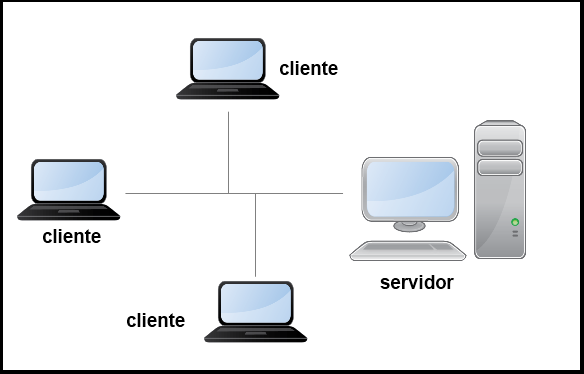
|  |  |
| --- | --- |
| INFORMACIÓN BÁSICA TEÓRICA | |
| NOMBRE DEL TEMA: **Transacciones en bases de datos multiusuario** | SUBTEMA: Fundamentos |
| **ASIGNATURA:** Teoría de Bases de Datos | |
| TIEMPO*:*  TRABAJO GRUPAL: TRABAJO INDIVIDUAL: x | |
| **CONTENIDO DE LA GUÍA** | |
|  | |
|  | |

**Introducción**

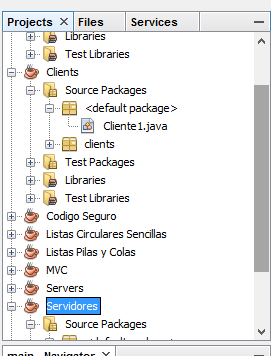
La idea es muy sencilla. Queremos que un computador de la empresa realice las funciones de servidor de la base de datos, al que de ahora en adelante llamaremos servidor. Este ordenador debe ser accesible por todos los ordenadores a través de la red de la empresa, que puede ser una red por cable o inalámbrica; a estos ordenadores los llamaremos clientes.

**Esquema:**



De este modelo podemos simplemente conectar un Servidor y mínimo un cliente utilizando dos equipos físicos pro medio de Wifi. En el caso de nosotros, es crear dos proyectos en java, uno que se llame Servidor y otro cliente de acuerdo a las guías que se han venido trabajando y de igual forma separar los dos proyectos. Uno en un servidor y otro en un cliente con conectividad por medio de sockets como los creamos en el primer corte.

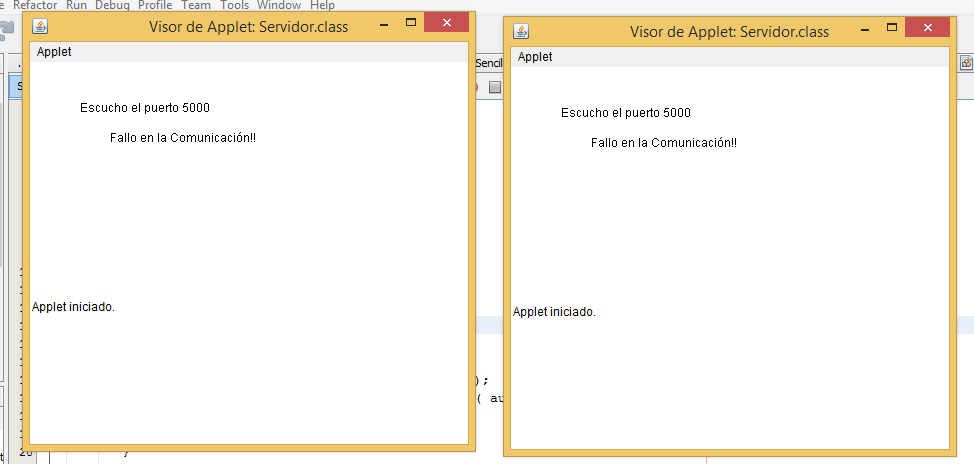
El esquema sería: Dos Apllet uno en el computador Servidor y el otro Applet en el computador Cliente, ver figura No. 1.



En la figura hay un proyecto Clients donde se encuentra un Applet llamado Cliente1.java, y otro con el nombre Servidor en el proyecto Servidores.

No obstante, como se dijo anteriormente, los proyectos se deben separar cada uno formará parte de un computador físico. Luego deben tener, como primera medida cerciorarse que se encuentren conectados haciendo ping desde cada equipo.

Una vez conectados, verifican que se pueda enviar datos desde el cliente al servidor desde los Applet. Ver figura No.2



El código que crea estos dos Applet ya lo tienen en el aula virtual, son las guías con las que venimos trabajando. Ahora, Se debe crear en cada Applet una Interface de usuario para realizar solicitudes desde el Cliente al Servidor que este puede responder (Solicitud respuesta).

Bien, para crear las interfaces de usuario pueden utilizar las clase de Swing o de AWT (deben consultar en el libro de Cesar Becerra que se encuentra en las guías).

Ahora, para entrar a los conceptos de bases de datos multiusuario, el código de cada Applet debe tener conexión a una Base de datos puede ser MySQL, en esta parte se debe revisar cómo se hace conexión a una BD utilizando java.

En consecuencia, como Uds, vienen trabajando en el aplicativo sobre la administración de la cuenta bancaria, ese aplicativo los deben ajustar al lado del servidor para que el cliente realice alguna consulta y el servidor responda, pero aquí es donde debe entrar los conceptos de Sistemas Transaccionales, entre ellos los que se encuentran en la guía que deben elaborar y que se encuentra en el aula virtual (Transacciones automáticas, Implícitas y ExplicitasTransacciones y recuentos de referencias, Iniciar una transacción, Confirmar una transacción, Revertir una transacción etc.)

Estos conceptos son los que Uds, deben consultar a manera de motivar el trabajo autónomo.

En estas consultas, van a encontrar que estas transacciones son propias del motor de bases de datos que utilicen, por ejemplo, si es SQL Server encontrarán lo siguiente:

Las transacciones implícitas permanecen abiertas hasta que se abre otra nueva transacción en la misma sesión o hasta que se ejecute la sentencia COMMIT TRANSACTION o ROLLBACK TRANSACTION.

Las transacciones explicitas son aquellas en donde se define de manera explícita el inicio y el fin de las mismas mediante las sentencias BEGIN TRANSACTION y COMMIT TRANSACTION (cuando finaliza sin errores) o ROLLBACK TRANSACTION (cuando se presenta algún error).

BEGIN TRANSACTION

INSERT INTO PRODUCTOS (ID, NOMBRE, VALOR) VALUES (1,’TORNILLOS’,500)

DELETE FROM DETALLE\_PEDIDOS WHERE ID\_PRODUCTO = 1

COMMIT TRANSACTION

La sentencia COMMIT TRANSACTION se encarga de confirmar la transacción y todas las operaciones Transact Sql que la conforman, por el contrario la sentencia ROLLBACK TRANSACTION se encarga de deshacer la transacción y todas las operaciones Transact Sql que la conforman.

Ya es cuestión que Uds, averigüen cómo se realiza en MySQL o Postgrees SQL etc. Recueden que deben seguir el diligenciamiento de la guía propuesta para que adicionen todo el desarrollo ingenieril con una metodología adecuada.

Este, es todo el desarrollo del trabajo para el segundo corte. A continuación les dejo un ejemplo de cómo funciona la arquitectura de transacciones en SQL Server iniciando con los conceptos de un motor de bases de datos que obviamente Uds ya conocen.

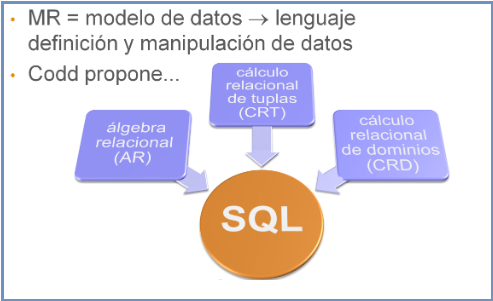
**Conceptos de un Motor de Base de Datos Relacional (BDR)**

Un motor de Base de Datos Relacional (BDR) es un sistema de software que administra, almacena y ofrece información de un conjunto de datos estructurados en Entidades y almacenados en Tablas. Los datos están organizados en Listas de Entidades. Cada Entidad consiste en una fila (o registro) en la tabla, el cual representa una instancia de información.  
Los atributos (o campos) de cada Entidad están definidos en la estructura de columnas de la Tabla. En una tabla todas las entidades comparten la misma estructura. Usualmente todos los registros de una tabla se diferencian en el valor de una tupla de campos. Esta tupla de campos se define como Llave Primaria.

Los registros se recuperan en forma de Consultas.

Las Consultas son peticiones de datos que generan una lista de selección de Registros según un criterio de búsqueda.

Usualmente se usa un lenguaje de definición de consultas estructurado (SQL). Ver siguiente figura:



El modelo relacional, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas.

**Los componentes de un Motor de Base de Datos Relacional**  
Los componentes principales de un motor de Base de Datos son:

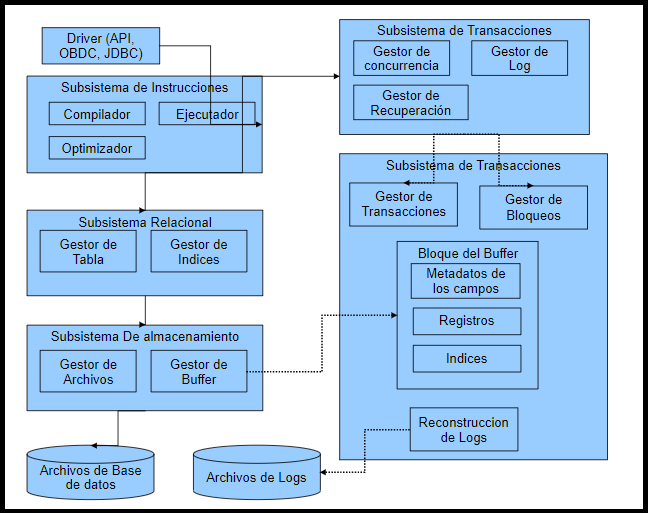
Drivers de Interface.

Subsistema de instrucciones SQL.

Subsistema de transacciones.

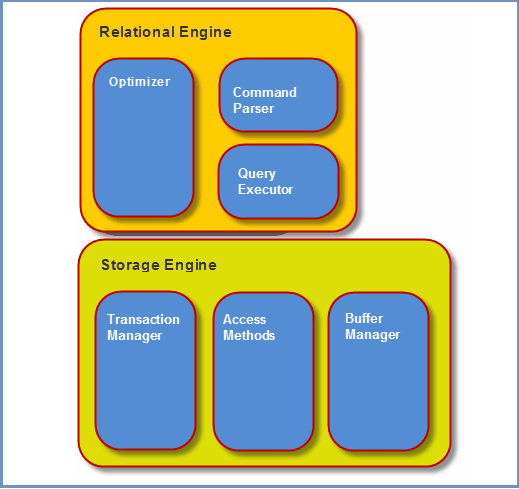
Subsistema relacional.

Subsistema de almacenamiento.

**Arquitectura de un sistema de BDR**  


**Como ejemplo tomaremos el motor de bases de datos SQL Server**

Esta básicamente formado por dos motores o “engines” internos. El primero es el motor relacional, o “Relational Engine”, el cual se encarga de procesar las consultas de la base de datos, optimizarlas y ejecutarlas. El segundo es el motor de almacenamiento, o “Storage Engine”, el cual es responsable de realizar todas las operaciones de I/O de la data, administrar las transacciones, y de manejar el “Buffer Pool”. A continuación se presenta un gráfico representando los dos motores y a sus componentes internos.

****

**Relational Engine**

Este es el primer motor interno de SQL Server, el cual está encargado de poder procesar las sentencias T-SQL enviadas desde los clientes. Este motor tiene a su vez tres componentes internos los cuales le permiten hacer su trabajo. A continuación explicaremos un poco más las funciones que tiene cada uno de estos componentes.

**Command Parser:** Este componente es el primero en recibir las sentencias T-SQL enviadas desde el cliente al servidor de base de datos. Lo primero que hace este componente es revisar que la sintaxis T-SQL sea correcta, si no lo es, enviara directamente un error al cliente. Si la sintaxis es correcta entonces el siguiente paso es generar un plan de ejecución, para poder realizar esto el “Command Parser”, primero genera un “hash” a partir de la sentencia T-SQL a procesar para poder determinar si es que ya existe algún plan que se ajuste a ésta dentro del “Plan Cache”, la cual es un área dentro del “Buffer Pool” utilizada para poder almacenar los planes de ejecución. Si no encuentra un plan adecuado, entonces genera un “Query Tree” basado en la sentencia T-SQL a procesar, un “Query Tree” es una estructura interna donde cada nodo en el árbol representa una operación en la sentencia SQL que necesita ser ejecutada. Una vez que se tiene generado el “Query Tree”, éste es pasado al siguiente componente, el “Query Optimizer”.

**Query Optimizer:** Este componente es la pieza más preciada y más compleja dentro de SQL Server. A este componente también se le denomina “Cost-BasedOptimizer”, debido a que evalúa múltiples formas de ejecutar una sentencia T-SQL, y escoge el método que le costó menos. La descripción anterior podría llevarnos a pensar que la función del “Query Optimizer” es encontrar el mejor plan para nuestra consulta, pero en realidad su función principal es encontrar un buen plan en un periodo de tiempo razonable, en vez de encontrar el mejor plan, se podría decir que el objetivo principal de este componente es encontrar el plan más eficiente, en relación al costo y al tiempo. Si el “Query Optimizer” intentara encontrar el \*mejor\* plan, quizás la generación del plan demoraría más que la misma ejecución de la consulta, entonces, es por esto que se intenta encontrar un balance entre costo y tiempo de generación del plan. Una vez que se tiene el plan de ejecución generado, éste es enviado al siguiente componente que es el “QueryExecutor”.

**Query Executor:** La función principal de este componente es ejecutar las consultas, para ser más específicos ejecutar los planes de ejecución interactuando directamente con el “Storage Engine” para obtener y modificar la data, y finalmente enviar el resultado al cliente.

**Storage Engine:** Una vez que el “Query Executor” comienza a ejecutar el plan de ejecución para la consulta, necesita trabajar en conjunto con el “Storage Engine” que es el encargado de manejar las operaciones de I/O de la data, además de las transacciones y el “Buffer Pool”. El “Query Executor” interactúa con este motor a través de uno de sus componentes que es el “Access Methods”.

**Access Methods:** Este componente tiene como objetivo fundamental el de permitir la comunicación entre el “Relational Engine” y el “Storage Engine”. Este componente recibe las instrucciones enviadas en el plan de ejecución y verifica si hay algún bloqueo con el “Transaction Manager”, si no es así procede con el “BufferManager”.

**Buffer Manager:** Este es el encargado de buscar las páginas de datos en el “BufferCache”, el cual es un área dentro del “Buffer Pool” donde se guardan las páginas de datos en la memoria, en el caso de que la pagina no se encuentre en esa sección de memoria, procederá a extraerla de la base de datos en el disco duro, y traerá a memoria las páginas de datos necesarias, una vez que ya tiene las paginas en el “Buffer Cache” recién envía el resultado devuelta al “AccessMethods”, el cual es el que forma el set de datos resultante y finalmente lo devuelve al componente “Query Executor” del “Relational Engine” para que este a su vez lo envíe al cliente. Es importante recordar que para que una página pueda ser leída y/o modificada, ésta debe encontrarse antes en la memoria del servidor “Buffer Cache”.

**Transaction Manager:** Este componente está divido internamente en dos sub-componentes los cuales son el “Lock Manager” y el “Log Manager”.

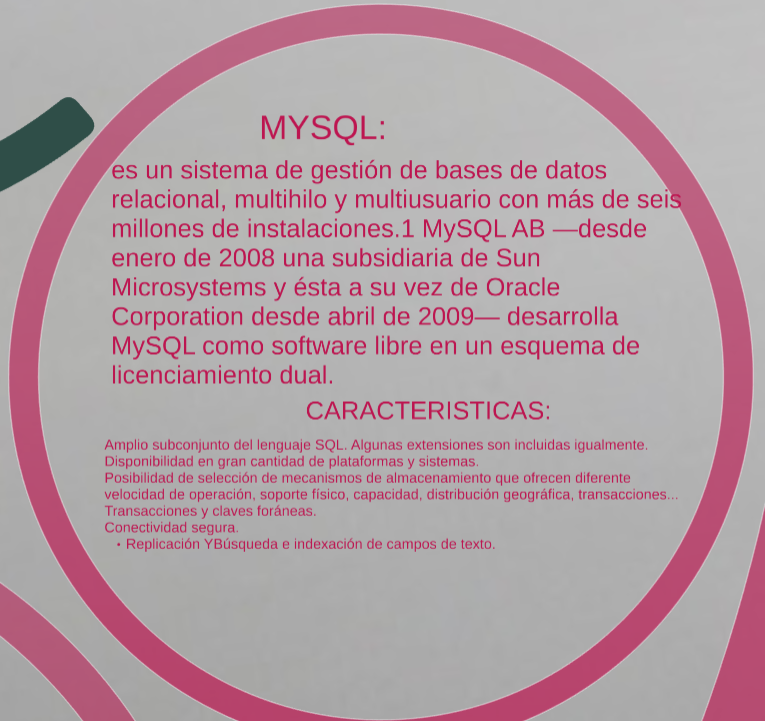
**Lock Manager:** Este es el responsable de mantener la concurrencia en la data a través de los diferentes tipos de bloqueos de la base de datos. Permite mantener la data consistente entre las diferentes sesiones a través de la administración de los diferentes bloqueos que se obtienen en cada una de las operaciones de la base de datos.

**Log Manager:** Este componente es crucial en el caso de que se hagan modificaciones a la data, debido a que antes de realizar algún cambio en las páginas de datos, este necesita estar guardado en el log de la base de datos, en su archivo físico en el disco duro. Esta es la única vez durante el ciclo normal de ejecución de transacciones T-SQL en la que el motor de base de datos ira al disco duro a guardar los cambios. Una vez que este componente confirma que el cambio ha sido guardado en el log de base de datos, el cambio a las paginas lo realizara el “Buffer Manager” dentro del “Buffer Cache”, lo cual permite tener una mejor performance durante las transacciones porque todas las operaciones se hacen en la memoria del servidor y muy poco se baja a disco.

Entre otros motores de BD se tienen:



**El motor MySQL por ejemplo tiene las siguientes características:**

****

**En esta parte, les recomiendo consultar cuáles son las instrucciones que MySQL utiliza para las transacciones con Bases de datos. La idea del ejercicio es que puedan implementar una aplicación para realizar un CRUD como lo indica la figura siguiente y después que el cliente pueda solicitar alguna transacción.**



Muchas gracias.