

BASES DE DATOS

Objetos de BD



Agenda



- ❑ Introducción
- ❑ Esquemas, tablas
- ❑ Índices
- ❑ Otros
- ❑ Ejercicios

Introducción

- Una BD consta de muchos objetos entre los que se cuentan:

Esquemas

Tablas

Índices

Secuencias

Vistas

Sinónimos

Tablas temporales

Vistas Materializadas

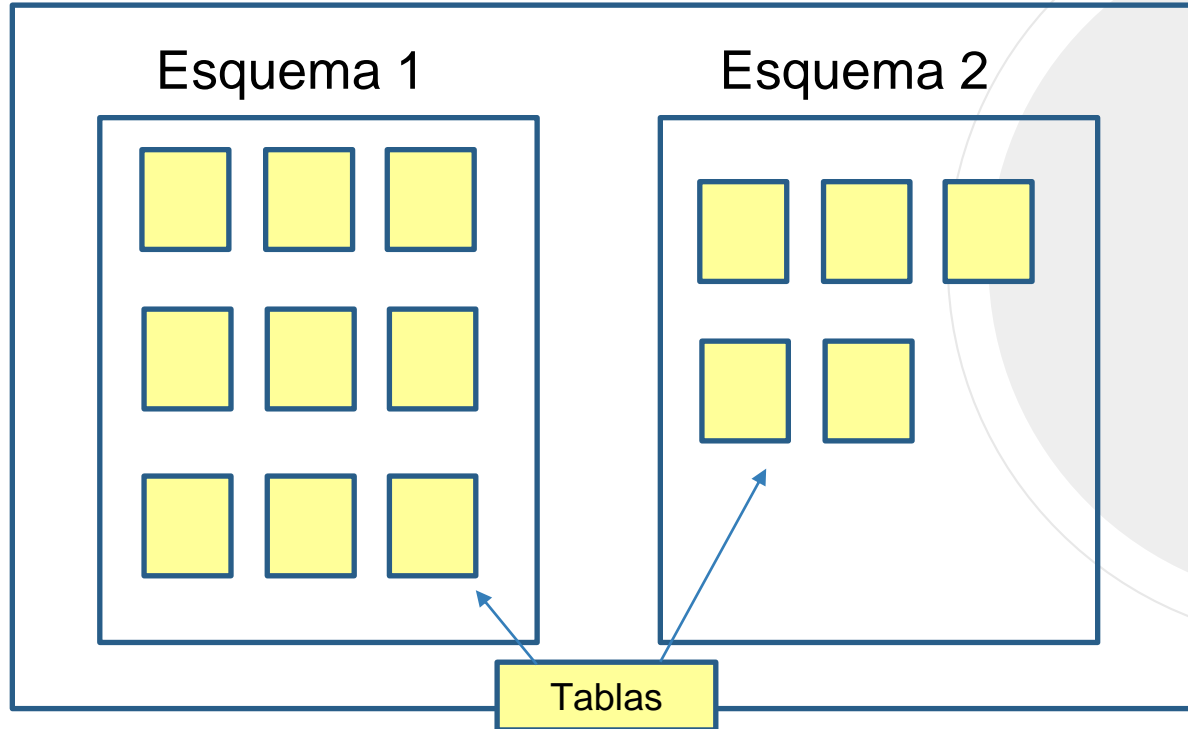
DBLinks

Store Procedures

Triggers

Estructura

BD



Esquemas

- Agrupación lógica de objetos.
- Su implementación depende del motor.
- Un nombre de objeto de BD debe ser único en cada esquema.
- En Sql Server el esquema default es **dbo**
- En Oracle no existe el esquema como tal.

Tablas

- Objeto que funciona como contenedor de información.
- Compuesta por filas y columnas.

Results Messages						
	manu_code	manu_name	lead_time	state	f_alta_audit	d_usualta_audit
	ANZ	Anza	5	CA	NULL	NULL
	HHH	HHHH	5	CO	2020-04-29 12:12:19.860	dbo
	HRO	Hero	4	CA	NULL	NULL
	HSK	Husky	5	CA	NULL	NULL
	KAR	Karsten	21	CA	NULL	NULL
	NKL	Nikolus	8	AZ	NULL	NULL
	NRG	Norge	7	AZ	NULL	NULL
	PRC	ProCycle	9	AZ	NULL	NULL

Tablas

```
CREATE TABLE alumnos (  
    Legajo int,  
    Nombre varchar(20),  
    Apellido varchar(20),  
    FechaNacimiento Date,  
    DNI int,  
    fechaCreacion Datetime,  
    UsuarioCreacion varchar(20)  
)
```

Tablas

Los tipos de datos dependen del motor utilizado.
Algunos de Sql Server son ...

Strings	Numéricos	Fecha
Char(n) Varchar(n)	Int Decimal(m,n) Bigint Smallint Tinyint Float Real	Date Datetime

Tablas

```
CREATE TABLE alumnos (  
    Legajo int,  
    Nombre varchar(20),  
    Apellido varchar(20),  
    FechaNacimiento Date,  
    Sexo char,  
    DNI int,  
    fechaCreacion Datetime,  
    UsuarioCreacion varchar(20)  
)
```



Tablas - Constraints

Integridad de Dominio
Integridad de Entidad
Integridad Referencial



Tablas - Constraints

- Tipos de datos
- Obligatoriedad
- Checks
- Default
- PKs, UKs
- Claves Foráneas



Tablas - Constraints

Tipos de datos y Obligatoriedad.

```
CREATE TABLE alumnos (  
    Legajo int,  
    Nombre varchar(20) NOT NULL,  
    Apellido varchar(20) NOT NULL,  
    FechaNacimiento Date NOT NULL,  
    sexo char NOT NULL,  
    DNI int NOT NULL,  
    fechaCreacion Datetime,  
    UsuarioCreacion varchar(20)  
)
```

Tablas - Constraints

CHECK

```
CREATE TABLE alumnos (  
    Legajo int,  
    Nombre varchar(20) NOT NULL,  
    Apellido varchar(20) NOT NULL,  
    FechaNacimiento Date NOT NULL,  
    sexo char NOT NULL CHECK(sexo in('F', 'M')),  
    DNI int NOT NULL,  
    fechaInscripcion Datetime,  
    Pais varchar(20),  
    CHECK(fechaNacimiento < fechaInscripcion)  
)
```

Tablas - Constraints

DEFAULT

```
CREATE TABLE alumnos (  
    Legajo int,  
    Nombre varchar(20) NOT NULL,  
    Apellido varchar(20) NOT NULL,  
    FechaNacimiento Date NOT NULL,  
    sexo char NOT NULL CHECK(sexo in('F', 'M')),  
    DNI int NOT NULL,  
    fechaInscripcion Datetime,  
    Pais varchar(20) DEFAULT 'Argentina',  
    CHECK(fechaNacimiento < fechaInscripcion)  
)
```

Tablas - Constraints

PRIMARY KEY

- Corresponde a la clave primaria.
- Identifica unívocamente cada fila.
- Puede ser simple o compuesta.
- Sólo puede haber una en cada tabla.
- No pueden contener valores nulos.

UNIQUE

- Corresponde a las claves alternativas o secundarias.
- Identifica unívocamente cada fila.
- Puede ser simple o compuesta.
- Puede haber mas de una en cada tabla.
- Puede contener valores nulos.

Tablas - Constraints

PRIMARY KEY. UNIQUE.

```
CREATE TABLE alumnos (  
    Legajo int PRIMARY KEY,  
    Nombre varchar(20) NOT NULL,  
    Apellido varchar(20) NOT NULL,  
    FechaNacimiento Date NOT NULL,  
    sexo char NOT NULL CHECK(sexo in('F', 'M')),  
    dni int NOT NULL,  
    fechaInscripcion Datetime,  
    Pais varchar(20) DEFAULT 'Argentina',  
    UNIQUE (sexo, dni)  
)
```


Tablas - Constraints

FOREIGN KEY

- Atributos que hacen referencia a una clave de otra tabla.
- Puede ser simple o compuesta.
- Pueden haber varias en cada tabla.
- Representan relaciones entre tablas.

Tablas - Constraints

FOREIGN KEY.

```
CREATE TABLE empleados(  
    Legajo int PRIMARY KEY,  
    Nombre varchar(20) NOT NULL,  
    Apellido varchar(20) NOT NULL,  
    FechaNacimiento Date NOT NULL,  
    nro_depto smallint REFERENCES departamentos (nro_depto),  
    gerente int NULL,  
    FOREIGN KEY (gerente) REFERENCES empleados (legajo_int)  
);
```

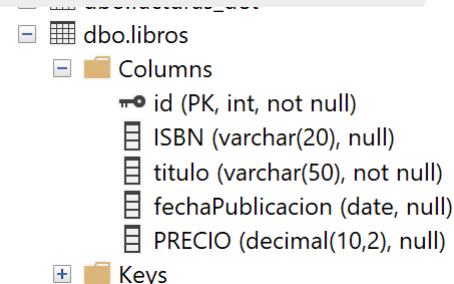
Operador ALTER

- Se utiliza para modificar objetos.
- En el caso de las tablas se puede utilizar para agregar, modificar o borrar columnas y/o constraints.

Operador ALTER

```
CREATE TABLE libros (id          int not null,  
                      ISBN       varchar(20),  
                      titulo     varchar(50),  
                      autor      varchar(50),  
                      fechaPublicacion date);
```

```
ALTER TABLE libros ADD PRIMARY KEY (id);  
ALTER TABLE libros ADD PRECIO DECIMAL(10,2) DEFAULT 0;  
ALTER TABLE libros ALTER COLUMN titulo varchar(50) not null;  
ALTER TABLE libros DROP COLUMN autor;
```



dbo.libros
Columns
id (PK, int, not null)
ISBN (varchar(20), null)
titulo (varchar(50), not null)
fechaPublicacion (date, null)
PRECIO (decimal(10,2), null)
Keys

Operador DROP

“

- Se utiliza para borrar objetos de la BD.
- Una vez borrados no se pueden recuperar.

DROP TABLE Libros;



Ejercicios

En base al ejercicio de las jugadoras de la Clase 4, cree las tablas con las sentencias DDLs que correspondan.

ÍNDICES

“Es una estructura de datos que permite acceder más rápidamente a la datos de las tablas”

- Asociado a una tabla.
- No produce acoplamiento.
- Tabla independiente del índice.
- Diferentes tipos.
- Algunos se crean en forma implícita.

ÍNDICES

“ Clasificación por tipo

- Único o Duplicado
- Simple o Compuesto

ÍNDICES

Clasificación por tipo

- **Único:** No puede estar repetido.
- **Duplicado:** Puede contener valores repetidos.
- **Simple:** Conformado por un solo atributo.
- **Compuesto:** Formado por varios atributos.

```
create unique index PK_items on items (order_num, item_num);
```

125 %



Messages

Commands completed successfully.

Completion time: 2022-02-22T11:16:03.5760138-03:00

|

ÍNDICES

“

Tipos según su estructura física

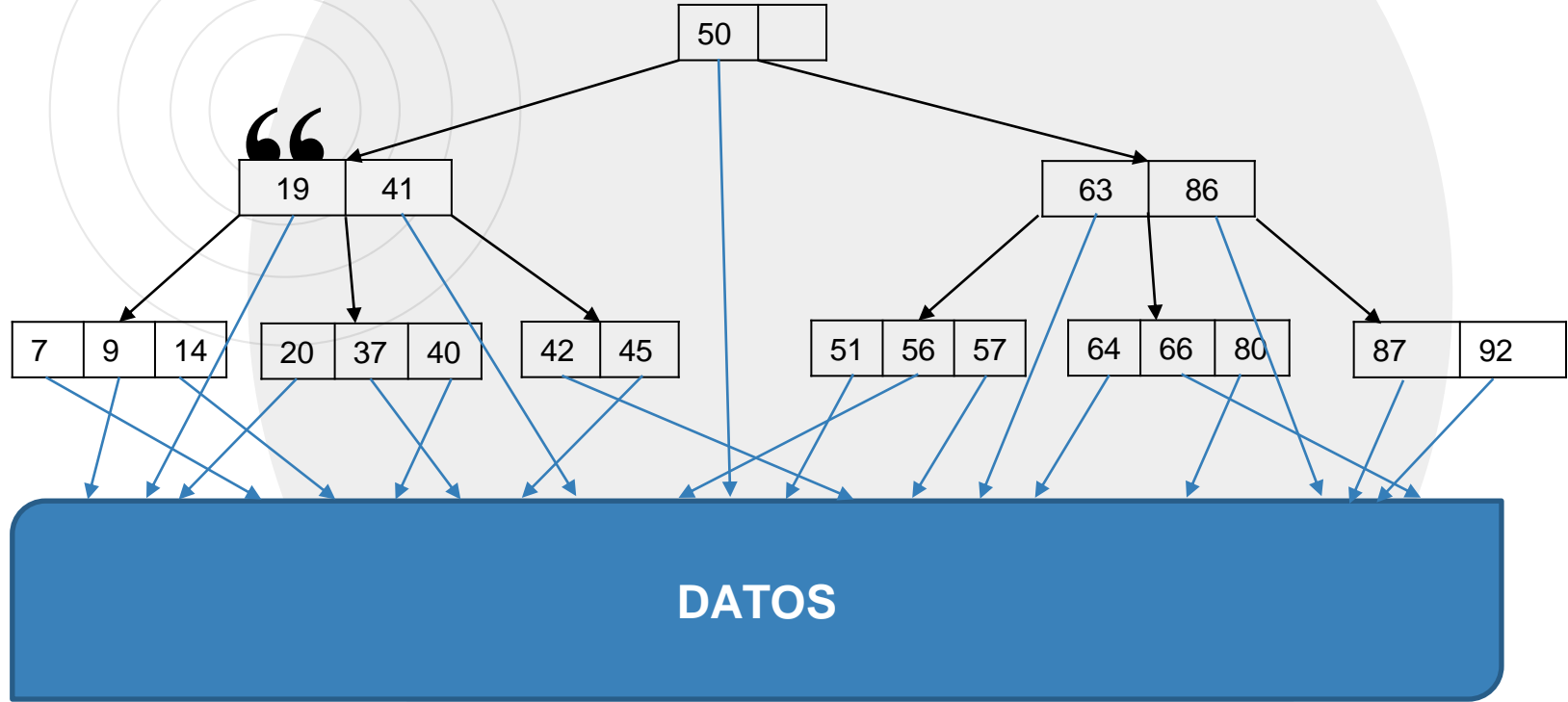
- **Btree, Btree+**
- **Cluster**
- **Hash**
- **Bitmap**

ÍNDICES - BTree

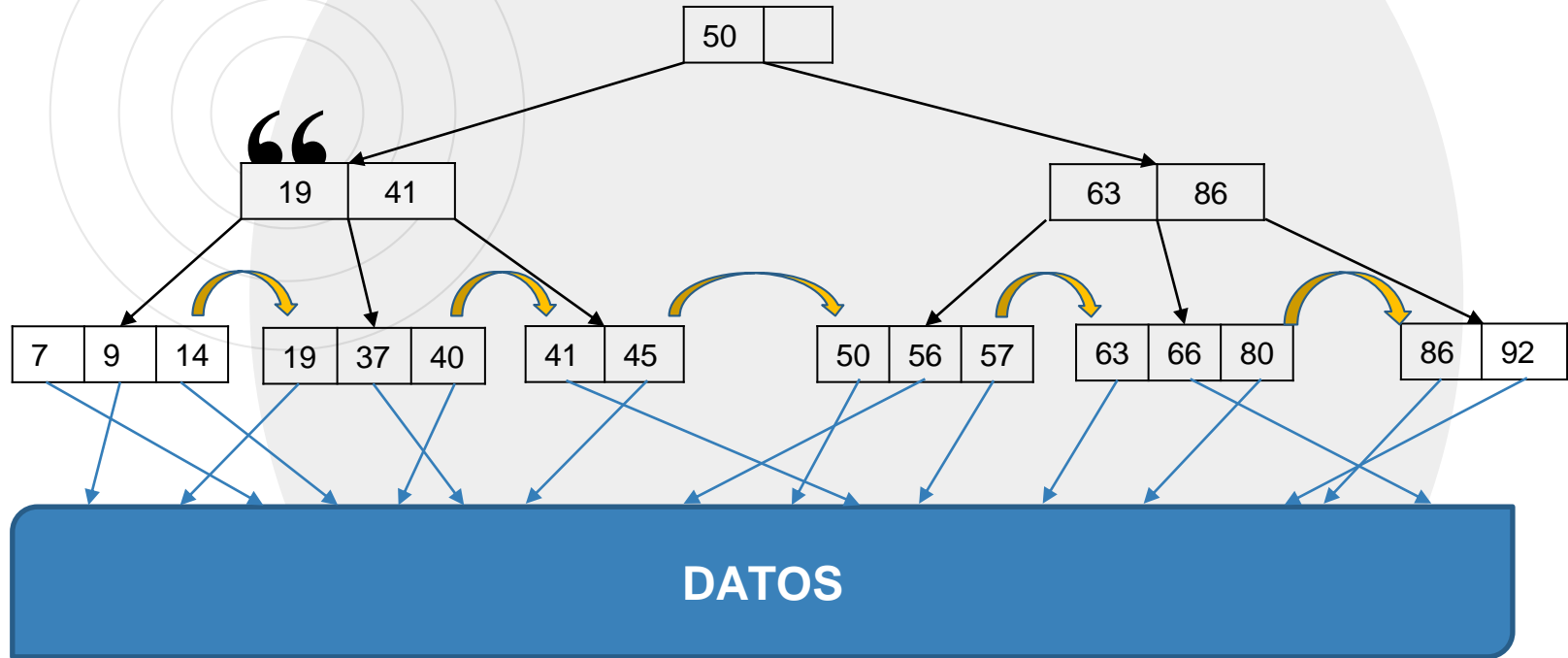
*“Es una estructura de datos de tipo árbol **balanceado** compuesto por nodos que contienen varios valores y punteros”*

- Posee un nodo raíz, nodos internos (ramas) y nodos hojas.
- Se dice que es de **orden n** cuando cada nodo que no sea hoja tiene entre $n/2$ y n hijos.

ÍNDICES – BTree



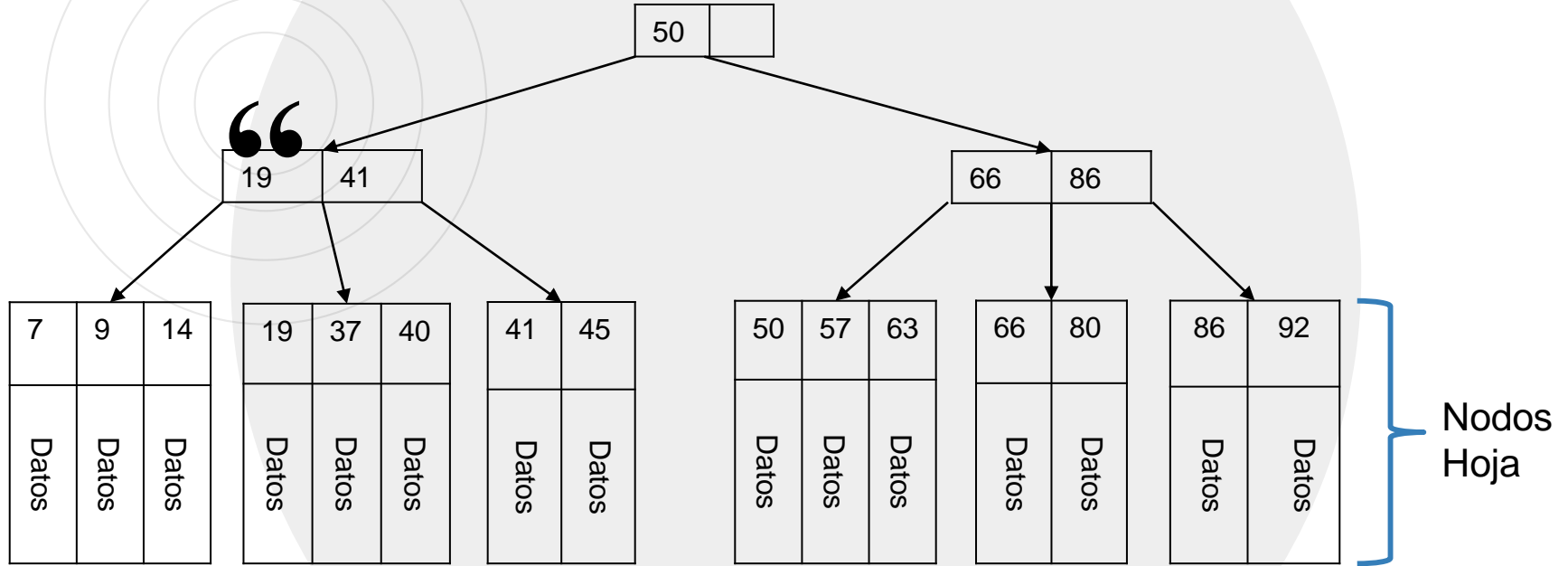
ÍNDICES – BTree+



ÍNDICES - Cluster

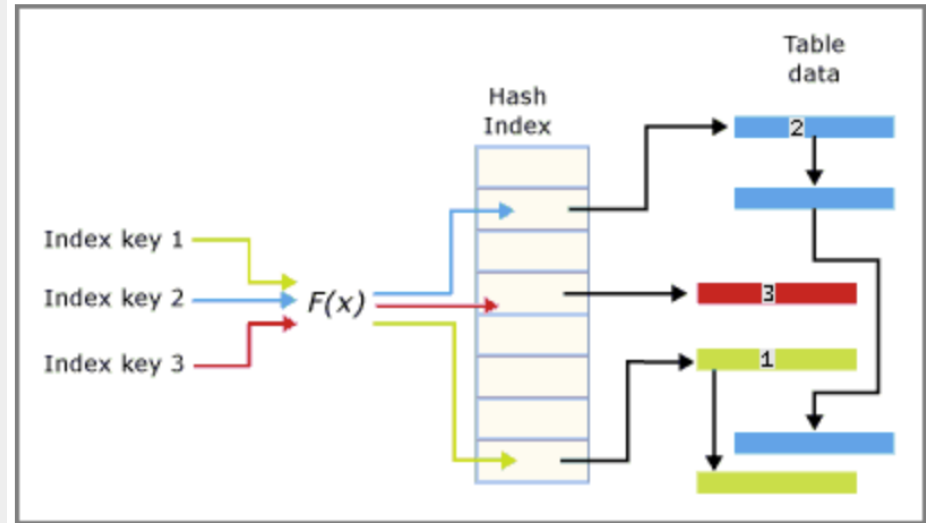
- **Es una estructura de datos de tipo árbol balanceado compuesto por nodos que contienen varios valores y punteros cuyas hojas contienen los datos del registro de la tabla correspondiente**
 - **Los registros de la tabla se ordenan por los valores del índice.**
-
- Solo puede haber un índice Cluster por tabla

ÍNDICES - Cluster



ÍNDICES - Hash

- ❑ Utilizados para búsquedas puntuales “=”
- ❑ Búsquedas con TODOS los atributos de la clave
- ❑ Menor tamaño que Btree
- ❑ Más rápidos que Btree



ÍNDICES - Hash

Conceptos

“

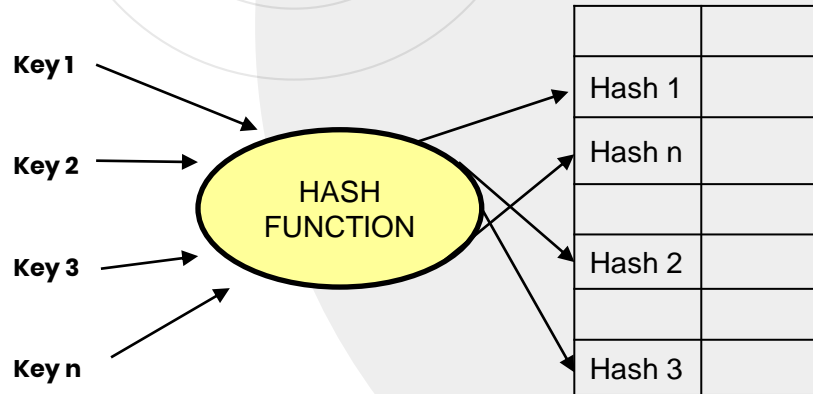
Bucket: Es una unidad de almacenamiento que ocupa normalmente un bloque de disco con varios registros. Cada bucket contiene una colección de pares (*clave, valor*)

Hash function: Funcion $h(k)$ cuyo resultado mapea a una dirección donde se encuentra un bucket.

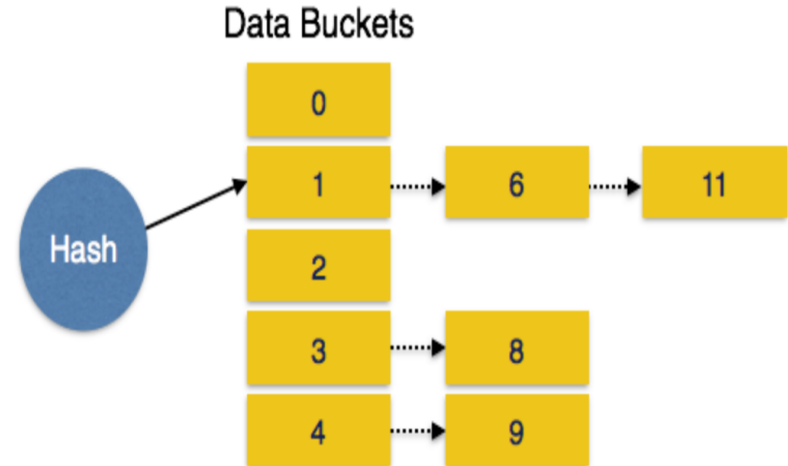
ÍNDICES - Hash

Tipos - Hashing estático

Open Hashing



Closed Hashing

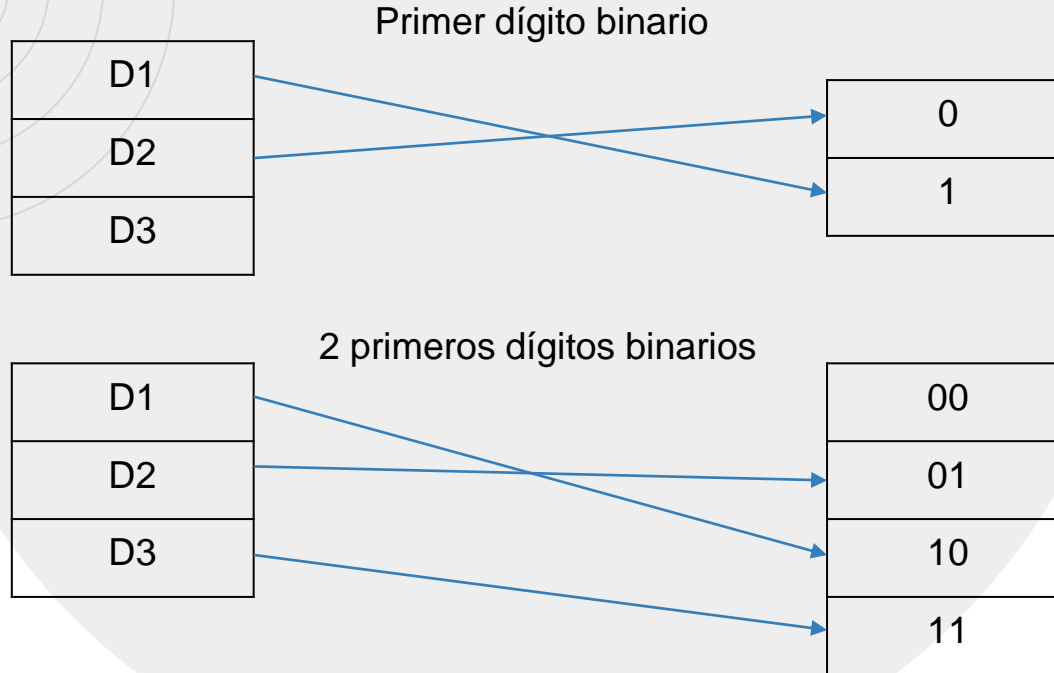


ÍNDICES - Hash

Tipos - Hashing dinámico o extendido

“

$h(D1) \rightarrow 1001$
 $h(D2) \rightarrow 0101$
 $h(D3) \rightarrow 1101$



ÍNDICES - Hash

“

Desventajas

- ☐ Ante el aumento de colisiones pierden eficiencia.
- ☐ No están ordenados.
- ☐ No son eficientes para búsquedas por rangos
- ☐ No son eficientes para búsquedas por NOT =
- ☐ No son eficientes para búsquedas con LIKE

ÍNDICES - Bitmap

- ❑ Utiliza mapas de bits
- ❑ Un índice de mapas de bits sobre el atributo A de la relación r consiste en un mapa de bits para cada valor que pueda tomar A.
- ❑ Cada mapa de bits tiene tantos bits como el número de registros de la tabla.
- ❑ Se utilizan en atributos con baja cardinalidad.

Fem	Masc
0	1
1	0
1	0
0	1
1	0
0	1
1	0
0	1
1	0
1	0

ÍNDICES - Bitmap

Legajo	Género	Estado civil
0001	M	Casado
0002	F	Soltero
0003	F	Casado
0004	M	Viudo
0005	F	Casado
0006	M	Soltero
0007	F	Soltero
0008	M	Soltero
0009	F	Casado
0010	F	Casado

Tabla



Fem	Masc
0	1
1	0
1	0
0	1
1	0
0	1
1	0
0	1
1	0
1	0

Bitmap de Género

Soltero	Casado	Viudo
0	1	0
1	0	0
0	1	0
0	0	1
0	1	0
1	0	0
1	0	0
1	0	0
0	1	0
0	1	0

Bitmap de estado civil

ÍNDICES - Bitmap

“ VENTAJAS

- ❑ Reducidos Tiempos de respuesta .
- ❑ Reducido tamaño de almacenamiento.
- ❑ Buena performance.
- ❑ Incluye en el índice valores **NULL**.
- ❑ Sistemas OLAP (On-Line Analytical Processing)
(DW)

Fem	Masc
0	1
1	0
1	0
0	1
1	0
0	1
1	0
0	1
1	0
1	0

ÍNDICES - Bitmap

“ *DESVENTAJAS* ”

- ❑ Costoso para aplicaciones con muchas actualizaciones concurrentes (OLTP).

ÍNDICES. EN GENERAL.

Ventajas

- ☐ Acceso mas rápido a la información.
- ☐ Integridad (Pk, Fk).

Desventajas

- ☐ Espacio en disco.
- ☐ Mantenimiento (insert, update, delete).

ÍNDICES - Ejemplos

```
CREATE TABLE facturas(  
  Nro_factura int,  
  nro_cliente int,  
  fechaEmision DATE NOT NULL,  
  fechaVto     DATE NOT NULL,  
  fechaPago    DATE NOT NULL);
```

create unique index facturas_idx1 on facturas (nro_factura)

create index facturas_idx2 on facturas (nro_cliente)

create index facturas_idx3 on facturas (fechaVto, nro_cliente)

create CLUSTERED index facturas_idx4 on facturas
(fechaEmision)



Preguntas ?



Ejercicios?



“

1. Tipos de índices
2. Tipos de índice según su organización física.
3. Diferencias entre índice Cluster y No cluster.
4. Diferencias entre Btree y Hash.
5. Índice Bitmap
6. Ventajas y desventajas de los índices.

SECUENCIAS

- Objeto que genera números secuenciales.
- Se utiliza para numerar registros en forma secuencial sin repeticiones.
- Se implementa mediante un tipo de columna o mediante un objeto separado en la BD.

SECUENCIAS

IDENTITY(inicio, incremento)

“

```
CREATE TABLE pedidos (  
    nro_pedido int PRIMARY KEY IDENTITY(1, 1),  
    cliente int NOT NULL,  
    fechaPedido DATE,  
    plazoEntrega smallint,  
    domicilioEntrega varchar(50)  
)
```

```
INSERT INTO pedidos (cliente, fechaPedido)  
VALUES (1001, '20220302')
```

Qué nota de raro en este INSERT ?

SECUENCIAS

```
CREATE TABLE pedidos (  
    nro_pedido int PRIMARY KEY,  
    cliente int NOT NULL,  
    fechaPedido DATE,  
    plazoEntrega smallint,  
    domicilioEntrega varchar(50)  
)
```

```
CREATE SEQUENCE seqPedidos  
INCREMENT BY 1  
START WITH 10  
MINVALUE 1  
MAXVALUE 9999  
CYCLE;
```

```
INSERT INTO pedido (nro_pedido, cliente, fechaPedido)  
VALUES (seqPedidos.nextval, 1001, '20220302')
```


SECUENCIAS/IDENTITYs - Diferencias

- ✓ *Los IDENTITYs se implementan mediante una propiedad en los atributos de una tabla.*
- ✓ *Las SEQUENCES son independientes de las tablas*
- ✓ *Los valores de las IDENTITYs se generan cuando se inserta una fila*
- ✓ *Los valores de las SEQUENCES se generan utilizando la cláusula NEXT VALUE*
- ✓ *Los valores de las IDENTITYs no son cíclicos y su valor máximo está dado por el tamaño del atributo.*
- ✓ *En las SEQUENCES pueden ser seteados los valores máximos y pueden ser cíclicas.*

VISTAS

- Es otro objeto de una BD
- Consta de una sentencia SELECT que simula ser una tabla.
- Referencia a otras tablas o vistas.
- Tiene un nombre
- No contiene datos almacenados
- No ocupa espacio en disco (salvo la metadata)

VISTAS

CARACTERÍSTICAS

- Permite implementar algún grado de seguridad.
- Enmascarar complejidad a los usuarios.
- Desacopla las aplicaciones de cambios a nivel lógico.
- Algunas permiten ejecutar sentencias insert, update y delete.
- No se pueden declarar con ORDER BY

VISTAS

```
CREATE VIEW Facturas_v AS
  SELECT f.factura_num, f.cliente_num,
         c.nombre, c.apellido,
         f.fecha_emision, f.fecha_vto,
         d.renglon, d.producto_cod,
         d.cantidad, d.precio_unit
  FROM facturas f JOIN clientes c ON f.cliente_num = c.cliente_num
                   JOIN facturas_det d ON f.factura_num = d.factura_num;
[ WHERE ]
```

VISTAS

WITH CHECK OPTION

“

```
CREATE VIEW clientesView AS  
SELECT cliente_num,  
       nombre, apellido,  
       empresa, domicilio,  
       ciudad, provincia_cod, estado  
FROM clientes where provincia_cod = 'BA'  
WITH CHECK OPTION;
```

VISTAS

WITH CHECK OPTION

```
INSERT INTO clientesView (cliente_num, nombre, apellido,  
                           empresa, domicilio, ciudad, provincia_cod, estado)  
VALUES (1234, 'BRUCE', 'LEE', 'MIJO', 'Avellaneda 345',  
        'CABA', 'CB', 'A');
```

5 %

Messages

Msg 550, Level 16, State 1, Line 10

The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that
The statement has been terminated.

VISTAS

WITH CHECK OPTION

“

```
UPDATE clientesView  
  set provincia_cod = 'CB'  
 where provincia_cod = 'BA'
```

%

Messages

Msg 550, Level 16, State 1, Line 15

The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION.
The statement has been terminated.

VISTAS MATERIALIZADAS

(Materialized View/Snapshot)

- Es un objeto que persiste los datos devueltos por un query en una tabla junto con los cambios producidos en las tablas origen.
- Estos datos se actualizan manual o automáticamente.
- Dependiendo del motor la actualización puede ser total o incremental.
- Los datos se almacenan en tablas físicas.
- Se utilizan principalmente en DW.

SINONIMOS

- Es un nombre de “alias” que referencia un objeto
- Provee un nivel de abstracción (nombre y locación)
- Simplifica nombres de objetos.
- Brinda transparencia ante cambios de su objeto base (nombre, BD).
- Proporciona Seguridad.
- Referencian: Tablas, Vistas, Store Procedures.

SINONIMOS

- Para crear un sinónimo que referencia objetos de BD en otros servers se necesita especificar el esquema y nombre del sinónimo y el servidor, BD, esquema y objeto al que hace referencia.

```
CREATE SYNONYM schema_1. synonym_name  
FOR server_name. database_name.schema_2. object_name
```

- Una vez creado se pueden realizar operaciones de SELECT, INSERT, UPDATE, DELETE o EXECUTE sobre el sinónimo.

SINONIMOS

Ejemplos



```
CREATE SYNONYM personas  
FOR DEV.RRHHDB01.rrhh.empleados
```

```
Select * from DEV.RRHHDB01.rrhh.empleados
```

```
Select * from personas
```

TABLAS TEMPORALES

- Son tablas que tienen una duración en el tiempo.
- Se pueden realizar las mismas operaciones de SELECT, INSERT, UPDATE y DELETE.
- Se almacenan en un “espacio temporal” de la BD por lo que no forman parte del catálogo.
- No pueden referenciar otras tablas ni ser referenciadas.
- Se utilizan en procesos o procesamientos Batch.
- Se utilizan como resultados intermedios de procesamiento de otras tablas o para reutilizar tablas ya procesadas.

TABLAS TEMPORALES

- Pueden ser Locales o Globales.

“

- Las tablas locales solo son vistas por la sesión que la creó.
 - En Sql Server deben tener el prefijo # en el nombre de la tabla.
 - Las tablas Locales o de sesión se eliminan cuando se cierra la sesión o por medio de un DROP explícito.
-
- Las tablas Globales son visibles a todas las sesiones.
 - En Sql Server deben tener el prefijo ## en el nombre de la tabla.
 - Global temporary tables son eliminadas cuando finaliza la sesión que la creó.

TABLAS TEMPORALES

- En SqlServer pueden ser creadas en forma implícita o explícita.

Explícita. Sin datos.

```
CREATE TABLE #empleados (  
    Legajo int PRIMARY KEY,  
    nombre varchar(20) NOT NULL,  
    apellido varchar(20) NOT NULL,  
    depto char(3),  
    salario decimal(10,2))
```

Implícita. Con datos.

```
SELECT *  
    INTO #empleados  
    FROM empleados
```

DB LINKs (Oracle, PostgreSQL)

- Es un objeto que habilita el acceso a objetos en otra base de datos.
- La conexión es en un solo sentido.
- Una vez creado, se puede utilizar para referirse a tablas y vistas en la otra BD.
- Se pueden realizar SELECTs, INSERTs, UPDATES y DELETEs sobre los objeto remotos.

```
create public database link parts_link  
connect to remoteUser identified by password  
using 'remoteDB';
```

STORE PROCEDURES

Son programas almacenados en la BD.

“

Se utilizan para agregar funcionalidad a la suministrada por la BD.

Agregar reglas de negocio.

Desarrollar procesos.

Permiten ser reutilizados.

En ocasiones mejoran la performance

No forman parte del SQL estándar.

STORE PROCEDURES

Son programas almacenados en la BD.

“

Dependiendo del DBMS existen diferentes tipos:

- Procedures
- Functions
- Packages (Oracle)

STORE PROCEDURES

PROCEDURES

- Pueden recibir parámetros de entrada y/o salida.
- No se pueden utilizar en sentencias sql.

```
CREATE PROCEDURE InsertaAlumno AS  
[parámetros tipos]  
Begin  
...  
...  
END
```

STORE PROCEDURES

FUNCTIONS

- Dependiendo del motor puede recibir parámetros de entrada y/o salida.
- Devuelve como resultado un valor.
- No pueden contener DMLs (Insert, Delete, Update)
- Se pueden utilizar en sentencias sql.

```
CREATE FUNCTION nombreFuncion ([parámetros tipos])  
    RETURNS tipoDato AS  
BEGIN  
    ...  
    ...  
END
```

STORE PROCEDURES

PACKAGES (Oracle)

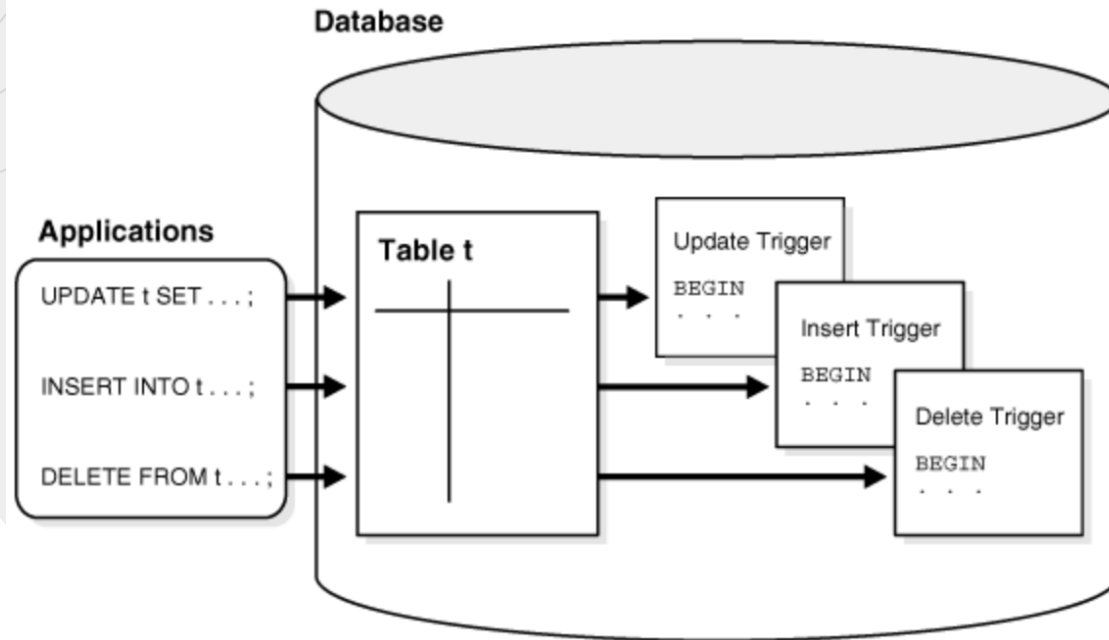
“

- Agrupa procedimientos y funciones de usuario.
- Consta de dos partes: una especificación y un cuerpo.
- La especificación define los módulos a utilizar y define ciertos objetos y variables globales del Package.
- El cuerpo define objetos, variables e implementa los módulos de la especificación y otros.

TRIGGERS

“

Es un objeto de BD compuesto por código que se ejecuta automáticamente ante ciertos eventos.



TRIGGERS

“

Dependiendo del DBMS existen diferentes tipos de triggers:

- DML: Insert, Delete, Update (sobre tablas y views)
- DDL: Create, Alter, Drop, Grant, Revoke (AFTER)
- Sistema: Logon, Logoff, Startup, Shutdown.

TRIGGERS

“

Ademas del evento se define el momento de ejecución del trigger.

Dependiendo del DBMS existen diferentes momentos:

- Before
- Instead of
- After

TRIGGERS

Motivos para utilizar Triggers.

- Generar valores de columnas derivados.
- Prevenir operaciones inválidas.
- Forzar autorizaciones de seguridad.
- Forzar la integridad referencial entre diferentes BD.
- Implementar reglas de negocio.
- Proveer auditoría.
- Mantener replicas entre tablas.
- Obtener estadísticas de operaciones.
- Modificar datos cuando las DML son ejecutadas contra Vistas.

Cuidado con los triggers en cascada !!



Ejercicios?



“

1. Qué tipo de trigger utilizaría para realizar modificaciones sobre los datos de una Vista?
2. Qué tipo de trigger utilizaría para registrar auditoría sobre las modificaciones de los datos de una Tabla?
3. De ejemplos prácticos o de negocios de utilización de triggers