Desarrollo de software en sistemas distribuidos

Servidores de Bases de datos SQL y NO SQL

Introducción

- En general para la gestión de un sistema de información es necesario la normalización de sus datos
- Lo ideal es que la información de una organización esté estructurada de un modo conocido para poder manejarla, almacenarla, recuperarla.
- Para este proceso se definen modelos de datos.

Introducción

- Un sistema es mucho mas que datos
- Surgen los Frameworks con el propósito de normalizar y estructurar el código del sistema
- En general, divide la aplicación en tres capas:
  - La lógica de presentación
  - La Lógica de datos
  - La lógica de dominio o de negocio

Gestión de datos

- Se debe tener en cuenta el propósito de la aplicación
  - Bases de datos SQL
  - Bases de datos NoSQL (Not Only SQL)

La diferencia fundamental: Resuelven escenarios completamente distintos.

Ideal SQL no lo es para NoSQL

SQL vs NoSQL

SQL	NoSQL
Permite combinar tablas y relacionar información	No lo permite o muy limitado
Facilita distribuir Bases de Datos Relacionadas	Distribuir grandes cantidades de información
Gestión de datos y sus relaciones	No existen estos conceptos

Cuándo utilizar qué tipo de base de datos

SQL	NOSql
Datos consistentes sin dar posibilidad de error	Las estructuras de datos manejadas son variables
Sistemas de gestión de información operativa	Análisis de grandes cantidades de datos en modo lectura
Auditoria y trazabilidad de los datos registrados	Captura y procesado de eventos
Claridad conceptual en la relación de los datos que se deben representar	Tiendas online con motores de inteligencia complejos

El lenguaje SQL

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una BD relacional

- Tiene diferentes roles, es un lenguaje
  - de consulta interactivo
  - de programación de bases de datos.
  - de definición y administración
  - de bases de datos distribuidas
- Ayuda a proteger los datos en un ambiente multiusuario (control de acceso).
- Garantiza la integridad y consistencia de datos en un ambiente multiusuario

¿Qué hace un servidor SQL?

- Controla y ejecuta comandos SQL
- Provee una visión lógica y física de los datos
- Genera planes optimizados de ejecución de los comandos SQL
- Manejan la recuperación, concurrencia, seguridad y consistencia de la base de datos.
- Controlan la ejecución de transacciones
- Obtienen y liberan lockeos durante la ejecución de la transacción en curso

Arquitectura de los servidores SQL

La arquitectura se puede definir desde dos puntos de vista:

- Desde el punto de vista del almacenamiento → Centralizado o Distribuido
- Desde el punto de vista del procesamiento → Arquitectura multihilada, tiene su propio scheduler independiente del SO. Por ejemplo: Subase y SQL Server

SQL como lenguaje

- Es un lenguaje de consulta interactivo
- Es un lenguaje de definición y administración (manipulación) de datos.
- Es un lenguaje de programación de bases de datos.
  - Triggers
  - Stored Procedures

Stored Procedure

- Es una colección de sentencias SQL y lógica procedural.
- Es un objeto mas de la base de datos y es almacenado en el catálogo
- Acepta parámetros de entrada
- Se gana enormemente en eficiencia y reducción de trafico en la red.

Ejemplo de SP

```
Create procedure Asignar_org (p_uni int,
p_cong int);
update congreso
set id_universidad = p_uni
where id_congreso= p_cong;
End procedure;

create procedure registrar( @nombre varchar(40),
                            @precio money,
                            @stock int,
                            @mensaje varchar(60)output
                           ) as

BEGIN
        if(exists(select*from PRODUCTO where nombre=@nombre))
                set @mensaje='ESTE PRODUCTO YA EXISTE'
        else
        begin
                insert into PRODUCTO VALUES(@nombre,@precio,@stock)
                set @mensaje='PRODUCTO REGISTRADO'
        end
END; // fin del proceso

//Se debe invocar explicitamente al sp

EXECUTE PROCEDURE registrar ("Mayonesa", "15.25", "50", "");
```

Triggers y reglas

- Un trigger es una acción especial definida por el usuario (usualmente en la forma de SP) que son automáticamente invocadas cuando ocurre un evento relacionado con los datos de la base de datos. Se habilitan implícitamente por las operaciones de DELETE, INSERT y UPDATE sobre las tablas para las cuales están definidos.
- Una regla es un caso especial de trigger que se usa para chequeos simples sobre los datos.

Triggers: modelo de ejecución

- Existen dos modelos de ejecución de los triggers:
  - after (después de la operación para la que fue definido)
  - before (antes de la operación para la que fue definido)
- Según el modelo, se altera el orden de ejecución de las restricciones de integridad

Ejemplo de triggers

```
//Trigger de Delete

CREATE TRIGGER Elimina_auditoria_clientes
AFTER DELETE ON clientes

FOR EACH ROWINSERT INTO auditoria_clientes(nombre_anterior, seccion_anterior, usuario,
                                           modificado, proceso, Id_Cliente)
VALUES (OLD.nombre, OLD.seccion, CURRENT_USER(), NOW(), 'Eliminado', OLD.id_cliente);

INSERT INTO ausentes (nombre, seccion) VALUES(OLD.id_cliente,OLD.seccion);

---------------------------------------
//Triger de insert

CREATE TRIGGER Inserta_auditoria_clientes
AFTER INSERT ON clientes

FOR EACH ROWINSERT INTO auditoria_clientes(nombre_nuevo, seccion_nueva, usuario,
                                          modificado, proceso, Id_Cliente)
VALUES (NEW.nombre, NEW.seccion, CURRENT_USER(), NOW(), 'Incluido', NEW.id_cliente);
```

SQL estático y SQL dinámico

- El SQL estático son sentencias definidas en el código y compiladas en el momento de su definición. Si los objetos que referencia no existen, no compila.
- El SQL dinámico es creado en tiempo de ejecución y ofrece mas flexibilidad a expensas de velocidad de ejecución

SQL incorporado y SQL programado

- SQL incorporado: las sentencias SQL están incorporadas al código fuente del programa, entremezcladas con las otras sentencias del lenguaje de programación
- SQL programado: las sentencias SQL se almacenan como objetos de la base de datos (triggers,Stored procedure, reglas)

SQL programado vs SQL embebido(dinámico o estático)

	Procedimientos almacenados	SQL embebido estático	SQL embebido dinámico
Nombres para las funciones	SI	NO	NO
Funciones compartidas	SI	NO	NO
Parametros de E/S	SI	NO	NO
Catalogado	SI	SI	NO
Logica procedural	En el objeto	Externa	Externa
Flexibilidad	Baja	Baja	Alta
Nivel de abstraccion	Alto	Bajo	Bajo
Estandarizacion	NO	SI	SI
Performance	Rapido	Mediano	Lento
Trafico	1 Request/Reply por varios comandos SQL	1 Request/Reply por cada comando SQL	1 Request/Reply por cada comando SQL

SQL y el acceso concurrente

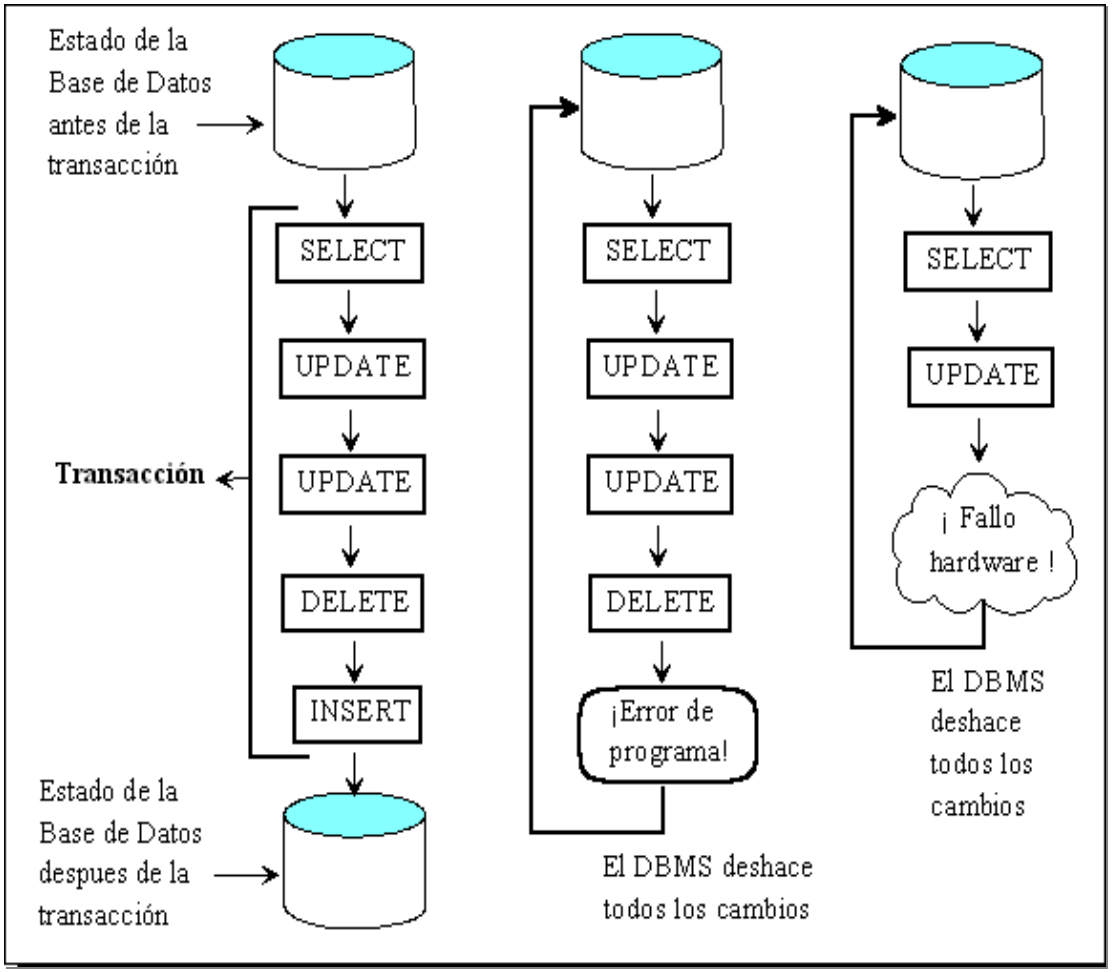
Ayuda a proteger los datos en un ambiente multiusuario (control de acceso), garantizando la integridad y consistencia de datos:

A través de los siguientes servicios:

- Control de ejecución de transacciones
- Obtener y liberar lockeos durante la ejecución de la transacción en curso
- Proteger la base de datos de accesos no autorizados.

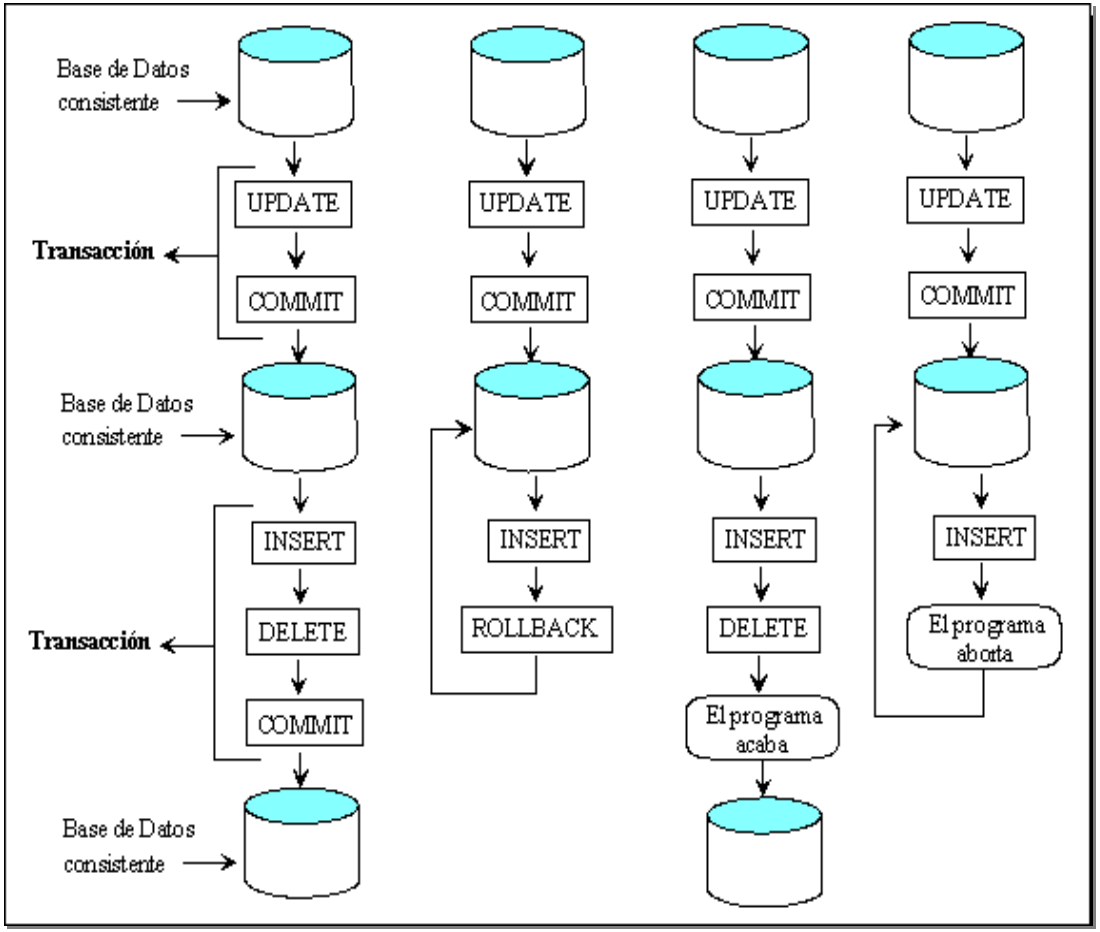
Transacciones

- Una transacción es una secuencia de una o más sentencias SQL que juntas forman una unidad de trabajo para el DBMS, el cuál asume que todas las sentencias deben completarse para asegurar la consistencia e integridad de la BD



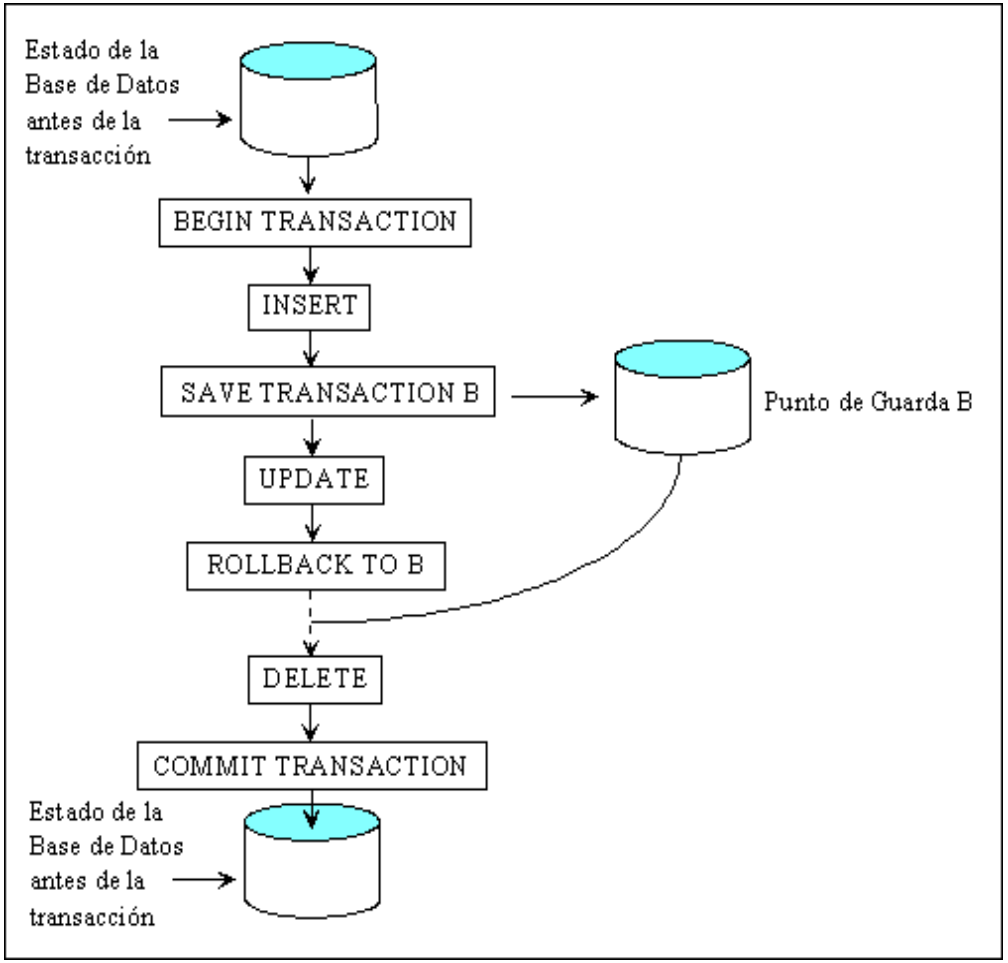
Transacciones Estándar

- SQL estándar soporta las transacciones de base de datos mediante las sentencias COMMIT y ROLLBACK.



Transacciones No Estándar

- En SQL interactivo se asume que cada sentencia en una transacción Sybase extiende el modelo de transacciones del estándar SQL incluyendo otras sentencias para el procesamiento de transacciones



Transacciones y Procesamiento Multiusuario

Cuando dos o mas usuarios acceden o ejecutan transacciones concurrentemente, se pueden presentar básicamente tres problemas:

Actualización Perdida: cuando dos aplicaciones leen datos de la BD, los utilizan para un cálculo y luego los actualizan. Ambas transacciones leen el valor antes de que lo cambie la otra…

Datos No Cumplimentados: cuando una aplicación lee actualizaciones no cumplimentadas de otra y actúa en base a los datos leídos

Datos Inconsistentes: cuando durante una consulta otra aplicación cumplimenta transacciones que afectan a los datos de la consulta.

Niveles de Cerramiento

El cerramiento puede ser implementado a distintos niveles de la base de datos. Los cuales pueden ser:

- Cerramiento a nivel de la base de datos.
- Cerramiento a nivel de tabla.
- Cerramiento a nivel de fila.
- Cerramiento a nivel de columnas.

Esquemas de Cerramientos

Para aumentar el acceso concurrente a una BD, se utilizan esquemas con mas de un tipo de cierre, siendo habitual en tal sentido implementar al menos dos tipos de cierre: Compartido y Exclusivo

Transacción A		Transacción B		
	Acción	**No Cerrado**	**Cierre Compartido**	**Cierre Exclusivo**
	No Cerrado	SI	SI	SI
	Cierre Compartido	SI	SI	NO
	Cierre Exclusivo	SI	NO	NO

Ambas pueden leer datos

SQL como middleware

- ¿Cómo resuelve cada cliente el hecho de acceder a datos en servidores SQL de múltiples vendedores?
- La respuesta es a través del middleware especifico de la base de datos (API)

Interfaz de Programa de Aplicación (API)

- El programa se comunica con el DBMS mediante un conjunto de llamadas de función denominadas interfaz de programa de aplicación (API).
  - Análisis Sintáctico.
  - Valida la sentencia contra el catálogo del sistema.
  - Optimiza la sentencia, determinando la mejor forma de ejecutarla.
  - Genera un plan de aplicación para la sentencia, esto es un secuencia de los pasos a seguir para su realización.
  - Ejecuta el plan de aplicación.

Interfaz de Programa de Aplicación (API)

```php
<?php
// mysqli
$mysqli = new mysqli("ejemplo.com", "usuario", "contraseña", "basedatos");
$resultado = $mysqli->query("SELECT '¡Hola, querido usuario de MySQL!' AS _message FROM DUAL");
$fila = $resultado->fetch_assoc();
echo htmlentities($fila['_message']);

// PDO
$pdo = new PDO('mysql:host=ejemplo.com;dbname=basedatos', 'usuario', 'contraseña');
$sentencia = $pdo->query("SELECT '¡Hola, querido usuario de MySQL!' AS _message FROM DUAL");
$fila = $sentencia->fetch(PDO::FETCH_ASSOC);
echo htmlentities($fila['_message']);

// mysql
$c = mysql_connect("ejemplo.com", "usuario", "contraseña");
mysql_select_db("basedatos");
$resultado = mysql_query("SELECT '¡Hola, querido usuario de MySQL!' AS _message FROM DUAL");
$fila = mysql_fetch_assoc($resultado);
echo htmlentities($fila['_message']);
?>
```

¿En que consiste realmente el API SQL?

- El API surge por las características procedurales que se apartan del estándar.
- Hay dos mecanismos para soportar el uso de SQL desde un lenguaje de programación:
  - SQL Embebido (ESQL)
  - Call-Level Interface (CLI)

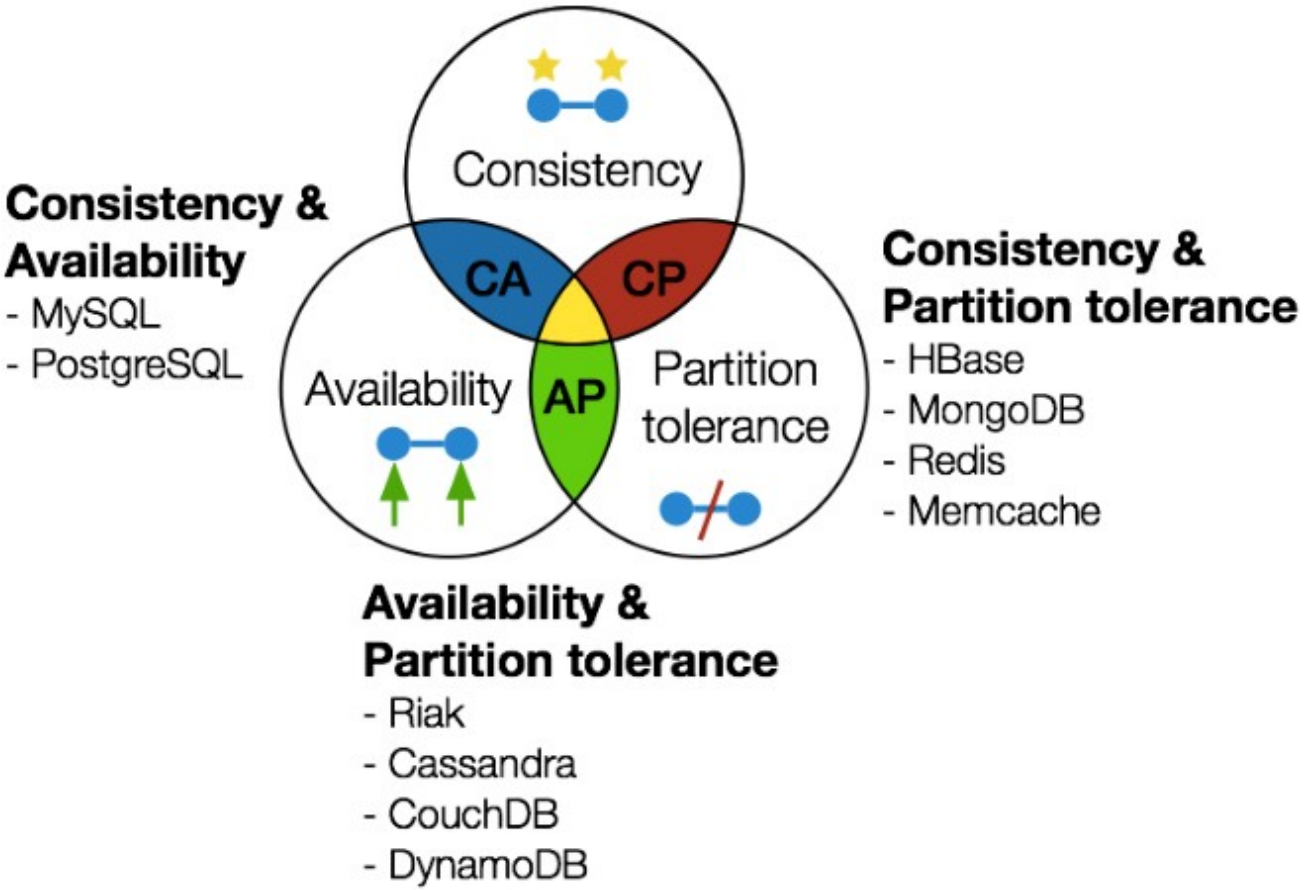
Open DataBase Connectivity

- El ESQL se sustenta el estándar ISO ya que no existen características procedurales. En cuanto al CLI hay distintas alternativas, una de las cuales es ODBC de MicroSoft.
- ODBC (Open DataBase Connectivity) es una API estandar multivendedor.
- JDBC, permite la ejecución de operaciones sobre bases de datos desde JAVA independiente del SO y del motor de BD.

Open Client

- Es el software propietario (de cada BD) que garantiza la conectividad entre cliente y servidor
- En definitiva dependerá de las aplicaciones cual de ellos usara y a su vez, la aplicación se verá condicionada por el API que utilice la herramienta de desarrollo

El teorema CAP



http://altenwald.org/2017/05/23/teorema-cap/

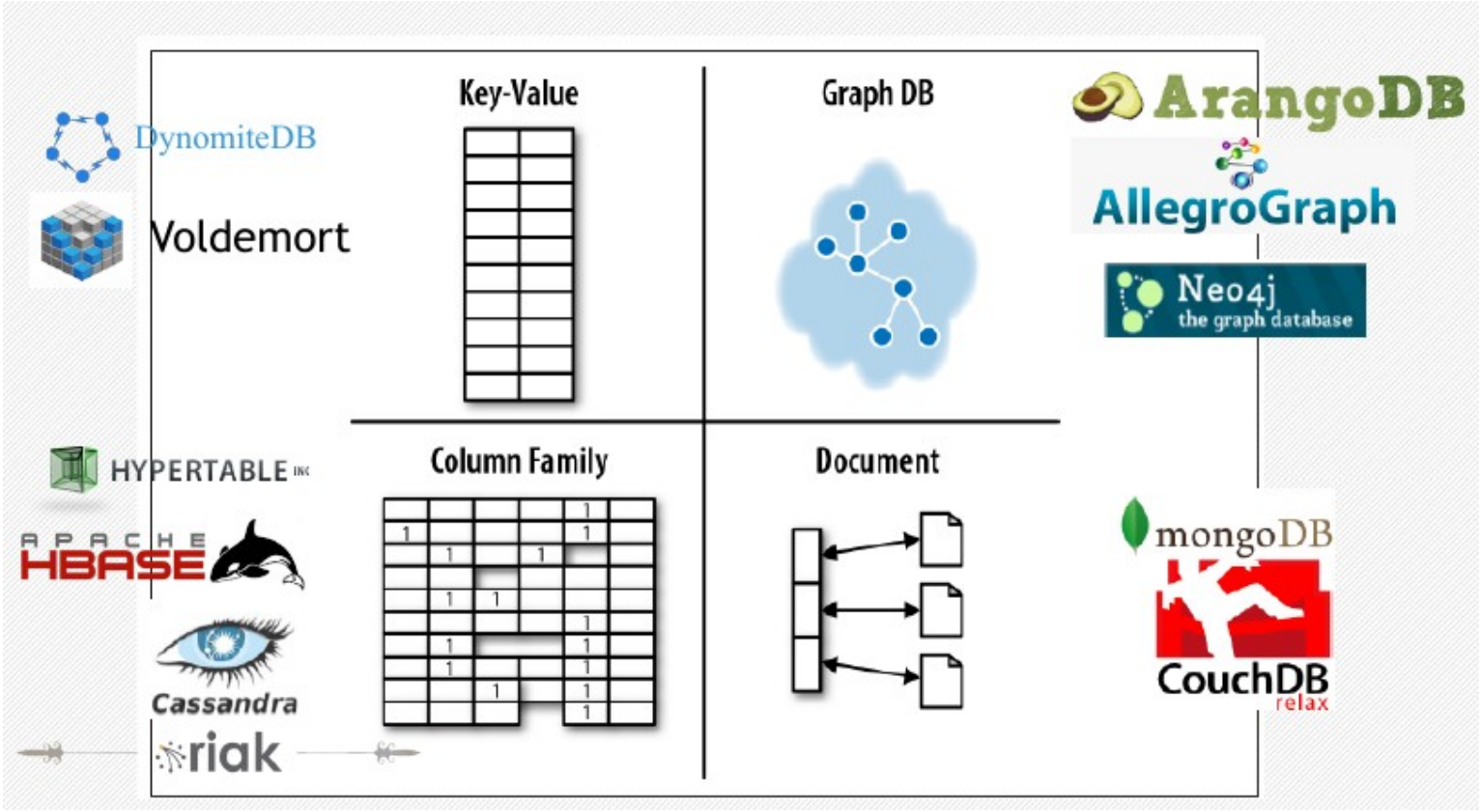
El teorema CAP – ACID vs BASE

- Basically Available (disponible básicamente), está operativo la mayoría del tiempo.
- Soft state (estado débil), los datos en las diferentes réplicas no tienen que ser mutuamente consistentes en todo momento.
- Eventually Consistent (eventualmente consistente), se asegura la consistencia solo después de que pase cierto tiempo.

ACID vs BASE: http://queue.acm.org/detail.cfm?id=1394128
TEOREMA DE CAP: http://altenwald.org/2017/05/23/teorema-cap/

Almacenamiento en BD NoSQL

- Taxonomía o clasificación



Bases de datos clave-valor

- La información se obtiene a partir de una clave única
- Se tiene una sola tabla enorme y convenientemente indexada
- Ejemplos
  - Cassandra (Apache) http://cassandra.apache.org/
  - BigTable (Google) tablas multidimensionales, celdas con versionado.
  - HBase (Hadoop)

Bases de datos documentales

- Almacenamiento de documentos, uno o más pares campo/valor (datos simples o listas). Cada documento tiene un id.
- El acceso a los datos es mediante vistas
- Posee una arquitectura distribuida con replicación bidireccional y operación off-line.
- Ejemplos
  - Lotus Domino (IBM)
  - CouchDB (Apache)
  - Mongo DB

MongoDB – Características

- Orientada a documentos
- Almacenados en BSON (binario de JSON)
- Documentos pertenecen a Colección
- Las documentos de una misma colección no están obligados a tener el mismo esquema
- Consultas pasando objetos JSON a través de Consola Java Script
- Drivers para ser utilizada desde Java, PHP, Python, Ruby, etc.

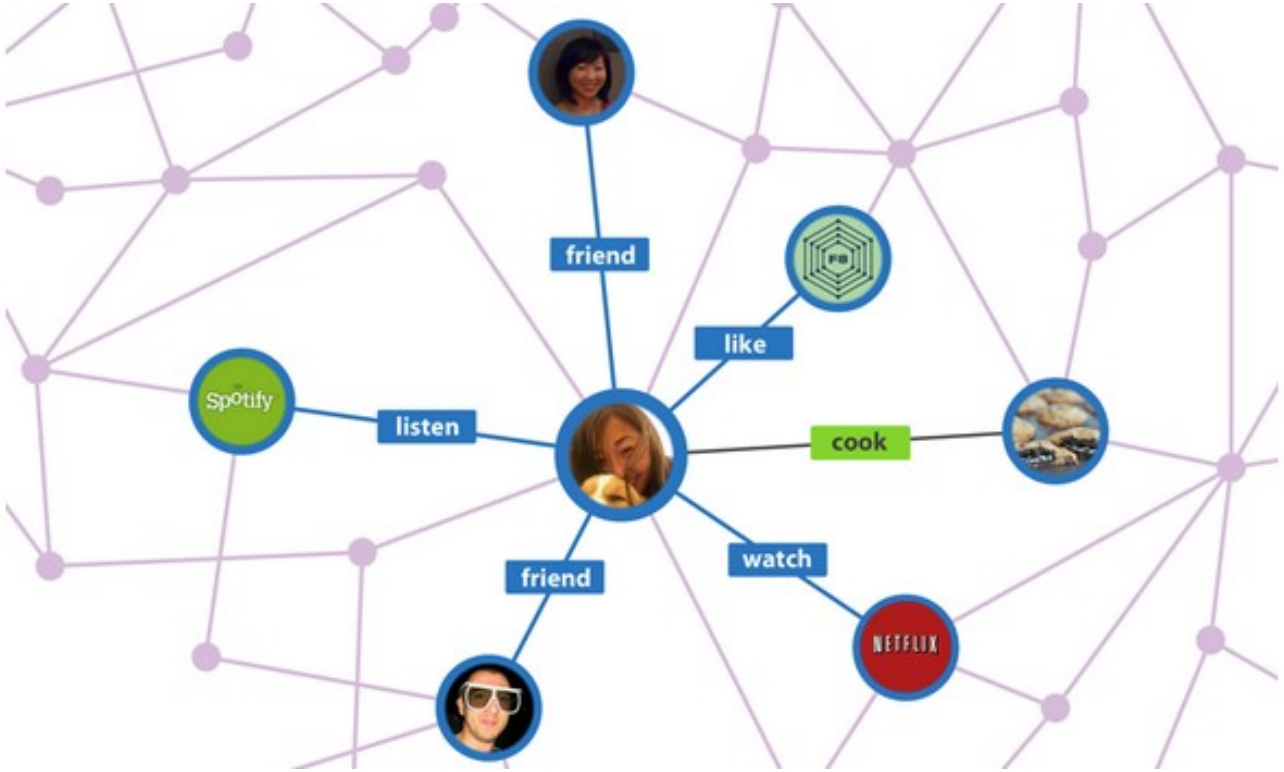
MongoDB – Características

- No existen las transacciones
- No existen los joins
- Las consultas de agregación no son sencillas aunque hay un framework
- Indices de tipo B-Tree

Bases de datos orientadas a grafos (BDOG)

- Grafos dirigidos
- Almacenan relaciones entre los nodos, preponderancia por sobre los nodos (particularmente útil en redes sociales)
- Ofrecen una API para consultas que recorren los grafos

Bases de datos orientadas a grafos (BDOG)



- **Ejemplo: Neo4j** http://www.neo4j.org/learn/cypher

Bases de datos orientadas a familias de columnas

- La información se almacenan en familias de columnas.
- Estructura: Mapa multidimensional

- Cassandra
- Amazon Redshift: como parte de Amazon Web Services (AWS),
- HBase

Bibliografía

- Patricia Bazán – “Aplicaciones, Servicios y Procesos distribuídos”
- Thomas Connolly, Carolyn Begg - “Database Systems”
- Hector Garcia-Molina - “Database Systems”