GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

UD 5. MODELO FÍSICO DML

Parte 2. Transacciones e índices

Bases de Datos (DAW/DAM)

CFGS Desarrollo de Aplicaciones Web (DAW)

CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)

Abelardo Martínez y Pau Miñana

Curso 2023-2024

Créditos

- Apuntes realizados por Abelardo Martínez y Pau Miñana.
- Basados y modificados de Sergio Badal (www.sergiobadal.com) y Raquel Torres.
- Las imágenes e iconos empleados están protegidos por la licencia LGPL y se han obtenido de:
  - https://commons.wikimedia.org/wiki/Crystal_Clear
  - https://www.openclipart.org

Contenidos

¿Qué veremos en esta parte?

- Transacciones
- Índices
- Soluciones boletín A: ejercicios sobre modelado físico DML

Contenidos

1. TRANSACCIONES
2. ÍNDICES
   1. Índices implícitos
   2. Índices explícitos
3. BIBLIOGRAFÍA

1. TRANSACCIONES

Concepto de transacción. Ejemplos

Una **transacción** está formada por una **serie de instrucciones DML** (INSERT, DELETE, UPDATE) **que pueden realizarse o no** dependiendo de cómo tengamos configurado nuestro SGBD.

Ejemplo Transacción 1

INSERT INTO ESTUDIANTES (‘12’, ‘PEDRO, ‘GARCÍA’, LÓPEZ’, ‘20/12/1978’);
INSERT INTO ESTUDIANTES (‘13’, ‘MARTA, ‘LÓPEZ’, LÓPEZ’, ‘22/10/1979’);
DELETE FROM ESTUDIANTES WHERE IDESTUDIANTE=13;

Ejemplo Transacción 2

DELETE FROM ESTUDIANTES WHERE IDESTUDIANTE=12;
INSERT INTO ESTUDIANTES (‘14’, ‘LUCÍA, ‘HERRERO’, LÓPEZ’, ‘02/11/1978’);

El concepto de transacción está muy relacionado con el mundo Oracle, por lo que solo lo veremos desde un punto de vista teórico (al menos en esta unidad).

¿Cuándo es definitiva una transacción?

Si tenemos el modo **autocommit activado**, cada instrucción será definitiva al momento de ejecutarla.

Ejemplo Transacción 1

INSERT INTO ESTUDIANTES (‘12’, ‘PEDRO, ‘GARCÍA’, LÓPEZ’, ‘20/12/1978’);
INSERT INTO ESTUDIANTES (‘13’, ‘MARTA, ‘LÓPEZ’, LÓPEZ’, ‘22/10/1979’);
DELETE FROM ESTUDIANTES WHERE IDESTUDIANTE=13;

Si tenemos el modo **autocommit desactivado**, las instrucciones serán definitivas (principalmente) cuando hagamos un **COMMIT** (confirmar) o un **ROLLBACK** (descartar):

Ejemplo Transacción 2

INSERT INTO ESTUDIANTES (‘12’, ‘PEDRO, ‘GARCÍA’, LÓPEZ’, ‘20/12/1978’);
INSERT INTO ESTUDIANTES (‘13’, ‘MARTA, ‘LÓPEZ’, LÓPEZ’, ‘22/10/1979’);
DELETE FROM ESTUDIANTES WHERE IDESTUDIANTE=13;
COMMIT;

Ejemplo Transacción 3

INSERT INTO ESTUDIANTES (‘16’, ‘LAILA’, ‘KEIZ’, ‘LULA’, ‘22/10/1979’);
DELETE FROM ESTUDIANTES WHERE IDESTUDIANTE=16;
ROLLBACK;

Inicio y fin de una transacción

Si tenemos el **modo autocommit desactivado**, una transacción **comienza** con la primera instrucción DML (INSERT, DELETE, UPDATE) que se ejecute y **finaliza** con alguna de estas circunstancias:

- Una operación COMMIT o ROLLBACK
- Una instrucción DDL (como ALTER TABLE por ejemplo)
- Una instrucción DCL (como GRANT)
- El usuario abandona la sesión
- Caída del sistema

AUTOCOMMIT ON
(por defecto)

AUTOCOMMIT OFF
(por defecto)

MySQL. Ejemplo

```
mysql> -- By default, in MySQL autocommit is turned ON
mysql> CREATE TABLE ESTUDIANTES (IDEST VARCHAR(2), NOMBRECOMP VARCHAR (20));
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ESTUDIANTES VALUES (12, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> -- Do another transaction with autocommit turned OFF
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ESTUDIANTES VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO ESTUDIANTES VALUES (20, 'Paul');
Query OK, 1 row affected (0.00 sec)
mysql> DELETE FROM ESTUDIANTES WHERE NOMBRECOMP = 'Heikki';
Query OK, 1 row affected (0.00 sec)
mysql> -- Now we undo those last 2 inserts and the delete.
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM ESTUDIANTES;
+------+--------+
| IDEST| NOMBRECOMP
|
+------+--------+
| 12 | Heikki |
+------+--------+
```

2. ÍNDICES

Índices. Tipos

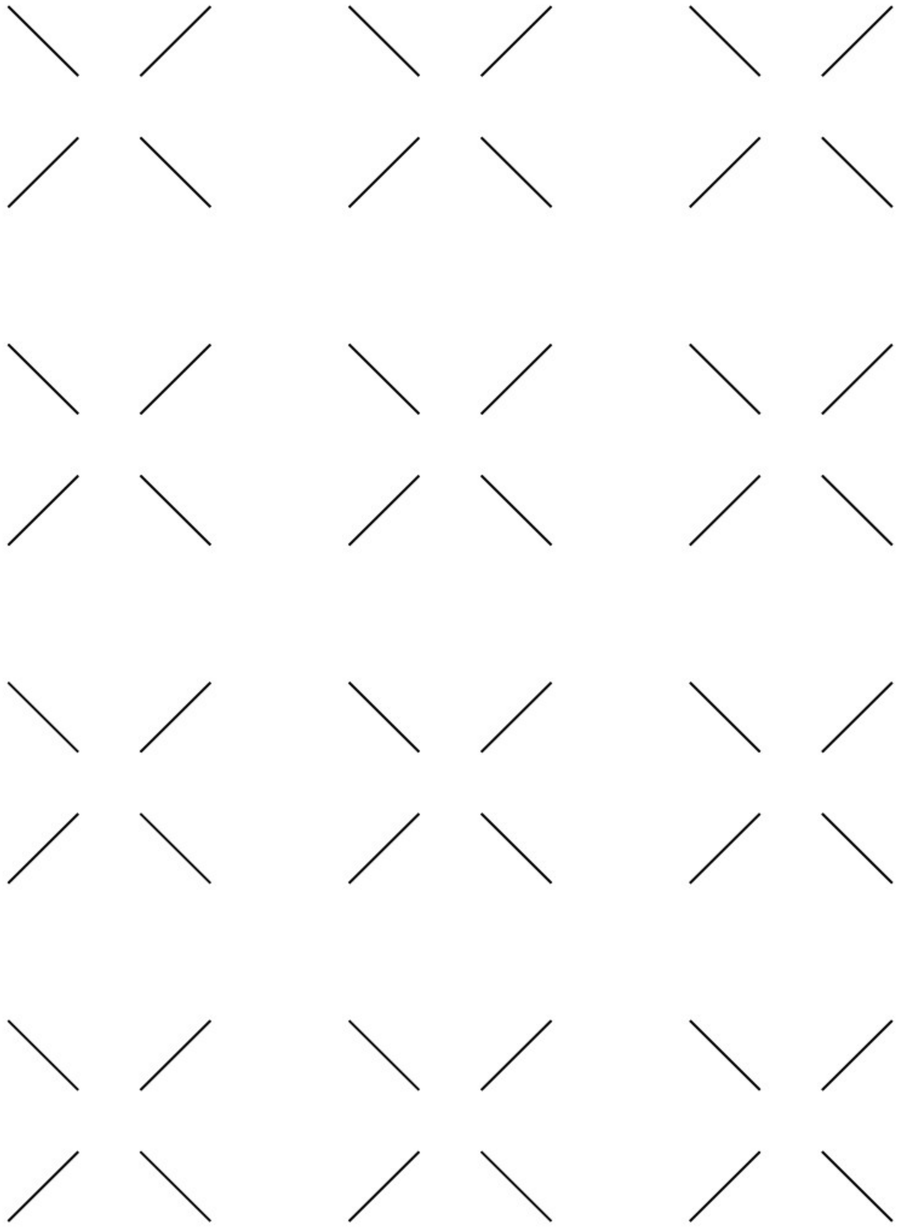
Los **índices** son un **objeto más de las bases de datos** -como pueden ser las tablas- que hacen que las bases de datos aceleren las operaciones de consulta y ordenación de los datos (DQL o SELECT).

De cada tabla, se puede crear uno o varios índices, siendo cada uno una copia de esa tabla ordenada en función de un campo o criterio concreto.

Hay 2 tipos de índices:

1) **Implícitos**. Se crean **AUTOMÁTICAMENTE**.
2) **Explícitos**. Son índices que podemos crear nosotros **MANUALMENTE**.

2.1 Índices implícitos



Ejemplo 1

Si creamos la tabla ESTUDIANTES con el **DNI como PK**, el **id de ciudad de residencia** y el **id de país de residencia como FKs** y el **numSS** (número de la seguridad social) **como UK** estamos, implícitamente, creando estos objetos en la base de datos:

- La tabla ESTUDIANTES
- Una copia de esa tabla ordenada por la PK (DNI)
- Una copia de esa tabla ordenada por la FK id_ciudad
- Una copia de esa tabla ordenada por la FK id_pais
- Una copia de esa tabla ordenada por la UK numSS

De esta manera, cada vez que hagamos una operación DML (INSERT, UPDATE, DELETE) sobre esa tabla tendremos que reconstruir todos esos objetos. Por ello, **los índices son recomendados solo cuando las operaciones DML son infinitamente menores que las de consulta (DQL o SELECT)**.

Ejemplo 2

¿Cuántos índices implícitos (automáticos) ves aquí?

```
mysql> show create table alpinistas;
+------------+-------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
| Table      | Create Table

                                                                           |
+------------+-------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
| alpinistas | CREATE TABLE `alpinistas` (
  `nif` char(9) NOT NULL,
  `nombre` varchar(30) DEFAULT NULL,
  `fecha_nacimiento` date DEFAULT NULL,
  `codigo` mediumint DEFAULT NULL,
  `fecha_ingreso` date DEFAULT NULL,
  PRIMARY KEY (`nif`),
  KEY `codigo_asociacion_fk` (`codigo`),
  CONSTRAINT `codigo_asociacion_fk` FOREIGN KEY (`codigo`) REFERENCES `asociacion` (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+------------+-------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
1 row in set (0,00 sec)
```

Ejemplo 2 (solución)

¿Cuántos índices implícitos (automáticos) ves aquí?

```
mysql> show create table alpinistas;
+------------+-----------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------------------------------+
| Table      | Create Table

                                                           |
+------------+-----------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------------------------------+
| alpinistas | CREATE TABLE `alpinistas` (
  `nif` char(9) NOT NULL,
  `nombre` varchar(30) DEFAULT NULL,
  `fecha_nacimiento` date DEFAULT NULL,
  `codigo` mediumint DEFAULT NULL,
  `fecha_ingreso` date DEFAULT NULL,
  PRIMARY KEY (`nif`),
  KEY `codigo_asociacion_fk` (`codigo`),
  CONSTRAINT `codigo_asociacion_fk` FOREIGN KEY (`codigo`) REFERENCES `asociacion` (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+------------+-----------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------------------------------+
1 row in set (0,00 sec)
```

Por cada FOREIGN KEY se crea una KEY (clave interna), por lo que tenemos **DOS ÍNDICES IMPLÍCITOS**

Ejemplo 3

¿Cuántos índices implícitos (automáticos) ves aquí?

```
mysql> show create table asociacion;
+------------+-----------------------------------------------------
-----------------------------------------------------+
| Table      | Create Table
                                                     |
+------------+-----------------------------------------------------
-----------------------------------------------------+
| asociacion | CREATE TABLE `asociacion` (
  `codigo` mediumint NOT NULL,
  `nombre` varchar(30) DEFAULT NULL,
  `ubicacion` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+------------+-----------------------------------------------------
-----------------------------------------------------+
1 row in set (0,00 sec)
```

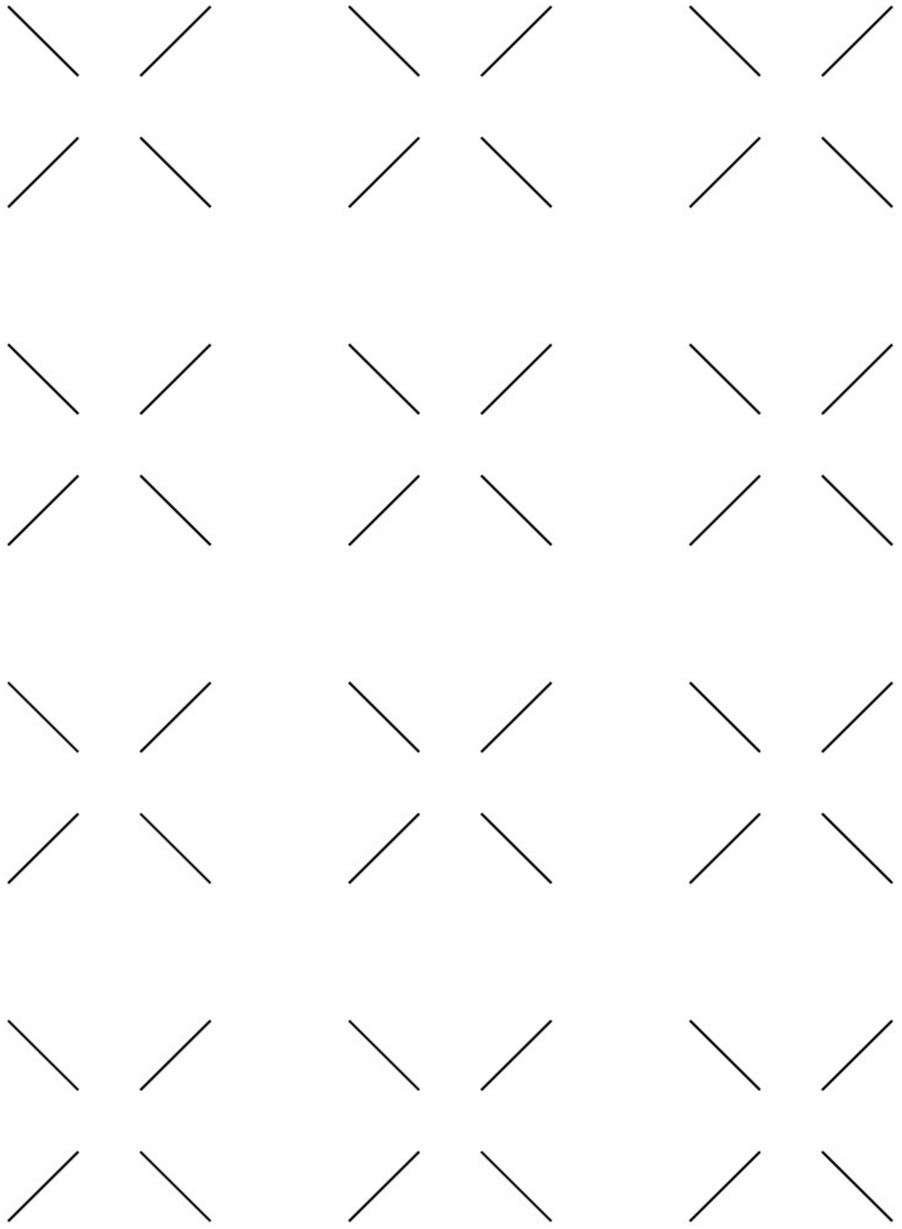
Ejemplo 3 (solución)

¿Cuántos índices implícitos (automáticos) ves aquí?

```
mysql> show create table asociacion;
+------------+-----------------------------------------------------
------------------------------------------------------------+
| Table      | Create Table
                                                            |
+------------+-----------------------------------------------------
------------------------------------------------------------+
| asociacion | CREATE TABLE `asociacion` (
  `codigo` mediumint NOT NULL,
  `nombre` varchar(30) DEFAULT NULL,
  `ubicacion` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+------------+-----------------------------------------------------
------------------------------------------------------------+
1 row in set (0,00 sec)
```

Tenemos **UN ÍNDICE IMPLÍCITO**

2.2 Índices explícitos



MySQL. Creación y borrado de índices explícitos

Se pueden **crear índices** de forma explícita (manual). Características:

- Se crean para aquellos campos sobre los cuales se realizarán búsquedas e instrucciones de ordenación frecuente.
- Se almacenan aparte de la tabla a la que hace referencia, lo que permite crearlos y borrarlos en cualquier momento.

Ésta es su sintaxis:

```
CREATE INDEX nombre
ON tabla (columna1 [,columna2...])
```

Para **mostrar** los **índices** de una tabla usamos el comando SHOW INDEXES FROM ESTUDIANTES;

Para **eliminar un índice** concreto usamos el comando DROP INDEX IX_FECHA ON ESTUDIANTES;

Ejemplos

- Puedo tener una tabla de ESTUDIANTES con los campos DNI, nombre, apellidos y fecha de nacimiento con un índice sobre el campo fecha de nacimiento, que me permita consultar rápidamente todos los estudiantes ordenados por ese campo.

```
CREATE INDEX ix_fecha
ON ESTUDIANTES (fecha_nacimiento);
```

- Otro índice EXPLÍCITO (manual) podría ser el siguiente: para los campos apellido1, apellido2 y nombre. Esto no es lo mismo que crear un índice para cada campo. Este índice es efectivo solo cuando se buscan u ordenan clientes usando los tres campos (apellido1, apellido2 y nombre) a la vez.

```
CREATE INDEX ix_nombre_completo
ON ESTUDIANTES (apellido1, apellido2, nombre);
```

Si realizamos una consulta sobre esa tabla ordenada por esos campos y en ese orden, la respuesta será de milisegundos **aunque la tabla tenga millones de registros**.

Ejemplos

```
mysql> create index ix_fecha on alpinistas(fecha_nacimiento desc);
Query OK, 0 rows affected (0,03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table alpinistas;
+------------+------------------------------------------------------------------------
--------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
| Table      | Create Table


+------------+------------------------------------------------------------------------
--------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
| alpinistas | CREATE TABLE `alpinistas` (
  `nif` char(9) NOT NULL,
  `nombre` varchar(30) DEFAULT NULL,
  `fecha_nacimiento` date DEFAULT NULL,
  `codigo` mediumint DEFAULT NULL,
  `fecha_ingreso` date DEFAULT NULL,
  PRIMARY KEY (`nif`),
  KEY `codigo_asociacion_fk` (`codigo`),
  KEY `ix_fecha` (`fecha_nacimiento` DESC),
  CONSTRAINT `codigo_asociacion_fk` FOREIGN KEY (`codigo`) REFERENCES `asociacion` (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+------------+------------------------------------------------------------------------
--------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
1 row in set (0,00 sec)

mysql> select * from alpinistas order by fecha_nacimiento desc;
```

Creo un índice sobre un campo por el que voy a ordenar muy frecuentemente.

Índices. Ventajas e inconvenientes

Ventajas:

- Los índices permiten optimizar las consultas sobre los elementos de la base de datos, **especialmente en tablas de gran tamaño**.
- Los índices **mejoran el tiempo de recuperación de los datos en las consultas** realizadas contra nuestra base de datos. Las búsquedas serán más rápidas, ya que evita la sobrecarga de la CPU y de disco.

Inconvenientes:

- La creación de índices implica -generalmente- un **aumento en el tiempo de ejecución sobre aquellas consultas de inserción, actualización y eliminación** realizadas sobre los datos afectados por el índice (ya que tendrán que actualizarlo).
- Los índices **necesitan un espacio para almacenarse**, por lo que también tienen un coste adicional en forma de espacio en disco.
- Crear índices en exceso resultará en un bajo rendimiento de las consultas.

¿Cuándo crear índices?

Se aconseja crear índices en campos que:

- Contengan una gran cantidad de valores
- Contengan una gran cantidad de nulos
- Sean parte habitual de cláusulas WHERE, GROUP BY u ORDER BY (que veremos más adelante)
- Sean parte de consultas muy frecuentes

No se aconseja en campos que:

- Pertenezcan a tablas pequeñas, ya que no aportan una mejora significativa
- No se usen a menudo en las consultas
- Como norma general, pertenezcan a tablas que se actualicen frecuentemente (operaciones DML) → **EXCEPCIONES**: aún así puede que se necesite porque una consulta utilice mucho ciertos campos (por ejemplo en una tabla de pedidos)
- Se utilizan en expresiones

3. BIBLIOGRAFÍA

Recursos

- MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/
- Oracle Database Documentation. https://docs.oracle.com/en/database/oracle/oracle-database/index.html
- MySQL Tutorial. https://www.w3schools.com/mysql/
- GURU99. Tutorial de MySQL para principiantes Aprende en 7 días. https://guru99.es/sql/
- Manual de SQL (Oracle SQL). Creación de otros objetos con DDL. Índices. https://jorgesanchez.net/manuales/sql/ddl-otros-sql2016.html
- Adictos al trabajo. Introducción a índices en MySQL. https://www.adictosaltrabajo.com/2015/09/11/introduccion-a-indices-en-mysql/