

Bases de Datos 1

Alejandra Lliteras

alejandra.lliteras@lifa.info.unlp.edu.ar



DBMS Relacional

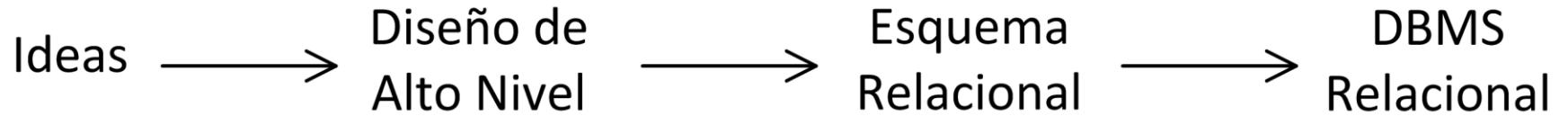


Figura extraída de:
García-Molina, H. (2008). *Database systems: the complete book*. Pearson Education India.

► DBMS

- Un sistema manejador de bases de datos (**Data Base Managment System**) es una herramienta para crear y manejar grandes volúmenes de datos de manera eficiente permitiendo persistirlos de manera segura y por largos períodos de tiempo

▶ DBMS

- Debe permitir
 - Crear nuevas bases de datos y sus esquemas
 - Lenguaje de definición de datos (DDL)
 - Consultar los datos y modificarlos
 - Lenguaje de consultas y manipulación de datos (DML)

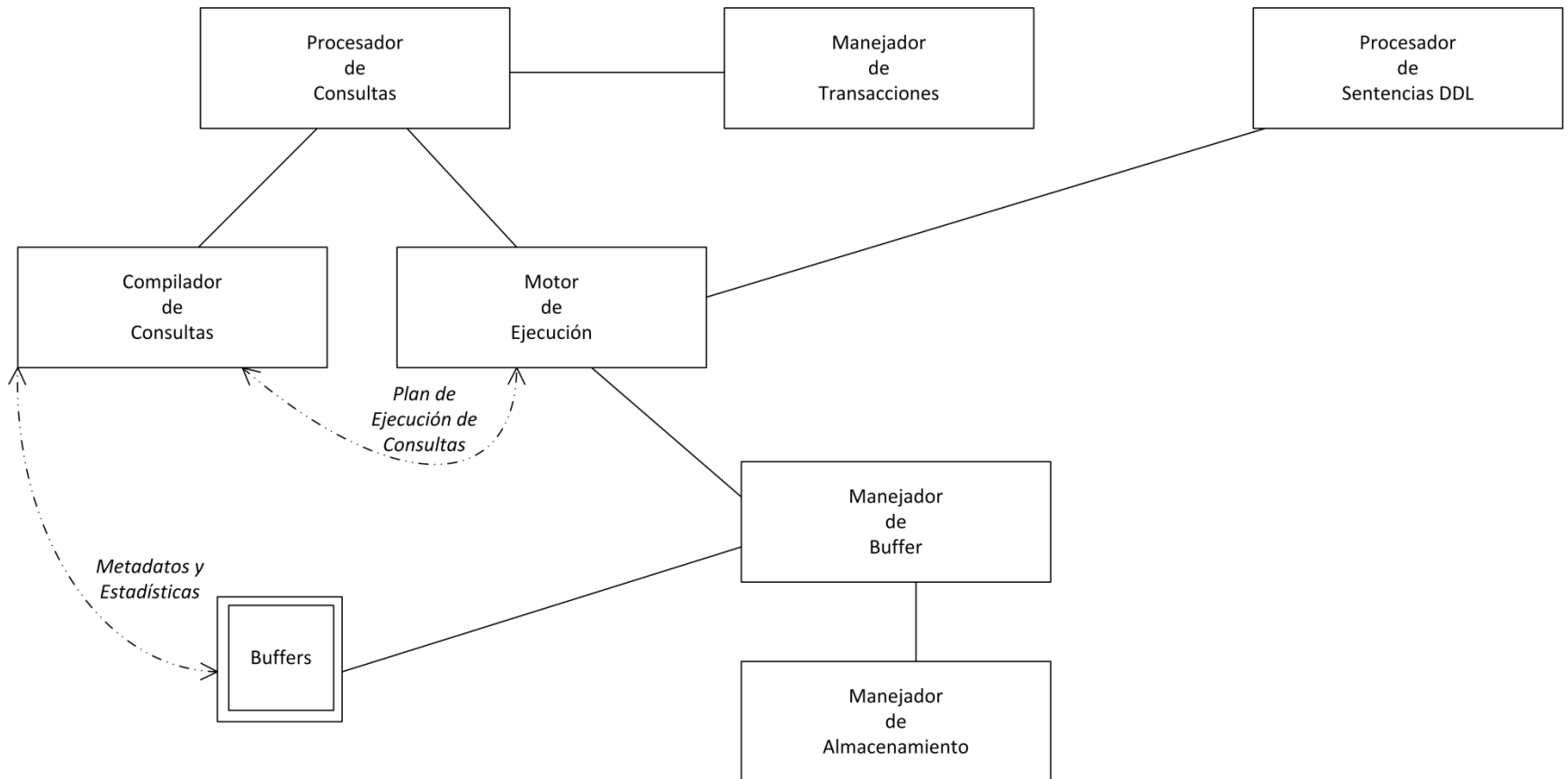
▶ DBMS

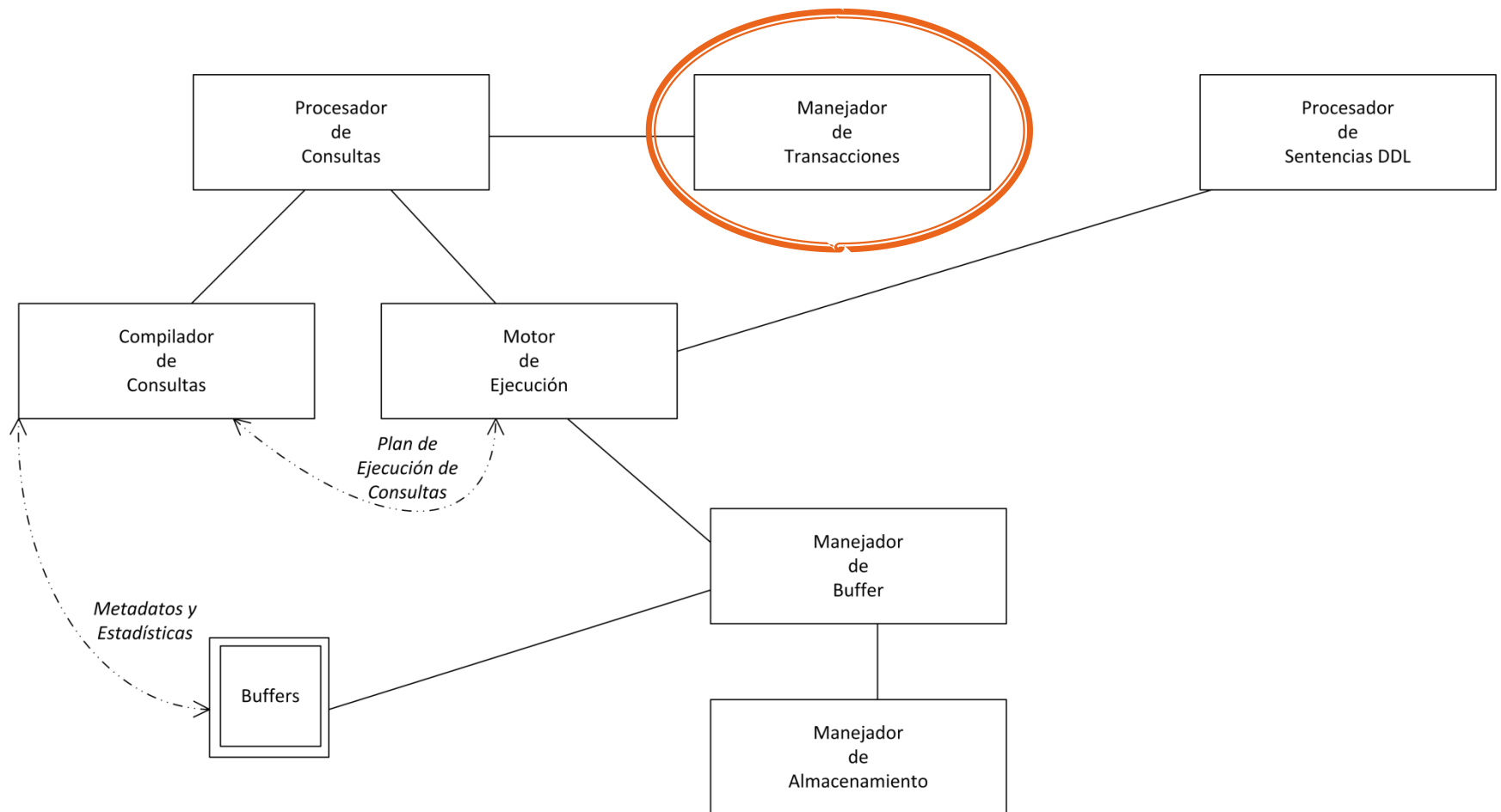
- Debe permitir (continuación)
 - Permitir el almacenamiento de grandes volúmenes de datos, por largos períodos de tiempo
 - Permitir el acceso eficiente a los datos (para consultas y modificaciones)
 - Permitir la recuperación de los datos ante fallos (Durabilidad)
 - Control de acceso de múltiples usuarios, sin que éstos tengan interacciones indeseadas (Aislamiento)
 - Acciones sobre los datos que se realizan de manera completa (Atomicidad)

▶ DBMS

- Es usado por
 - Administradores del sistema
 - Crean bases de datos y esquemas
 - Usuarios convencionales
 - que consultan los datos o los modifican

► Esquemático simplificado de un DBMS





Transacciones

Definición de transacción

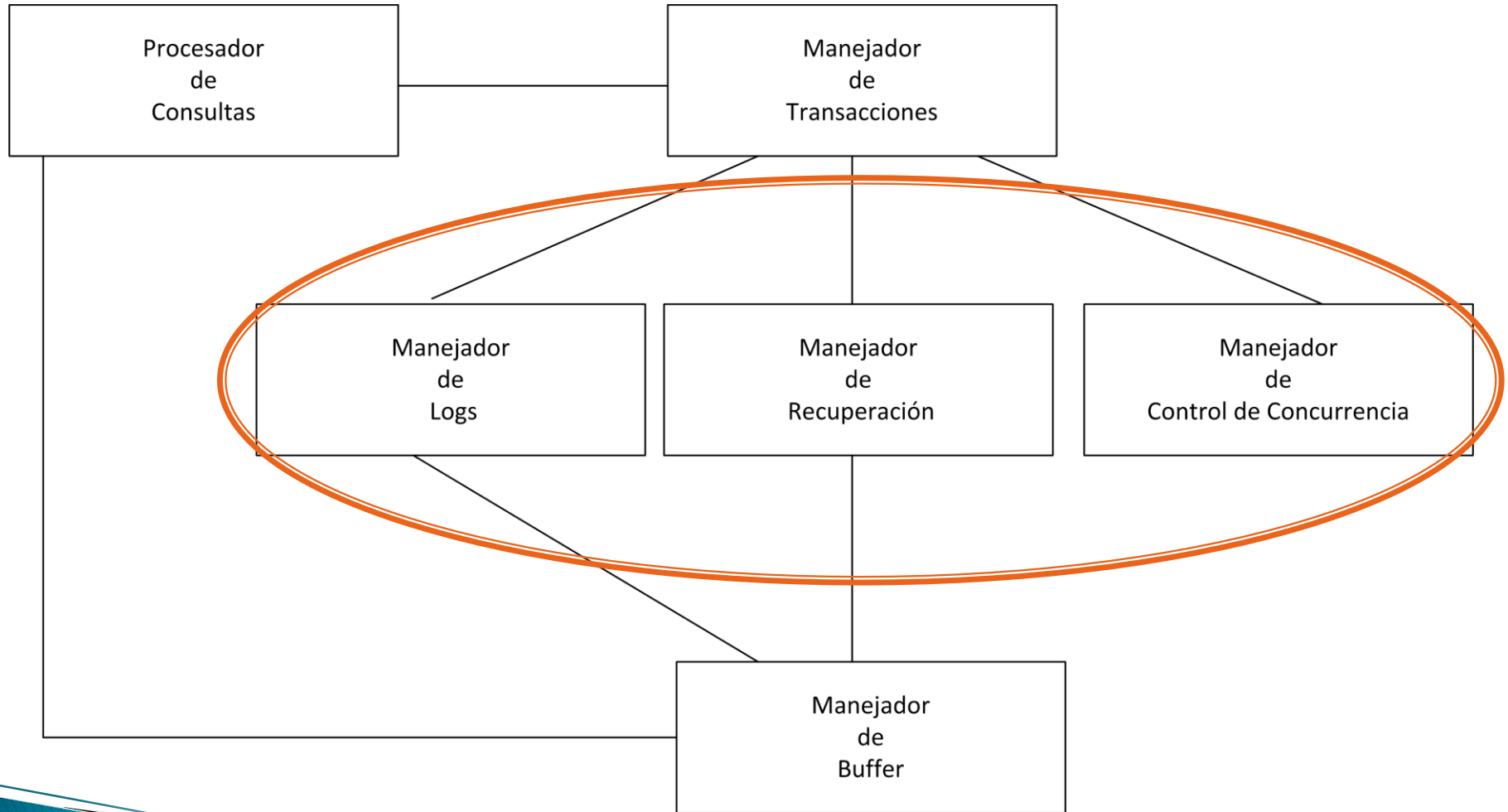
- ▶ Es una unidad de ejecución de un programa que accede y posiblemente modifica datos
- ▶ Es una colección de operaciones que forman una única unidad lógica de trabajo
- ▶ Tiene un **inicio** y un **fin** definido
 - La transacción consiste de cada una de las sentencias que se ejecutan entre el inicio de la transacción y su fin.

Transacciones

Un DBMS debe asegurar:

- ▶ que la ejecución de las transacciones se realice adecuadamente a pesar de la existencia de fallos:
 - o se ejecuta la transacción completa o no se ejecuta nada
- ▶ Debe asegurar la concurrencia de las transacciones

Transacciones



Transacciones

Un DBMS debe asegurar:

- ▶ Que la ejecución de las transacciones se realice adecuadamente a pesar de la existencia de fallos:
 - o se ejecuta la transacción completa o no se ejecuta nada
- ▶ Debe asegurar la concurrencia de las transacciones

Transacciones

Propiedades **ACID**

- ▶ **Atomicidad (Atomicity):**
 - o todas las operaciones de una transacción se realiza o ninguna de ellas, lo hace.
- ▶ **Consistencia (Consistency):**
 - La ejecución aislada de la transacción, conserva la consistencia de la base de datos
- ▶ **Aislamiento (Isolation):**
 - Cuando dos o mas transacciones se ejecutan concurrentemente, cada una de ellas ignora al resto
- ▶ **Durabilidad (Durability):**
 - cuando termina una transacción exitosamente, los cambios quedan en la base de datos aun cuando haya fallos en el sistema

Transacciones

Estados de una transacción

▶ **Activa:**

- Es el estado inicial, la transacción permanece en este estado mientras se ejecuta

▶ **Parcialmente comprometida:**

- Después de ejecutarse la última operación

▶ **Fallida:**

- Luego de darse cuenta que no puede continuar con la ejecución normal

▶ **Abortada:**

- Después de haber retrocedido la transacción y restablecido la base de datos a su estado anterior al comienzo de la transacción

▶ **Comprometida:**

- Al completarse con éxito

Transacciones

Diagrama de transición de estados

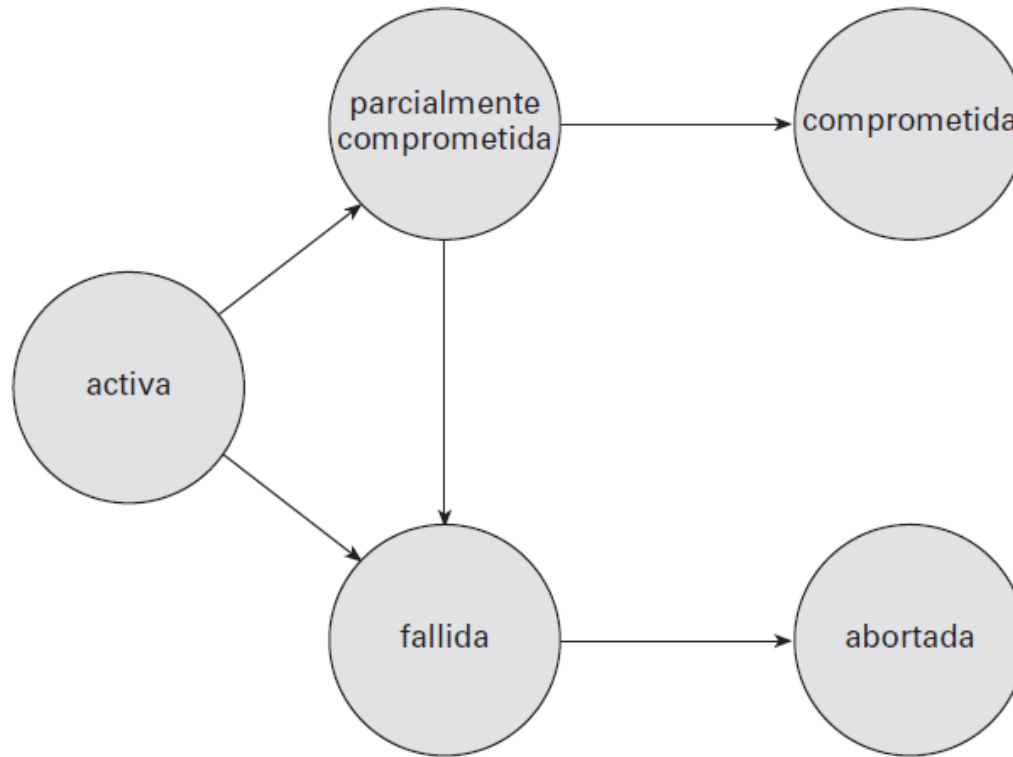
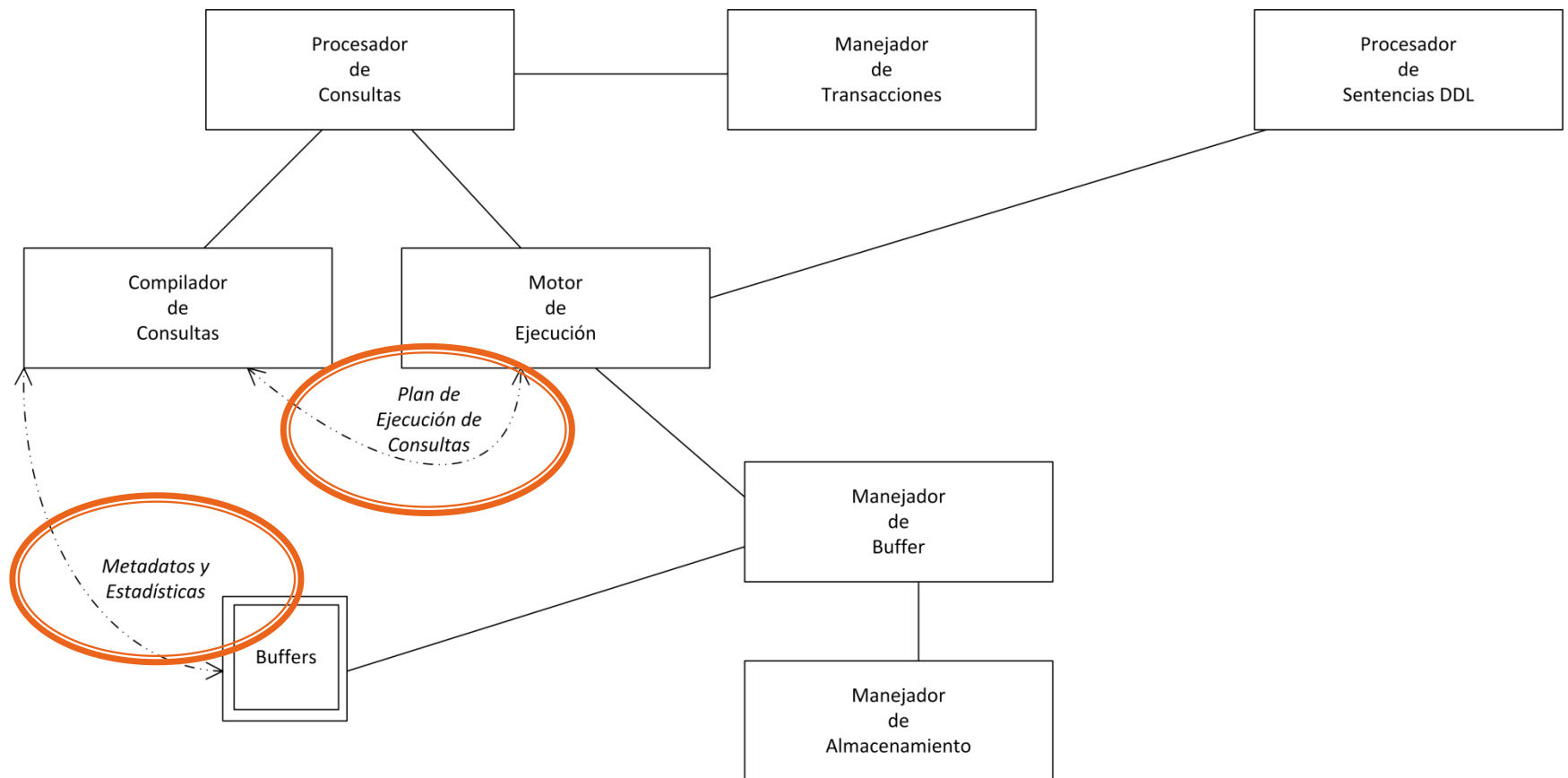


Figura extraída de:
Elmasri, R., & Navathe, S. B. (2007). Fundamentos de sistemas de bases de datos.

Transacciones

Una transacción **ha sido**

- **abortada**, cuando llega al estado abortada
- **comprometida**, cuando llega al estado comprometida
- **terminada**, cuando se encuentra en estado comprometida o abortada



Procesador de consultas

► Plan de ejecución de consulta

- El plan de ejecución de consulta, surge a partir del análisis y la optimización que se realiza en el compilador de consultas, donde se emplean las estadísticas que genera el DBMS
 - **Estadísticas:** información generada y almacenada por el DBMS sobre las propiedades de los datos

MySQL

MySQL

▶ DBMS relacional y Open Source

- 2008: SUN Microsystem
- 2010: Desarrollado, distribuido y mantenido por ORACLE Corporation

▶ Algunas características

- Portabilidad y algunos aspectos internos
 - Escrito en C y C++
 - Corre en diferentes plataformas
(<http://www.mysql.com/support/supportedplatforms/database.html>)
 - Motor de almacenamiento transaccional y no transaccional
- Seguridad
 - Sistema de contraseña y privilegios
 - Encriptación de contraseña
- Escalabilidad
 - Usada en grandes bases de datos (50 millones de registros)
 - Hasta 64 índices por tabla. Cada índice de 1 a 16 columnas
- Conectividad
 - Mediante diversos protocolos
 - TCP IP

MySQL

▶ DBMS relacional y Open Source

- Desarrollado, distribuido y mantenido por ORACLE Corporation

▶ Algunas características


- Se opera por línea de comando o bien por alguna interface gráfica (por ejemplo: **mySQL Workbench**, **PHP Admin**, entre otros)
- Es parte de la arquitectura LAMP
 - Sistemas de infraestructura de internet que usa las Herramientas: **Linux** (como SO), **Apache** (como servidor web), **MySQL/MariaDB** (gestor de BBDD) y **Perl**, **PHP**, o **Python** (lenguaje de programación)

MySQL

► Versiones

- Con licencia GPL¹ (General Public License)
- Comercial

¹ *Garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.*



MySQL

- ▶ Permite múltiples formatos de almacenamiento

Por ejemplo:

Engine	Support	Comment	Transactions
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO
MyISAM	YES	MyISAM storage engine	NO
CSV	YES	CSV storage engine	NO

Para ver todos los formatos soportados por la versión de Mysql, ejecutar el comando

SHOW ENGINES;

MySQL

- ▶ Los múltiples formatos de almacenamiento son administrados por Motores de almacenamiento
 - Son componentes de MySql que manejan las operaciones de SQL para los diferentes tipos de tablas.
 - **InnoDB** es el componente de propósito más general.
 - Oracle recomienda su uso excepto para tablas muy especializadas.
 - La sentencia “CREATETABLE” por defecto crea tablas con InnoDB.
 - MySQL utiliza una arquitectura de storage engines conectable, lo que permita cargar y/o descargar nuevos engines en un servidor que se está ejecutando.

MySQL

► InnoDB

- **Soporta transacciones**
- Realiza el **bloqueo a nivel de fila** de una tabla
- Permite definir claves foráneas
- Integridad de datos
 - Transacciones
 - Restricciones entre las relaciones
- Es recomendable para tablas con alta cantidad de inserts y deletes, debido al bloqueo a nivel de fila

MySQL

► MyISAM

- No soporta transacciones
- Hace bloqueo a nivel de tabla
 - Lo que la hace lenta para tablas con muchos inserts/deletes
- No posee restricción de clave foránea
- Es beneficioso para tablas con alto nivel de lecturas

MySQL

Algunos elementos del lenguaje



Algunos elementos del lenguaje

- ▶ Tablas
 - Clave primaria
 - Clave foránea
- ▶ Stored procedures
 - Parámetros
 - Estructuras de control
 - Cursores
 - Función last_insert_id
- ▶ Trigger
- ▶ Vistas

Stored Procedures

- Conjunto de sentencias SQL que se almacenan el servidor
- Se lo crea indicando
 - Nombre
 - Parámetros (de ser necesario)
 - IN, OUT, INOUT
 - Cuerpo del Procedimiento almacenado
- Se lo invoca mediante
 - `CALL nombreSP`
- Eliminación
 - Al borrar toda la base, los stored procedures se eliminan
 - Drop
 - `DROP PROCEDURE IF EXISTS nombreSP`

Stored Procedures

- **Acceso homogéneo:**
 - Cuando es necesario realizar las mismas operaciones en la base de datos desde diferentes aplicaciones clientes, escritas en diferentes lenguajes y quizás hasta en diferentes plataformas.
- **Asegurar consistencia en las operaciones**
 - Escribiendo una vez y asegurando para cada ejecución que la secuencia de instrucciones retornara de manera consistente los resultados. Puede usarse para no permitir acceder a tablas directamente, sino que solo por medio de estas rutinas
- **Pueden ayudar a la performance**
 - porque se disminuye el tráfico entre el servidor y el cliente
 - Sin embargo, incrementa la actividad en el servidor de la bases de datos ya que el trabajo se ejecuta en el
 - Esto empeora si múltiples cliente peticionan a un solo servidor
- **Consultas complejas**
 - Es posible usar estructuras y funciones adicionales para resolver consultas complejas dentro de un sp

Stored Procedures

- ▶ Privilegios que tiene que tener el usuario para:
 - Crearlo
 - CREATE ROUTINE
 - Modificarlo
 - ALTER ROUTINE
 - Ejecutarlo
 - EXECUTE

SP –Parámetro IN

- ▶ Son parámetros de entrada
- ▶ Se puede usar y modificar su valor dentro del SP, pero los cambios no se verán reflejados fuera de este

```
1  delimiter //
2  •  create procedure recuperarEmpleado (in nroEmpleadoParam int)
3  begin
4  select * from empleado where nroEmpleado = nroEmpleadoParam;
5  end//
6
7
```

SP –Parámetro OUT

- ▶ Son parámetros de salida
- ▶ Se puede asignar un valor dentro del SP, y usarlo dentro del mismo. Los cambios se verán reflejados fuera del SP

```
DROP procedure IF EXISTS `pruebajson`.`parametrosSalida2`;  
  
DELIMITER //  
CREATE PROCEDURE `pruebajson`.`parametrosSalida2` (out valor int)  
BEGIN  
    set valor = 6 *2;  
END//  
  
delimiter ;
```

SP –Parámetro OUT

- ▶ Son parámetros de salida
- ▶ Se puede asignar un valor dentro del SP, y usarlo dentro del mismo. Los cambios se verán reflejados fuera del SP

```
DELIMITER //
```

```
CREATE PROCEDURE `pruebajson`.`parametrosSalida` (out valor int)
```

```
BEGIN
```

```
select count(*) into valor from empleado;
```

```
END//
```

```
delimiter ;|
```

SP –Parámetro INOUT

- ▶ Son parámetros de entrada salida
- ▶ Se puede usar y modificar su valor dentro del SP, y los cambios se verán reflejados fuera de este

```
DELIMITER //
```

```
CREATE PROCEDURE `pruebajson`.`parametrosEntradaSalida` (inout valor int)
```

```
BEGIN
```

```
set valor = valor *2;
```

```
END//
```

```
delimiter ;
```

SP – Función LAST_INSERT_ID

- ▶ Retorna el valor del último autoincremental agregado inmediatamente anterior a su invocación
- ▶ Si la inserción es errónea, el valor que retorna es indefinido

Sp- Estructuras de control

```
delimiter //
CREATE PROCEDURE estructuraDeControl()
BEGIN
    DECLARE cantidadDeIteraciones INT; -- declaracion de un a
    SET cantidadDeIteraciones = 0; -- variable cantidadDeIter
    loop_label: LOOP -- el loop se indica con la palabra clav

        SET cantidadDeIteraciones = cantidadDeIteraciones + 1;
        IF cantidadDeIteraciones = 5 THEN
            ITERATE loop_label; -- si iteró 5 veces, se detiene
        END IF; -- del if que controla cantidadDeIteraciones =

        SELECT cantidadDeIteraciones;

        IF cantidadDeIteraciones >= 6 THEN
            LEAVE loop_label; -- termina el ciclo loop
        END IF; -- fin del if que controla cantidadDeIteracion
    END LOOP; -- fin del loop
END; // -- fin del sp

call estructuraDeControl();
```


SP-Cursores

- ▶ Permiten guardar en ellos valores obtenidos de ejecutar una sentencia SQL
- ▶ Es posible recorrerlos e ir recuperando de a uno sus valores
- ▶ Operaciones:
 - **DECLARE:** permite declarar un cursor
 - **OPEN:** permite abrir un cursor que ya ha sido declarado
 - **FETCH:** permite recuperar el valor de un cursor que ya ha sido abierto previamente
 - **CLOSE:** permite cerrar un cursor que ha sido al menos, declarado previamente.

TRIGGER

- ▶ Son disparadores que se pueden usar
 - BEFORE (antes de)
 - AFTER (después de)

De las siguientes operaciones

- INSERT
 - DELETE
 - UPDATE
- 

TRIGGER

```
CREATE TRIGGER nombre_disp momento_disp evento_disp  
ON nombre_tabla FOR EACH ROW sentencia_disp
```

► Donde

- nombre_disp es el nombre que se le asigna
- momento_disp: BEFORE o AFTER
- evento_disparador: INSERT-DELETE-UPDATE
- nombre_tabla: tabla sobre la cual se generará el evento
- sentencia_disp: conjunto de sentencias del cuerpo del trigger

TRIGGER

- ▶ Cuando el evento disparador es **UPDATE**, se puede usar:
 - **OLD.nombreColumna** (hace referencia al valor **antes** de actualizarse de la columna en cuestión)
 - **NEW.nombreColumna** (hace referencia al valor **después** de actualizarse de la columna en cuestión)

TRIGGER

- ▶ Cuando el evento disparador es **INSERT**, se puede usar:
 - NEW.nombreColumna
- ▶ Cuando el evento disparador es **DELETE**, se puede usar:
 - OLD.nombreColumna

TRIGGER

- ▶ Cómo eliminar un trigger
 - DROP TRIGGER nombreTrigger
 - Al eliminar una tabla, todos los triggers asociados son eliminados
- ▶ ¿Qué privilegios debe tener un usuario para crear triggers?

```
Grant TRIGGER on pruebajson.* to 'usuarioAdministrativo'@'localhost';
```

- ▶ Una columna OLD es de solo lectura y requiere privilegios de SELECT
- ▶ Una columna de NEW requiere privilegio de SELECT, pero además, es posible modificarla si se usa con BEFORE, para ello es necesario además privilegio de UPDATE

TRIGGER

► Limitaciones

- No pueden ejecutar un stored procedure (call)
- No pueden usar sentencias que explicita o implícitamente abran o cierren una transacción

Vistas

- Habitualmente el modelo lógico suele ser muy complejo para el usuario.
- Existen consideraciones de seguridad para que un usuario no acceda al todo el modelo.
- En ocasiones es mejor darle al usuario una “versión” personalizada del modelo que se ajuste mejor a sus necesidades de consulta.
- No se debe permitir al usuario realizar operaciones sobre los datos (insert/update/delete).

Vistas

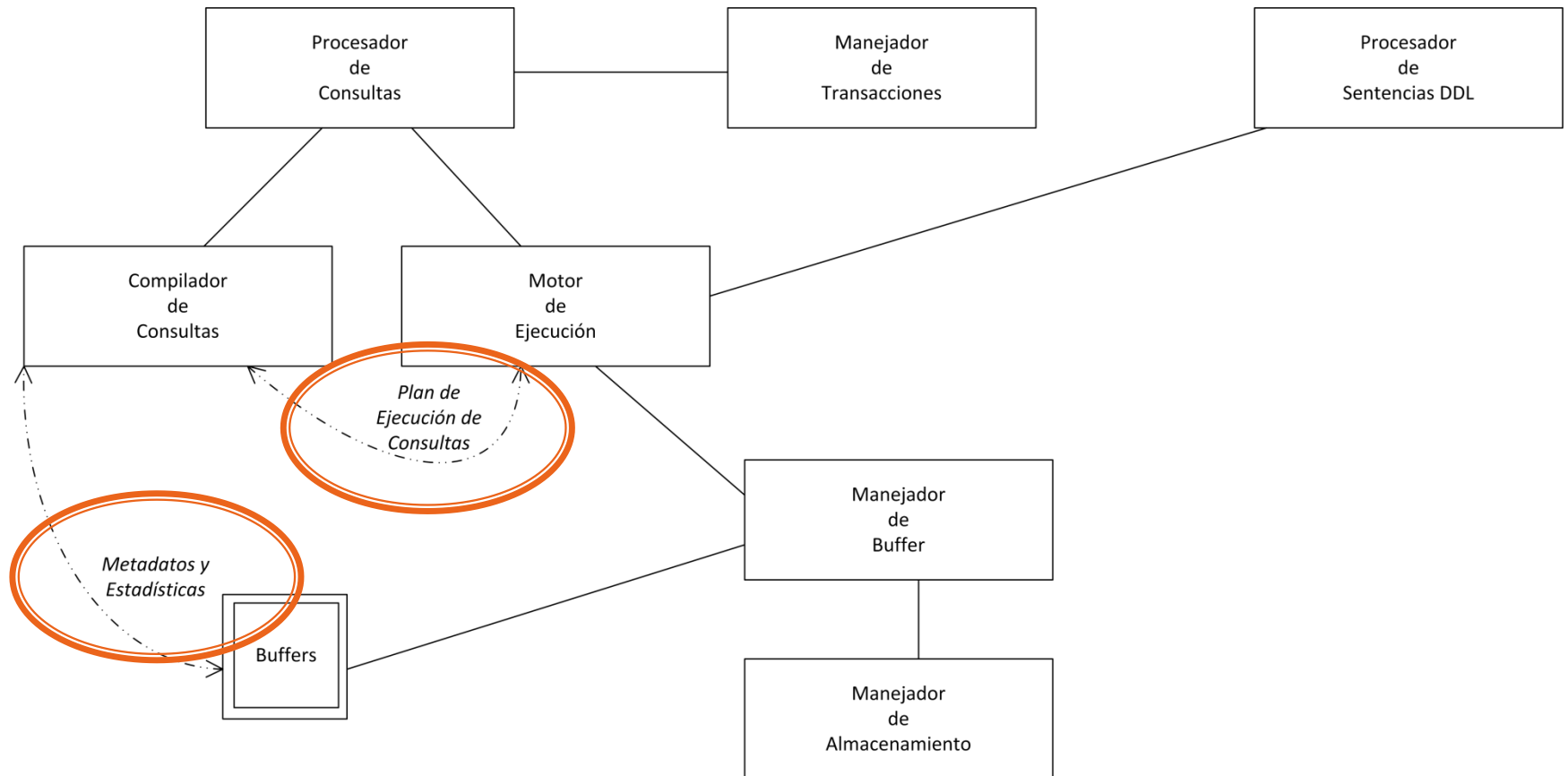
- Una vez creada, la definición de una vista es “congelada”. Esto significa que cambios posteriores a las tablas de la vista no afectarán la vista.
 - Por ejemplo, si luego de crear una vista con `SELECT *`, si luego se agregan nuevas columnas, éstas no aparecerán en la vista.
- Las vistas pertenecen a una base de datos, por lo que si se elimina la base, se elimina la vista.
- Los nombres de las columnas deben ser únicos.

Vistas

- No se pueden utilizar subqueries en la cláusula FROM.
- El SELECT no puede referirse a variables del usuario o del sistema.
- Si se crea dentro de un programa (sp) no se pueden utilizar los parámetros del programa.
- Cuando se está definiendo la vista, todas las tablas y/o otras vistas a las que se menciona deben existir.
- No se pueden utilizar tablas temporales ni crear vistas temporales.
- No se pueden asociar triggers con las vistas.
- Se pueden utilizar ORDER BY, pero se lo ignora si el select viene acompañado de uno propio.

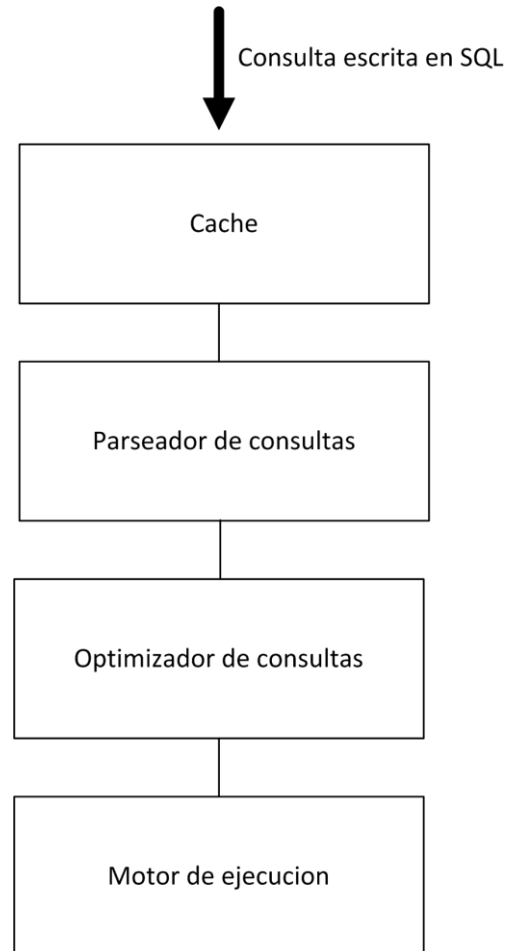
Vimos que...

Plan de Ejecución



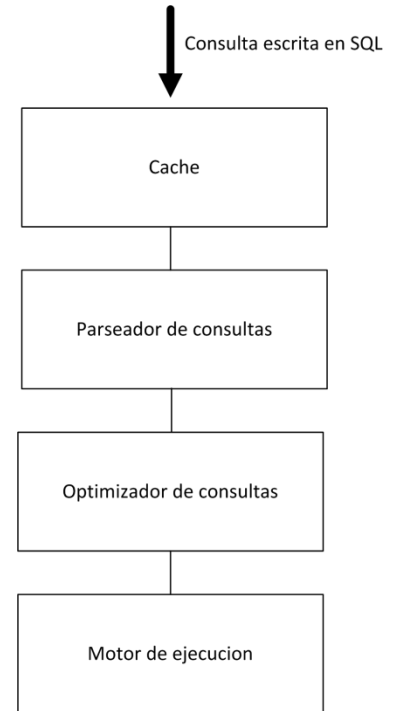
MySQL

Procesador de consultas



Procesador de consultas

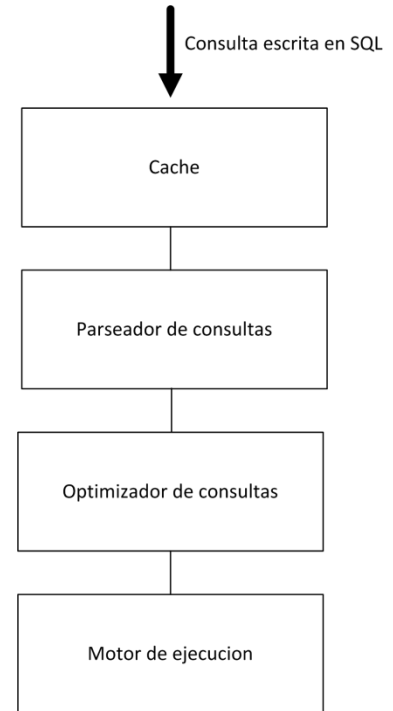
- ▶ Ante una consulta que llega al procesador
 - ▶ Si esta en la caché, entonces retorna los resultados de la misma
 - ▶ Si no esta en la caché
 - ▶ La parsea
 - ▶ Intenta optimizarla
 - ▶ La Ejecuta



Procesador de consultas

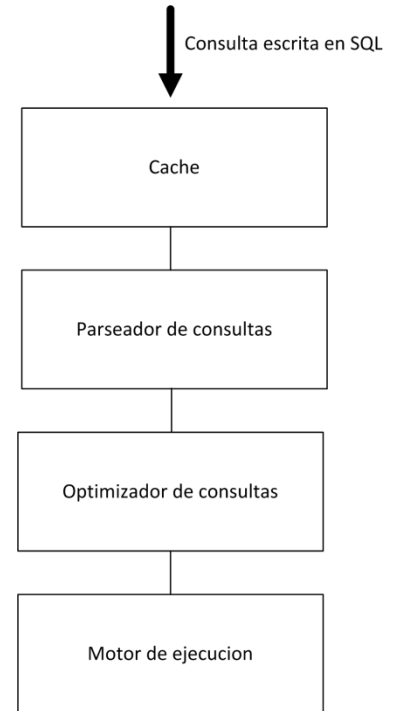
► Cache

- Se habilita o deshabilita su uso
- En caso de estar habilitado su uso, las operaciones de insert, delete y update, borran la cache



Procesador de consultas

- ▶ Optimizador
 - ▶ Usa el plan de ejecución
 - ▶ Contempla las estadísticas de las tablas involucradas



Optimización de consultas



Optimización de consultas

- ▶ Con el uso de las aplicaciones que emplean bases de datos relacionales:
 - La cantidad de tuplas en las tablas crecen
 - La cantidad de usuarios que acceden a la misma se incrementa

Esto puede conllevar a la degradación en la performance de las consultas

Análisis de consultas

- ▶ Usando el comando EXPLAIN para analizar el plan de ejecución

```
1 | EXPLAIN SELECT * FROM categoriesG
```

```
***** 1. row *****
```

```
      id: 1
  select_type: SIMPLE
        table: categories
         type: ALL
possible_keys: NULL
         key: NULL
        key_len: NULL
         ref: NULL
         rows: 4
       Extra:
1 row in set (0.00 sec)
```

- **Información obtenida a partir del comando “explain”**

- **Id:** es un identificador secuencial para cada uno de los select del query.
- **Select_type:** indica el tipo de select
 - **Simple:** se trata de un simple select.
 - **Primary:** es el select más externo.
 - **Derived:** el select es parte de un subquery en el from.
 - **Subquery:** es el primer select en un subquery.
 - **Dependent subquery:** el select es un subquery que depende de uno más externo.
 - **Uncacheable subquery:** el subquery no se almacena en la caché.
 - **Union:** el select es la segunda parte de un union.
 - **Dependent union:** es la segunda parte de un union que depende de un query más externo.
 - **Union result:** el select es el resultado de un union.

- **Información obtenida a partir del comando “explain”**

- Table: es la tabla a la que se hace referencia.
- Type: indica como une las tablas MySQL
 - System: la tabla tiene 0 o 1 sola fila.
 - Const: la tabla tiene una sola fila que coincide y además está indizada.
 - Eq_ref: se utilizan todas las partes de un índice.
 - Ref: todas las filas que coinciden con un campo del índice de otra tabla son leídas
 - index_merge: el join utiliza una serie de índices
 - Range: un índice se utiliza para encontrar filas dentro de un rango.
 - All: todas las filas son leídas.

- **Información obtenida a partir del comando “explain”**

- Possible_keys: muestra las claves que podrían ser utilizadas para encontrar los resultados.
- Key: indica que clave es utilizada por Mysql.
- REF: muestra que columnas son utilizadas en las comparaciones con los campos de KEY.
- Rows: muestra la cantidad de filas examinadas.
- Extra: muestra posible información útil por ejemplo (using filesort).
- Explain extended:
 - Show warnings;

Aclaración

▶ IMPORTANTE

- ▶ Al crear un stored procedure y ejecutarlo, este ya queda compilado, por ello, las estadísticas que se actualicen luego de esto no serán contempladas en sus ejecuciones posteriores
 - En este caso, se debe recompilar el SP para asegurar que tome nuevas estadísticas y se mantenga optimizado.

Bibliografía de la clase

- ▶ Garcia–Molina, H. (2008). *Database systems: the complete book*. Pearson Education India.
- ▶ Elmasri, R., & Navathe, S. B. (2007). Fundamentos de sistemas de bases de datos.
- ▶ <https://dev.mysql.com/doc/refman/5.7/en/>

Algunos anuncios

► **Miércoles 17 de Octubre**

- Expo Ciencia Facultad de Informática
 - Dónde: Hall central
 - Horario: de 9hs. a 17hs.
 - Nuestro proyecto: **“Conservación de especies del mar de la costa atlántica argentina: una experiencia mediada por tecnología móvil”**
 - Y también los invitamos a sumarse para la convocatoria que se abre a fin de año
 - MAS INFO: alejandra.lliteras@lifa.info.inlp.edu.ar

◦ **Jueves 18 de octubre**

- MySQL (usando el motor)
- *Luego de la clase*: parcialito de promoción de los temas de la clase 1 y de la clase 2: modelo de entidades y relaciones + entrega de un ejercicio de la práctica 1 que se anuncia ese día.
 - (la bibliografía de los temas figura en los slides correspondientes)