5,00
2303	27/02/11	110	Pedro	Maracay	4011	Raqueta	2	65,00
2303	27/02/11	110	Pedro	Maracay	3141	Funda	2	10,00

DDL

Tabla1(id_Orden , fecha date , id_Cliente)

Tabla4(id_Cliente, nombre_Cliente, estado)

Tabla2 (id_orden, no_Articulo, cantidad)

Tabla3(no_Articulo, nombre_Articulo, precio)

DDL

```
CREATE TABLE cliente (
id_cliente varchar(13) not null,
nombre varchar(50) not null,
ap_Pat varchar(50) not null,
ap_Mat varchar(50) null,
estado varchar(25) not null
);
```

DDL

- Permanentes
- Temporales -> CREATE TEMPORARY TABLE
- Externas -> CREATE EXTERNAL TABLE

DDL - Valores por defecto

```
CREATE TABLE cliente (
id_cliente varchar(13) not null PRIMARY KEY,
nombre char(50) not null,
ap_Pat char(50) not null,
ap_Mat char(50) null,
estado varchar(25) not null DEFAULT ’cdmx’
);
```

DDL - Restricciones

- Check
- Not null
- Unique
- Primary keys
- Foreign keys

```
CREATE TABLE articulo (
num_Articulo int PRIMARY KEY,
nombre_Articulo varchar(30) not null,
precio numeric CHECK (precio > 0)
);
```

Nivel columna

```
CREATE TABLE articulo (
num_Articulo int PRIMARY KEY,
nombre_Articulo varchar(30) not null,
precio numeric CONSTRAINT verifica_Precio
CHECK (precio > 0) );
```

DDL

```
CREATE TABLE articulo (
num_Articulo int,
nombre_Articulo varchar(30) not null,
precio numeric CHECK (precio > 0),
CONSTRAINT articulo_PK PRIMARY
KEY(num_articulo, nombre_Articulo)
);
```

Nivel tabla

DDL

```
CREATE TABLE articulo (
num_Articulo int,
nombre_Articulo varchar(30) not null,
precio numeric not null,
CONSTRAINT verifica_Precio CHECK (precio > 0),
CONSTRAINT articulo_PK PRIMARY
KEY(num_articulo,nombre_Articulo));
```

DDL - Restricciones

- No action
- Restrict
- Set null
- Cascade

Aplican a borrado y actualización

DDL

```
CREATE TABLE orden (
id_Orden int not null,
fecha date not null DEFAULT now(),
id_Cliente varchar(13),
CONSTRAINT orden_PK PRIMARY KEY(id_orden),
CONSTRAINT orden_cliente_FK FOREIGN KEY (id_Cliente)
REFERENCES cliente(id_Cliente) ON DELETE CASCADE
ON UPDATE RESTRICT);
```

DDL - Valores secuenciales

```
CREATE SEQUENCE nombre_Sec [AS tipo_Dato]
[INCREMENT valor]
[MINVALUE valor]
[MAXVALUE valor]
[START valor]
[ [NO] CYCLE ];

SELECT nextval('nombre_Sec');
```

DDL

```
CREATE SEQUENCE ejemplo
INCREMENT 1
MINVALUE 1
MAXVALUE 4
START 1
CYCLE;


SELECT nextval('ejemplo');
```

DDL - Índices

Estructura de datos que facilita el acceso a la información.

- **Clustered**
- **Non clustered**

DDL

```
CREATE TABLE student
(
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    gender VARCHAR(50) NOT NULL,
    DOB datetime NOT NULL,
    total_score INT NOT NULL,
    city VARCHAR(50) NOT NULL
 )
```

DDL

```
INSERT INTO student

VALUES
(6, 'Kate', 'Female', '03-JAN-1985', 500, 'Liverpool'),
(2, 'Jon', 'Male', '02-FEB-1974', 545, 'Manchester'),
(9, 'Wise', 'Male', '11-NOV-1987', 499, 'Manchester'),
(3, 'Sara', 'Female', '07-MAR-1988', 600, 'Leeds'),
(1, 'Jolly', 'Female', '12-JUN-1989', 500, 'London'),
(4, 'Laura', 'Female', '22-DEC-1981', 400, 'Liverpool'),
(7, 'Joseph', 'Male', '09-APR-1982', 643, 'London'),
(5, 'Alan', 'Male', '29-JUL-1993', 500, 'London'),
(8, 'Mice', 'Male', '16-AUG-1974', 543, 'Liverpool'),
(10, 'Elis', 'Female', '28-OCT-1990', 400, 'Leeds');
```

DDL

```
SELECT * FROM student
```

Los registros serán recuperados en el siguiente orden:

id	name	gender	DOB	total_score	city
1	Jolly	Female	1989-06-12 00:00:00.000	500	London
2	Jon	Male	1974-02-02 00:00:00.000	545	Manchester
3	Sara	Female	1988-03-07 00:00:00.000	600	Leeds
4	Laura	Female	1981-12-22 00:00:00.000	400	Liverpool
5	Alan	Male	1993-07-29 00:00:00.000	500	London
6	Kate	Female	1985-01-03 00:00:00.000	500	Liverpool
7	Joseph	Male	1982-04-09 00:00:00.000	643	London
8	Mice	Male	1974-08-16 00:00:00.000	543	Liverpool
9	Wise	Male	1987-11-11 00:00:00.000	499	Manchester
10	Elis	Female	1990-10-28 00:00:00.000	400	Leeds

DDL

```
CREATE NONCLUSTERED INDEX IX_tblStudent_Name
ON student(name ASC)
```

DDL

```
SELECT * FROM student
```

Los registros serán recuperados en el siguiente orden:

id	name	gender	DOB	total_score	city
1	Jolly	Female	1989-06-12 00:00:00.000	500	London
2	Jon	Male	1974-02-02 00:00:00.000	545	Manchester
3	Sara	Female	1988-03-07 00:00:00.000	600	Leeds
4	Laura	Female	1981-12-22 00:00:00.000	400	Liverpool
5	Alan	Male	1993-07-29 00:00:00.000	500	London
6	Kate	Female	1985-01-03 00:00:00.000	500	Liverpool
7	Joseph	Male	1982-04-09 00:00:00.000	643	London
8	Mice	Male	1974-08-16 00:00:00.000	543	Liverpool
9	Wise	Male	1987-11-11 00:00:00.000	499	Manchester
10	Elis	Female	1990-10-28 00:00:00.000	400	Leeds

DDL

name	Row Address
Alan	Row Address
Elis	Row Address
Jolly	Row Address
Jon	Row Address
Joseph	Row Address
Kate	Row Address
Laura	Row Address
Mice	Row Address
Sara	Row Address
Wise	Row Address

DDL

- Unique
- Non unique
- Compuestos
- Basados en funciones: CREATE INDEX first_name_idx ON user_data (UPPER(first_name));

DDL

CREATE [UNIQUE] INDEX nombre_Ind ON nombre_tabla [USING tipo] (columnas)

CLUSTERED nombre_Tab USING nombre_Ind;

Los sinónimos proporcionan independencia de datos y transparencia de ubicación.

CREATE SYNONYM nombre_Sin
FOR origen

DDL - Vistas

Es una especie de tabla virtual, ya que está compuesta por filas y columnas, y puede estar formada por toda la información de una(s) tabla(s) o parte de ella(s).

DDL

S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

Vista donde sólo mostramos algunas columnas y las observaciones con id < 5

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

DDL

CREATE [OR REPLACE] [TEMP | TEMPORARY] VIEW nombre_Vist [nombres_Columnas] AS consulta;

DDL - Mod. estructura

- Alter: Permite realizar diversas modificaciones a un objeto de la base de datos, por ejemplo, agregar/editar/eliminar una columna a una tabla, editar/agregar/eliminar constraints, modificar almacenamiento, etc.

ALTER TABLE nombre_Tabla accion

DDL

- **Drop: Permite eliminar objetos en la base de datos.**

DROP objeto nombre_Objeto;

Ejercicio

Sea la siguiente relación: R(A,B,C,D,E,F,G,H,I) con las siguientes dependencias:

A -> B, C

D -> A, E

F -> G, H

{F, I} -> D

{D, I} -> F

- **Determinar las CK de la relación**
- **Determinar la PK de la relación**
- **Indicar en que forma normal se encuentra la relación R**
- **Normalizar hasta 3FN**

DML

La sentencia insert nos permite agregar/crear información en una tabla.

DML

```
INSERT INTO nombre_Tabla VALUES (val1, val2, …)

INSERT INTO nombre_Tabla [col1, col2, …]
VALUES (val1, val2, …)

INSERT INTO nombre_Tabla [col1, col2, …] SELECT
(col1, col2, …)
FROM nombre_Tabla
[WHERE …]
```

Consideraciones

- Considerar tipo de dato
- Cuidado con las llaves foráneas
- Restricciones not null

La sentencia update nos permite actualizar información de una tabla.

```
UPDATE nombre_tabla SET
nombre_columna = valor
[WHERE ...]
```

Consideraciones

- Considerar tipo de dato
- Cuidado con las llaves foráneas
- Puede emplearse la sentencia SELECT siempre y cuando se seleccione sólo una columna

La sentencia delete nos permite borrar información de una tabla.

DELETE FROM nombre_tabla [WHERE ...]

DML

La sentencia merge nos permite agregar, actualizar y borrar información de un solo movimiento.

DML

```
MERGE <table_destino> [AS TARGET]
USING <table_origen> [AS SOURCE]
ON <condicion_compara_llaves>
[WHEN MATCHED THEN
<accion cuando coinciden> ]
[WHEN NOT MATCHED [BY TARGET] THEN
<accion cuando no coinciden por destino> ]
[WHEN NOT MATCHED BY SOURCE THEN
<accion cuando no coinciden por origen
```

En postgres no hay una sentencia MERGE. Pero puede “hacerse”, ya sea con triggers, procedures, o con la instrucción ON CONFLICT

Transacciones

Conjunto de operaciones realizadas de manera ordenada -> Agrupación de consultas

Control de transacciones:

- Commit: Confirma una transacción
- Rollback: Deshace completamente una transacción o permite regresar a un checkpoint
- Checkpoint: Puntos de control dentro de una transacción

Propiedades de las transacciones (ACID):

- Atomicidad: La transacción es tratada como una unidad atómica, o se ejecutan todas sus operaciones o ninguna. No hay transacciones parcialmente ejecutadas.
- Consistencia: La BD debe permanecer en un estado consistente después de una transacción. No deben presentarse efectos adversos.

Transacciones

- Aislamiento: Toda transacción debe tratarse como si fuera la única en proceso.
- Durabilidad: Los datos que son confirmados por una transacción, deben quedar almacenados sin importar las fallas que puedan presentarse en el sistema.

Niveles de aislamiento:

- Lecturas no confirmadas: Permite lecturas sucias, lo que implica que una operación realizada dentro de una transacción puede partir de cambios que aún no son confirmados por otra transacción.

Transacciones

TRANSACCION 1

SELECT EDAD WHERE ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 20

SELECT EDAD WHERE ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 30

TRANSACCION 2

UPDATE ALUMNO SET EDAD = 30
WHERE NOMBRE = ‘MONSERRAT’;

— 30

ROLLBACK;

TENEMOS LECTURAS SUCIAS

Transacciones

- **Lecturas confirmadas: Garantiza que cualquier lectura de datos está confirmada a la hora de la lectura, evita lecturas sucias. Default en postgres.**

Transacciones

TRANSACCION 1

SELECT EDAD FROM ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 20

TENEMOS LECTURAS NO REPETIBLES

SELECT EDAD FROM ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 30

TRANSACCION 2

UPDATE ALUMNO SET EDAD = 30
WHERE NOMBRE = ‘MONSERRAT’;

— 30

COMMIT;

Transacciones

TRANSACCION 1

SELECT EDAD FROM ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 20

SELECT EDAD FROM ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 20

TRANSACCION 2

UPDATE ALUMNO SET EDAD = 30
WHERE NOMBRE = ‘MONSERRAT’;

— 30

ROLLBACK;

Transacciones

TRANSACCION 1

SELECT EDAD FROM ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 20

SELECT EDAD FROM ALUMNO
WHERE NOMBRE =
‘MONSERRAT’;

— 20

TRANSACCION 2

UPDATE ALUMNO SET EDAD = 30
WHERE NOMBRE = ‘MONSERRAT’;

— 30

Transacciones

- **Lecturas repetibles: La transacción mantiene bloqueos en todas los registros a los que hace referencia. Evita lecturas no repetibles.**

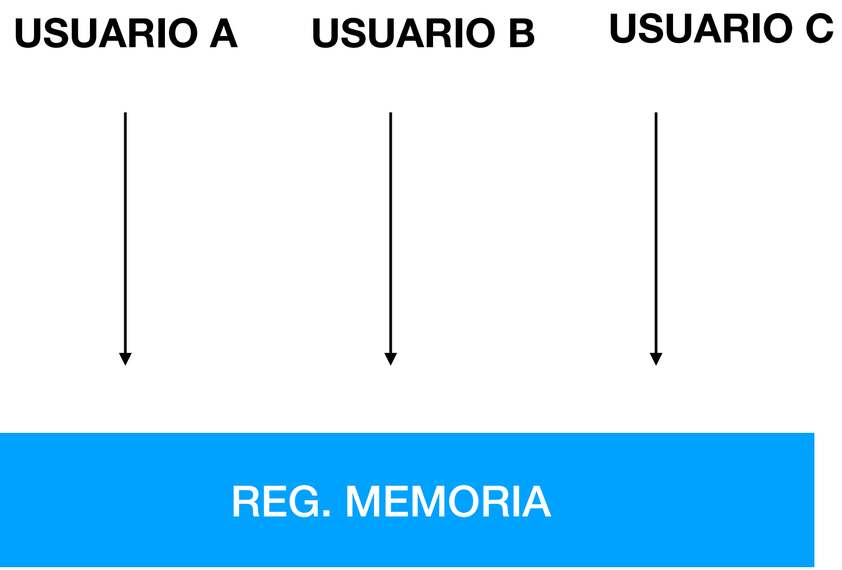
Transacciones

- **Lecturas serializables: Nivel de aislamiento más alto, en el que las transacciones dan la apariencia de ejecutarse de forma secuencial.**

Transacciones

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

Transacciones



Control de concurrencia:

- Optimista: Asume que difícilmente, las transacciones se van a interferir.

Transacciones

- **Pesimista: Bloquea una operación hasta asegurarse que no se causa algún conflicto**