Laboratorio de Bases de Datos (EBB)

Unidad I - Introducción

Departamento de Electricidad, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

Primer Cuatrimestre 2017

Introducción [1 | 2]

- Bases de Datos
  - Tipos de Bases de Datos
  - Objetos de Bases de Datos
- Trabajo con un SGBDR
  - Diseño de aplicaciones
  - Implementación de BDs
  - Administración de BDs

Introducción [2 | 2]

- Creación de Bases de Datos
  - Almacenamiento y Transacciones
  - Creación y modificación de BDs
  - Creación de tipos datos y tablas
  - Planificación de la capacidad

Tipos de Bases de Datos [1 | 5]

- **SGBD**
  - Sistema Gestor de Bases de Datos
  - Compuesto por un conjunto de aplicaciones que se encargan de la creación y acceso a las BDs
  - Si se soportan bases de datos relacionales, se habla de un SGBDR
  - Ej. de SGBDR: Oracle, SQL Server, MySQL, PostgreSQL, etc

- Las siglas equivales en inglés para SGBD y SGBDR son DBMS y RDBMS respectivamente.

Tipos de Bases de Datos [2 | 5]

- Al instalar un SGBDR, el mismo se ejecuta como servicio
- Ventajas:
  - Continúa la ejecución después que se desconecta el usuario
  - Comienza a ejecutarse antes que se conecte el usuario
  - Generalmente se ejecuta bajo una cuenta de sistema o una creada para este fin
  - Se puede administrar remotamente

Servicio de MySQL y SQL Server

Tipos de Bases de Datos [3 | 5]

- Al instalar SQL Server también se instalan herramientas gráficas para su uso/administración:
  - Management Studio
- Para MySQL, una de las herramientas gráficas que se puede utilizar es “Workbench”

Conexión con Management Studio y Workbench

Tipos de Bases de Datos [4 | 5]

- **Base de Datos**
  - Conjunto de datos que pertenen a un mismo contexto que se pueden guardar para ser usados posteriormente
  - Existen programas llamados sistemas gestores de bases de datos (SGBD) que permiten guardar y posteriormente acceder a los datos en forma rápida y estructurada

Tipos de Bases de Datos [5 | 5]

- **BDs del sistema**: guardan información del SGBDR (no hay que borrarlas)
  - SQL Server: master, model, tempdb, msdb, distribution
  - MySQL: information_schema, performance_schema, sys, mysql
- **BDs del usuario**: son las BDs de producción
  - El SGBDR puede manejar muchas de éstas

BDs de sistema y usuario en Management Studio y Workbench

- En el caso de MySQL, para ver las BDs de sistema se puede usar el comando “show databases”, o desde la herramienta Workbench editar las preferencias, entrada “SQL Editor”, opción “Show metadata and internal schemas”.

Objetos de Bases de Datos [1 | 11]

- **Tablas**: colección de filas con la misma cantidad de columnas
- **Tipos de datos**: valores permitidos para cada columna
- **Restricciones (*constraints*)**: reglas que rigen los valores permitidos para las columnas. Proporcionan el mecanismo de integridad de datos

- Una base de datos no es únicamente un conjunto de tablas y sus respectivos datos.

Objetos de Bases de Datos [2 | 11]

- **Índices**: estructura de almacenamiento que brinda un acceso rápido a los datos y fuerza la integridad de los mismos
- **Valores por defecto**: valores que toman las columnas cuando no se proporciona alguno
- **Reglas**: información que define los valores para las columnas

Objetos de Bases de Datos [3 | 11]

- **Vistas**: forma de ver los datos de una o más tablas
- **Procedimientos almacenados**: colección de sentencias SQL, con un nombre, que se ejecutan como un todo. Admiten parámetros de entrada y/o salida
- **Desencadenadores (*triggers*)**: procedimientos almacenados que se ejecutan automáticamente después de producida una acción

Objetos de Bases de Datos [4 | 11]

- **Funciones**: rutinas formadas por una o más sentencias SQL
- **Usuarios**: entidad que puede solicitar un recurso del servidor de BD
- **Roles (grupos)**: conjunto de usuarios de una misma BD que comparten los mismos permisos

Objetos de Bases de Datos [5 | 11]

- **Esquemas**: espacio de nombres para los objetos de la BD (SQL Server)

Objetos de Bases de Datos [6 | 11]

- Los objetos se organizan en una determinada jerarquía:
  - **SQL Server**: servidor - BD - esquema - objeto
  - **MySQL**: servidor - BD - objeto

Esquemas en SQL Server

- En versiones anteriores a SQL Server 2005, la jerarquía de un objeto cualquier era:

  servidor - BD - usuario dueño - objeto

- Esto tenía la desventaja que si se borraba un usuario implicaba que se debían renombrar los objetos correspondientes. Al reemplazar al usuario dueño con un esquema, el agrupamiento de objetos no depende del dueño (borrar un usuario no implica renombrar los objetos dependientes).

- El empleo de esquemas es propio de SQL Server. MySQL no cuenta con este concepto.

Objetos de Bases de Datos [7 | 11]

- Sobre los nombres de objetos en MySQL:
  - Pueden ir entre comillas simples invertidas (' ` ')
  - En los nombres de BDs y tablas no se puede emplear '.', '\' ni '/', ya que las BDs se implementan como directorios y las tablas como archivos
  - Las palabras reservadas y los nombres de funciones son insensibles a mayúsculas/minúsculas, mientras que los nombres de BDs y tablas sí lo son

- El empleo de la comilla simple invertida permite emplear casi cualquier carácter en el nombre de los objetos (hasta se pueden especificar nombres reservados).

- Cada vez que se crea una BD en MySQL, en su directorio de datos se crea un directorio cuyo nombre es el de la BD. Por cada tabla que se cree en la BD, se crea un archivo con este nombre dentro del directorio que corresponde a la BD. Por esta razón, como MySQL se puede instalar en Linux, y el mismo es sensible a mayúsculas y minúsculas, se debe tener cuidado con los nombres de tablas y bases de datos.

Objetos de Bases de Datos [8 | 11]

- Formas para referenciar los objetos:
  - **Absoluta**: se especifica toda la jerarquía a la cual pertenece un determinado objeto
  - Ej: server1.ventas.dbo.facturas
  - **Relativa**: se especifica sólo una parte de la jerarquía a la cual pertenece un determinado objeto

- Cada objeto creado tiene un nombre único. Cuando se emplean nombres completos se dice que el objeto está completamente cualificado.

- No siempre hay que especificar el nombre completo. Se pueden emplear nombres relativos.
  - Si no se especifica el servidor, se asume el servidor al cual se realizó la conexión actualmente.
  - Si no se especifica la base de datos, se asume la actual.
  - ¿Si no se especifica el esquema? (ver la siguiente diapositiva)

Objetos de Bases de Datos [9 | 11]

- **Esquema (SQL Server):**
  - Esquema **dbo**: es el predeterminado para una BD recién creada que tienen todos los usuarios (no se puede borrar ni modificar)
  - En una BD se pueden crear los esquemas que sean necesarios
  - Todos los usuarios de una BD pertenecen a un único esquema

- El esquema “dbo” es el esquema predeterminado para una BD recién creada, y es el que tienen por defecto los usuarios que se van creando.

- Si al momento de especificar un objeto no se emplea el nombre completamente cualificado del mismo, SQL Server sigue los siguientes pasos:

  - 1. Si el usuario tiene un esquema predeterminado, SQL Server busca en el mismo.

  - 2. Si no se encuentra en el esquema predeterminado, o si el usuario no tiene uno, SQL Server busca en el esquema dbo.

Objetos de Bases de Datos [10 | 11]

- **Esquema (resolución de nombres):**
  - 1. Si el usuario tiene un esquema predeterminado, SQL Server busca en el mismo
  - 2. Si no se encuentra en el esquema predeterminado, o si el usuario no tiene uno, se busca en dbo
  - 3. Si un usuario no tiene un esquema predeterminado, se busca en el esquema dbo directamente

- Ejemplo: un usuario, cuyo esquema predeterminado se llama Person, ejecuta la siguiente consulta:

```
SELECT * FROM Contact
```

- SQL Server primero busca la tabla `Contact` en el esquema `Person` (`Person.Contact`). Si el esquema `Person` no tiene la tabla `Contact`, SQL Server intenta resolver el nombre de la tabla a `dbo.Contact`.

Objetos de Bases de Datos [11 | 11]

- **Creación de un esquema:**

```
USE <BD>
GO
CREATE SCHEMA <esquema>
GO
```

- **Asignación del esquema predeterminado:**

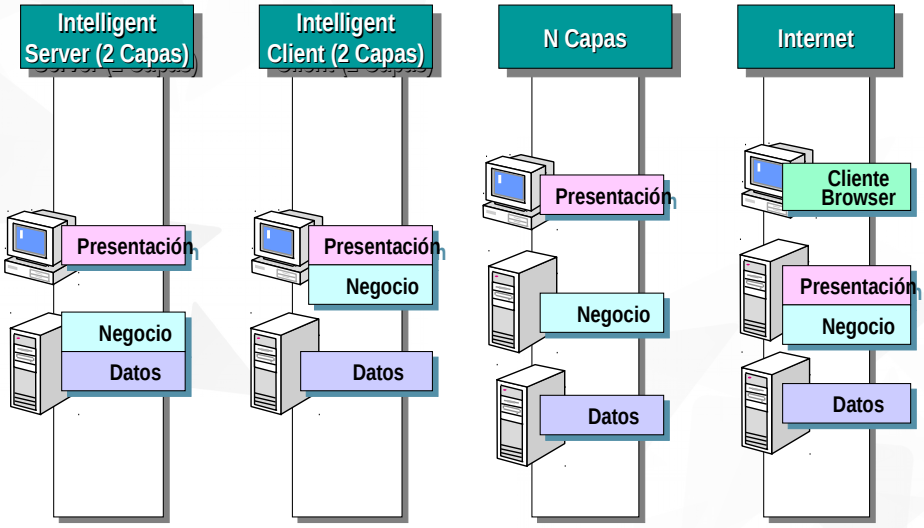
```
ALTER USER <usuario> WITH DEFAULT_SCHEMA = <esquema>
```


- Estas operaciones también se pueden hacer desde el Management Studio.

Diseño de aplicaciones [1 | 4]

- Capas para implementar un modelo C/S:
  - **Presentación**: lógica para presentar los datos a los usuarios (lógica de interfaz). Casi siempre la implementa el cliente
  - **Negocio**: lógica de la aplicación y reglas de negocio. El SGBDR puede estar involucrado en esta capa
  - **Datos**: definición de la BD, integridad, procedimientos almacenados, etc. El SGBDR se involucra estrechamente con esta capa

Diseño de aplicaciones [2 | 4]



Diseño de aplicaciones [3 | 4]

- Arquitecturas típicas:
  - **Intelligent Server (2 capas)**: la mayoría de los procesos ocurre en el servidor con la lógica de interfaz en el cliente. La lógica de negocios es implementada casi por completo en la BD
  - **Intelligent Client (2 capas)**: la mayoría de los procesos ocurre en el cliente con los datos residiendo en el servidor. La performance es mala por el tráfico en la red

Diseño de aplicaciones [4 | 4]

- Arquitecturas típicas:
  - **N capas**: el procesamiento es dividido en un servidor de BD, uno o varios servidores de aplicación y los clientes. Es complejo pero muy escalable
  - **Internet**: el proceso es dividido en 3 capas con los servicios de negocio y de presentación residiendo en un servidor web, y los clientes usando simples navegadores

Implementación de BDs [1 | 4]

- Implementar una BD implica:
  - Diseñar la BD
  - Crear la BD y sus objetos
  - Probar y poner a punto la aplicación y la BD
  - Planear el desarrollo
  - Administrar la aplicación

Implementación de BDs [2 | 4]

- **Diseñar la BD**
  - Uso eficiente del HW que permita crecimiento futuro, identificando y modelando los objetos de la BD y la lógica de la aplicación, la información de cada objeto y sus relaciones
- **Crear la BD y sus objetos**
  - Tablas, integridad y entrada de datos, procedimientos almacenados, vistas, índices, seguridad, etc

Implementación de BDs [3 | 4]

- **Probar y poner a punto la aplicación y la BD**
  - Las tareas se deben ejecutar de forma rápida y correcta. Junto a un buen diseño, correcto uso de índices y RAID se consigue un buen rendimiento
- **Planear el desarrollo**
  - Analizar la carga de trabajo con el sistema en producción para recomendar una óptima configuración de índices

Implementación de BDs [4 | 4]

- **Administrar la aplicación**
  - Configurar los servidores y clientes, monitorizar el rendimiento en todo momento, gestionar los trabajos, alertas y operadores, manejar la seguridad y las copias de respaldo, etc

Administración de BDs [1 | 3]

- **Administrar una BD involucra:**
  - Instalar y configurar el SGBDR
  - Establecer la seguridad de la red
  - Construir las BDs
  - Manejar las actividades del día a día

Administración de BDs [2 | 3]

- Construir las BDs:
  - Reservar espacio en disco
  - Crear trabajos automatizados

Administración de BDs [3 | 3]

- Manejar las actividades del día a día:
  - Importación y exportación de datos
  - Copias de respaldo y restauraciones
  - Monitorizar y poner a punto la BD
  - Automatizar todo lo anterior

Almacenamiento y Transacciones [1 | 33]

- **Transacción:**
  - Una o más unidades de trabajo donde todas tienen éxito o ninguna
  - Puede ser un único cambio en la BD, o un conjunto de cambios relacionados que deben completarse todos o ninguno
  - Ejemplo: extracción de una cuenta y depósito en otra

- Cualquier cambio que se realice en una BD (agregado, modificación, eliminación de datos y/o estructura) se hace en el contexto de una transacción. Una transacción puede ser un único cambio en la BD, o una serie de cambios relacionados que deben completarse todos o ninguno.

- Todas las transacciones que se realizan sobre una BD se registran secuencialmente en el registro de transacciones (ver más adelante).

Almacenamiento y Transacciones [2 | 33]

- Propiedades de una transacción:
  - **A**tomicidad
  - **C**onsistencia
  - A**I**slamiento
  - **D**urabilidad

- **Atomicidad**: se completan todas las operaciones de la transacción, o ninguna. Esto evita que las modificaciones se realicen parcialmente.
- **Consistencia**: debe modificar los datos según las reglas definidas.
- **Aislamiento**: cualquier otra actividad concurrente con la transacción no tiene efecto sobre esta última.
- **Durabilidad**: cuando se completa una transacción, sus resultados se guardan.
- A estas propiedades se las conoce por el acrónimo ACID en inglés.

Almacenamiento y Transacciones [3 | 33]

- Las transacciones pueden ser:
  - **Explícitas**: el comienzo y el final están explícitamente definidos
  - **Implícitas**: algunos comandos SQL terminan implícitamente una transacción (`INSERT`, `ALTER`, `SELECT`, `UPDATE`, etc)

- Las transacciones implícitas terminan sin un COMMIT (ver más adelante).

- Por defecto, ambos motores trabajan en el modo “autocommit”, con lo cual las transacciones implícitas se ejecutan sin necesidad de un COMMIT, y no pueden retrotraerse.

Almacenamiento y Transacciones [4 | 33]

Transacciones explícitas

SQL Server	MySQL
`BEGIN TRANSACTION`	`START TRANSACTION`
`sentencias`	`sentencias`
`COMMIT | ROLLBACK TRANSACTION`	`COMMIT | ROLLBACK`

Almacenamiento y Transacciones [5 | 33]

- **COMMIT**: termina una transacción, y como resultado todos los cambios se vuelven permanentes
- **ROLLBACK**: retrotrae todo lo hecho hasta el momento por las tareas que forman la transacción

- Todas las transacciones pueden terminar con éxito (commit), en cuyo caso sus cambios se vuelven permanentes en la BD, o bien pueden fracasar o cancelarse (roll back), en cuyo caso sus cambios se retrotraen de la BD.

Almacenamiento y Transacciones [6 | 33]

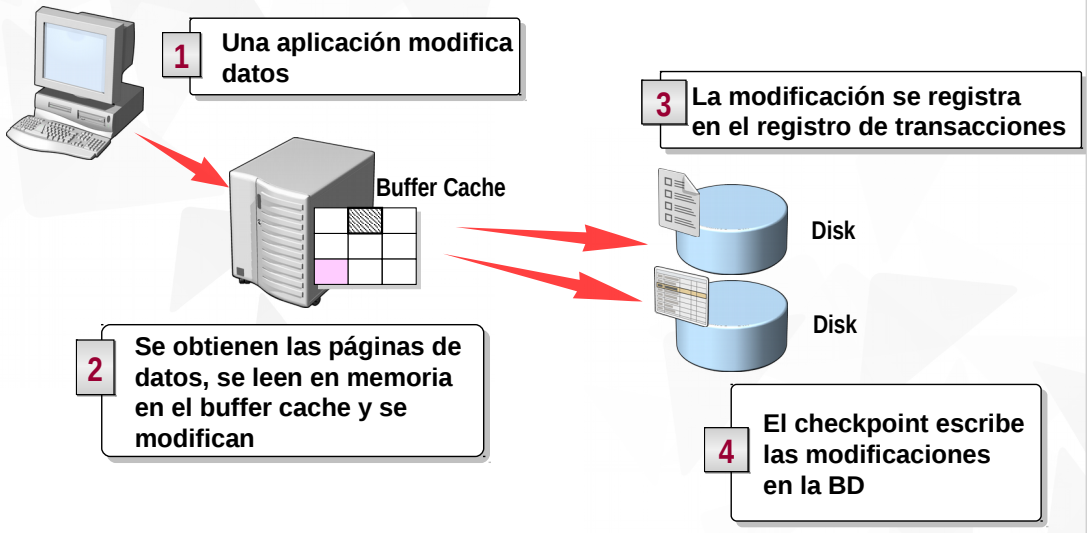
- En SQL Server una BD puede estar formada por 3 tipos de archivos:
  - Archivos primarios de datos
  - Archivos secundarios de datos
  - Registro de transacciones

- **Archivos primarios de datos**: toda BD tiene un único archivo primario de datos (extensión predeterminada .mdf). Este archivo, además de tener los datos de la BD, tiene información sobre la misma (metainformación).

- **Archivos secundarios de datos**: una BD puede tener uno o más archivos secundarios de datos (extensión predeterminada .ndf). Generalmente se usan estos archivos para aumentar la capacidad.

- **Registro de transacciones**: una BD debe tener al menos un archivo de registro de transacciones (extensión predeterminada .ldf). El registro de transacciones registra, **secuencialmente**, todas las transacciones, permitiendo recrear la secuencia de cambios en la BD en caso de producirse un problema.

Almacenamiento y Transacciones [7 | 33]



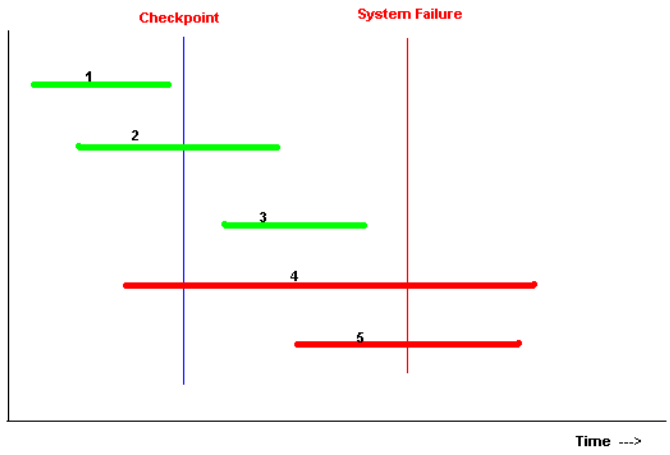
- 1. Una aplicación envía una modificación de datos.
- 2. Cuando se ejecuta la modificación, se cargan (si es que no fueron cargadas ya previamente) desde el disco a memoria las páginas de datos afectadas en un área reservada llamada "*buffer cache*". SQL Server modifica estas páginas en memoria (páginas "sucias").
- 3. Cuando la transacción termina (COMMIT), SQL Server escribe los cambios en el registro, marcándola como terminada. Esto no significa que los datos se copian del registro a la BD, sino que se han completado con éxito todos los elementos de la transacción.
- 4. El proceso de *checkpoint* escribe todas las páginas modificadas al disco, no sólo las transacciones terminadas.

Almacenamiento y Transacciones [8 | 33]

- Proceso *checkpoint* (SQL Server):
  - Copia todas las páginas modificadas al disco (no sólo las transacciones terminadas)
  - Se puede ejecutar de distintas formas:
    - Comando `CHECKPOINT`
    - Intervalo de tiempo
    - Al hacer un backup (modelo Simple)
    - Detención del motor de BD

- El proceso de *checkpoint* puede producirse de diferentes formas:
  - Ejecutando el comando CHECKPOINT
  - Cuando se llega al intervalo de tiempo en el cual se ejecuta el proceso
  - Estando en el modelo de recuperación Simple, se hace un backup de la BD (ver más adelante)
  - Estando en el modelo de recuperación Simple, se modifica la estructura del archivo de la BD (ver más adelante)
  - Se detiene el motor de BD

Almacenamiento y Transacciones [9 | 33]



Almacenamiento y Transacciones [10 | 33]

- Transacción 1:
  - Terminó antes del *checkpoint* y antes de la falla de SQL Server
  - Después de la falla, SQL Server no deberá hacer nada (el *checkpoint* escribió todas las modificaciones a la BD)

Almacenamiento y Transacciones [11 | 33]

- Transacción 2:
  - El *checkpoint* escribió las modificaciones hasta ese momento en la BD
  - Terminó antes de producirse la falla
  - Después de la falla, SQL Server termina de escribir el resto de la transacción (*roll forward*)

Almacenamiento y Transacciones [12 | 33]

- Transacción 3:
  - Comenzó después del *checkpoint*
  - Terminó antes de producirse la falla
  - Después de la falla, SQL Server escribe toda la transacción (*roll forward*)

Almacenamiento y Transacciones [13 | 33]

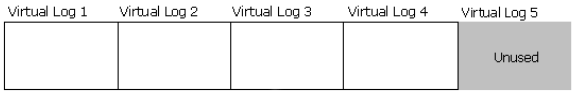
- Transacción 4:
  - El *checkpoint* escribió las modificaciones hasta ese momento en la BD
  - Al producirse la falla no había terminado
  - Después de la falla, SQL Server debe retrotraer todas las modificaciones hechas por la transacción (*roll back*)

Almacenamiento y Transacciones [14 | 33]

- Transacción 5:
  - Comenzó después del *checkpoint*
  - Al producirse la falla no había terminado
  - Después de la falla, SQL Server debe volver a ejecutar la transacción ya que no escribió nada en la BD

Almacenamiento y Transacciones [15 | 33]

- Arquitectura física del registro de transacciones:
  - Implementado mediante 1 o más archivos
  - Cada archivo se divide en Virtual Log Files (VLF)
  - Las transacciones se agregan dentro de los VLF

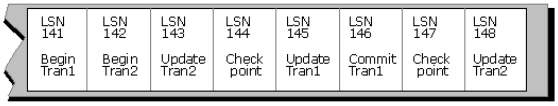


- Cada archivo se divide en partes más chicas llamadas Virtual Log File (VLF). Estos VLF no tienen tamaño fijo, y no hay una cantidad fija de los mismos.

- Las transacciones se van agregando dentro de los VLF, y cuando se llena uno se continúa con el siguiente.

Almacenamiento y Transacciones [16 | 33]

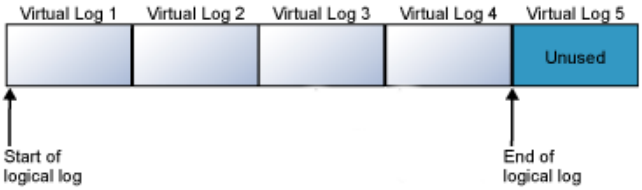
- Arquitectura lógica del registro de transacciones:
  - Cada transacción se identifica mediante un número de secuencia (LSN)
  - Funciona como un anillo (ver más adelante)



- Cada transacción nueva que se agrega lo hace a continuación de la última, con un LSN mayor. Las transacciones en el registro pueden registrar ya sea la operación realizada, o las imágenes antes y después de realizar la operación.

Almacenamiento y Transacciones [17 | 33]

- Ejemplo:
  - Físicamente dividido en 5 VLF
  - El quinto VLF todavía no se empezó a llenar
  - Lógicamente, la primer transacción comienza al principio del primer VLF, y la última está al final del cuarto VLF



- En este ejemplo se puede ver que el tamaño lógico del registro de transacciones es distinto al tamaño físico.

Almacenamiento y Transacciones [18 | 33]

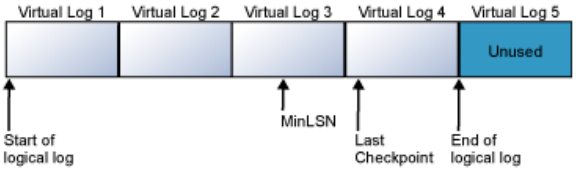
- Truncamiento del registro de transacciones:
  - Si nunca se borraran las transacciones del registro, el tamaño lógico crecería hasta igualar al físico
  - Al proceso de borrar las transacciones del registro que no sirven para restaurar una BD se lo llama “truncamiento”

- En algún momento, las transacciones viejas que ya no sirven para restaurar la BD se tienen que borrar para poder agregar más.

- Una transacción deja de ser útil en el registro de transacciones cuando se cumplen todas las condiciones siguientes:
  - La transacción está terminada
  - Las páginas de datos afectadas por la transacción ya fueron escritas a disco por un *checkpoint*
  - No es necesaria la transacción para realizar un backup (completo, diferencial o del registro de transacciones)
  - No es necesaria la transacción para ninguna funcionalidad que lea el registro (ejemplo: replicación)

Almacenamiento y Transacciones [19 | 33]

- Truncamiento del registro de transacciones:
  - Porción activa del registro
  - Transacción MinLSN



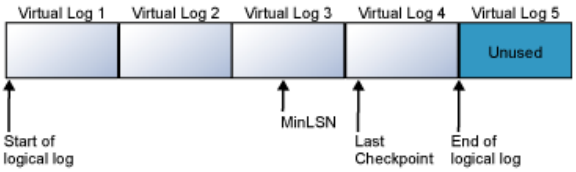
- Dentro del registro de transacciones, una transacción que se necesite para poder restaurar la BD se llama transacción activa. Un VLF que tenga al menos una transacción activa se llama VLF activo, y constituye la porción activa del registro de transacciones.

- Dentro de la porción activa, a la primer transacción se la identifica por el mínimo número de secuencia (MinLSN).

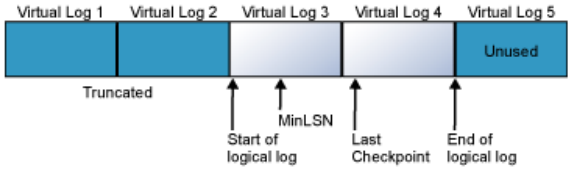
- En este ejemplo, la transacción MinLSN está en el VLF 3. Los VLF 1 y 2 contienen transacciones inactivas, por lo que se pueden truncar estas transacciones de los mismos (la porción activa del registro nunca se puede truncar).

Almacenamiento y Transacciones [20 | 33]

- Registro antes de ser truncado:



- Registro después de ser truncado



- Al truncar el registro, no se reduce el tamaño del archivo físico, sino el del lógico.

- Cuando el tamaño lógico iguala al físico, las nuevas transacciones que se agreguen lo hacen al comienzo del archivo físico. Este ciclo se repite indefinidamente siempre que el tamaño lógico no iguale al físico. Si el tamaño lógico iguala al físico, ocurre 1 de 2 cosas:
  - Si está habilitada la opción FILEGROWTH para el registro de transacciones, y hay espacio en el disco, el archivo se expande en la cantidad especificada, pudiéndose agregar nuevos registros.
  - Si no está habilitada la opción FILEGROWTH, o el disco donde está el registro de transacciones no tiene más espacio, se genera un error.

Almacenamiento y Transacciones [21 | 33]

- Truncar el registro de transacciones es fundamental para evitar que el mismo se llene
- Cuando se trunca el registro de transacciones, se borran los VLFs inactivos
- Para que se pueda truncar el registro de transacciones, se debe ejecutar antes un *checkpoint*

- Para que se pueda truncar el registro se debe ejecutar antes un proceso de *checkpoint*, con lo cual una vez copiadas las páginas modificadas a la BD, la parte inactiva del registro se marca como reutilizable y puede ser liberada para ser truncada.

Almacenamiento y Transacciones [22 | 33]

- **Motor de almacenamiento (MySQL):**
  - Componente que maneja las operaciones SQL para distintos tipos de tablas
  - MySQL permite cargar y descargar distintos motores de almacenamiento mientras se ejecuta el servicio
  - A los motores de almacenamiento también se los conoce como motores de tablas o tipos de tablas

- MySQL utiliza una arquitectura en la cual se pueden cargar y descargar motores de almacenamiento mientras se está ejecutando el servicio.

- Para determinar qué motores soporta el servidor, se puede usar:

```
SHOW ENGINES;
```

- Para la versión 5.7, el motor de almacenamiento por defecto es InnoDB.

Almacenamiento y Transacciones [23 | 33]

- Motores de almacenamiento en MySQL:
  - **InnoDB**: motor transaccional (conforme a ACID), con capacidades de aplicar y retrotraer transacciones, recuperación en caso de problemas con la BD. Guarda los datos usando índices agrupados, soporta integridad referencial mediante claves propagadas
  - **MyISAM**: usa bloqueos a nivel tabla, limitando las operaciones de lectura/escritura. Usado generalmente en escenarios de sólo lectura o *data warehousing*

Almacenamiento y Transacciones [24 | 33]

- Motores de almacenamiento en MySQL:
  - **Memory**: guarda todos los datos en RAM, permitiendo un acceso rápido. Antes tenía el nombre de HEAP. Se lo está dejando de usar
  - **CSV**: sus tablas son archivos de texto con valores separados por comas. Permite importar/exportar los datos en formato CSV. No emplea índices

Almacenamiento y Transacciones [25 | 33]

- Motores de almacenamiento en MySQL:
  - **Archive**: emplea tablas compactas, sin indexar. Pensado para guardar y recuperar grandes cantidades de datos históricos
  - **Blackhole**: acepta pero no guarda datos (similar al dispositivo /dev/null en Unix). Pensado para escenarios de replicación, donde se envían las sentencias DML a los servidores esclavos, pero el servidor principal no mantiene una copia de los datos

Almacenamiento y Transacciones [26 | 33]

- Motores de almacenamiento en MySQL:
  - **NDB (NDBCLUSTER)**: pensado para aplicaciones que requieren el mayor nivel de disponibilidad y uso
  - **Mrg_MyISAM (Merge)**: permite agrupar lógicamente un conjunto de tablas MyISAM idénticas y referenciarlas como un único objeto. Resulta útil en escenarios de "*data warehousing*"

Almacenamiento y Transacciones [27 | 33]

- Motores de almacenamiento en MySQL:
  - **Federated**: ofrece la capacidad de conectar servidores MySQL separados para crear una BD lógica a partir de varios servidores físicos. Muy bueno para entornos distribuidos
  - **Example**: pensado para quienes quieran escribir su propio motor de almacenamiento. Se pueden crear tablas, pero no se puede guardar ni recupear nada

Almacenamiento y Transacciones [28 | 33]

- Motores de almacenamiento en MySQL:
  - En una misma BD se pueden usar distintos tipos de tablas
  - El tipo de tabla se especifica en la sentencia `CREATE TABLE`. Para cambiar el tipo de tabla:

```
ALTER TABLE <tabla> ENGINE=<motor>;
```

Almacenamiento y Transacciones [29 | 33]

- Algunas diferencias entre InnoDB y MyISAM:

InnoDB	MyISAM
Soporta bloqueos a nivel fila	Soporta bloqueos a nivel tabla
Soporta claves propagadas	No soporta claves propagadas
Soporta transacciones (se puede hacer `commit` y `roll back`)	No soporta transacciones (no se puede hacer `commit` ni `roll back`)

Almacenamiento y Transacciones [30 | 33]

- Registro de transacciones (InnoDB):
  - MySQL escribe al registro "**redo**" y al archivo de datos
  - La escritura en el registro "**redo**" se realiza cada vez que se produce una modificación de datos, mientras que la escritura en el archivo de datos se realiza en forma periódica, cuando haya tiempo

- La escritura en el registro "redo" es mucho más rápida (en forma serial) que la del archivo de datos (en forma aleatoria).

- El registro "redo", por defecto, se encuentra en el directorio $DATADIR, y está formado por los archivos "ib_logfile0" e "ib_logfile1".

Almacenamiento y Transacciones [31 | 33]

- Registro de transacciones (InnoDB):
  - Si se interrumpe una transacción se deben retrotraer todos los cambios hechos
  - Registro "**undo**": se emplea cuando se deben retrotraer los cambios hechos por una transacción incompleta

- Una transacción se puede interrumpir, por ejemplo, porque se detiene el servicio de MySQL, el cliente se desconecta antes de llegar al COMMIT, se envía el comando ROLLBACK, etc.

- El registro "undo" está formado por los archivos "ibdata".

Almacenamiento y Transacciones [32 | 33]

- Registro de transacciones (InnoDB):
  - En caso de producirse un problema con el servidor:
    - Primero se aplican las transacciones del registro "**redo**" cuando arranca el servicio de MySQL
    - Luego se retrotraen las transacciones que no se completaron, utilizando el registro "**undo**"

- En caso de producirse un problema con el servidor, en una BD que utilice tablas InnoDB no se necesita hacer nada especial: el proceso de restauración de InnoDB finaliza automáticamente cualquier transacción completada antes de producirse el fallo y retrotrae los cambios hechos por las transacciones no completadas.

Almacenamiento y Transacciones [33 | 33]

- El registro de transacciones contiene (InnoDB):
  - **LSN (Log Sequence Number)**: ID único para cada transacción
  - **Prev LSN**: cada transacción tiene un enlace a la transacción anterior
  - **ID de transacción**: identifica a la transacción en la BD que generó el registro
  - **Checkpoints**: lista de transacciones abiertas cuando se creó el *checkpoint*

- El servidor utiliza los *checkpoints* para saber hasta dónde debe volver hacia atrás en el registro de transacciones para empezar a procesarlo.

Creación y modificación de BDs [1 | 17]

- Para crear una BD se puede especificar una sentencia SQL o usar alguna herramienta gráfica:
  - SQL Server: Management Studio
  - MySQL: Workbench

- Desde SQL Server 2005, la herramienta gráfica se llama “Management Studio” (en versiones anteriores se llamaba “Enterprise Manager”.

- En el caso de MySQL, una de las herramientas gráficas que se puede utilizar es “Workbench”, pero no es la única.

Creación y modificación de BDs [2 | 17]

- Para crear una BD en SQL Server se debe especificar:
  - Nombre
  - Tamaño
  - Archivos donde residirá
- En el caso de MySQL:
  - Nombre

Creación y modificación de BDs [3 | 17]

- Creación de una BD en SQL Server:

```
CREATE DATABASE Prueba ON
  PRIMARY (NAME = prueba_data,
  FILENAME = 'C:\data\prueba.mdf',
  SIZE = 10MB, MAXSIZE = 15MB, FILEGROWTH = 20%
)
LOG ON (
  NAME = prueba_log,
  FILENAME = 'C:\data\prueba.ldf',
  SIZE = 3MB, MAXSIZE = 5MB, FILEGROWTH = 1MB
)
```

Creación de BD en SQL Server

- Cada vez que se crea, modifica o borra una BD, es conveniente realizar una copia de seguridad a la BD master.

- Si sólo se especifica el nombre de la BD durante la creación, SQL Server hace una copia de la BD model en la nueva BD, con lo cual esta última queda con las mismas características que la BD model.

Creación y modificación de BDs [4 | 17]

- **PRIMARY**: especifica los archivos del grupo de archivos primario. Una BD puede tener varios grupos de archivos, pero sólo uno puede ser el primario (si se omite, el primer archivo listado es el archivo primario).
- **NAME**: nombre lógico del archivo (nombre con el cual SQL server referencia al archivo).
- **FILENAME**: nombre físico del archivo (para el SO). La extensión suele ser .mdf. Debe ser local al servidor.

- Un grupo de archivos es una colección lógica de archivos de datos que permite administrarlos como una única unidad.
- SQL Server tiene un grupo de archivos primario, el cual contiene el archivo de datos primario (tiene las tablas del sistema, y por lo general tiene la extensión .mdf). También puede tener grupos de archivos secundarios, los cuales contienen archivos de datos cuya extensión suele ser .ndf.
- El registro de transacciones no va en ningún grupo de archivos.

Creación y modificación de BDs [5 | 17]

- **SIZE**: tamaño de los archivos de datos y del registro de transacciones. Se puede especificar en MB (por defecto) o en KB (tamaño mínimo = 512 KB para ambos archivos).
- **MAXSIZE**: tamaño máximo al que puede crecer un archivo. Se lo puede especificar en MB o KB. Si no se especifica hay un crecimiento sin restricciones (hasta que se llene el disco).

- El tamaño especificado para el archivo primario de datos debe ser más grande que la BD model.

Creación y modificación de BDs [6 | 17]

- **FILEGROWTH**: incremento de crecimiento del archivo (no debe exceder a MAXSIZE). Un valor de 0 indica sin crecimiento. Se puede especificar en MB, KB o porcentajes. El valor por defecto es 10% y el mínimo = 64 KB (1 extent).

Creación y modificación de BDs [7 | 17]

- Creación de una BD en MySQL:

```
CREATE DATABASE Prueba;
```

Creación de BD en MySQL

- Al crear una BD en MySQL, se crea un directorio con el nombre de la misma. En este ejemplo, se crearía el directorio /var/lib/mysql/Prueba.

Creación y modificación de BDs [8 | 17]

- Después de crear una BD en SQL Server, se pueden cambiar sus opciones:
  - **Read-Only**: para lectura únicamente
  - **Auto shrink**: determina si los archivos reducen su tamaño automáticamente o no
  - **Auto close**: determina si se cierra automáticamente cuando se desconecta el último usuario
  - Etc

Opciones de BD en SQL Server

- Para modificar las opciones de una BD se pueden emplear sentencias T-SQL o usar la herramienta gráfica. Por ejemplo, para modificar las opciones se puede emplear el procedimiento almacenado `sp_dboption`. Para ver las opciones se puede usar el procedimiento `sp_help`.

Creación y modificación de BDs [9 | 17]

- En SQL Server, cuando crecen los archivos, o aumentan las modificaciones de datos, se puede aumentar el tamaño de los archivos de datos o del registro de transacciones
- MySQL no permite estas modificaciones

Creación y modificación de BDs [10 | 17]

- Se puede controlar el tamaño de la BD:
  - **Configurando el crecimiento automático de los archivos**: se reducen las tareas de administración
  - **Incrementando manualmente el tamaño máximo de los archivos**
  - **Agregando manualmente archivos secundarios**: se pueden crear archivos secundarios en diferentes discos para aumentar el tamaño (no usar si se tienen sistemas de almacenamiento con RAID)

Creación y modificación de BDs [11 | 17]

- Incremento del tamaño máximo de un archivo:

```
ALTER DATABASE Prueba
MODIFY FILE (NAME = 'Prueba_log', SIZE = 10MB)
```

- Agregado de un archivo de datos secundario:

```
ALTER DATABASE Prueba
ADD FILE (NAME = Prueba_data2,
FILENAME='C:\data\Prueba2.ndf',
SIZE = 10MB, MAXSIZE = 20MB)
```


- Estas modificaciones deben realizarse desde la BD master.

- En MySQL la sentencia `ALTER DATABASE` permite cambiar las características de una BD (el juego de caracteres válidos y las reglas para comparar esos caracteres), las cuales se guardan en el archivo db.opt dentro del directorio de la BD.

Creación y modificación de BDs [12 | 17]

- Cuando crece la BD o la misma tiene mucha modificación de datos, se debe expandir el registro de transacciones
- Si el registro de transacciones se llena, SQL Server no las puede guardar, con lo cual no se procesan, y la BD queda sin actividad
- Conclusión: se debe monitorizar periódicamente el registro de transacciones

- Igual que con los archivos de datos, se puede expandir el registro de transacciones usando el Management Studio o la sentencia `ALTER DATABASE`.

Creación y modificación de BDs [13 | 17]

- Situaciones que producen gran actividad en el registro de transacciones:
  - Guardar información en una tabla que contiene índices
  - Transacciones masivas (`INSERT`, `UPDATE` y `DELETE`) que llevan a la modificación de muchas filas
  - Agregar o modificar datos de tipo `text` o `image` con la opción `WITH LOG`

Creación y modificación de BDs [14 | 17]

- Cuando hay mucho espacio reservado, o bien el espacio de datos decrece, conviene reducir la BD o los archivos de la misma
- En SQL Server se puede reducir una BD usando un asistente o mediante la sentencia `DBCC SHRINKDATABASE`

Creación y modificación de BDs [15 | 17]

- Opciones de `DBCC SHRINKDATABASE`:
  - NombreBD: nombre de la BD a reducir
  - Porcentaje: % de espacio libre luego de la reducción
  - NOTRUNCATE: se retiene el espacio liberado en los archivos
  - TRUNCATEONLY: libera el espacio no usado SO

- Ejemplo:

```
DBCC SHRINKDATABASE (Prueba, 25)
```

Creación y modificación de BDs [16 | 17]

- SQL Server también permite reducir un archivo de datos:

```
DBCC SHRINKFILE (Prueba_data, 10)
```

- Opciones:
  - Nombrearchivo: nombre del archivo a reducir
  - Tamaño: tamaño final del archivo, luego de la reducción (MB)
  - EMPTYFILE: migra los datos de ese archivo a los otros archivos del mismo grupo de archivos

Creación y modificación de BDs [17 | 17]

- Para borrar una BD se puede usar la sentencia `DROP DATABASE`:

```
DROP DATABASE Prueba[;]
```

- No se puede eliminar una BD cuando:
  - Está siendo restaurada
  - Un usuario está conectada a la misma
  - Se está publicando como parte de una replicación

Borrado de BD en SQL Server y MySQL

- En SQL Server, después de borrar una BD se debe realizar un respaldo de la BD master.

Creación de tipos de datos y tablas [1 | 13]

- SQL Server y MySQL proveen diferentes tipos de datos:
  - `char(N), varchar(N), nchar(N), nvarchar(N)`
  - `datetime, smalldatetime`
  - `decimal(p,s), numeric(p,s), float, real,`
  - `int, smallint, tinyint`
  - `money`
  - `text, ntext, image`

Creación de tipos de datos y tablas [2 | 13]

- SQL Server también permite definir tipos de datos, los cuales están basados en los del sistema
- Creación:

```
CREATE TYPE dbo.isbn FROM smallint NOT NULL
```

- Borrado:

```
DROP TYPE dbo.isbn
```


- MySQL no soporta la creación de tipos de datos de usuario.

Creación de tipos de datos y tablas [3 | 13]

- Guías para la creación de tipos de datos:
  - Si la longitud de la columna varía, usar tipos variables (`varchar`)
  - Para tipos de datos numéricos, tener en cuenta el almacenamiento y la precisión. En general, usar `decimal`
  - Si el almacenamiento supera los 8000 bytes, se deben usar los tipos `text` o `image`

Creación de tipos de datos y tablas [4 | 13]

- Al crear una tabla, se debe especificar:
  - Nombre (único dentro de la BD)
  - Nombres y tipos de datos de las columnas (únicos dentro de la tabla)
  - Si cada columna admite o no valores nulos (`NULL` o `NOT NULL`)

- La cantidad de tablas que se pueden crear dentro de una misma BD, como así también la cantidad de columnas por tabla, depende del motor de BD y su versión.

Creación de tipos de datos y tablas [5 | 13]

- Creación de tablas (SQL SERVER): `CREATE TABLE`

```
CREATE TABLE Socio (
  idSocio  idSocio NOT NULL,
  apellido cadena NOT NULL,
  nombre cadena NOT NULL,
  foto image NULL
)
```

Creación de tabla en SQL Server

Creación de tipos de datos y tablas [6 | 13]

- Creación de tablas (MySQL):

```
CREATE TABLE [IF NOT EXISTS13] Socio (
  idSocio  INT NOT NULL,
  apellido VARCHAR(30) NOT NULL,
  nombre VARCHAR(30) NOT NULL,
  foto BLOB NULL
) ENGINE = <motor de almacenamiento>;
```

Creación de tabla en MySQL

- Un motor de almacenamiento es el componente de SW subyacente que utiliza el SGBDR para crear, leer, actualizar y borrar datos de una BD. Algunos SGBDR, como MySQL, soportan múltiples motores de almacenamiento: InnoDB, MyISAM, etc.

- Para los tipos de datos numéricos, MySQL emplea un ancho para visualización. Para más información: https://blogs.oracle.com/jsmyth/entry/what_does_the_11_mean

Creación de tipos de datos y tablas [7 | 13]

- Borrado de tablas:
  - SQL Server: `DROP TABLE Socio`
  - MySQL: `DROP TABLE [IF EXISTS] Socio;`

Borrado de tabla en SQL Server y MySQL

Creación de tipos de datos y tablas [8 | 13]

- Agregado / borrado de columnas:

```
ALTER TABLE Socio
  ADD domicilio VARCHAR(30) null[;]

ALTER TABLE Socio
  DROP COLUMN foto[;]
```

Modificación de tabla en SQL Server y MySQL

Creación de tipos de datos y tablas [9 | 13]

- Generación automática de valores:
  - **SQL Server**: se puede emplear la propiedad `IDENTITY` o la función `NEWID()` con el tipo de datos `UNIQUEIDENTIFIER`
  - **MySQL**: se puede emplear la propiedad `AUTO_INCREMENT`

Creación de tipos de datos y tablas [10 | 13]

- Propiedad `IDENTITY`:
  - Se fija el valor de la semilla
  - Se fija el valor del incremento
  - Ej:

```
CREATE TABLE Socio (
 idSocio idSocio IDENTITY(1, 1) NOT NULL,
 ...
```


- Se puede usar `@@IDENTITY` para determinar el valor más reciente.

Creación de tipos de datos y tablas [11 | 13]

- Función `NEWID()`:
  - Se aseguran valores globales únicos
  - Se usa con la restricción `DEFAULT`.
  - Ej:

```
CREATE TABLE Socio (
  idSocio UNIQUEIDENTIFIER NOT NULL DEFAULT NEWID(),
  ...
```

Creación de tipos de datos y tablas [12 | 13]

- Propiedad `AUTO_INCREMENT`:
  - Ej:

```
CREATE TABLE Socio (
  idSocio  INT AUTO_INCREMENT NOT NULL,
  ...
  PRIMARY KEY(idSocio)
```


- MySQL exige que la columna con la propiedad AUTO_INCREMENT sea o forme parte de la clave primaria.

- Se puede usar `LAST_INSERT_ID()` para determinar el valor más reciente.

Creación de tipos de datos y tablas [13 | 13]

- Consideraciones sobre las columnas identidad:
  - Cada tabla admite una sola
  - No se puede modificar el valor
  - SQL Server no admite valores nulos
  - Se usa sólo con tipos de datos numéricos
  - No asegura unicidad (usar índices)

Planificación de la capacidad [1 | 3]

- Al planificar una BD se debe tener en cuenta el espacio que ocupará en disco en forma de archivos:
  - Tablas
  - Registro de transacciones
  - Índices

- También ocupan espacio otros objetos que se crean en la BD: procedimientos almacenados, desencadenadores, vistas, etc).

Planificación de la capacidad [2 | 3]

- Factores a considerar:
  - Tamaño de la BD model (SQL Server) y cantidad de datos en las tablas
  - Número y tamaño de índices
  - Tamaño del registro de transacciones
  - Tamaño de las tablas del sistema

Planificación de la capacidad [3 | 3]

- Luego de planificar la cantidad de espacio que necesita la BD, se debe estimar la cantidad de espacio que ocuparán los datos en las tablas:
  - Se puede calcular estimando para cada tabla la cantidad de filas y el tamaño de cada una

Resumen [1 | 1]

- Tipos de Bases de Datos
- Objetos de Bases de Datos
- Arquitecturas de diseño de aplicaciones
- Actividades para implementar una BD
- Almacenamiento y Transacciones
- Creación y modificación de BDs
- Creación de tipos datos y tablas
- Planificación de la capacidad