UT7.
ACID, BASE y CAP

Modelo ACID

Bases de datos distribuidas

- Consistencia
- Transacciones y propiedades ACID
- Protocolos para la gestión de transacciones
- Transacciones en BD relacionales y BD NoSQL

Consistencia

- Uno de los objetivos más importantes de un SGBD es garantizar la consistencia de los datos.
- La consistencia tiene que ver con la integridad, calidad y corrección de los datos, y se puede ver comprometida en diferentes situaciones:
  - Errores de programación
  - Violación de restricciones de integridad
  - El acceso simultáneo de diferentes usuarios
  - La existencia de datos replicados
  - Averías

Consistencia

- Las averías pueden ser de naturaleza diversa:
  - Caídas del SGBD debidas, por ejemplo, a una avería software o un corte de luz
  - La destrucción total o parcial de la BD a causa de desastres o de fallos de los dispositivos de almacenamiento
  - Fallos en la red de comunicaciones que pueden causar el particionamiento del sistema.
  - Pérdida de mensajes
- Para garantizar la consistencia de la BD el SGBD se basa, principalmente, en transacciones y en una serie de mecanismos que las gestionan.

Transacciones y propiedades ACID

Una transacción es un conjunto de operaciones (de lectura y/o escritura) sobre la BD que se ejecutan como una unidad indivisible de trabajo. La transacción acaba su ejecución confirmando o cancelando los cambios que se han llevado a cabo sobre la BD.

- Las transacciones deben cumplir las propiedades ACID:
  - **Atomicidad**: el conjunto de operaciones que constituyen la transacción es la unidad indivisible de ejecución.
  - **Consistencia**: la ejecución de la transacción tiene que preservar la consistencia de la BD.
  - **aIslamiento**: una transacción no puede ver interferida su ejecución por otras transacciones que se estén ejecutando de forma concurrente con ella.
  - **Definitividad/Durabilidad**: los resultados producidos por una transacción que confirma tienen que ser definitivos en la BD.

Aislamiento

- La propiedad de aislamiento se puede ver vulnerada debido a la presencia de interferencias.
- Existen diferentes tipos de interferencias:
  - Actualización perdida
  - Lectura no confirmada
  - Lectura no repetible
  - Análisis inconsistente (fantasmas)

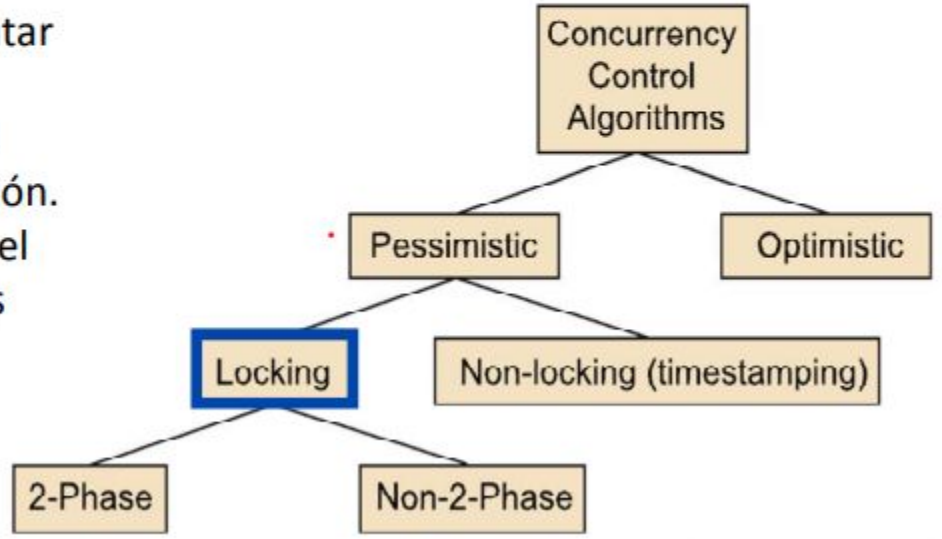
Protocolos para gestión de transacciones

- Para mantener las propiedades de consistencia y aislamiento el SGBD implementa:
  - Protocolos de control de concurrencia
  - Protocolos para la gestión de réplicas (BD distribuidas)
- Para mantener las propiedades de atomicidad y definitividad el SGBD implementa:
  - Protocolos de recuperación: requieren de dietarios (*log*) y copias de seguridad de la BD (*backups*).
  - Protocolo de confirmación en dos fases (BD distribuidas)
- A pesar de ello, el mantenimiento de las propiedades ACID también es responsabilidad de las aplicaciones (y en consecuencia de sus desarrolladores).

Protocolos para gestión de transacciones

Protocolos para el control de concurrencia

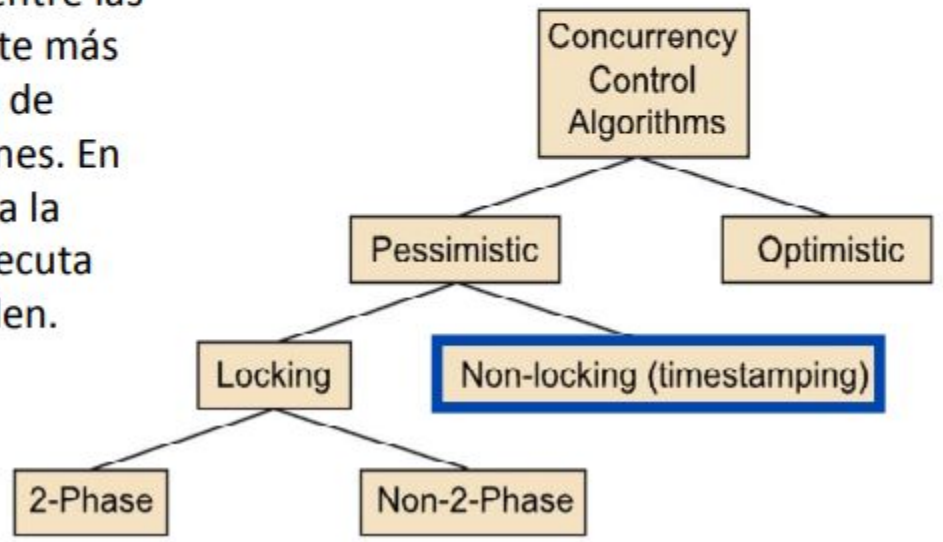
Reservas (*locking*): la transacción, antes de ejecutar una operación tiene que adquirir una reserva que le permita efectuar la operación. Puede causar esperas y en el caso peor abrazos mortales (*deadlocks*).



Protocolos para gestión de transacciones

Protocolos para el control de concurrencia

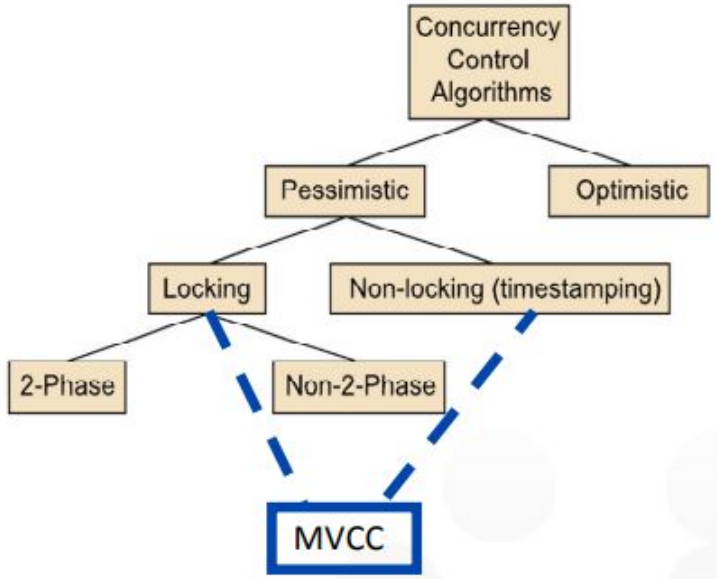
Marcas de tiempo *(timestamping):* impone relación de orden entre las transacciones, en su variante más simple, basadas en el inicio de ejecución de las transacciones. En caso de conflicto, se cancela la primera transacción que ejecuta una operación fuera de orden.



Protocolos para gestión de transacciones

Protocolos para el control de concurrencia

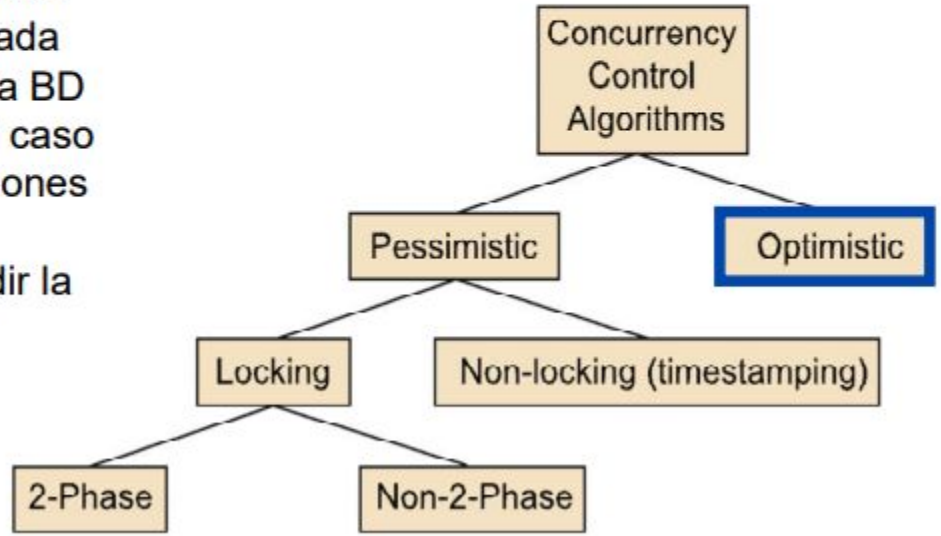
Mecanismo de control de concurrencia multiversión *(multiversion concurrency control,* MVCC*):* enfoque híbrido. Las transacciones de sólo lectura tienen una misma visión de la BD (quizá "antigua"). Las transacciones que efectúan escrituras siguen una estrategia basada en bloqueos, y trabajan con versiones privadas que se hacen públicas a la finalización de la transacción (si ésta confirma sus resultados).



Protocolos para gestión de transacciones

Protocolos para el control de concurrencia

Técnicas optimistas (*optimistic techniques*): una vez finalizada la transacción el gestor de la BD comprobará si es válida, en caso de no serlo se tomaran acciones al respecto (por ejemplo, cancelar la transacción, pedir la intervención del usuario/aplicación etc.).



Protocolos para gestión de transacciones

- Los protocolos para el control de concurrencia más utilizados por los SGBD relacionales son las reservas y el MVCC.
- En el caso de NoSQL hay diversidad de propuestas:
  - Neo4J y MongoDB utilizan reservas.
  - CouchDB implementa el MVCC.
  - Riak utiliza *vector clocks*.

Transacciones en BD relacionales

- Las BD relacionales trabajan con transacciones que verifican las propiedades ACID.
- A pesar que una transacción puede incorporar varias operaciones, el modo de trabajo por defecto acostumbra a ser que cada operación efectuada sobre la BD sea una transacción (*autocommit* activado).
- Es posible relajar el nivel de aislamiento de las transacciones, esto implica que:
  - Se pueden producir interferencias
  - Se mejora el rendimiento
  - Disminuye la sobrecarga del SGBD

Transacciones en BD No SQL

- Soportar un modelo de transacciones ACID en BD que almacenan grandes volúmenes de datos, que están distribuidas y con replicación de datos es complejo y puede causar problemas de rendimiento. Es por ello que:
  - Se han propuesto modelos transaccionales alternativos (modelo BASE)
  - Se evitan transacciones que incluyan diversas operaciones (éste es el caso de las BD NoSQL basadas en modelos de agregación)
- A pesar de ello, las BD NoSQL de grafos, en general, se ajustan a modelos de transacciones ACID.

Gestión de réplicas

- La replicación consiste en almacenar todos o una parte de los fragmentos diseñados (y por lo tanto, los datos que éstos contienen) en más de un nodo de la BDD.
- La replicación permite maximizar la disponibilidad de los datos, así como mejorar el rendimiento de la BDD.
- El problema principal es que es necesario garantizar la consistencia de las réplicas.
- La consistencia se refiere a que todas las réplicas de unos mismos datos deben contener los mismos valores.

Gestión de réplicas

- El mantenimiento de la consistencia de las réplicas (o gestión de réplicas) se puede realizar:
  - De manera síncrona o asíncrona
  - A través de una réplica destacada (réplica o copia primaria) o sin réplica destacada (todas las réplicas son iguales)

Gestión de réplicas

El mantenimiento de la consistencia de las réplicas (o gestión de réplicas) se puede realizar:

★ De manera síncrona o asíncrona

- A través de una réplica destacada (réplica o copia primaria): las políticas que consideran diferentes tipos de réplicas se conocen de forma genérica como master-slave
- Sin réplica destacada (todas las réplicas son iguales): las que consideran todas las réplicas iguales se conocen bajo la denominación de replicación P2P (o replicación multimaster).

Teorema de CAP

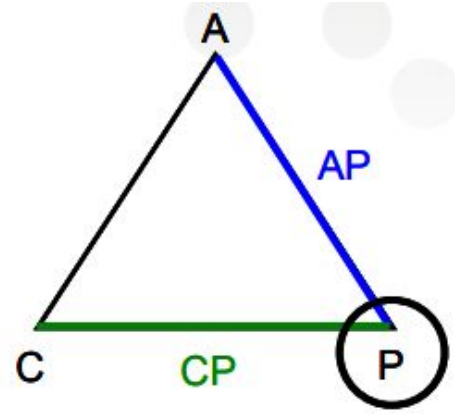
- **Consistencia** (*consistency*): los usuarios del sistema tienen que poder recuperar siempre los mismos datos en un mismo instante de tiempo.
- **Disponibilidad** *(availability):* las peticiones de servicio enviadas por los usuarios a un nodo que está disponible deben obtener su debida respuesta.
- **Tolerancia a particiones** (*tolerance to network partitions)*: el sistema debe proporcionar servicio a los usuarios a pesar de que se puedan producir situaciones de avería que causen que el sistema quede particionado en diferentes componentes.

El teorema CAP enuncia que es imposible garantizar simultáneamente las tres características.

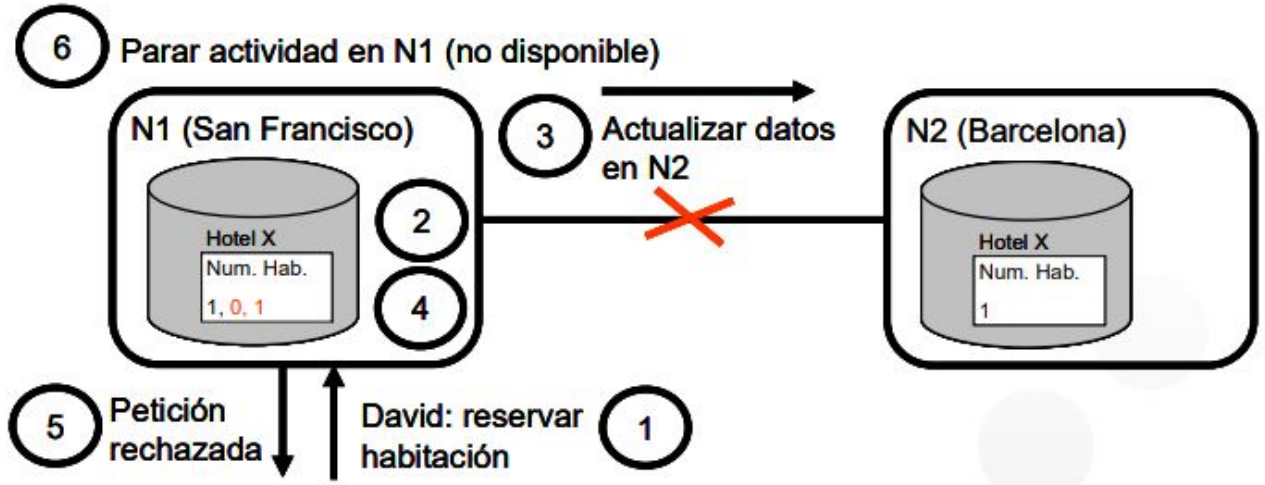
Teorema de CAP

- En un sistema altamente distribuido se van a producir situaciones de avería que causen que el sistema se particione.
- En este escenario, garantizar P constituye una necesidad. En consecuencia, hay que decidir que se quiere priorizar:
  - La consistencia (C además de P)
  - La disponibilidad (A además de P)

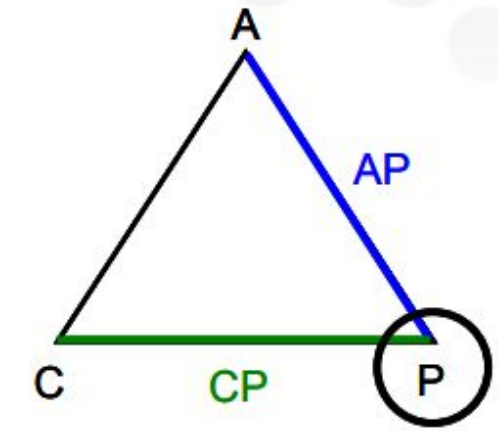
Teorema de CAP: CP



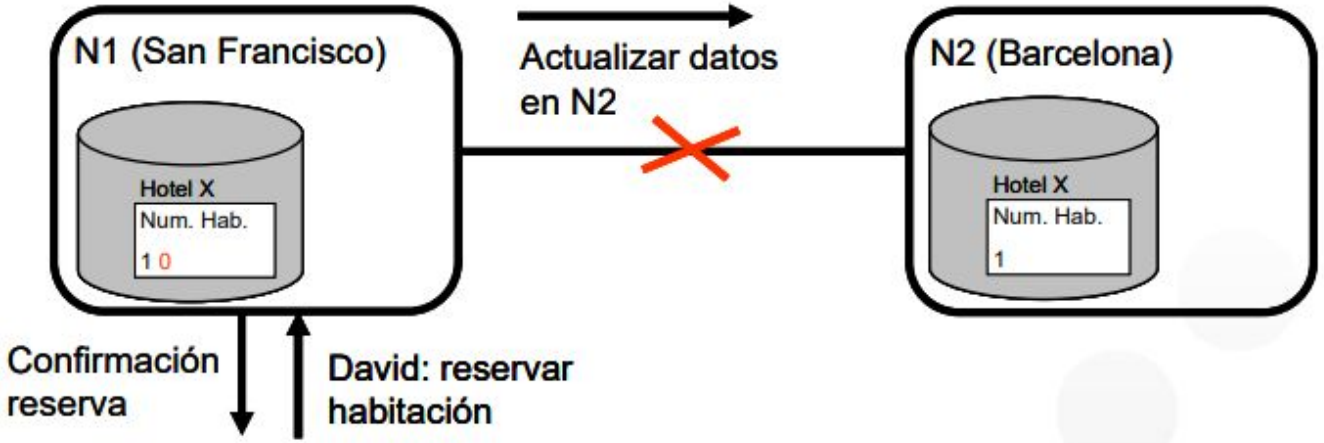
CP: el sistema siempre contiene una visión consistente de los datos, aunque no esté totalmente disponible en presencia de particiones.



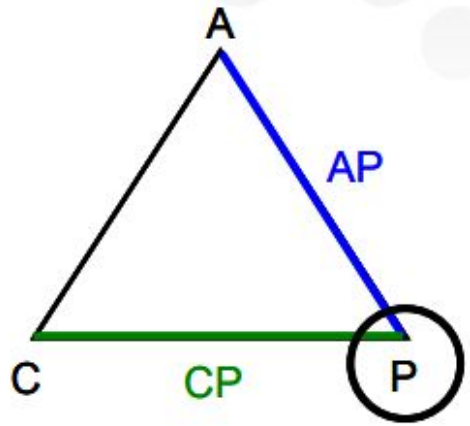
Teorema de CAP: AP



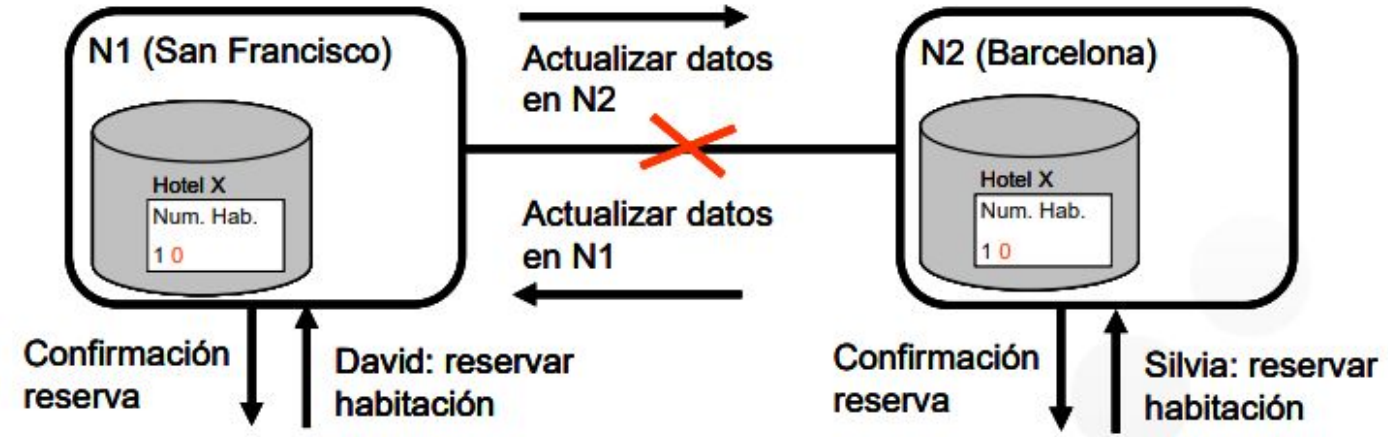
AP: el sistema siempre está disponible, aunque temporalmente puede mostrar datos inconsistentes en presencia de particiones.



Teorema de CAP: AP



AP: el sistema siempre está disponible, aunque temporalmente puede mostrar datos inconsistentes en presencia de particiones.



MODELO BASE

- Los sistemas que garantizan AP, se basan en un modelo de transacciones diferente al modelo ACID, en concreto se trabaja con el denominado modelo BASE:
  - **Disponibilidad limitada** (*Basic Availability*)
  - **Estado flexible** *(Soft-state)*
  - **Consistencia final en el tiempo** (*Eventual consistency*)

ACID, BASE y CAP

- Los modelos ACID y BASE representan dos enfoques para el mantenimiento de la consistencia situados en polos opuestos: ACID constituye un enfoque pesimista, mientras que BASE es optimista.
- La noción de consistencia (C) en CAP y ACID difieren:
  - En teorema CAP la C se refiere a la consistencia de las réplicas de unos mismos datos.
  - La consistencia en el modelo ACID es una noción que abarca múltiples elementos. Se conoce como consistencia estricta (*strict consistency*).

Consideraciones finales

- Las BD NoSQL son, en general, sistemas CP o AP:
  - Sistemas CP: MongoDB, BigTable, Hbase, Redis
  - Sistemas AP: Riak, Dynamo, Cassandra, CouchDB
- En ocasiones pueden trabajar indistintamente en ambas modalidades.
- Las BD relacionales distribuidas son sistemas CA.