Bases de datos

Tema III – El lenguaje estructurado de consulta - SQL

Universidad Nacional del Litoral
FACULTAD DE INGENIERÍA
Y CIENCIAS HÍDRICAS

SQL *Structured Query Language*

Componentes del SQL

- Catálogo o diccionario de datos
- Los comandos
- Las palabras reservadas
- Los conectores lógicos y predicados
- El lenguaje de definición de datos
- El lenguaje de manejo de datos
- El lenguaje de control de datos

SQL *Structured Query Language*

Comandos del SQL

ALTER TABLE
BEGIN TRANSACTION
COMMIT TRANSACTION
CREATE
DELETE
DROP
GRANT
INSERT
REVOKE
ROLLBACK
SELECT
UPDATE

SQL *Structured Query Language*

Las palabras reservadas

Son las palabras claves que forman los comandos de SQL, así como a las palabras calificativas que se usan con ellos.

No pueden utilizarse como identificadores o sea que no se pueden usar como nombres de tablas, de vistas o de columnas sin aplicarles símbolos especiales definidos por el implementador, con el fin de que se distingan de sus correspondientes palabras claves.

SQL *Structured Query Language*

Tipos de datos y operadores

SQL funciona con tres tipos principales datos:
- Cadenas alfanuméricas (CHAR, CHARACTER VARING)
- Numérico
  - Entero
    - Cortos (byte, smallint, short)
    - Largos (integer, bigint, numeric)
  - Decimal
- Fecha (DATE)

Operaciones básicas de valores:

Suma	(+)
Resta	(-)
Multiplicación	(*)
División	(/)

SQL *Structured Query Language*

Conectores lógicos y predicados

Conectores:

AND
OR
NOT

Predicados:

comparación	(=, <>, <, >, <=, =>)
entre	(...BETWEEN...AND...)
incluido (o no) en	(IN, NOT IN)
como	LIKE
nulos	NULL
cuantificadores existenciales	(ALL, SOME, ANY) (EXISTS, NOT EXISTS)

Lenguaje de definición de datos

CREATE DATABASE nombre_de_la_base [adicionales]

Dentro de adicionales, pueden ir distintos elementos que dependerán del motor sobre el que se ejecute la orden. Estos elementos pueden ser:

- **si llevará o no archivo de *log*;**
- ***dbspace* o *device* donde se alojarán los datos,**
- ***dbspace* o *device* donde se alojará el log,**
- **espacio inicial tanto para datos como para log,**
- **forma de crecimiento de los dispositivos,**
- **etc**

Lenguaje de definición de datos

```
CREATE TABLE nombre_de_la_tabla
    (nombre_columna1   tipo_de_datos(tamaño_de_los_datos),
    nombre_columna2    tipo_de_datos(tamaño_de_los_datos),
    ...
    nombre_columnaN    tipo_de_datos(tamaño_de_los_datos));
```


Lenguaje de definición de datos

```
CREATE TABLE  owner.nombre_tabla (
                Columna1  tipo_dato (tamaño)
        [NOT NULL] [DEFAULT valor_por_default]
        [CONSTRAINT nombre_constraint_de_columna]
        [UNIQUE] [PRIMARY KEY]
        [REFERENCES nombre_tabla (columna/s)]
        [CHECK (condición)],
                .....,
                ColumnaN .....,
    [CONSTRAINT nombre_constraint_de_tabla]
    [UNIQUE (columna/s)]
    [PRIMARY KEY (columna/s)]
    [CHECK (condición)]
    [FOREIGN KEY (columna/s) REFERENCES nombre_tabla (columna/s)]
);
```


Lenguaje de definición de datos

Conceptos asociados

Páginas

Densidad de filas (fillfactor, max_row_per_page)

Extents

Lenguaje de definición de datos

```
CREATE VIEW nombre_de_la_vista
        ((nombre_columnas_de_la_vista))
        AS
        (SELECT columna(s) FROM tabla(s) WHERE condición(es));
```


Lenguaje de definición de datos Índices

- **Son estructuras de datos que permite acelerar la interacción con los datos de las tablas.**
- **Pueden crearse tantos índice como se deseen en una tabla cualquiera.**
- **Puede tener un índice para cada columna de la tabla, así como un índice para una combinación de columnas.**
- **Cuántos índices sean creados y de qué tipo para una determinada tabla, dependerán de los tipos de consultas esperadas que se dirigirán a la base de datos y además del tamaño de la misma.**
- **El crear demasiados índices puede presentar tantos inconvenientes como el crear muy pocos.**

SQL *Structured Query Language*

Lenguaje de definición de datos
Índices

- **Mejoran el rendimiento reduciendo las entradas y salidas del disco. Realizan una función análoga a las de los índices de un libro acelerando la recuperación.**
- **Aseguran la unicidad. Puede crearse un índice único sobre una columna. Después, si se hace algún intento para insertar una fila que duplique el valor que ya está en esa columna, la inserción será rechazada. Por consiguiente, un UNIQUE INDEX actúa como una comprobación de la unicidad de la columna.**

SQL *Structured Query Language*

Lenguaje de definición de datos
Índices

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED] INDEX
        nombre_del_indice
    ON  nombre_de_la_tabla (nombre(s)_de_la(s)_columna(s));
```


SQL *Structured Query Language*

Lenguaje de definición de datos
Modificación de tablas

Para añadir otra columna, la sintaxis es:

```
ALTER TABLE nombre_de_la_tabla
        ADD COLUMN nombre_de_la_columna  tipo_de_datos;
```

Para cambiar el ancho de una columna ya existente, la sintaxis es:

```
ALTER TABLE  nombre_de_la_tabla
        ALTER COLUMN nombre_de_la_columna
        TYPE tipo_de_datos    nueva_anchura;
```

Para modificar el nombre de una columna:

```
ALTER TABLE nombre_tabla
        RENAME nombre_viejo TO nombre_nuevo;
```


SQL *Structured Query Language*

Lenguaje de definición de datos
Modificación de tablas

Para eliminar una columna:

```
ALTER TABLE  nombre_de_la_tabla
        DROP COLUMN nombre_de_la_columna;
```

Para eliminar o agregar restricciones:

```
ALTER TABLE nombre_de_la_tabla
        ADD CONSTRAINT ... (igual definición que en la tabla);

ALTER TABLE nombre_de_la_tabla
        DROP CONSTRAINT nombre_constraint;
```


SQL *Structured Query Language*

Lenguaje de definición de datos
Eliminación

La supresión de bases de datos:

```
DROP DATABASE nombre_de_la_base;
```

La supresión de tablas:

```
DROP TABLE  nombre_de_la_tabla;
```

La supresión de vistas:

```
DROP VIEW nombre_de_la_vista;
```

La supresión de índices:

```
DROP INDEX  nombre_del_índice ON nombre_de_la_tabla;
```


SQL *Structured Query Language*

Lenguaje de manejo de datos

La Inserción

```
INSERT INTO nombre_de_la_tabla (nombre_de_columna1,
                                nombre_de_columna2, ...)
        VALUES ('valor1', 'valor2', ...);

INSERT INTO nombre_de_la_tabla (nombre_de_columna1,
                                nombre_de_columna2, ...)
        (subconsulta);
```

Los valores que se inserten deberán ajustarse al tipo de datos de la columna en la que se van a insertar. Los valores CHAR y los tipo DATE deben ir entre comillas o apóstrofos mientras que los valores NUMERIC y NULL simplemente se indican.

SQL *Structured Query Language*

Lenguaje de manejo de datos

La actualización

```
UPDATE  nombre_de_la_tabla
   SET    columna1 = valor_nuevo,
          columna2 = valor_nuevo,
          ...
          columnaN = valor_nuevo,
      [WHERE condición];
```

La cláusula SET indica las columnas que hay que actualizar y qué valores hay que poner en ellas.
Actúa en todas las filas que satisfacen la condición especificada por la cláusula WHERE. La cláusula WHERE es opcional, pero si se omite, se actualizarán todas las filas de la tabla. Puede contener una subconsulta.
Se pueden actualizar múltiples columnas en cada fila, con un único comando UPDATE.

SQL *Structured Query Language*

Lenguaje de manejo de datos

El Borrado

```
DELETE  FROM  nombre_de_la_tabla
    [WHERE condición];
```

No se pueden borrar parcialmente las filas.

La cláusula WHERE puede ser compleja y puede que incluya condiciones múltiples, conectores, y/o subconsultas.

Sin cláusula WHERE se suprimen todas las filas y dejará solamente las especificaciones de las columnas y el nombre de la tabla.

El comando TRUNCATE TABLE elimina todos los datos de la tabla indicada sin dejar rastros en el log, y consecuentemente no pueden deshacerse los cambios (el borrado). Es más rápido pero peligroso.

SQL *Structured Query Language*

Lenguaje de manejo de datos

La recuperación de datos

```
SELECT columna1, columna2, ......,columnaN
    FROM nombre_de_la_tabla;
```

Para recuperar solamente un conjunto de filas que se hayan especificado, se indicará la característica común de las filas que se quieren obtener mediante la cláusula WHERE.

La cláusula WHERE corresponde al predicado de selección del álgebra relacional. Consta de un predicado que incluye columnas de la tabla o tablas que aparecen en la cláusula FROM.

WHERE es opcional, y en caso de omitirla, devuelve todas las filas de la tabla.

SQL
Implementación de integridad de datos con integridad declarativa

- La **Integridad Declarativa** es un mecanismo para implementar la integridad de datos. El server la mantiene a través del uso de cláusulas de restricciones y de valores por defecto que se incorporan en la sentencia **create table** o **alter table**, y mediante las cuales se condicionan los valores de los datos. Estas cláusulas son:
  - **Default** constraint
  - **Check** constraint
  - **Unique** constraint
  - **Primary key** constraint
  - **Reference** constraint
- Existen constraints de nivel columna y de nivel tabla.

SQL
Cláusula default

- La cláusula **default** pertenece al nivel columna.
- Una cláusula **default** es aplicada sobre un **insert** si no existe un valor que llene la columna.
- La cláusula **default** se utiliza para especificar el valor que ocupará la columna.
- Los **defaults** pueden ser constantes, null o *user* (char de 30 para el ultimo caso).
- Ejemplo:

```
create table publishers
(pub_id    char(4)                    not null,
 pub_name  varchar(40)                null,
 city      varchar(20)  default "Akron" null,
 state     char(2)      default "OH"    null)
```


SQL
Vista general de los Check Constraints

- Check constraints:
  - Implementan integridad a los dominios.
  - Pueden ser aplicados a nivel columna o a nivel tabla.
- Son utilizados para especificar:
  - Una lista o conjunto de valores (a, b, c)
  - Un rango de valores (entre 0 y 255)
  - Un formato de edición (dos caracteres y un número: “AB3”)
  - Condiciones que debe cumplir un valor simple
- Es validado durante un **update** o **insert.**
- Especifica una “condición booleana”, restricción que el servidor de datos impone sobre todas las filas insertadas o modificadas en la tabla.

Check Constraints de Nivel Columna

Ejemplo:

```
create table publishers
(pub_id     char(4)     not null
   constraint chk_pubid
   check (pub_id in ("1389", "0736", "0877")
   or pub_id like "99[0-9][0-9]"),
 city       varchar(20) null,
 state      char(2)     null)
```

- Si el default viola la condición del check constraint, se rechazará cualquier inserción que use el valor default pero éste será aceptado ante la ausencia de dato para la columna en cuestión.

Check Constraints de nivel Tabla: Ejemplo

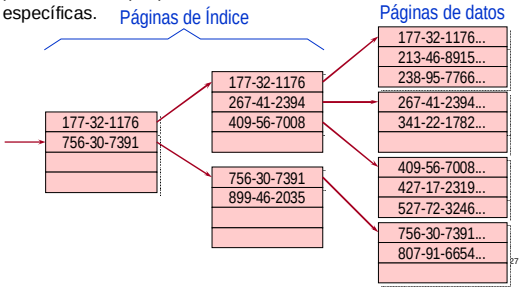
Crear una tabla *discounts* con chequeos que validen lowqty <= highqty para insertar o modificar una fila

```
create table discounts
(discounttype     varchar(40)  not null,
 stor_id          char(4)      null,
 lowqty           smallint     null,
 highqty          smallint     null,
 discount         float        not null,

   constraint chk_low_high
   check (lowqty <= highqty)
)
```


Los índices

- Un índice es una estructura de almacenamiento separada creada para una tabla que permite acceder directamente a filas de datos específicas.

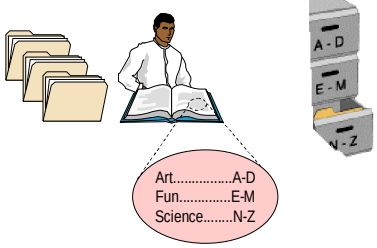


Los índices

- Existen dos tipos de índices:
  - Clustered
  - Nonclustered
- Una tabla puede tener:
  - Un máximo de un índice clustered
  - Un máximo de 249 índices nonclustered
- El orden físico de los datos depende del tipo de índice:
  - Si es un índice clustered, los datos estarán ordenados por la o las columnas indexadas
  - Si todos los índices son nonclustered o si no tiene índices, los datos estarán en el orden en el cual han sido insertados

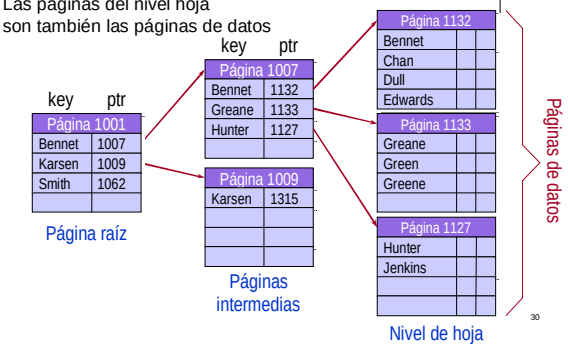
Almacenamiento Clustered de una Tabla

- Las filas de datos en la tabla están físicamente ordenadas
- Los usuarios definen cómo las filas de datos son ordenadas especificando la clave física (índice)



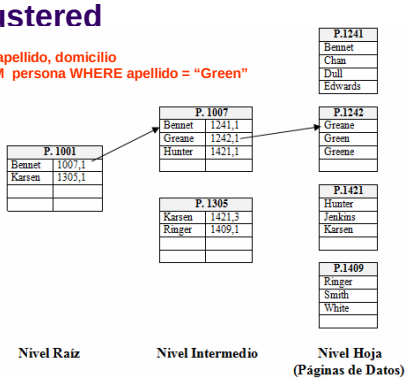
Estructura de un índice Clustered

Las páginas del nivel hoja son también las páginas de datos



Estructura de un índice Clustered

SELECT apellido, domicilio
FROM persona WHERE apellido = "Green"



Operaciones con tablas Clustered

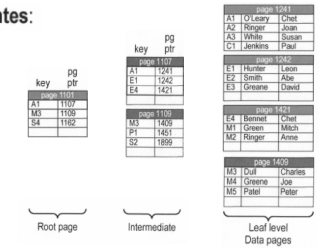
INSERT

- Las filas son colocadas en el orden físico de acuerdo al valor de la clave.
- Las otras filas se mueven hacia abajo.
- Si no hay espacio se realiza una división de página (Split page)
  - Se localiza una nueva página en un extent en uso. Si no existe, se reserva uno nuevo.
  - Los punteros de la pagina anterior y siguiente adyacentes son cambiados incorporando la nueva página a la cadena.
  - Aproximadamente la mitad de las filas son movidas a la nueva página con la nueva fila insertada en el orden correspondiente.
  - Cambian los punteros de los niveles de índices. Si existen índices Nonclustered deben cambiar sus referencias a las ubicaciones de la nueva página

Operaciones con tablas Clustered

INSERT insert into employee (emp_id, emp_lname, emp_fname) values ("M6", "Coates", "Paul");

Antes:

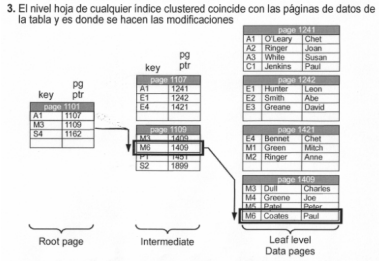


Operaciones con tablas Clustered

INSERT insert into employee (emp_id, emp_lname, emp_fname) values ("M6", "Coates", "Paul");

Después:

1. Los datos son insertados en la tabla ordenada
2. Como el nuevo dato cabe en una página de índice existente, el nivel intermedio del índice no es afectado
3. El nivel hoja de cualquier índice clustered coincide con las páginas de datos de la tabla y es donde se hacen las modificaciones



Operaciones con tablas Clustered

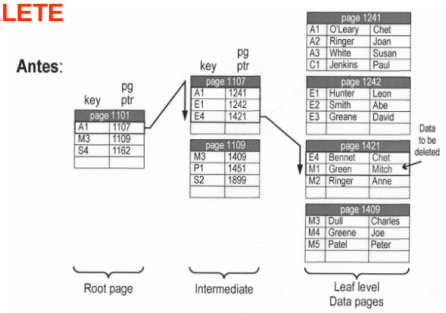
DELETE

- Las filas son movidas hacia arriba.
- Si se borra la última fila de la página ésta es liberada.
- Los punteros de la página anterior y siguiente son actualizados.
- Si otra página en el extent está siendo utilizada, la página queda disponible
- Si todas las páginas en el extent están vacías, se libera el extent.

Operaciones con tablas Clustered

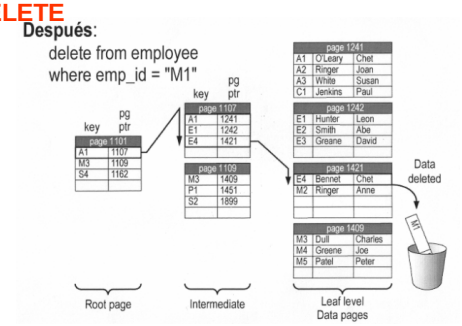
DELETE

Antes:



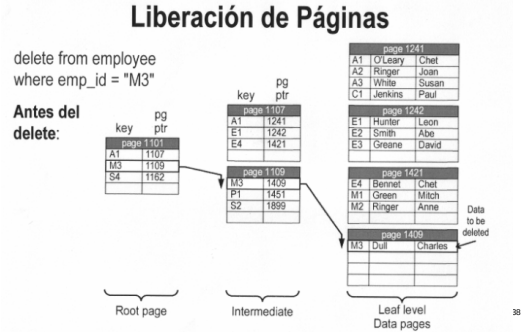
Operaciones con tablas Clustered

DELETE



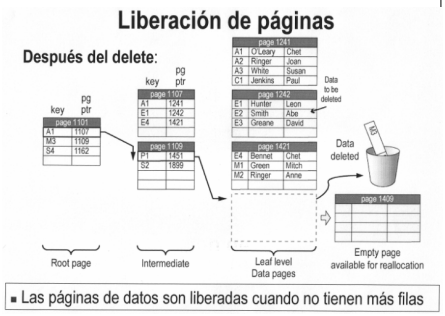
37

Operaciones con tablas Clustered



38

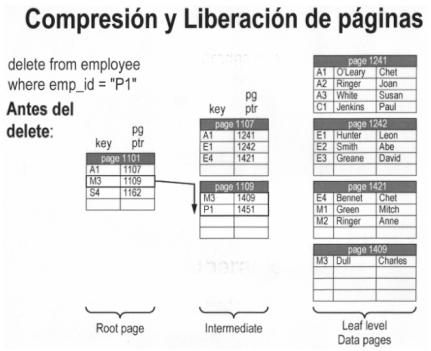
Operaciones con tablas Clustered



- Las páginas de datos son liberadas cuando no tienen más filas

39

Operaciones con tablas Clustered



40

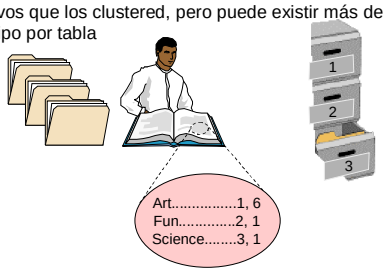
Operaciones con tablas Clustered

- La página de índice es liberada cuando tiene una fila y existe espacio suficiente en la página anterior

41

Almacenamiento de Índices Nonclustered

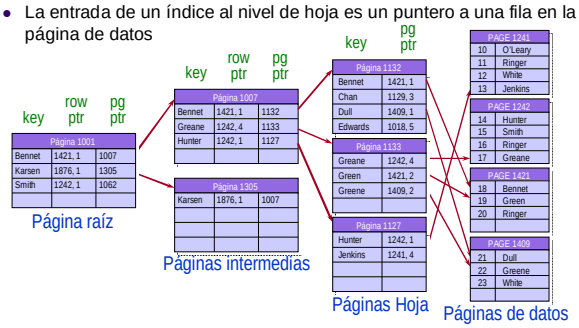
- La clave del índice está almacenada en orden aleatorio
- Los índices nonclustered son grandes y pueden ser menos efectivos que los clustered, pero puede existir más de uno de este tipo por tabla



42

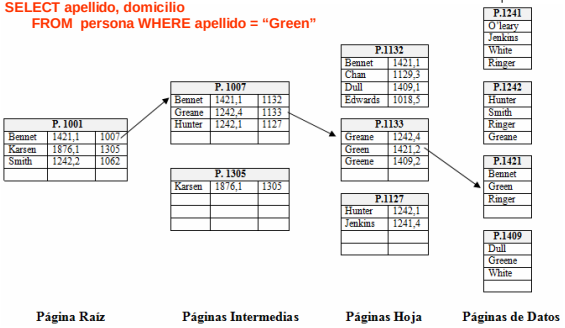
Estructura de un índice Nonclustered

- Las páginas de nivel hoja no son las mismas que las de datos
- La entrada de un índice al nivel de hoja es un puntero a una fila en la página de datos



Estructura de un índice Nonclustered

SELECT apellido, domicilio
FROM persona WHERE apellido = "Green"



Operaciones con tablas Nonclustered (heap)

INSERT

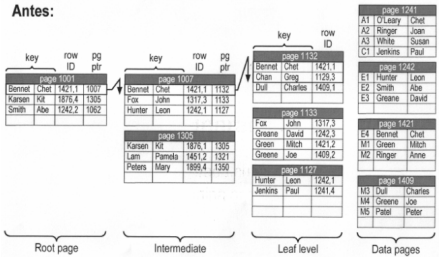
- **Se adiciona siempre en la última página de la tabla.**
- **Si está llena se utiliza una nueva en el extent actual.**
- **Si el extent está lleno se fija por páginas disponibles en otro extent.**
- **Si no hay, se reserva otro extent.**

45

Operaciones con tablas Nonclustered (heap)

INSERT

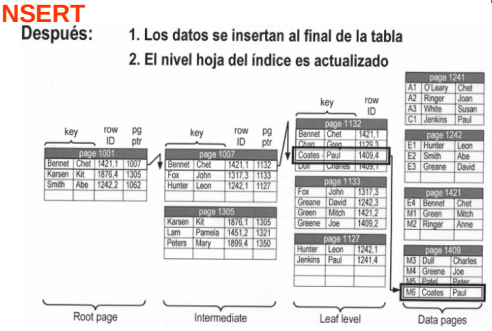
insert employee (emp_id, emp_lname,emp _fname)
values ("M6", "Coates", "Paul")



46

Operaciones con tablas Nonclustered (heap)

INSERT



47

Operaciones con tablas Nonclustered (heap)

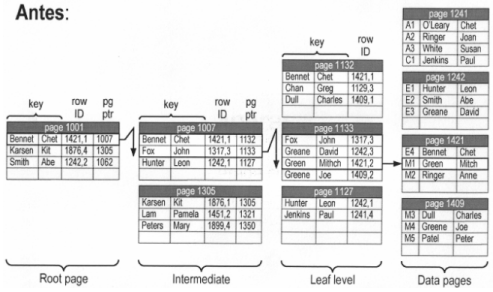
DELETE

- **Las filas que siguen a la borrada son movidas una posición hacia atrás dentro de la página.**
- **Si se borra la última fila de la página ésta es liberada.**
- **Si otra página en el extent está siendo utilizada, la página queda disponible**
- **Si todas las páginas en el extent están vacías, se libera el extent.**

48

Operaciones con tablas Nonclustered (heap)

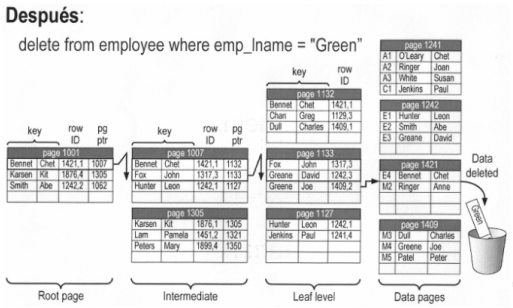
DELETE



49

Operaciones con tablas Nonclustered (heap)

DELETE



50

Operaciones con tablas Nonclustered (heap)

UPDATE

- **Si la longitud de la fila no cambia, la fila actualizada reemplaza a la existente y los datos no son movidos.**
- **Si la longitud cambia y hay espacio en la página, la fila queda en el lugar pero otra fila es movida hacia arriba o abajo para mantener filas contiguas en la página.**
- **Si la fila no cabe en la página, la fila es borrada de la página actual y una nueva fila es insertada en la última página de la tabla.**

51

Creación de índices con Constraints

- Unique Constraint
  - Asegura que no haya dos filas que tengan el mismo valor
  - Permite un solo valor NULL en la columna
  - Crea un **unique index nonclustered** por defecto

52

Unique Constraint

- Ejemplo de **unique constraint** a nivel tabla:

```
create table sales
(stor_id    char(4)     not null,
 ord_num    varchar(20) not null,
 date       datetime    not null,
      constraint unq_c_storid_ordnum unique
      clustered (stor_id, ord_num))
```

53

Creación de índices con Constraints

- Primary Key Constraint
  - Asegura que no existan dos filas con el mismo valor
  - No admite ningún valor NULL en la columna
  - Crea un unique index ***clustered por defecto***

54

Primary Key Constraint

- Ejemplo a nivel tabla:

```
create table sales
(stor_id    char(4)     not null,
 ord_num    varchar(20) not null,
 date       datetime    not null,
      constraint pky_storid_ordnum primary key
      nonclustered (stor_id, ord_num))
```

55

Constraints de Integridad Referencial Declarativa

- Use **create table** para mantener la integridad referencial cuando una clave ajena es insertada o modificada o una clave primaria es borrada o actualizada.
- Nivel de columna:

```
create table table_name
(column_name datatype
   [constraint constraint_name]
   references ref_table [ref_column]
   [,…])
```

- Nivel de tabla:

```
[constraint constraint_name]
foreign key (column_name [{, column_name}..])
references [[database.]owner.] ref_table
[(ref_column [{, ref_column}...])]
```

- Para claves compuestas se usan constraints de nivel de tabla.

56

Ejemplo de Constraint a nivel columna

```
create table titles
(title_id   tid         not null,
 title      varchar(80) null,
 pub_id     char(4)     null
 constraint ref_pub_id references publishers(pub_id),
 notes      varchar(200) null)
```

- Crear una tabla *titles* con constraint a nivel de columna sobre *pub_id*
- La tabla *publishers* debe existir y tener un índice unique sobre *pub_id*
- Un valor de foreign key no puede ser insertado o modificado con un valor de *pub_id* inexistente en *publishers*
- Un *pub_id* en *publishers* no puede ser borrado si un valor de foreign key lo referencia

57

Ejemplo de Constraint a nivel tabla

```
create table salesdetail
(stor_id    char(4)     not null,
 ord_num    varchar(20) not null,
 title_id   tid         not null,
 qty        smallint    not null,
 discount   float       not null,

            constraint ref_sales
            foreign key (stor_id, ord_num)
            references sales (stor_id, ord_num),

            constraint ref_titles
            foreign key (title_id)
            references titles (title_id))
```

58

Constraints de Integridad Referencial: Modificando y Borrando Primary Keys

- La integridad declarativa no es usada para efectuar caídas en forma de cascada ya sean modificaciones o borrado cuando una clave primaria es modificada o borrada

stores

stor_id	stor_name
5023	Thoreau Reading
8042	Bookbeat
7066	Barnum's

Valores de Primary key

sales

stor_id	order_num
5023	ABC-123-DEF-425-1Z3
5023	NF-123-ADS-642-9G3
5023	ZZ999-ZZZ-999-0AD
7066	234518
8042	Asoap132
8042	Fsoap867

Qué puede ocurrir en los datos de la clave ajena si la clave primaria es borrada o modificada? … →

59

Restricciones sobre Actualizaciones o Borrado de valores Clave

…

- **No pueden borrarse o modificarse** filas que contengan valores de columnas que estan referenciados en un constraint de integridad referencial si las filas tienen valores que están correspondiendo a valores de foreign key.
- No pueden eliminarse tablas referenciadas hasta que la tabla que hace la referencia es eliminada o el constraint de integridad referencial es removido.

60

Alter Table

- **Alter Table** es usado para:
  - Agregar columnas a una tabla
  - Agregar o eliminar constraints de una tabla
  - Renombrar tablas
  - Renombrar columnas
  - Cambiar el tipo de dato
- La única manera de modificar un constraint es usando **alter table**
- Si un constraint es agregado o modificado o un default es reemplazado usando **alter table**, existiendo datos en la tabla, éstos no son afectados

Ejemplos de Alter Table

- Reemplazando un default:

```
alter table publishers
replace state default "CA"

alter table publishers
replace state default null
```

- Agregando / eliminando un check constraint a nivel columna:

```
alter table roysched
add constraint chk_hirange
check (hirange < 75000)

alter table roysched
drop constraint chk_hirange
```

(continúa…)

Ejemplos de Alter Table

- Agregando / eliminando un unique constraint:

```
alter table sales
add constraint unq_stid_onum
unique (stor_id, ord_num)

alter table sales
drop constraint unq_stid_onum
```

- Agregando / eliminando un primary key constraint:

```
alter table sales
add constraint pky_c_stid_onum
primary key (stor_id, ord_num)

alter table sales
drop constraint pky_c_stid_onum
```

(continúa…)

Ejemplos de Alter Table

- Agregando / eliminando un constraint referencial:

```
alter table roysched
add constraint ref_title
foreign key(title_id)
references titles(title_id)

alter table roysched
drop constraint ref_title
```


SQL *Structured Query Language*

Lenguaje de control de datos

GRANT - REVOKE

Permiso	Objeto
SELECT	Tabla, Vista o Columna
UPDATE	Tabla, Vista o Columna
INSERT	Tabla, Vista
DELETE	Tabla, Vista
REFERENCES	Tabla, Columna
EXECUTE	Stored procedure

GRANT privilegio_nivel_base TO usuario [WITH GRANT OPTION];

Este comando permite otorgar permisos en el ámbito de base de datos completa.

GRANT privilegio_nivel_tabla ON nombre_de_la_tabla TO usuario [WITH GRANT OPTION];

Este comando permite otorgar permisos a nivel de tabla.

SQL *Structured Query Language*

Lenguaje de control de datos

GRANT - REVOKE

GRANT ALL PRIVILEGES TO user1 WITH GRANT OPTION;
Todos los privilegios al USER1 y puede a su vez otorgarlos a otros.

GRANT ALL PRIVILEGES ON tabla1, tabla2 TO user2;
Todos los privilegios al USER2 sobre las tablas "tabla1" y "tabla2".

GRANT ALL PRIVILEGES ON tabla1(col1), tabla2(col1) TO user2, user3, user4;
Todos los privilegios al USER2, USER3 y USER4 sobre las columnas col1 de la tabla1 y col1 de la tabla2.

GRANT SELECT ON tabla1 TO PUBLIC;
Privilegio de consulta a *cualquier usuario* sobre la "tabla1".

SQL *Structured Query Language*

Lenguaje de control de datos
GRANT - REVOKE

```
GRANT {ALL [PRIVILEGES] | lista_de_permisos }
    ON
    {
      nombre_de_tabla [(lista_de_columnas)]
      | nombre_de_vista [(lista_de_columnas)]
      |  nombre_procedimiento_almacenado
    }
    TO { PUBLIC | lista_de_usuarios  |  nombre_del_rol }
    [WITH GRANT OPTION]
```


SQL *Structured Query Language*

Lenguaje de control de datos
GRANT - REVOKE

```
REVOKE
    [GRANT OPTION FOR] {ALL [PRIVILEGES] | lista_de_permisos }
    ON
      {
        nombre_de_tabla [(lista_de_columnas)]
        |  nombre_de_vista [(lista_de_columnas)]
        |  nombre_procedimiento_almacenado
      }
    FROM  { PUBLIC | lista_de_usuarios  |  nombre_del_rol }
```


SQL *Structured Query Language*

Lenguaje de control de datos
Transacciones

```
BEGIN TRANSACTION;

COMMIT TRANSACTION;

ROLLBACK TRANSACTION;
```


SQL *Structured Query Language*

ACID (**A**tomicity, **C**onsistency, **I**solation and **D**urability)

Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Es un acrónimo **A**tomicidad, **C**onsistencia, a**I**slamiento y **D**urabilidad.

Atomicidad: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

Consistencia: Integridad. Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido.

SQL *Structured Query Language*

ACID (**A**tomicity, **C**onsistency, **I**solation and **D**urability)

Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.

Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Cumpliendo estos 4 requisitos un sistema gestor de bases de datos puede ser considerado ACID Compliant.