

Conceptos Generales De Bases de Datos

UTN - FRBA
Ing. en Sistemas de Información
Gestión de Datos

Prof.: Ing. Juan Zaffaroni

Conceptos

Que es una Base de Datos?

Una BD es un conjunto de datos persistentes e interrelacionados que es utilizado por los sistemas de aplicación de una empresa, los mismos se encuentran almacenados en un conjunto independiente y sin redundancias.

Conceptos

Que es un Sistema de Administración de Base de Datos? (DBMS – Data Base Manager System)

Es un programa que permite administrar los contenidos de una/s base/s de datos almacenada en disco. También llamado Motor de Base de Datos.

El DBMS ofrece a los usuarios una percepción de la base de datos que está en cierto modo por encima del nivel del hardware y que maneja las operaciones del usuario expresadas en el nivel más alto de percepción.

El DBMS también interpreta y ejecuta todas los comandos SQL.

Entre los motores de Base de Datos más utilizados podemos nombrar los siguientes: Oracle, MS SqlServer, DB2, MySql, PostgreSQL, Informix, Sybase, entre otros.

Componentes de un Sistema de Base de Datos

DATOS

Integrados -> Minimiza Redundancias
Compartidos-> Múltiples usuarios accediendo en forma concurrente.

TECNOLOGÍA

Hardware
Sistema Operativo

PROGRAMAS/PROCESOS

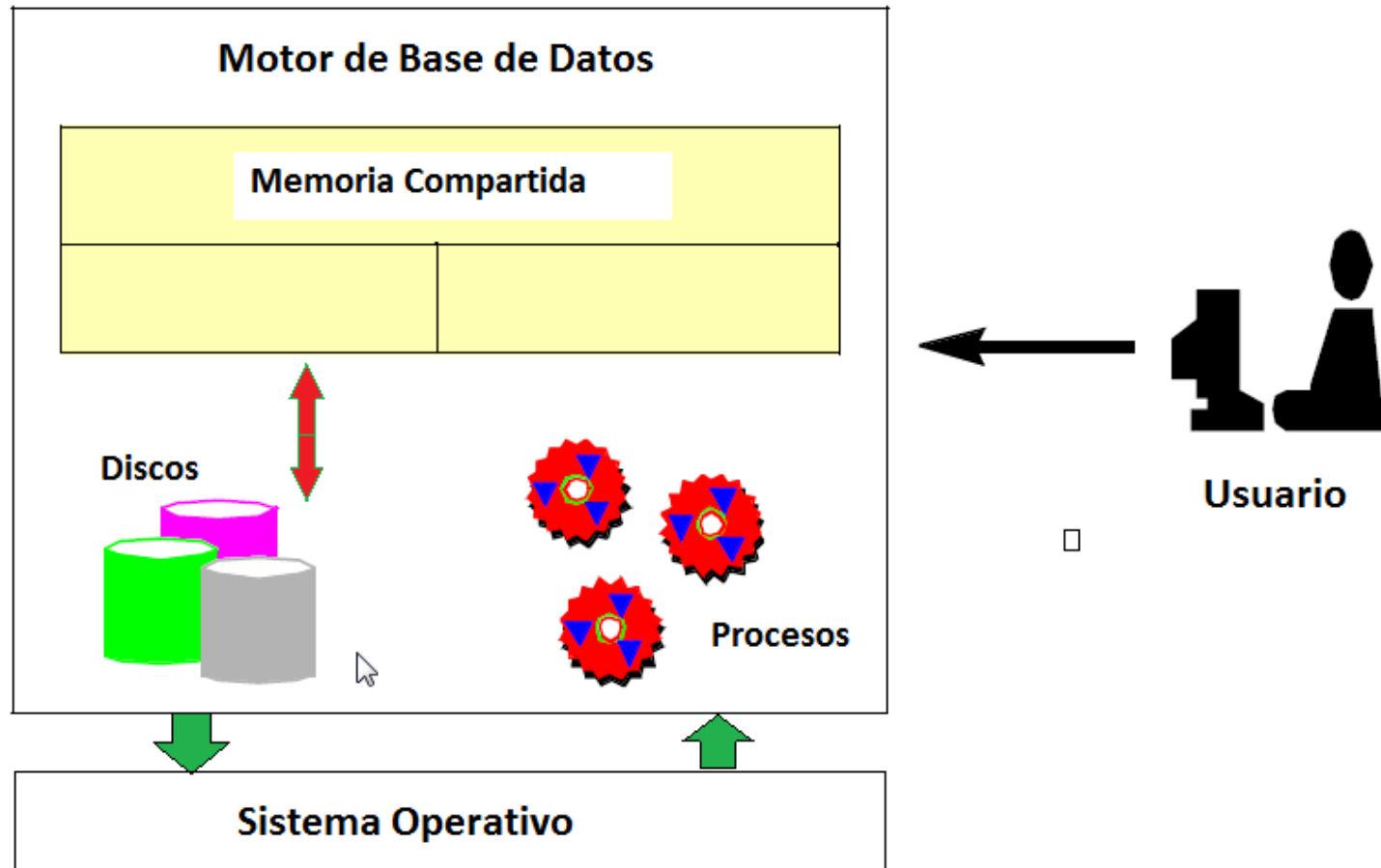
DBMS es el componente más importante.

Los procesos controlan todas las actividades del DBMS y abstraen a los usuarios de los detalles tecnológicos.

USUARIOS

Programador de Aplicaciones
DBA – DataBase Administrator
Usuarios Finales
Otros perfiles funcionales.

Componentes de un Sistema de Base de Datos



Conceptos

Los componentes principales de un DBMS son los siguientes:

Procesos

- Controlan el motor de Base de Datos
- Proveen funcionalidades específicas asociadas a seguridad, integridad, operación, organización interna entre otros.

Memoria Compartida

Es usada para

- Cachear datos del disco para tener un acceso más rápido..
- Mantener y controlar los recursos necesarios para los procesos.
- Proveer mecanismos de comunicación para los procesos clientes o propios del motor para conversar con otras aplicaciones y coordinar actividades.

Conceptos

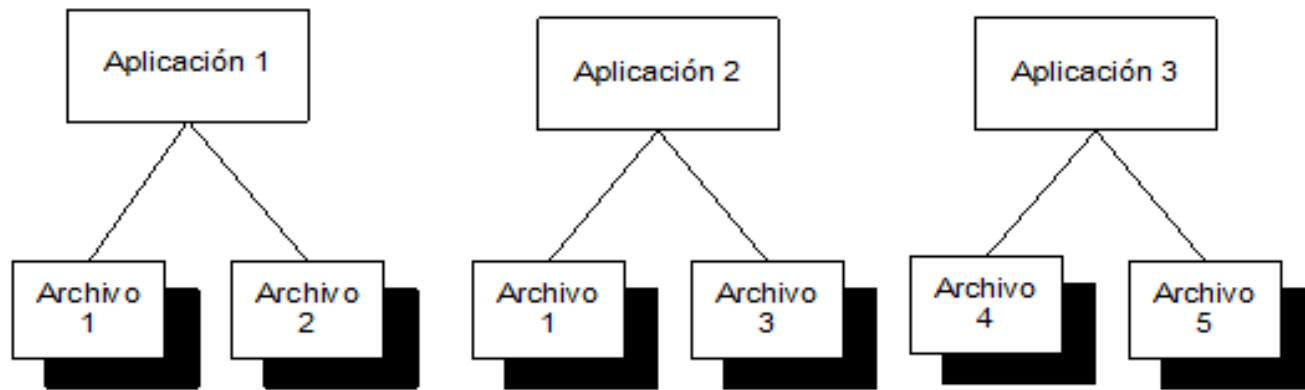
Los componentes principales de un DBMS (Cont.)

Unidades de Discos

- Es la colección de una o más unidades de espacio de disco asignadas al DBMS.
- Toda la información de las bases de datos, más la información propia del sistema necesarias para mantener el DBMS están almacenadas en en esta compomente.

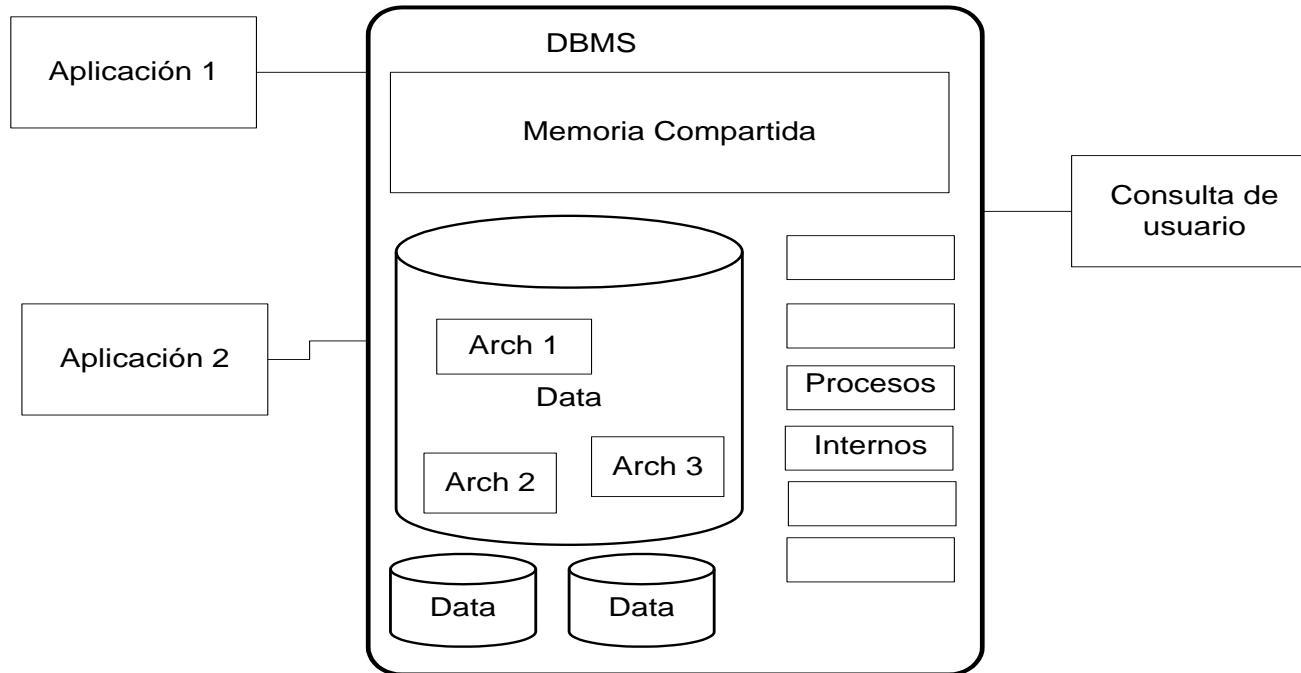
Aplicaciones Tradicionales

Enfoque orientado a la aplicación



En el ejemplo, supongamos un sistema simple que consta de tres programas o aplicaciones a la vista del usuario. Cada programa manipula a nivel físico los archivos, peticionando al S.O. las operaciones de lectura/escritura sobre los mismos. El desarrollador debe tener en cuenta durante el diseño y construcción factores relativos tales como: concurrencia entre la aplicación 1 y 2, ya que ambas utilizan el Archivo 1.

Enfoque de Base de Datos



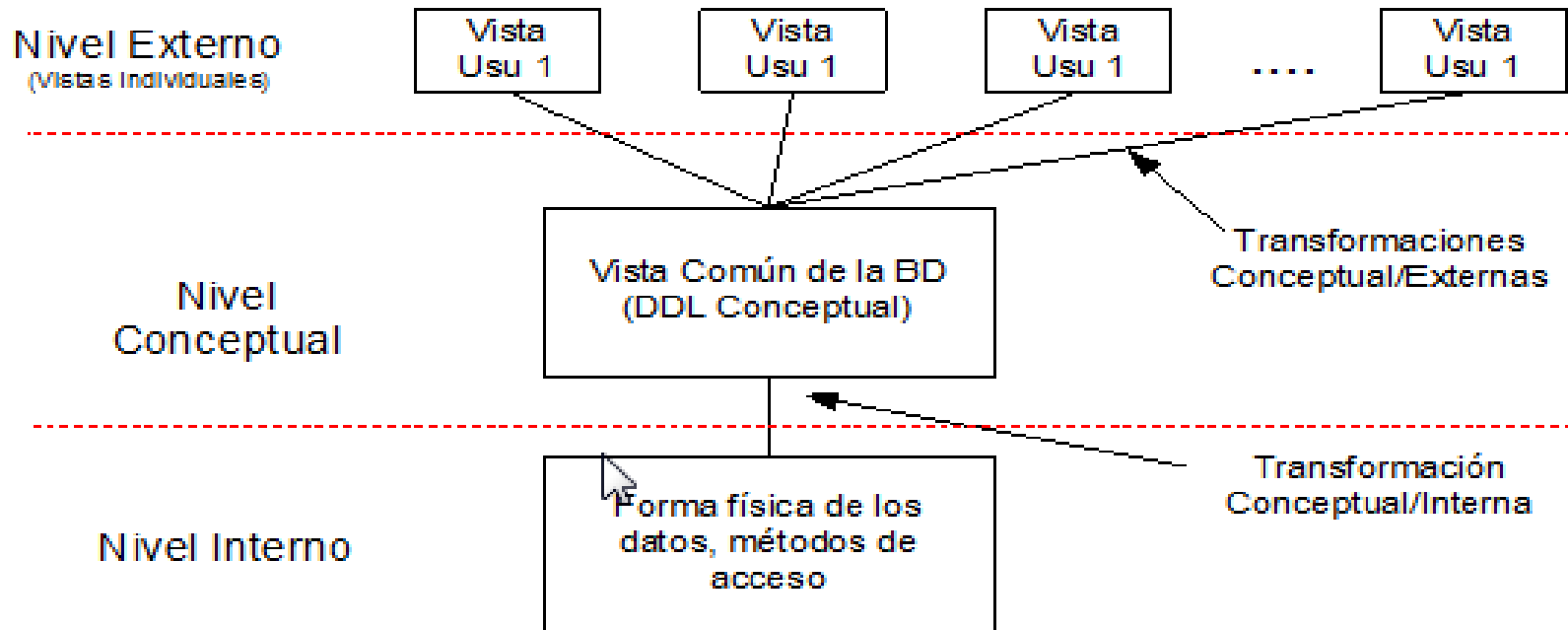
En este ejemplo los archivos de datos (tablas) residen en la Base de Datos, y los programas o consultas de usuarios son enviadas mediante algún mecanismo de comunicación estándar al DBMS, que es quien realiza realmente la consulta/actualización y devuelve el resultado al programa. El DBMS se encarga entre otras cosas de la seguridad, concurrencia, etc. Las peticiones no se hacen al S.O. sino al DBMS mediante instrucciones SQL.

Ventajas del Enfoque de Base de Datos

- Seguridad
- Estándares de documentación y normalización de nomenclaturas.
- Redundancia mínima:
- Consistencia de datos
 - Transacciones
- Concurrencia en acceso a datos.
- Integridad de los datos
 - Restricciones
 - Objetos de BD
- Criterios y normativas para organización de datos
- Independencia

Arquitectura de un sistema de bases de datos (ANSI/SPARC)

Esta Arquitectura fue propuesta por el Grupo de Estudio en Sistemas de Administración de Bases de Datos de ANSI/SPARC, la misma se ajusta bastante bien a la mayoría de los sistemas de BD



Arquitectura de un sistema de bases de datos (ANSI/SPARC)

Nivel externo o VISTAS

Esta es la percepción que tienen los usuarios respecto de la BD. Los usuarios pueden ser programadores, usuarios finales, o el DBA.

La percepción que tenemos como usuarios de una base de datos es solamente una vista externa.

Una vista externa es el contenido de una Base de datos como lo ve algún usuario en particular (para este usuario la vista es la Base de Datos).

Nivel conceptual o LOGICO

Representa de una forma 'entendible' de toda la información contenida en una base de datos. En lugar de describir datos personales, describe a los datos de toda la organización.

La vista conceptual se define mediante un esquema conceptual. Este esquema conceptual se escribe en DDL. Contiene definiciones del contenido de la base, , tipos de datos, restricciones, reglas de integridad, etc.

Arquitectura de un sistema de bases de datos (ANSI/SPARC)

Nivel interno o FISICO

En este nivel se define como se almacenan los datos en disco, es una representación de bajo nivel de toda la base de datos. Por ejemplo, se especifican las estructuras internas de disco y memoria, se definen métodos de acceso, en que secuencia física se encuentran los registros, entre otras.

Transformación Externa/Conceptual

Dada una determinada vista externa, esta transformación define la correspondencia entre dicha vista externa y la vista conceptual.

Transformación Conceptual / Interna.

Define la correspondencia entre la vista conceptual y la base de datos almacenada, y especifica la representación en el nivel interno de las filas y columnas del modelo conceptual.

Funciones del motor de base de datos y del DBMS en su conjunto

Diccionario de Datos

Es un conjunto de tablas de la base de datos del sistema, que define cada uno de los objetos dentro del mismo. Es lo que se llama 'metadatos'.

Estas tablas no pueden ser alteradas por ningún usuario de la base de datos. Por ejemplo de una Tabla, tiene almacenado todos los parámetros propios de la definición de la tabla, sus campos y sus restricciones (constraints), así como las relaciones entre las tablas.

Algunas sentencias SQL relacionadas con esta funcionalidad

- CREATE – Creación de objetos de BD (Tablas, índices, vistas, etc.)
- ALTER – Modificación de objetos de BD
- DROP – Eliminación de objetos de BD

Funciones del motor de base de datos y del DBMS en su conjunto

Control de la seguridad

La seguridad involucra la autenticación: asegurar que el usuario existe y su clave es correcta permitiendo el acceso a la BD y la autorización determinar los permisos que tiene el usuario para ejecutar acciones sobre diferentes objetos de la BD.

Para ello los DBMS poseen lo que se llama el **catálogo**. El catalogo mismo esta formado por entidades e interrelaciones (por lo que en un esquema relacional van a ser tablas).

Entidades a tener en cuenta para administrar la seguridad:

- Usuarios (internos, externos, mixtos)
- Roles o Perfiles (built-in roles / user defined roles)
- Acciones posibles a realizar sobre los objetos.
- Objetos de la BD sobre los que se otorgarán permisos.

Algunas sentencias SQL relacionadas con la seguridad

- GRANT – Para otorgar permisos.
- REVOKE – Para revocar permisos

Otros objetos relacionados con la seguridad

- Vistas, Triggers, Stored Procedures, Sinónimos, Funciones.

Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos para garantizar la integridad de datos

El DBMS cuenta con distintos mecanismos para poder controlar la integridad de los datos que se contienen en la/s Base/s de Datos.

Para asegurar la integridad cuenta con distintas restricciones y objetos que pueden ser creados y a partir de ello el DBMS cuenta con procesos que se encarga de controlar que se cumplan dichas restricciones asegurando la integridad de los datos.

Otros objetos relacionados con la integridad

- CONSTRAINTS
 - UNIQUE, NOT NULL, CHECK, DEFAULT
 - PRIMARY KEY
 - FOREIGN KEY
- Triggers
- Indices (únicos)
- Views (with check option)
- Stored Procedures
- Funciones

Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos para garantizar la consistencia de datos

El DBMS cuenta con distintos mecanismos para poder asegurar la consistencia de los datos que existentes en la/s Base/s de Datos.

Conceptos relacionados con la consistencia de datos

- **TRANSACCIONES**

- Es un conjunto de sentencias SQL que se ejecutan atómicamente en una unidad lógica de trabajo. Partiendo de que una transacción lleva la base de datos de un estado correcto a otro estado correcto, el motor posee mecanismos de manera de garantizar que la operación completa se ejecute o falle, no permitiendo que queden datos inconsistentes.

- Cada sentencia de alteración de datos (insert, update o delete) es una transacción en sí misma (singleton transaction).

- **LOGS TRANSACCIONALES:** Es un registro donde el motor almacena la información de cada operación llevada a cabo con los datos

- **RECOVERY:** Método de recuperación ante caídas. (Ver slide 24)

Funciones del motor de base de datos y del DBMS en su conjunto

Transacciones (Cont.)

Para definir una transacción debemos definir un conjunto de instrucciones precedidas por la sentencia **BEGIN TRANSACTION**, de esta manera las sentencias a continuación se ejecutarán de forma atómica. Pero para lo que es el concepto de transacción, una transacción puede finalizar correctamente o puede fallar; en el caso de finalizar correctamente, todos los datos se actualizarían en la base y en caso de fallar se deberían deshacer todos los cambios hasta el comienzo de la transacción. Para manejar estas acciones contamos con la sentencia **COMMIT TRANSACTION** para actualizar los datos en la base de datos, y con la sentencia **ROLLBACK TRANSACTION** para deshacer nuestra transacción.

(Ej de Sintaxis en SqlServer)

Funciones del motor de base de datos y del DBMS en su conjunto

Transacciones (Cont.)

Ejemplo:

```
--Declaración de variables
DECLARE @IMPORTE REAL;
DECLARE @FACTURA INT;
DECLARE @FECHA VARCHAR(8);
DECLARE @CLIENTE INT;

--Harcodeo de inicialización
SET @FACTURA = 353;
SET @FECHA = '20120601';
SET @CLIENTE = 15;

BEGIN TRANSACTION --Comienzo de transacción

INSERT INTO entregas(numero, fecha, cliente)
VALUES (@FACTURA, @FECHA, @CLIENTE)

SET @IMPORTE = Calcular_Importe(@FACTURA)

IF @IMPORTE > 0 THEN
    INSERT INTO facturas(n_factura, imp_total)
    VALUES (@FACTURA, @IMPORTE)
    COMMIT TRANSACTION --Se insertan los datos en la base
ELSE
    PRINT 'ERROR! IMPORTE NO VALIDO'
    ROLLBACK TRANSACTION --Se vuelven al punto de inicio
END
```

Para nuestro ejemplo, se realiza una inserción en la tabla de entregas con la factura entregada y luego se calcula el importe a partir de una función. Si el importe es mayor a 0 la factura se inserta en la tabla de facturas y la transacción se confirma con un *commit*; caso contrario se imprime el mensaje de error y la transacción se vuelve al estado inicial mediante un *rollback*.

Funciones del motor de base de datos y del DBMS en su conjunto

Transacciones (Cont.)

Algunos motores de base de datos permiten el manejo de Transacciones Anidadas.

Ejemplo:

```
CREATE TABLE #numeros (num int)
```

```
BEGIN TRAN T1
```

```
    INSERT INTO #numeros VALUES (1)
```

```
    BEGIN TRAN T2
```

```
        INSERT INTO #numeros VALUES (2)
```

```
    COMMIT TRAN --Confirma T2
```

```
ROLLBACK TRAN--Deshace T1 que contiene a T2; entonces también la deshace
```

En nuestro caso, la transacción T2 finaliza correctamente dentro de T1; pero en el momento de confirmar la transacción T1 se produce un rollback que deshace a T2 volviendo a la tabla a un estado inicial sin valores cargados.

(Ej de Sintaxis en SqlServer)

Funciones del motor de base de datos y del DBMS en su conjunto

Transacciones (Cont.)

Algunos motores de base de datos permiten establecer puntos intermedios de guardado de información.

Ejemplo:

```
CREATE TABLE #numeros (num int)

BEGIN TRAN

INSERT INTO #numeros VALUES (2)

SAVE TRAN N2 --Guardo estado actual a N2

    BEGIN TRAN
        INSERT INTO #numeros VALUES (3)
    ROLLBACK TRAN N2 --Deshago la transacción actual hasta N2

INSERT INTO #numeros VALUES (4)

COMMIT TRAN
```

Es posible realizar más de un *Save Tran* en cada transacción y se puede elegir a cual se desea volver en cual momento.

(Ej de Sintaxis en SqlServer)

Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos de recuperación (RECOVERY)

Es un mecanismo provisto por los motores de Base de Datos que se ejecuta en cada inicio del motor de forma automática como dispositivo de tolerancia a fallas.

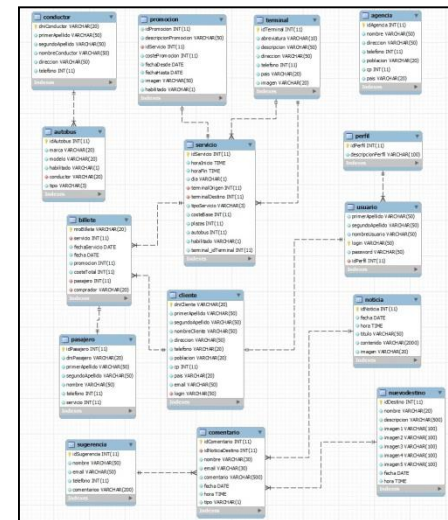
Sus objetivos son los siguientes:

- Retornar al Motor de Base de datos al punto consistente más reciente. (checkpoint = punto en el que el motor sincronizó memoria y disco)
- Utilizar los logs transaccionales para retornar el motor de base de datos a un estado lógico consistente, realizando un "rolling forwards" de las transacciones ocurridas con éxito luego del checkpoint más reciente y realizar un "rolling back" de las transacciones que no hayan sido exitosas.

Propiedades de un RDBMS

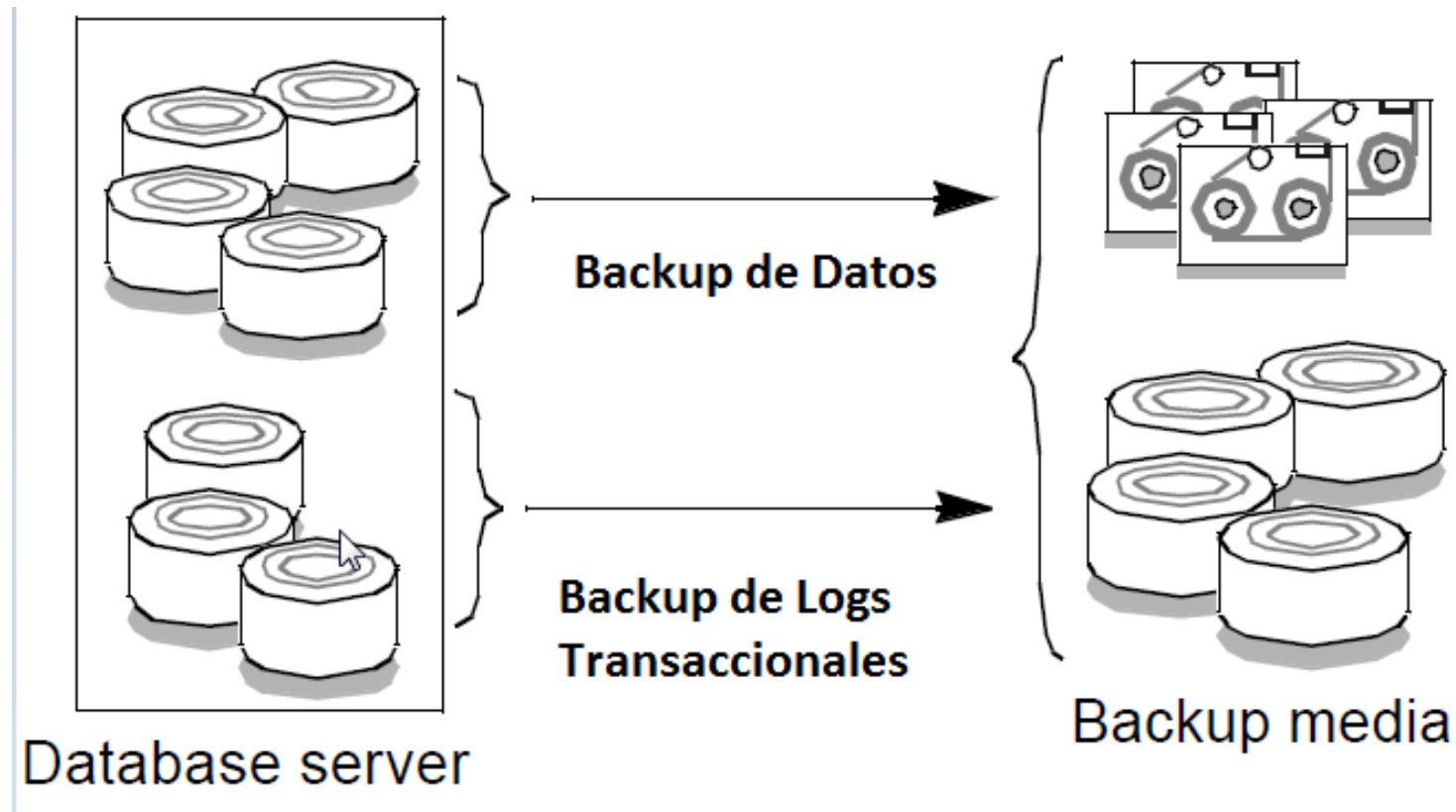
- Foco en la ejecución de **Transacciones**

- **A**tomicidad
- **C**onsistencia
- **I**solation (Aislamiento)
- **D**urabilidad



Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos de resguardo y restauración (BACKUP y RESTORE)



Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos de resguardo y restauración (BACKUP y RESTORE)

Los motores de Bd cuentan con distintas herramientas que permiten realizar estas acciones las cuáles son utilizadas generalmente en empresas medianas y chicas. Existen herramientas de terceras partes especializadas en el tema las cuáles son muy utilizadas sobre todo en grandes empresas. (Por ej. Veritas, IBM Tivoli)

Backup es la copia total o parcial de la información de una base de datos la cual debe ser guardada en algún otro sistema de almacenamiento masivo.

Restore es la acción de tomar un back up ya realizado y restaurar la estructura y datos sobre una base dada.

Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos de resguardo y restauración (BACKUP y RESTORE)

De acuerdo a su clasificación podemos distinguir los siguientes tipos:

Backup diferencial acumulativo / incremental: solo se guardan las modificaciones a partir de cierta fecha.

Backup completo: se guardan todos los datos.

Backup en caliente: se realiza mientras el aplicativo está en uso.

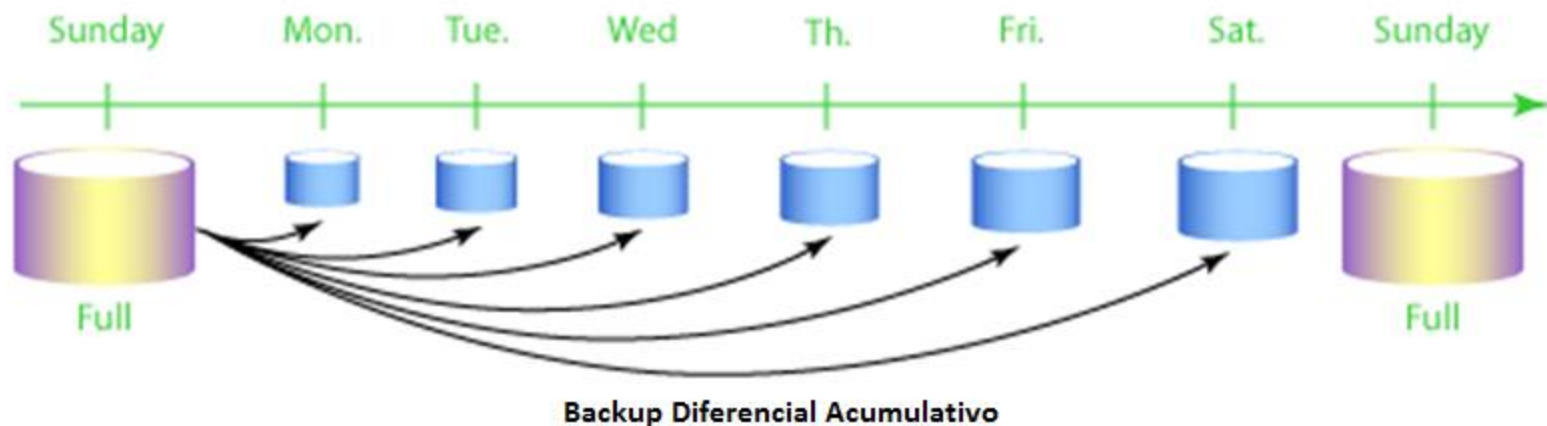
Backup en frío: se realiza con el aplicativo bajo.

Backup de Logs Transaccionales: se realizan sobre los logs de transacciones.

Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos de resguardo y restauración (BACKUP y RESTORE)

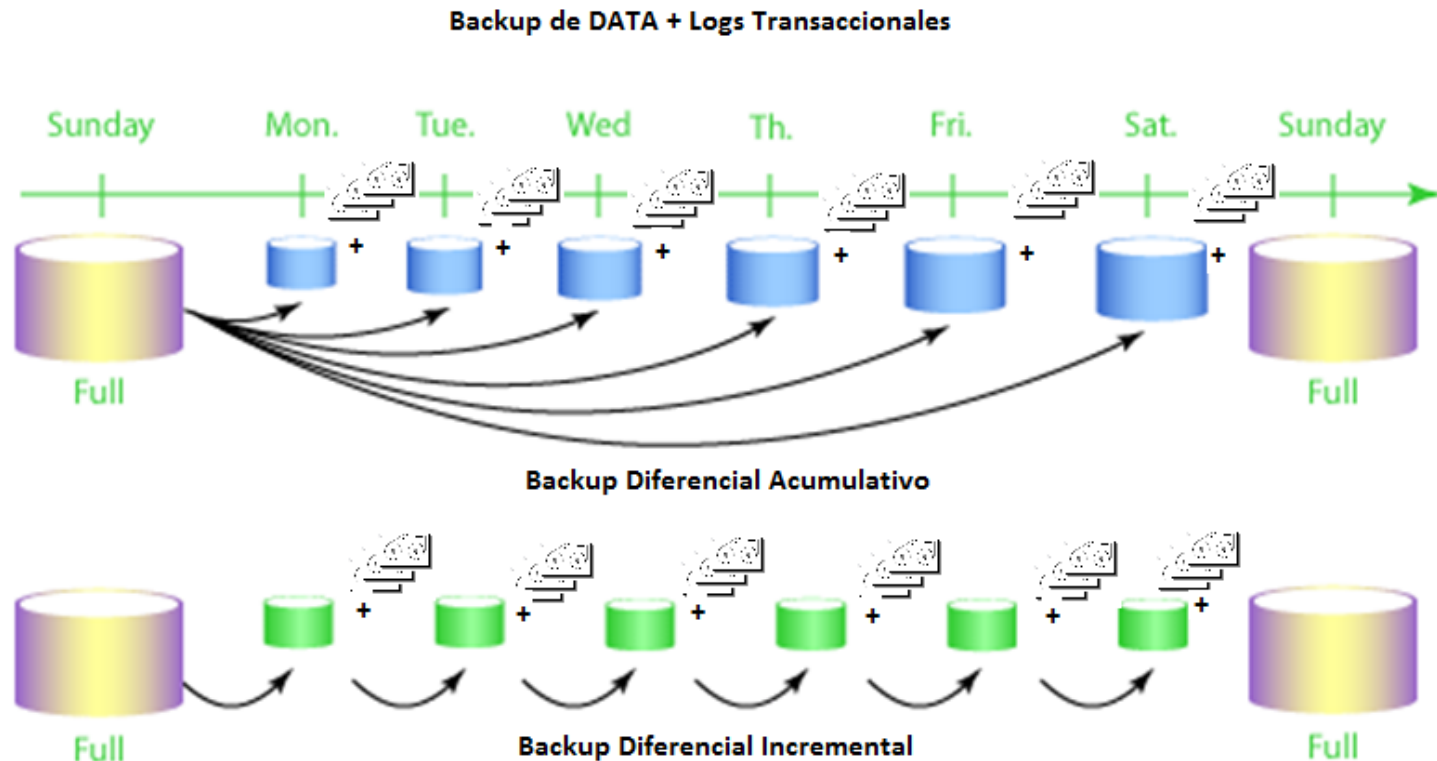
POLITICAS DE BACKUP



Funciones del motor de base de datos y del DBMS en su conjunto

Mecanismos de resguardo y restauración (BACKUP y RESTORE)

POLITICAS DE BACKUP



Funciones del motor de base de datos y del DBMS en su conjunto

Control de Concurrency

Los motores permiten controlar el acceso concurrente a sus recursos a través de bloqueos y de la definición de niveles de aislamiento (isolation levels)

Bloqueos

- Granularidad de Bloqueos
 - Nivel de Base de Datos.
 - Nivel de Tabla
 - Nivel de Página
 - Nivel de Fila
 - Nivel de Clave de Índice
- Tipos de Bloqueos
 - Compartido (Shared)
 - Exclusivo (Exclusive)
 - Promovible (Promotable-Update).

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento

Read uncommitted.

El read uncommitted no chequea lockeos por select en la tablas a consultar, lo que mejora el rendimiento pero afecta la integridad en cuanto a que existen **lecturas sucias** (*datos actualizados en una transacción que luego se deshacen por un rollback*) y **no existen lecturas repetibles** (*en la misma transacción poder hacer dos veces la misma consulta asegurando siempre el mismo resultado*).

Pueden existir lecturas sucias, lecturas repetibles, y lecturas fantasmas (Filas que aparecen durante la transacción que fueron insertadas en otra transacción concurrente).

Ejemplo User 1
 CREATE TABLE ##nums (valor INT)
 INSERT INTO ##nums VALUES (1)

 User 2
 SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento

Read uncommitted. (Cont.)

	USER 1	USER 2
1	CREATE TABLE ##nums (valor INT)	
2	INSERT INTO ##nums VALUES (1)	
3		SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
4	BEGIN TRANSACTION T1	
5		BEGIN TRANSACTION T2
6	INSERT INTO ##nums VALUES (2)	
7	INSERT INTO ##nums VALUES (3)	
8	INSERT INTO ##nums VALUES (4)	
9		SELECT * FROM ##NUMS
10	ROLLBACK TRANSACTION T1	
11		SELECT * FROM ##NUMS
12		
13		COMMIT TRANSACTION T2

La transacción T1 se deshace, por lo que la inserción de valores 2, 3 y 4 no se realiza. Pero la lectura de T2 a la tabla de ##nums devuelve lecturas sucias en el instante 9, ya que no se establecen bloqueos.

Además se observa que en el momento 11 se realiza la misma consulta dando valores distintos, observando que no se asegura tampoco una lectura repetible.

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento (Cont.)

Read committed. *(Por Defecto en SQLServer)*

El read committed asegura que sólo leerá datos confirmados por otra transacción, o sea realizará lecturas sucias.

Pero si bien el read commit ante una lectura de datos chequea la existencia de lockeos exclusivos en la tabla a consultar, no asegura lecturas repetibles. Ya que una vez que leyó los datos, los datos podrían ser lockeados o modificados por otra transacción concurrente.

¿Qué significa lecturas repetibles?

En una misma transacción durante su transcurso puede tener dos llamadas a un mismo select, las cuales arrojarían resultados distintos.

User 1

```
CREATE TABLE ##nums (valor INT)
```

```
INSERT INTO ##nums VALUES (1)
```

User2

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```


Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento Read committed. (Cont.)

	USER 1	USER 2
1	CREATE TABLE ##nums (valor INT)	
2	INSERT INTO ##nums VALUES (1)	
3		SET TRANSACTION ISOLATION LEVEL READ COMMITTED
4	BEGIN TRANSACTION T1	
5		BEGIN TRANSACTION T2
6	INSERT INTO ##nums VALUES (2)	
7	INSERT INTO ##nums VALUES (3)	
8	ROLLBACK TRANSACTION T1	SELECT * FROM ##NUMS
9	BEGIN TRANSACTION T3	
10	INSERT INTO ##nums VALUES (4)	
11		SELECT * FROM ##NUMS
12	COMMIT TRANSACTION T3	
13		COMMIT TRANSACTION T2

En nuestro ejemplo, el hilo de la transacción 2 queda bloqueado en el instante 8; ya que intenta poner un lockeo compartido para leer los datos insertados por T1 y queda a la espera debido a que están lockeados de forma exclusiva. En el instante 8 cuando la transacción hace el rollback, se desbloquea T2 y devuelve únicamente los datos confirmados; en este caso sólo el valor 1. Para el segundo caso de nuestro ejemplo, en el instante 11 el hilo de T2 queda bloqueado esperando la confirmación de la transacción T3 que sucede en el instante 9. Por lo que en el instante 12 T2 devuelve los valores 1 y 4 commiteados.

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento

Repeatable read

Las lecturas repetibles aseguran que no existan lecturas sucias y que las lecturas puedan ser repetibles, pero no evitan las lecturas fantasmas.

¿Qué quiere decir esto?

El nivel de aislamiento repeatable read establece lockeos en los selects para todos los registros consultados, durante la duración de la transacción. Cuando otra transacción intenta modificar o borrar algún registro que esté lockeado por este select quedará en espera. Pero las lecturas fantasmas podrán aparecer, ya que la inserción de nuevos registros por una segunda transacción no asegurará el lockeo de los mismos hasta que no sean nuevamente consultados por la transacción original.

Su uso es muy particular; se tiende a utilizar este tipo de nivel de aislamiento para conservar dentro de una misma transacción dos resultados iguales en un select a un cierto rango de números.

Ejemplo **User 1**

```
CREATE TABLE ##nums (valor INT)
```

```
INSERT INTO ##nums VALUES (1)
```

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
```

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento

Repeatable read. (Cont.)

	USER 1	USER 2
1	CREATE TABLE ##nums (valor INT)	
2	INSERT INTO ##nums VALUES (1)	
3	SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	
4	BEGIN TRANSACTION T1	
5		BEGIN TRANSACTION T2
6	SELECT * FROM ##NUMS	
7		UPDATE ##NUMS SET valor=2 WHERE valor=1
8	SELECT * FROM ##NUMS	
9		
10	COMMIT TRANSACTION T1	
11		COMMIT TRANSACTION T2

En nuestro ejemplo, la lectura del instante 6 realiza un SELECT a toda la tabla estableciendo lockeos en todos los registros que lee. En el instante 7, existe una instrucción de UPDATE la cual bloqueará el hilo de ejecución T2 ya que quiere modificar un registro lockeado por la T1. En el instante 8 realizará la misma consulta sobre los valores de prueba y arrojarán mismo resultado. Una vez confirmada la transacción T1 en el instante 8, el hilo de ejecución T2 se liberará y podrá actualizar los datos. Por lo que este nivel de aislamiento nos asegurará lecturas no sucias y repetibles.

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento

Serializable Read

El tipo de lectura serializable es el único que asegura que no existan lecturas sucias, lecturas fantasmas y que las lecturas puedan ser repetibles. Se realizará un bloqueo de un rango de índice, según lo que se coloque en el where, y si no es posible bloqueará toda la tabla.

El problema es que se aplicará un nivel de bloqueo que puede afectar a los demás usuarios en los sistemas multiusuario.

Ejemplo: **User 1**

```
CREATE TABLE ##nums (valor INT)
```

```
INSERT INTO ##nums VALUES (1)
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

Funciones del motor de base de datos y del DBMS en su conjunto

Niveles de Aislamiento Serializable read. (Cont.)

	USER 1	USER 2
1	CREATE TABLE ##nums (valor INT)	
2	INSERT INTO ##nums VALUES (1)	
3	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	
4	BEGIN TRANSACTION T1	
5		BEGIN TRANSACTION T2
6	SELECT * FROM ##NUMS	
7		INSERT INTO ##NUMS VALUES (2)
8		COMMIT TRANSACTION T2
9	SELECT * FROM ##NUMS	
10	COMMIT TRANSACTION T1	

En nuestro ejemplo, la lectura del instante 6 se realiza un SELECT a toda la tabla estableciendo lockeos en la tabla leída o el índice correspondiente.

En el instante 7, existe una instrucción de INSERT la cual bloqueará el hilo de ejecución T2 ya que quiere insertar un registro. El instante 9 realizará la misma consulta sobre los valores de prueba y arrojarán mismo resultado. Una vez confirmada la transacción en el instante 10, el hilo de ejecución T2 se liberará y podrá actualizar los datos. Por lo que este nivel de aislamiento nos asegurará lecturas no sucias, no permite registros fantasmas y lecturas repetibles.

Funciones del motor de base de datos y del DBMS en su conjunto

ISOLATION LEVELS – Resumen

	Dirty Read	Repeatable Read	Phantom Read
Read Uncommitted	SI	NO	SI
Read Committed	NO	NO	SI
Repeatable Read	NO	SI	SI
Serializable Read	NO	SI	NO

Dirty Read/ Lectura Sucia: Datos actualizados en una transacción concurrente que luego se deshacen por un rollback.

Phantom Reads/Lecturas Fantasma: Filas que aparecen durante la transacción que fueron insertadas en otra transacción concurrente.

Lectura Repetible: En la misma transacción poder hacer dos veces la misma consulta asegurando siempre el mismo resultado.

Funciones del motor de base de datos y del DBMS en su conjunto

Manejo de Lockeos - Deadlock

Para nuestro siguiente caso, tenemos un deadlock. Cada transacción quiere leer de datos no confirmados de otra.

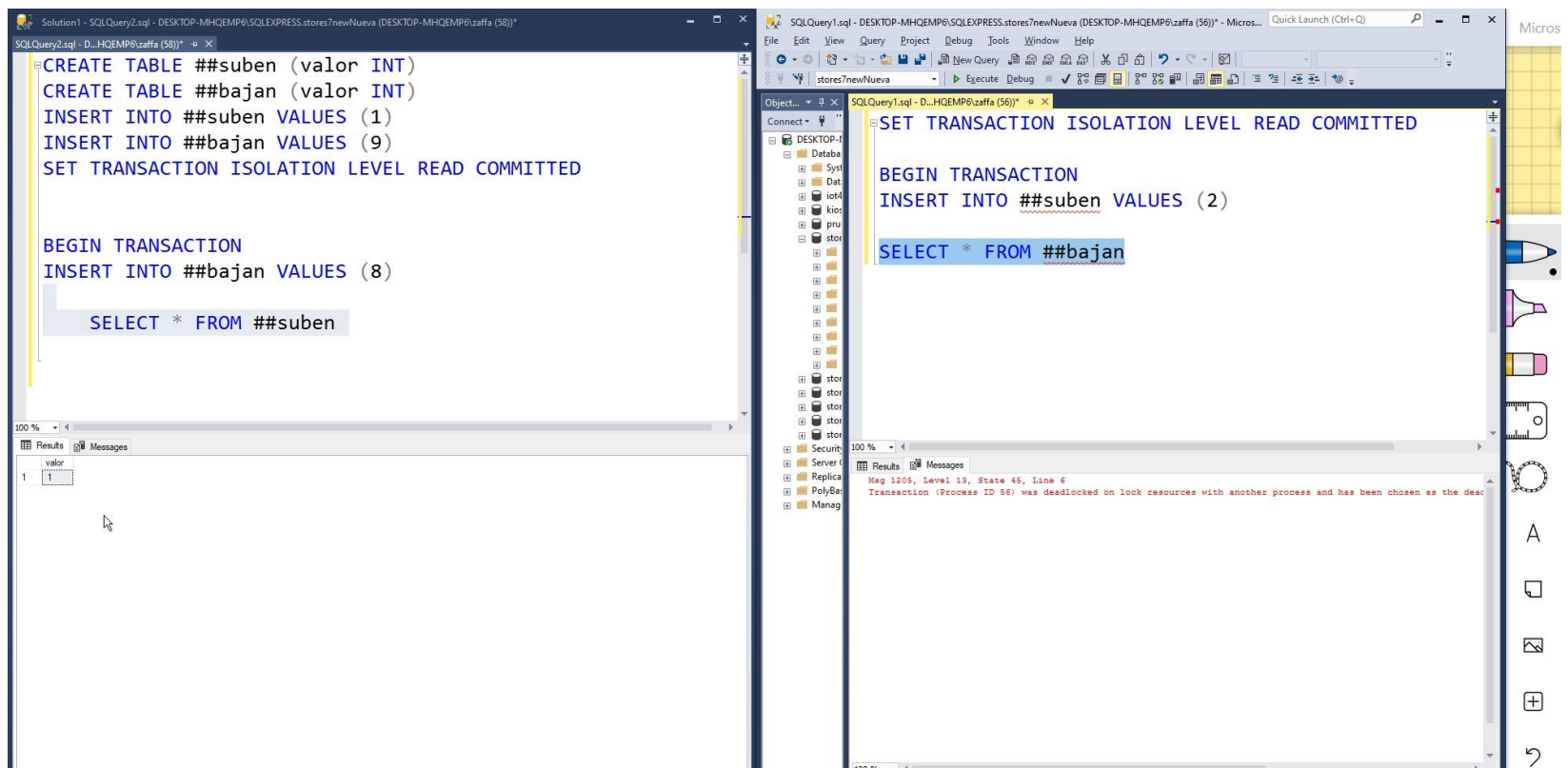
```
CREATE TABLE ##suben (valor INT)
CREATE TABLE ##bajan (valor INT)
INSERT INTO ##suben VALUES (1)
INSERT INTO ##bajan VALUES (9)
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```

1	BEGIN TRAN T1	
2		BEGIN TRAN T2
3	INSERT INTO ##bajan values (8)	
4		INSERT INTO ##suben VALUES (2)
5	SELECT * FROM ##suben	
6		SELECT * FROM ##bajan
7	COMMIT TRAN T1	
8		COMMIT TRAN T2

Funciones del motor de base de datos y del DBMS en su conjunto

Manejo de Lockeos - Deadlock

Se observa que el motor mata a la transacción mas nueva y que generó el deadlock.



The image displays two side-by-side SQL Server Enterprise Manager windows illustrating a deadlock scenario.

Left Window (SQLQuery2.sql - D:\HQUEMP6\zaffa (58)):

```
CREATE TABLE ##suben (valor INT)
CREATE TABLE ##bajan (valor INT)
INSERT INTO ##suben VALUES (1)
INSERT INTO ##bajan VALUES (9)
SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRANSACTION
INSERT INTO ##bajan VALUES (8)

SELECT * FROM ##suben
```

Right Window (SQLQuery1.sql - D:\HQUEMP6\zaffa (56)):

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRANSACTION
INSERT INTO ##suben VALUES (2)

SELECT * FROM ##bajan
```

The right window's Messages pane shows the following error:

Msg 1206, Level 13, State 45, Line 6
Transaction (Process ID 56) was deadlocked on lock resources with another process and has been chosen as the deadlocked transaction. Rollback initiated. Information about the deadlock and deadlocked transactions is stored in the system log.

Funciones del motor de base de datos y del DBMS en su conjunto

Facilidades de auditoría.

A los DBMS se les puede activar la opción de guardar en un log un registro del uso de los recursos de la base para auditar posteriormente.

Logs del sistema

Mediante este tipo de logs, el DBA puede llegar a determinar cual fue, por ejemplo, el problema que produjo una caída del sistema.

Acomodarse a los cambios, crecimientos, modificaciones del esquema.

El motor permite realizar cambios a las tablas constantemente, casi en el mismo momento en que la tabla está siendo consultada.

Otras Funcionalidades

Mirroring de Discos

Data Replication

Data Encryption

Monitoring Features

Event Alarms

Optimizador de Queries

Particionamiento de Tablas e Índices