Database Availability and Integrity in NoSQL

Fahri Firdausillah [M031010012]

What is NoSQL

- Stands for **Not Only SQL**
- Mostly addressing some of the points: **non-relational**, **distributed**, **horizontal scalable**, **schema-free**, **easy replication support**, **eventually consistent**, and **huge data amount**
- This presentation will talk much about **replication** and **horizontal scalable** for database availability, then **eventually consistent** and **schema-free** for database integrity

List of NoSQL Products

- **Cassandra** used on:
  - Digg, Facebook, Twitter, Reddit, Rackspace, Cloudkick, Cisco
- **Hadoop** used on:
  - Amazon Web Services, Pentaho, Yahoo!, The New York Times
- **CouchDB** used on:
  - CERN, BBC, Interactive Mediums
- **MongoDB** used on:
  - Foursquare, bit.ly, SourceForge, Fotopedia, Joomla Ads
- **Riak** used on:
  - Widescript, Western Communications, Ask Sponsored Listings

Database Availability Outline

- Database Availability Means
- CAP Theorem (BASE vs ACID)
- Partitioning and Replication
- Replication Diagram
- “Ring” of Consistent Hashing
- Next …. → Database Integrity

What is NoSQL

- Stands for **Not Only SQL**
- Mostly addressing some of the points: **non-relational**, **distributed**, **horizontal scalable**, **schema-free**, **easy replication support**, **eventually consistent**, and **huge data amount**
- This presentation will talk much about **horizontal scalable** for database availability and **eventually consistent** for database integrity

What is NoSQL

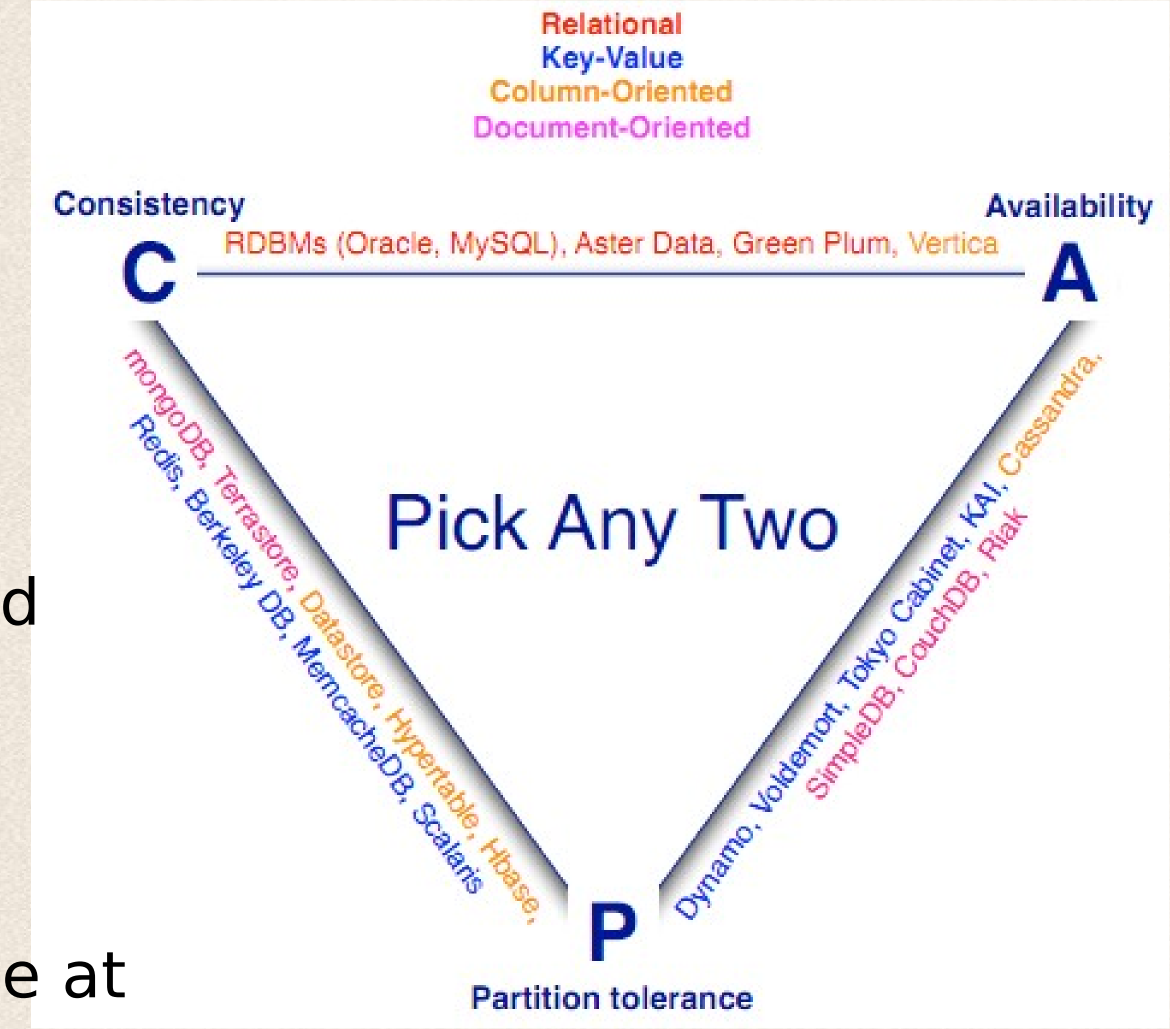
- Stands for **Not Only SQL**
- Mostly addressing some of the points: **non-relational**, **distributed**, **horizontal scalable**, **schema-free**, **easy replication support**, **eventually consistent**, and **huge data amount**
- This presentation will talk much about **horizontal scalable** for database availability and **eventually consistent** for database integrity

Database Availability Mean

- IBM divide database availability into 3 section:
  - **High Availability**: database and application is available in scheduled period, when maintenance period system is temporarily down.
  - **Continuous Operation**: system available all the time with no scheduled outages.
  - **Continuous Availability**: combination of HA & CO, data is always available, and maintenance is done without shutdown the system

* *Database Availability Considerations*, IBM RedBook [2001]

CAP Theorem (1)


Relational
Key-Value
Column-Oriented
Document-Oriented
Consistency
Availability
C
A
RDBMs (Oracle, MySQL), Aster Data, Green Plum, Vertica
mongoDB, Terrastore, Datastore, Hypertable, Hbase,
Redis, Berkeley DB, MemcacheDB, Scalaris
Pick Any Two
Dynamo, Voldemort, Tokyo Cabinet, KAI, Cassandra,
SimpleDB, CouchDB, Riak
P
Partition tolerance

Consistency, ***Availability*** and ***Partition Tolerance.***

A shared-data system can have at most two of those three.

** Visual Guide to NoSQL Systems, Nathan Hurst, 2010* [8]

CAP Theorem (2)

- **Consistent, Available (CA) Systems** have trouble with partitions and typically deal with it with replication.
- **Consistent, Partition-Tolerant (CP) Systems** have trouble with availability while keeping data.
- **Available, Partition-Tolerant (AP) Systems** achieve “eventual consistency” through replication and verification consistent across partitioned nodes.

ACID and BASE

- **ACID**

  Atomicity: All or nothing

  Consistency: Any transaction should result in valid tables

  Isolation: separate transactions

  Durability: Database will survive a system failures.

ACID and BASE cont'd

- **BASE**

  Basically **A**vailable - system seems to work all the time

- **S**oft State - it doesn't have to be consistent all the time
- **E**ventually Consistent - becomes consistent at some later time

Horizontal Scale

- Data explosion (especially in web application) force database system to scale

- **1st solution : Vertical scale**

  Improving server specification by adding more processor, RAM, and storage device. Limited and expensive.

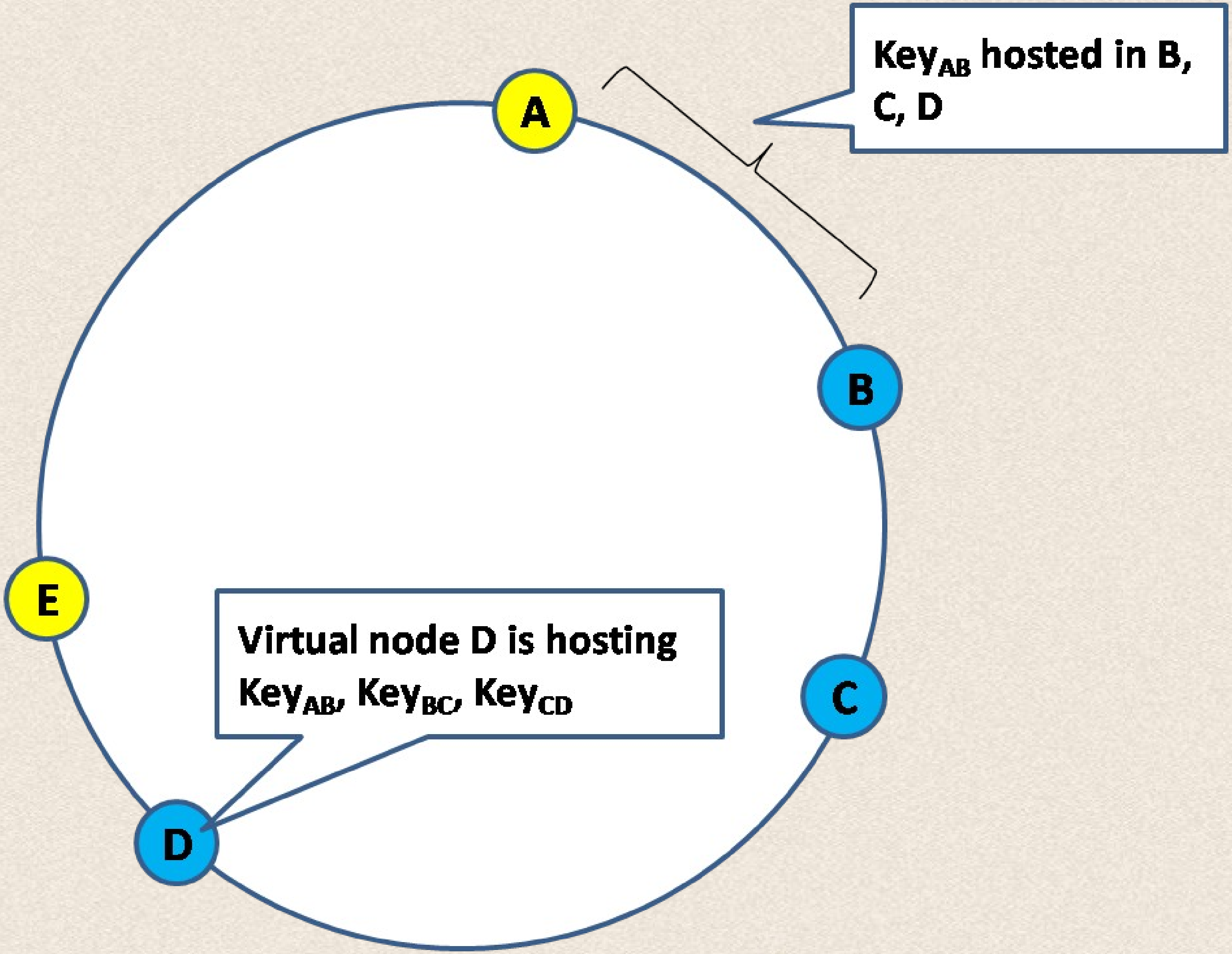
- **2nd solution : Horizontal scale**

  Adding more cheap computer as server expansion. Do sharding and partitioning which is hard to implement and expensive using relational databases (RDBMS)

Partitioning & Replication

- Partitioning
  - Sharing the data between different nodes (data host)
  - Each node placed on a ring
  - Advantage : ability to scale incrementally
  - Issues : non-uniform data distribution
- Replication
  - Multiple nodes
  - Multiple datacenters
  - High availability and durability

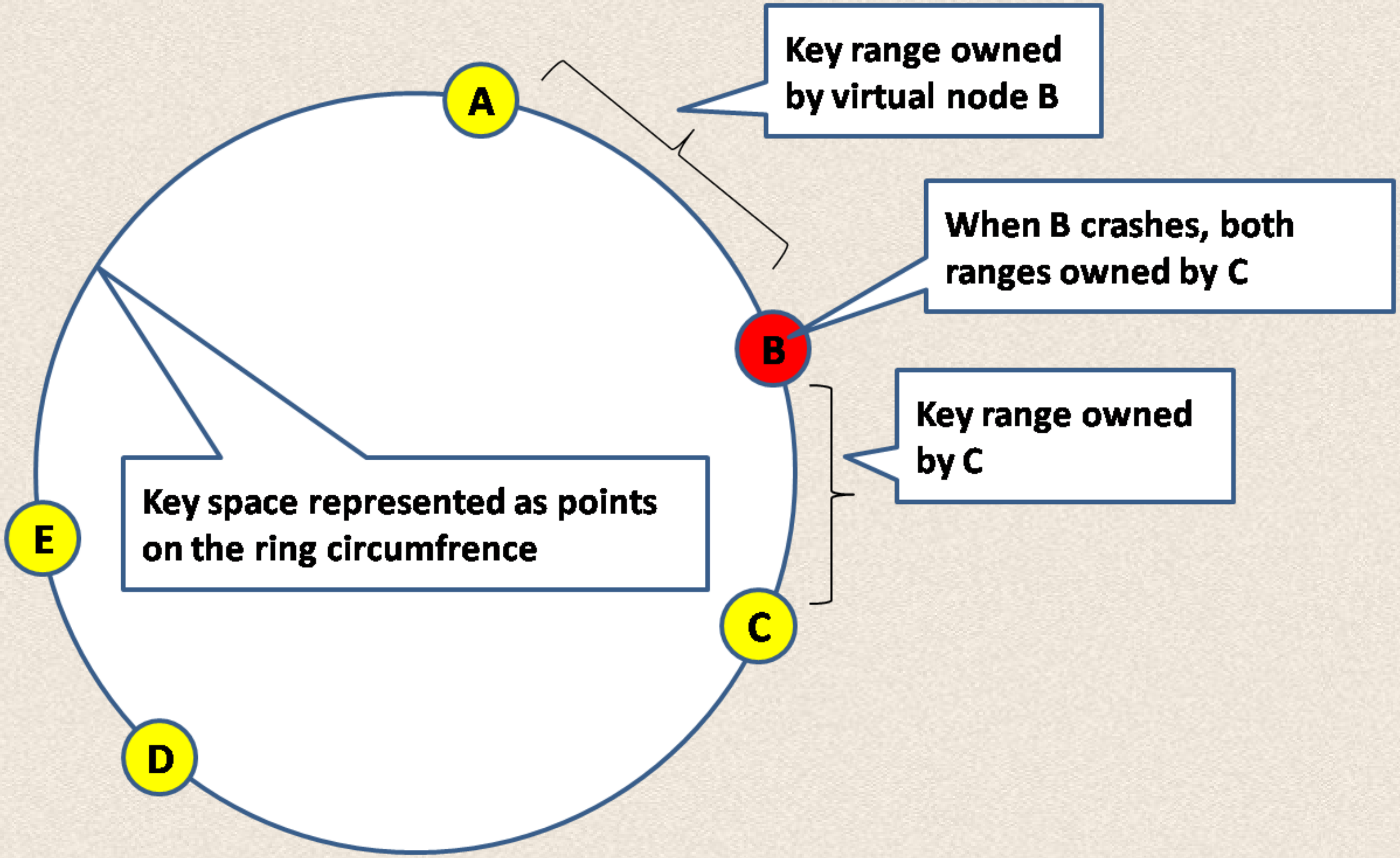
Data Replication



Key_{AB} hosted in B, C, D

Virtual node D is hosting Key_{AB}, Key_{BC}, Key_{CD}

Ring of Consistent Hashing



Key range owned by virtual node B

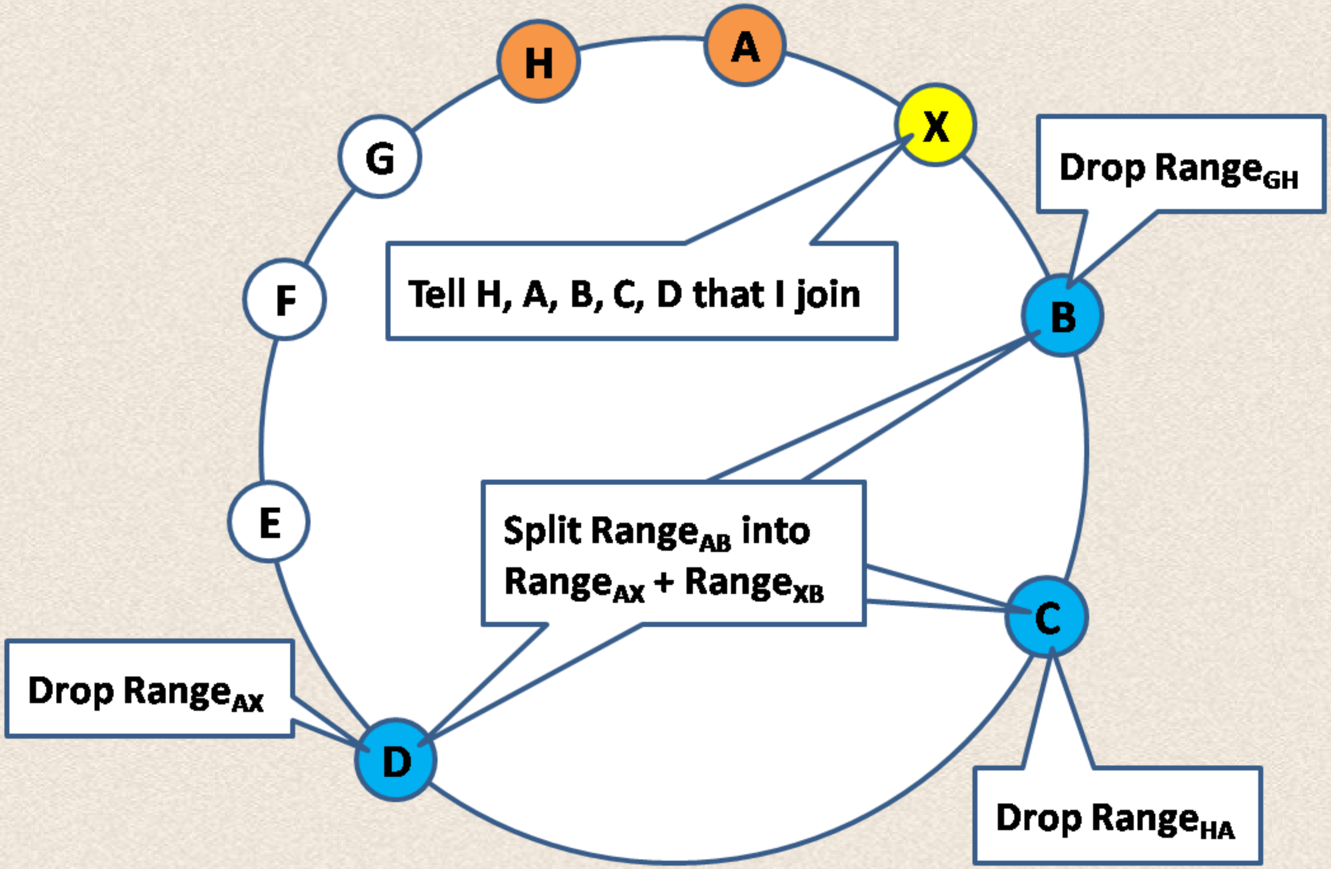
When B crashes, both ranges owned by C

Key range owned by C

Key space represented as points on the ring circumfrence

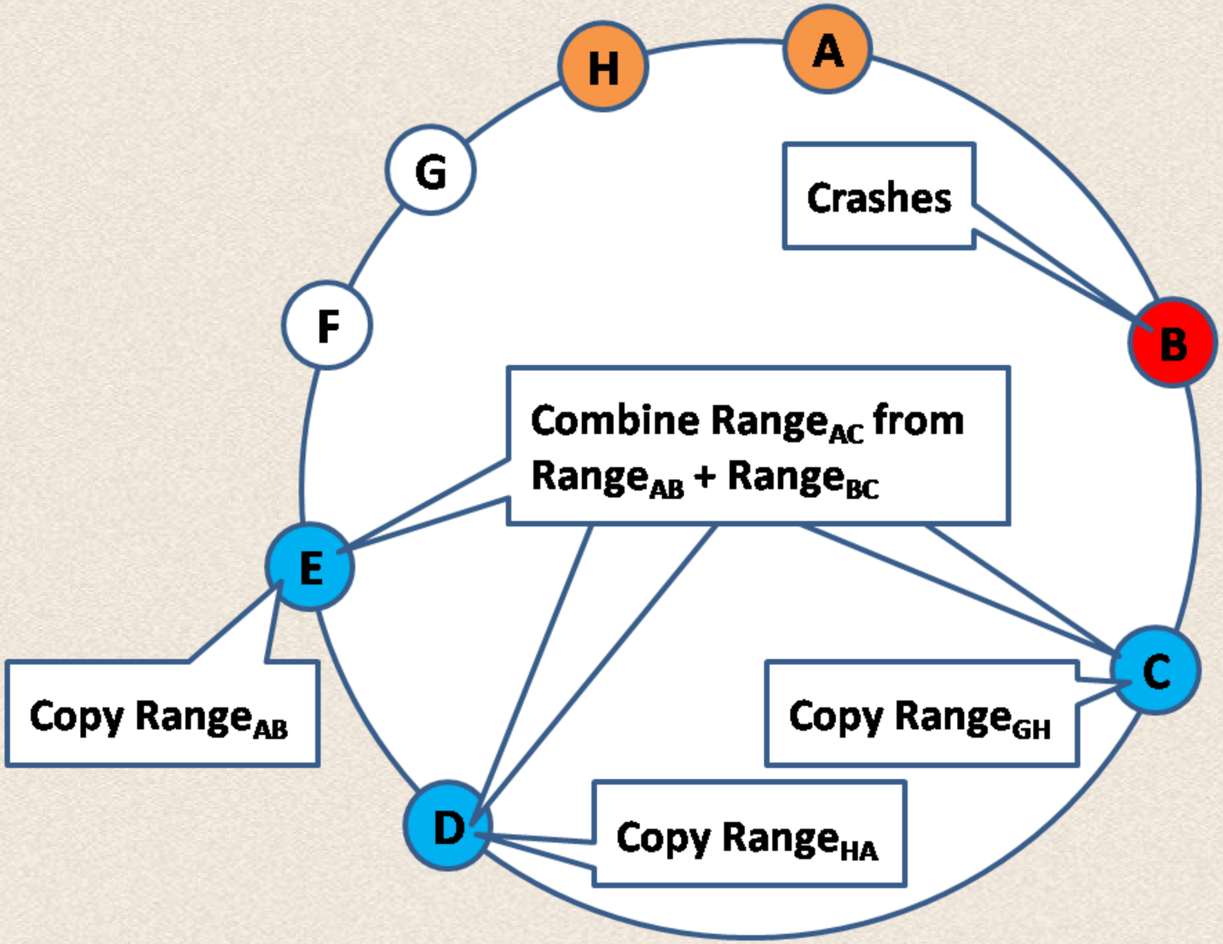
When a New Node Join Network

H, A, X, B, C, D will update the membership synchronously
And then asynchronously propagate the membership changes to other nodes


H
A
X
G
Drop Range$_{GH}$
F
Tell H, A, B, C, D that I join
B
E
Split Range$_{AB}$ into Range$_{AX}$ + Range$_{XB}$
C
Drop Range$_{AX}$
D
Drop Range$_{HA}$

When Existing Node Leaves Network

Asynchronously propagate the membership changes to other nodes



Database Integrity Outline

- Database Integrity Means
- Do We Really Need Consistency?
- Eventually Consistent
- Variations of Eventually Consistency
- Problem in Strict Schema
- Schema-Free

Database Integrity Means

- Ensure data entered into the database is **accurate**, **valid**, and **consistent**. *Three basic types of integrity constraints*:
  - **Entity integrity**, allowing no two rows to have the same identity within a table.
  - **Domain integrity**, restricting data to predefined data types.
  - **Referential integrity**, requiring the existence of a related row in another table, e.g. a customer for a given customer ID.

Do We Really Need Consistency?

- In strict OLTP environment (e.g. banking and ERP) data consistency is heart of the system.
- But even in Amazon (e-commerce) real-time consistency is not really needed.
- In large shared data environment such Facebook, Digg, Yahoo, Google, etc. data consistency can be relaxed
- Systems with strong ACID have poor performance.

Eventually Consistent

- Specific form of weak consistency
- If no new updates are made, eventually all accesses will return the last updated value.
- System does not guarantee subsequent accesses will return the updated value.
- A number of conditions need to be met before the value will be returned.

Variations of Eventually Consistency

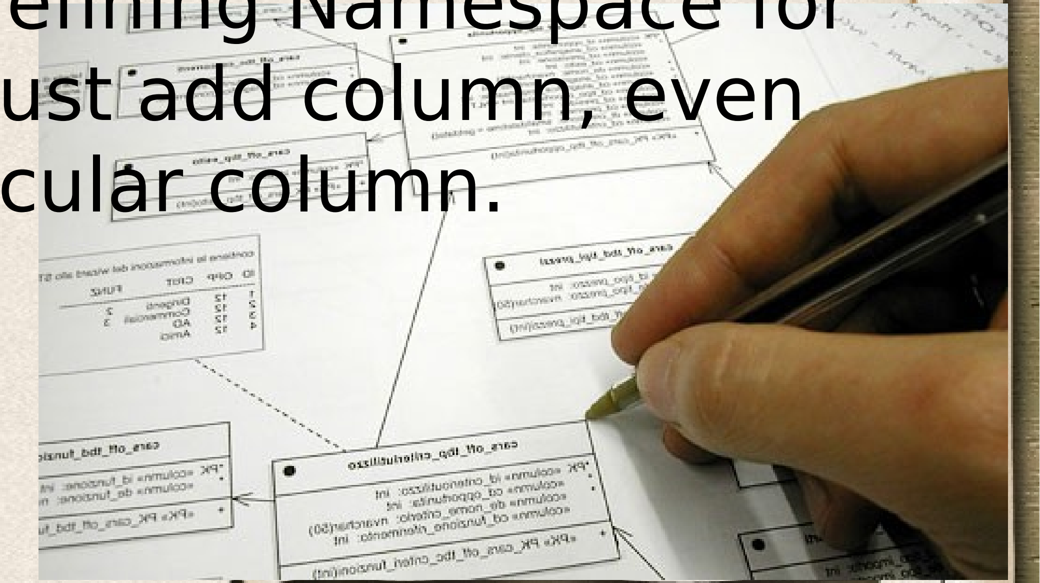
- Causal consistency
- Read-your-writes consistency
- Session consistency
- Monotonic read consistency
- Monotonic write consistency

Problem in Strict Schema

- Agile methodology is about changing adoption
- Dynamic Frameworks (e.g. Ruby on Rails, Django, and Grails, Symfony) are now widely used
- In many cases it is hard to migrate across database
- Adding more columns leaves null values in [illegible].

Schema-Free

- Enable to add column in row level. Not restricted to column level.
- Each rows only use column they need (saving space).
- All we need to do is defining Namespace for tables. Then we can just add column, even another table in particular column.
- No more integration **headache**



Conclusion & (not a) Summary

- NoSQL is yet another form of database.
- NoSQL don't intend to replace RDBMS.
- It is database alternative in Large data shared environment.
- Relaxing consistency will boost database availability and performance.
- There is no *Free Lunch* and *Silver Bullet* in database technologies.

Thank You