DOAG
Deutsche ORACLE-Anwendergruppe e.V.

DOAG-Regionaltreffen München/Südbayern
Donnerstag, 17. Februar 2011 um 17:00 Uhr

NOSQL –
Only a Hype ? Or the trend for the next future ?

NO:SQL – NOT ONLY SQL

Agenda

1	**Why?**
2	ACID versus BASE – Horizontal versus vertical scalability
3	CAP Theorem
4	NoSQL – most important ideas
5	Some NoSQL DB examples
6	Oracle NoSQL features
7	Summary and discussion

Timeline: ~ 40 min + 5min discussion

The Big Battle?

NO:SQL

Cassandra

RainStor

mongoDB

Apache CouchDB

Apache Solr

Lotus

ORACLE

MySQL

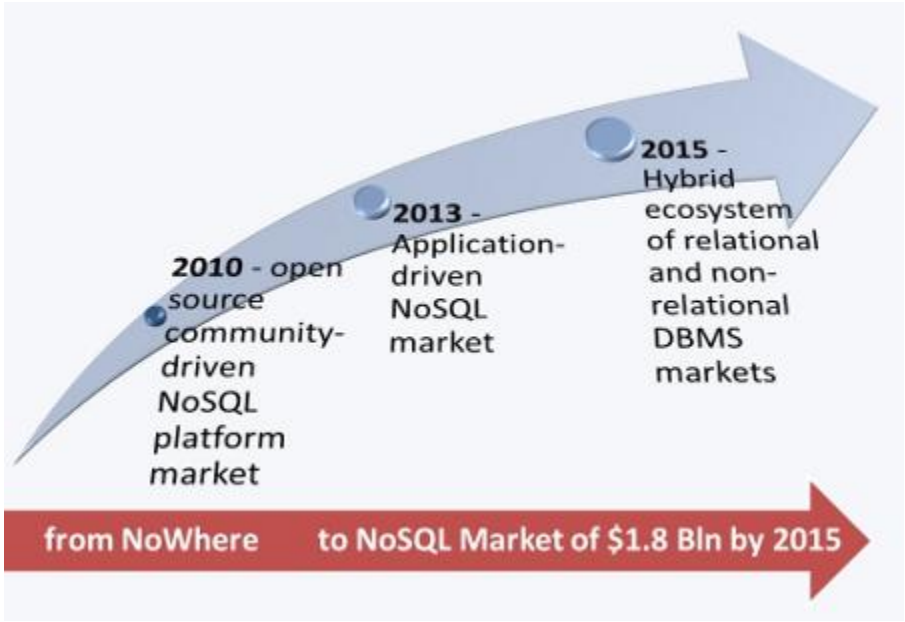
Microsoft SQL Server 2008

DB2

SYBASE

Access 2003

NoSQL Market Forecast 2011-2015



….. According to Market Research Media's latest forecasts, the worldwide NoSQL market is expected to reach $1.8 Billion by 2015 at a CAGR of 32% between 2011 and 2015. NoSQL market will generate $5.6 Billion revenues over the period 2011 – 2015 ……

See: http://www.marketresearchmedia.com/2010/11/11/nosql-market/

NoSQL – Not Only SQL – Why?

- Current popular problems in the database world
  - Excessive growth of data
  - Performance and scalability

Consequences:

⇒ Increasing hardware costs (especially the increasing higher requirements of storage hardware)!

⇒ Unbelievable high license cost for strictly required features for commercial high-volume database like EE partioning, inMemory TimesTen

I‘m cheaper!

NoSQL‘i

NoSQL – Not Only SQL – Why?

- Current popular problems in the programming world
  - Lack of experience with SQL databases
  - Unsolved problem of easy object to relational mapping
  - Complex test automation for SQL statements
  - Mindset to SQL => old-fashioned, to declarative

Consequences:

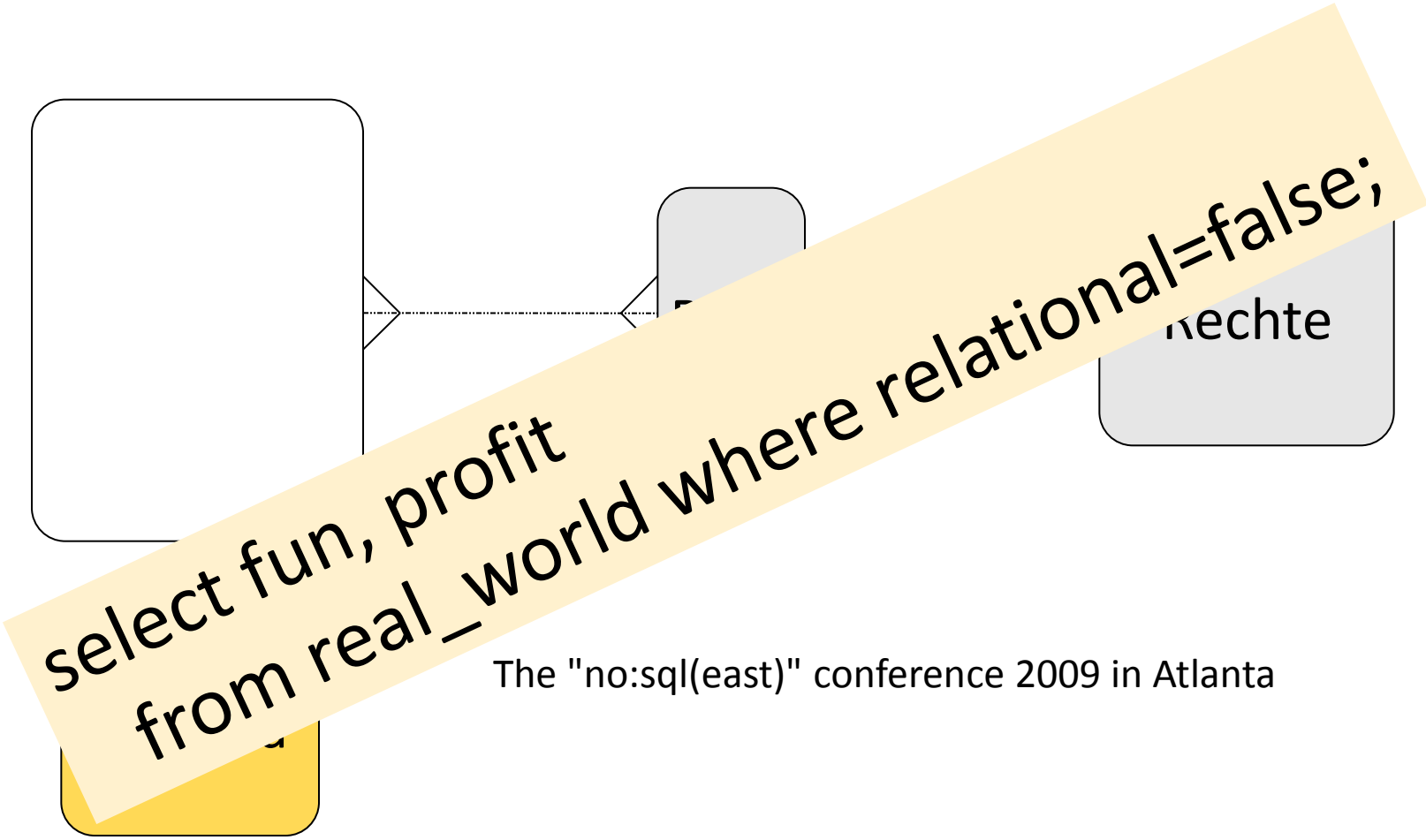
⇒ Unbelievable complex persistence layers to unlink java from the database

⇒Thousands of ideas to avoid the usage of SQL

Lean programming!

Relationali

No:SQL?



The "no:sql(east)" conference 2009 in Atlanta

NoSQL – Not Only SQL – Why?

- Current popular problems in the distributed server world
  - How we can grantee unlimited scalability?
  - How to a upgrade to a new software version?

Consequences:

⇒ Unbelievable high license and hardware costs …….

I‘m cheaper!

NOSQL‘i

NoSQL – Not Only SQL – Why?

- Current popular problems in the 24/7 world
  - How change a database schema without downtime?
  - How to change/delete a huge amount of values at runtime?

Consequences:

⇒ Again unbelievable high license and hardware costs…….

I‘m easyer!

NOSQL‘i

NOSQL – Definition

- A pure NoSQL DB's is:
  - non relational
  - schema free
  - from the scratch distributed multi-node architecture
    - Goal: Shared nothing cluster solutions
  - easy replication mechanisms
  - Simple API (**Not only SQL = NoSQL**)
  - No ACID consistency model
  - Open Source

BUT ! The big disadvantage at the moment

=> mostly no easy language to search
=> Complex Joins? Not implemented

NOSQL – Polyglot Persistence

- Fight for free Database Software

- Use the right database for your problem
  - Not only the default one!

- More effort to evaluate the requirement and search after the right solution for your problem

ACID versus BASE

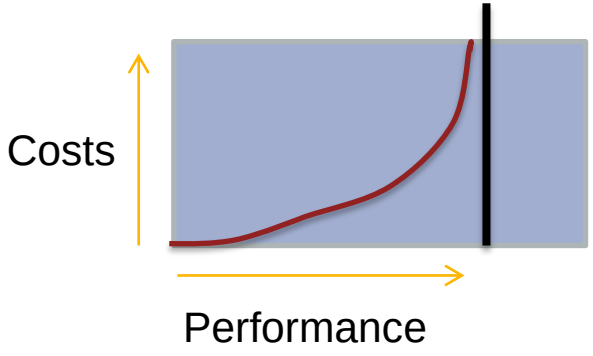
Horizontal versus vertical scalability

Our Oracle religion - ACID

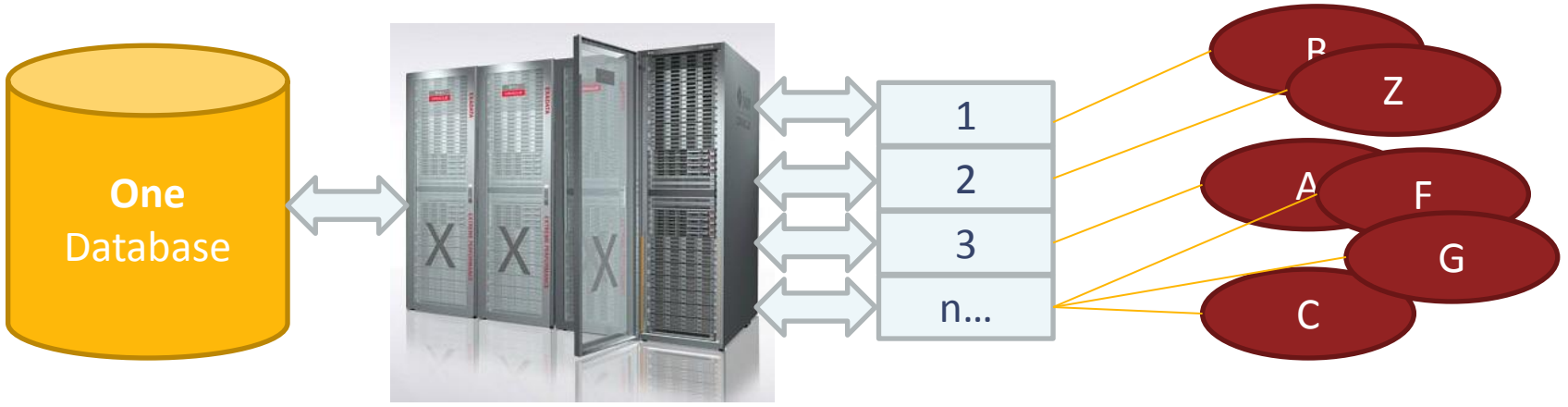
- **ACID** are the transactional properties of database management systems
  - **A**tomicity
    - All of the operations in the transaction will complete, or none will.
  - **C**onsistency
    - The database will be in a consistent state when the transaction begins and ends.
  - **I**solation
    - The transaction will behave as if it is the only operation being performed upon the database.
  - **D**urability
    - Upon completion of the transaction, the operation will not be reversed.

Horizontal scalability

- One huge central database
  - Bigger hardware helps to solve the performance issues



Only scalable to one absolute value



ONE DATABASE ! | n Storage cell machines | n Instance | n App. Server

BUT?

Do we really need strong ACID this in every Situation?

GPS Tracking

Amazon

E-Mail Archive

Xing

Ebay

Data ware house

Facebook

IT Monitoring

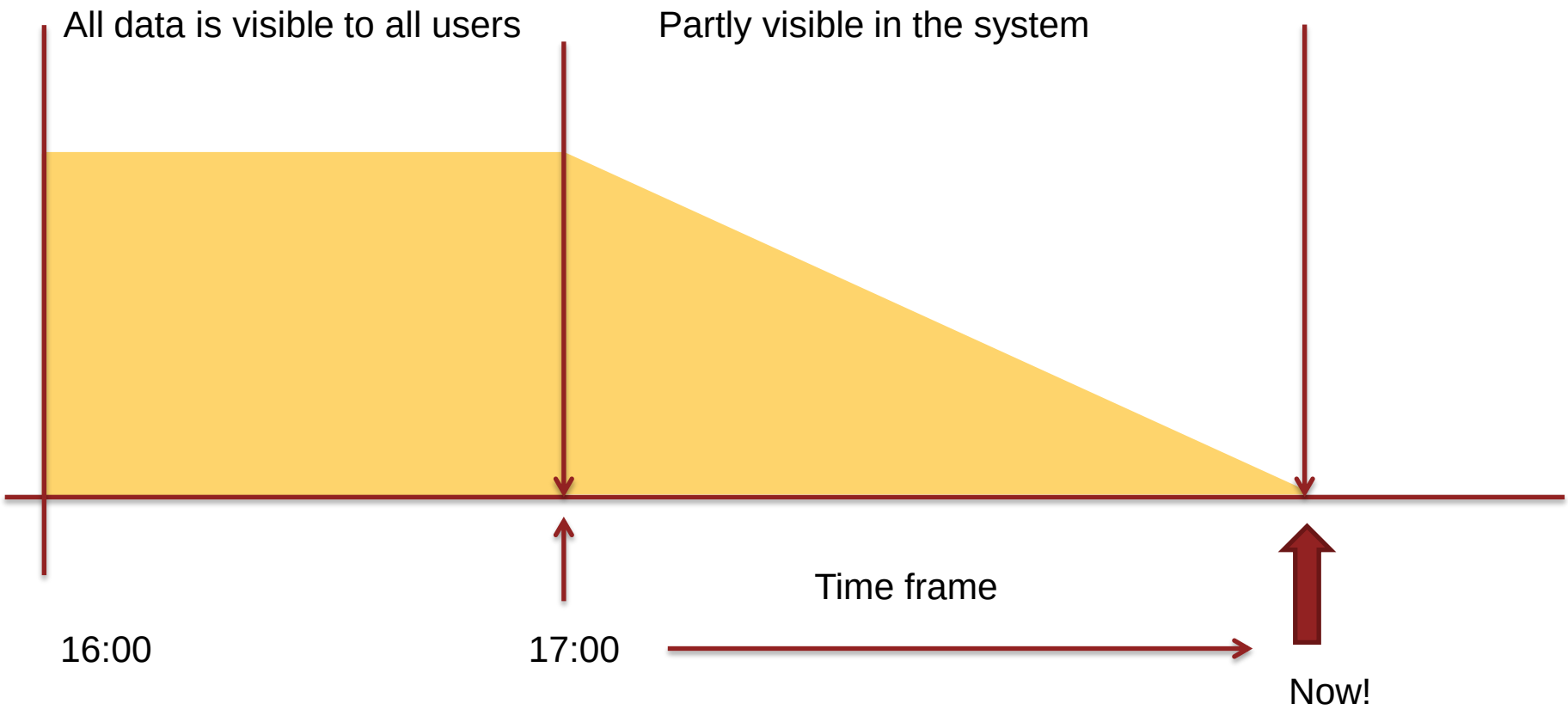
Online News
Spiegel.de

traffic guidance systems

syslog

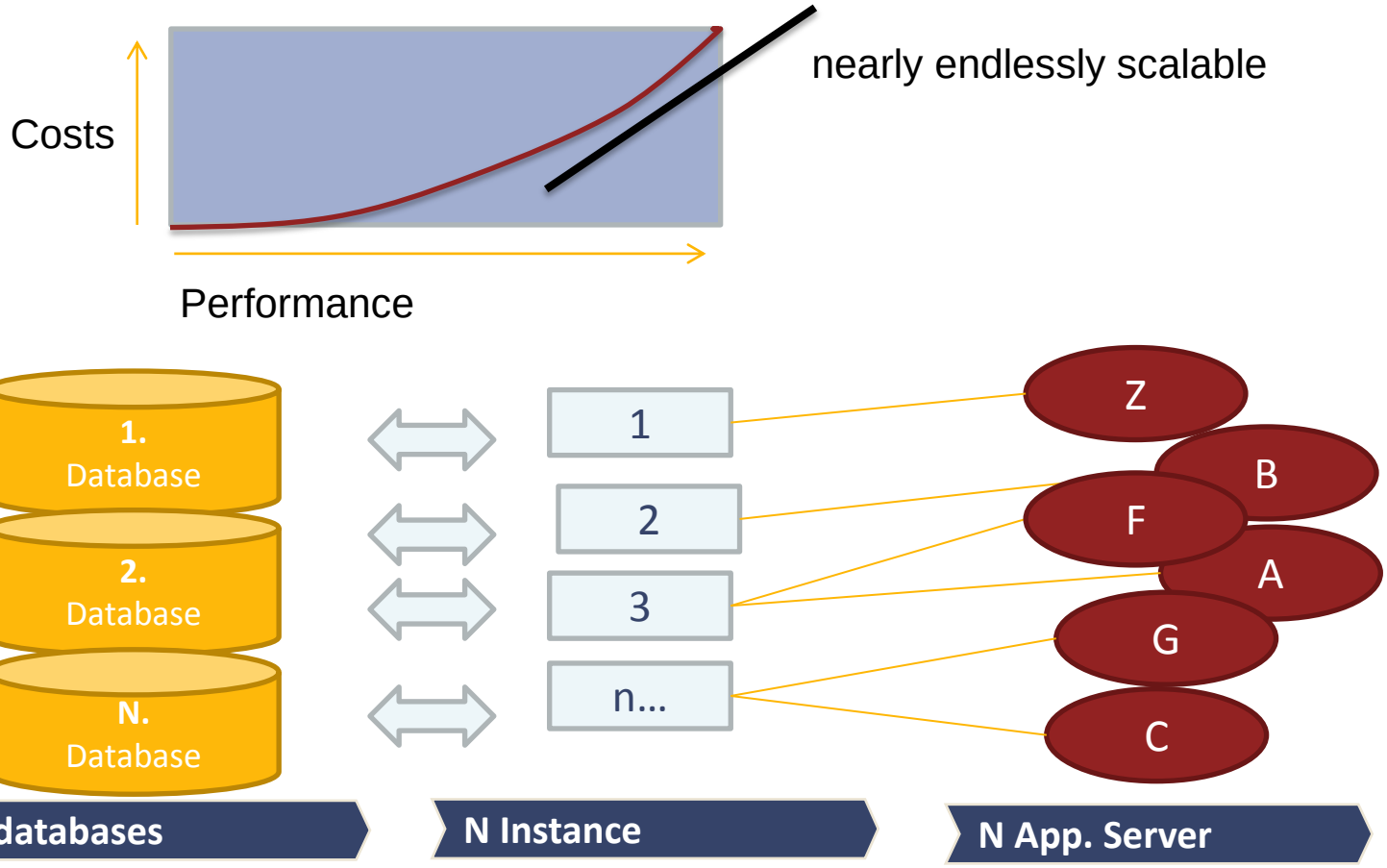
Toll collect

The persistency of the data in a system



Vertical scalability

- Small components working together in parallel
  - All Components are independent



CAP Theorem

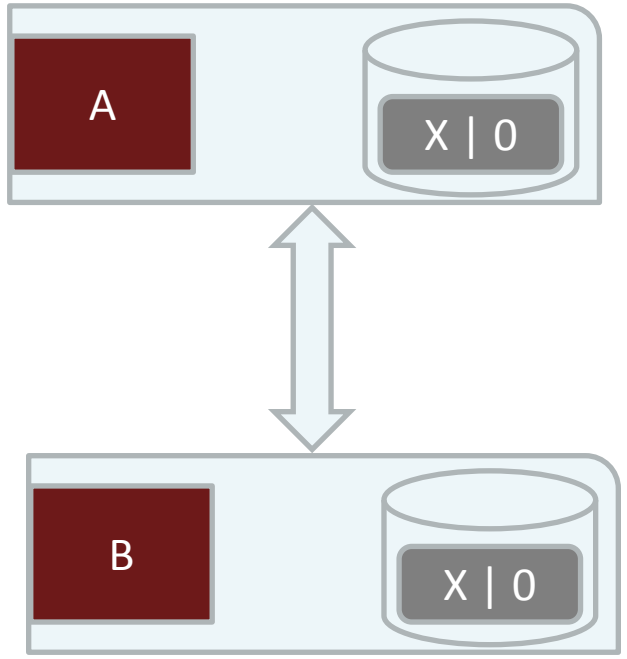
Multi node environments

CAP Theorem

- Start situation - Two nodes

Requirements:

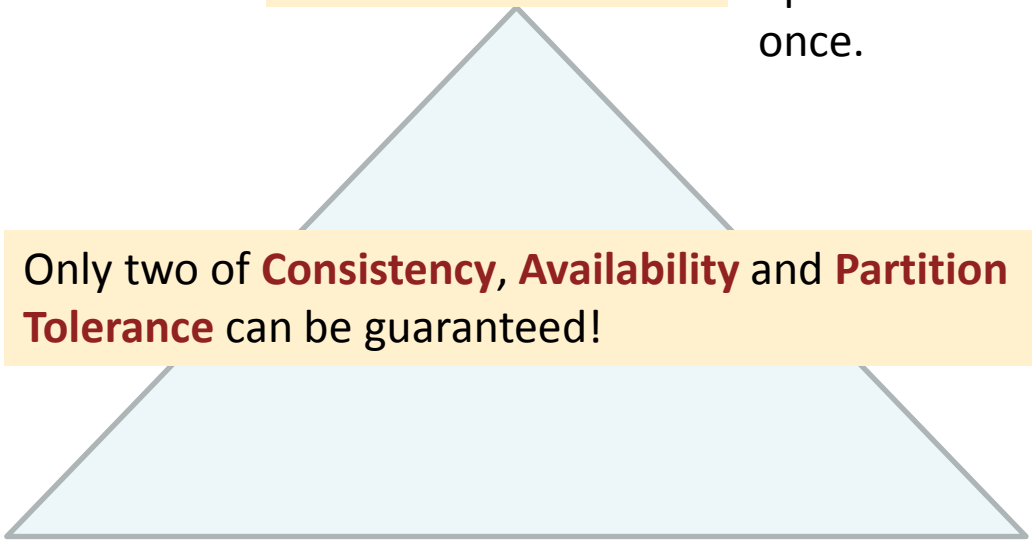
- **Consistency**
- **Availability**
- **Partition Tolerance**



Eric Brewer : ACM Symposium 2000

Brewer's CAP Theorem for a distributed model

- Is it possible to solve all this requirements?



Consistency

The client perceives that a set of operations has occurred all at once.

Only two of **Consistency**, **Availability** and **Partition Tolerance** can be guaranteed!

Partition Tolerance

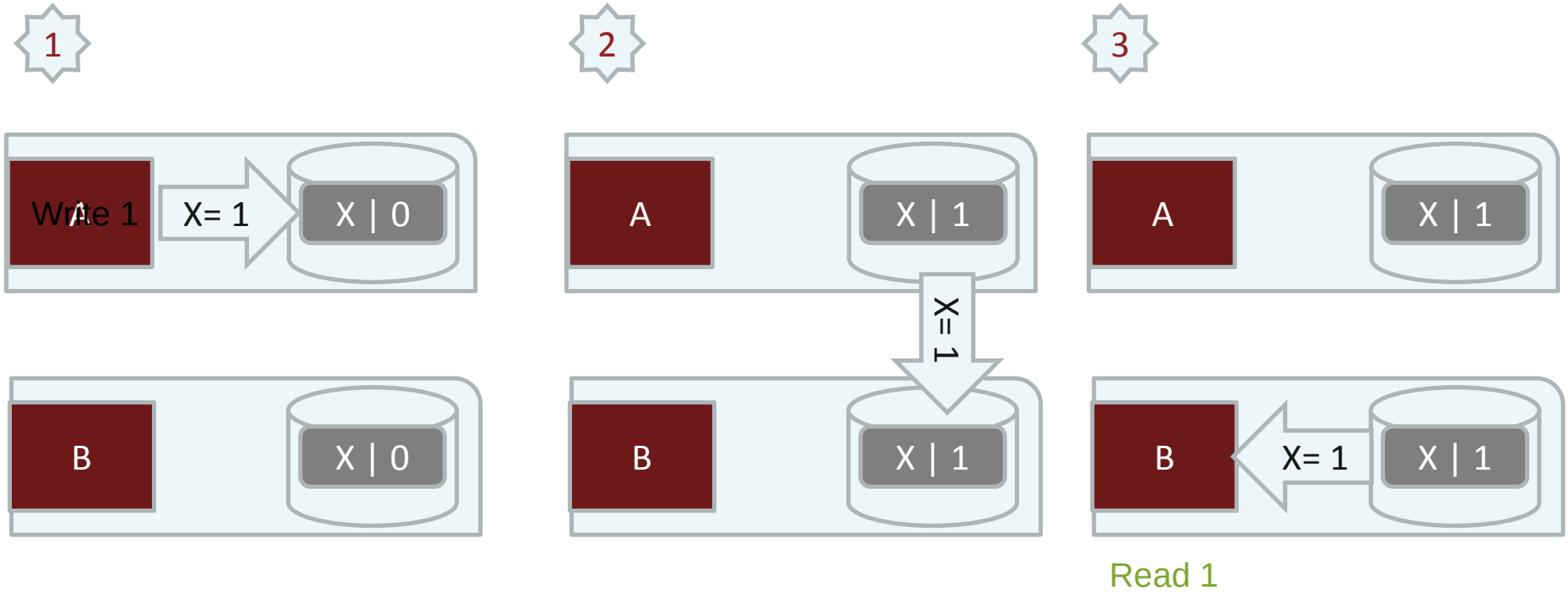
Operations will complete, even if individual components are unavailable.

Availability

Every operation must terminate in an intended response.

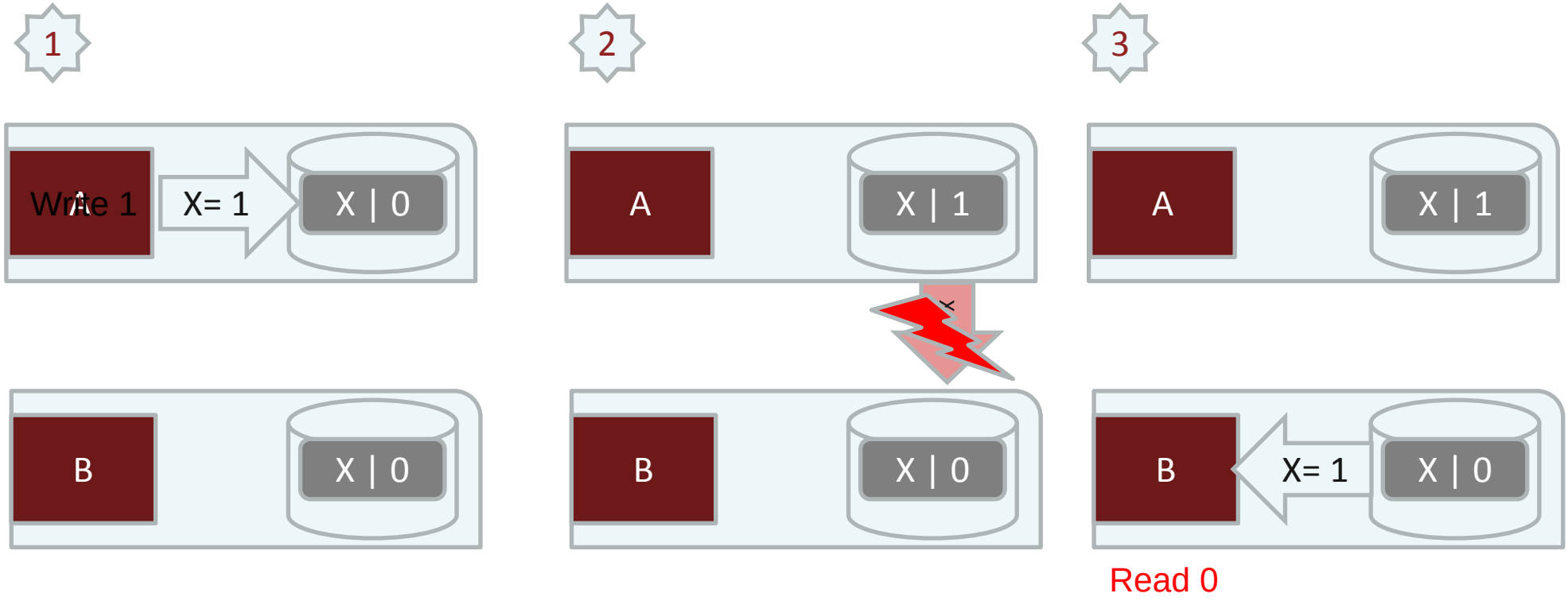
Theory of the CAP Theorem (1)

- Sunny Day Scenario for a update



Theory of the CAP Theorem (2)

- Error Scenario for a update



Theory of the CAP Theorem (3)

- How we solve the problem?
  - Drop Partition Tolerance
    - Only Node survived
  - Side effect => we drop Availability
  - Drop Consistency
    - Live with inconsistencie

A other consistency model - BASE

- A latency tolerant alternative to ASCI
  - Basically Available
  - Soft state
  - Eventual consistency
    - Given a sufficiently long period of time, over which no updates are sent, we can expect that during this period, all updates will, eventually, propagate through the system and all the replicas will be consistent

ACID vs. BASE

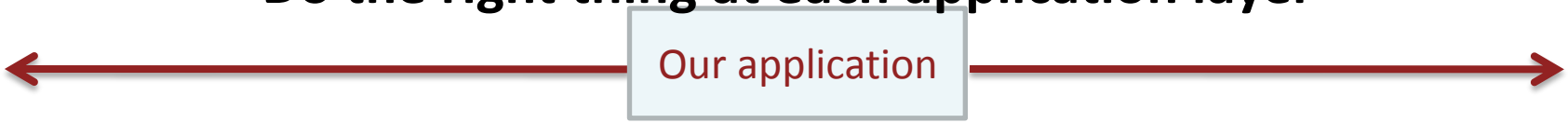
ACID

- Strong consistency
- Isolation
- Focus on “commit”
- Nested transactions
- Availability?
- Conservative (pessimistic)
- Difficult evolution (e.g. schema)

BASE

- Weak consistency
  - stale data OK
- Availability first
- Best effort
- Approximate answers OK
- Aggressive (optimistic)
- Simpler!
- Faster
- Easier evolution

Do the right thing at each application layer



Source: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

NoSQL architecture

Some base ideas

Some basic ideas and concepts

- Map and Reduce
- Consistent Hashing
- MVCC-Protocol
- Vector Clocks
- Paxos
- Wide Column Stores

Map and Reduce

- Base ideas from the functional programming
  - List processing
  - Programing model – two functions
    - Processes input key/value pair
    - Produces set of intermediate pairs

```
map (input) -> list(intermediate_value)
```

    - Combines all intermediate values for a particular key
    - Produces a set of merged output values (usually just one)

```
reduce (out_key, list(intermediate_value)) -> list(out_value)
```

Source:http://labs.google.com/papers/mapreduce-osdi04-slides

Map and Reduce

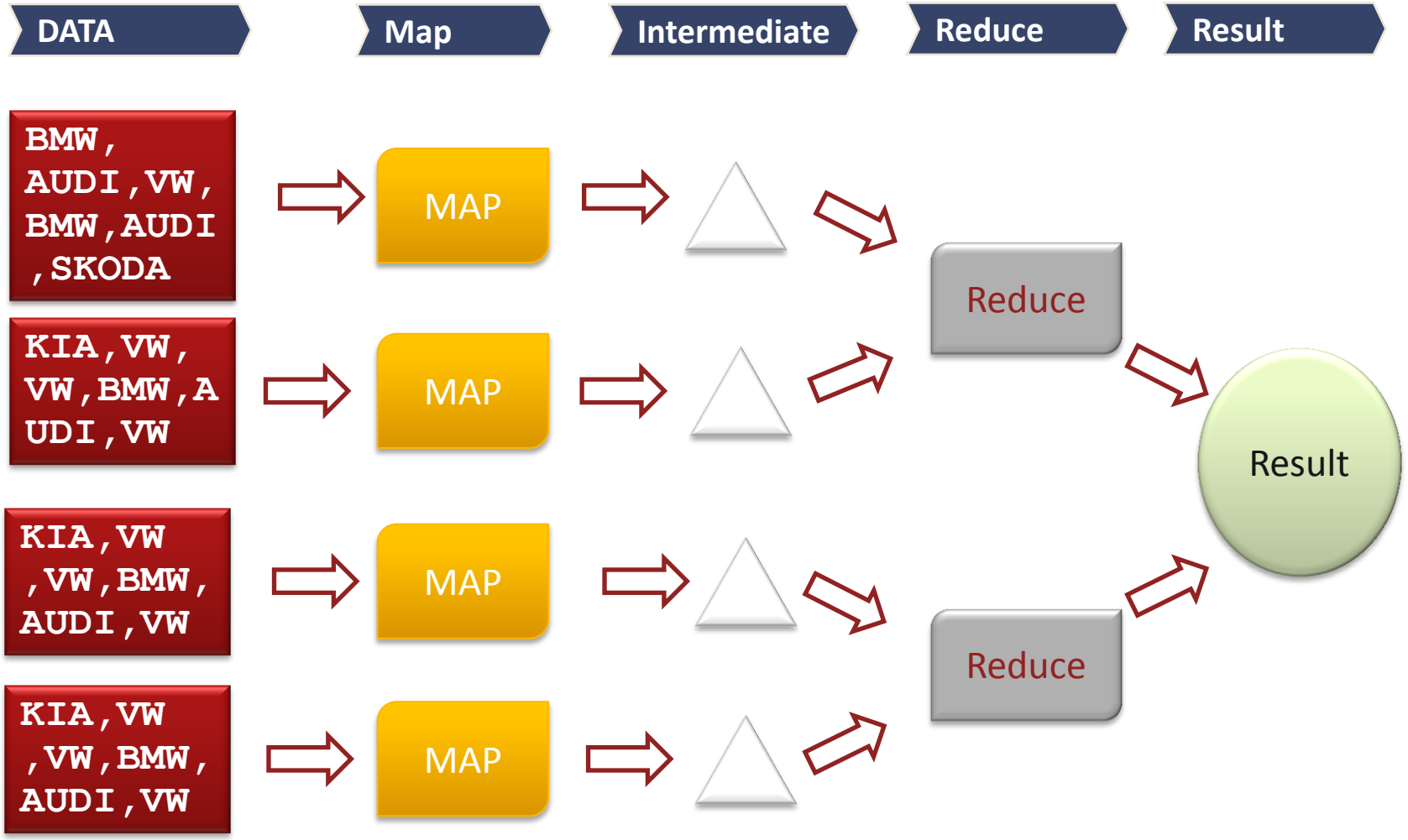
- Example – Count of Tokens

```
select   company
       , count(*)
 from cars_in_stock
group by company;
```

```
--data
token1 = 'BMW, AUDI,VW,BMW,AUDI,SKODA'
token2 = 'KIA, VW  ,VW,BMW,AUDI,VW'

map (token) : f() {count word} => List[]

-- first Run for each token list
map( tocken1 )    => list1 [ ('BMW,1,1')   , ('AUDI,1,1')
                           , ('VW,1')      , ('SKODA,1')                  ]
map( tocken2 )    => list2 [ ('BMW,1')     , ('AUDI,1')
                           , ('VW,1,1,1'),              , ('KIA,1')     ]
-- Intermediate values
map (list1+list2) => list3 [ ('BMW,1,1,1'),  ('AUDI,1,1')
                           , ('VW,1,1,1,1'), ('SKODA,1') , ('KIA,1')     ]

reduce : f() { ++ } => List []

-reduce to the result
reduce(list3)  => result  [ ('BMW,3')      , ('AUDI,3')
                          , (VW,4)         , ('SKODA,1') ,  ('KIA,1')    ]
```

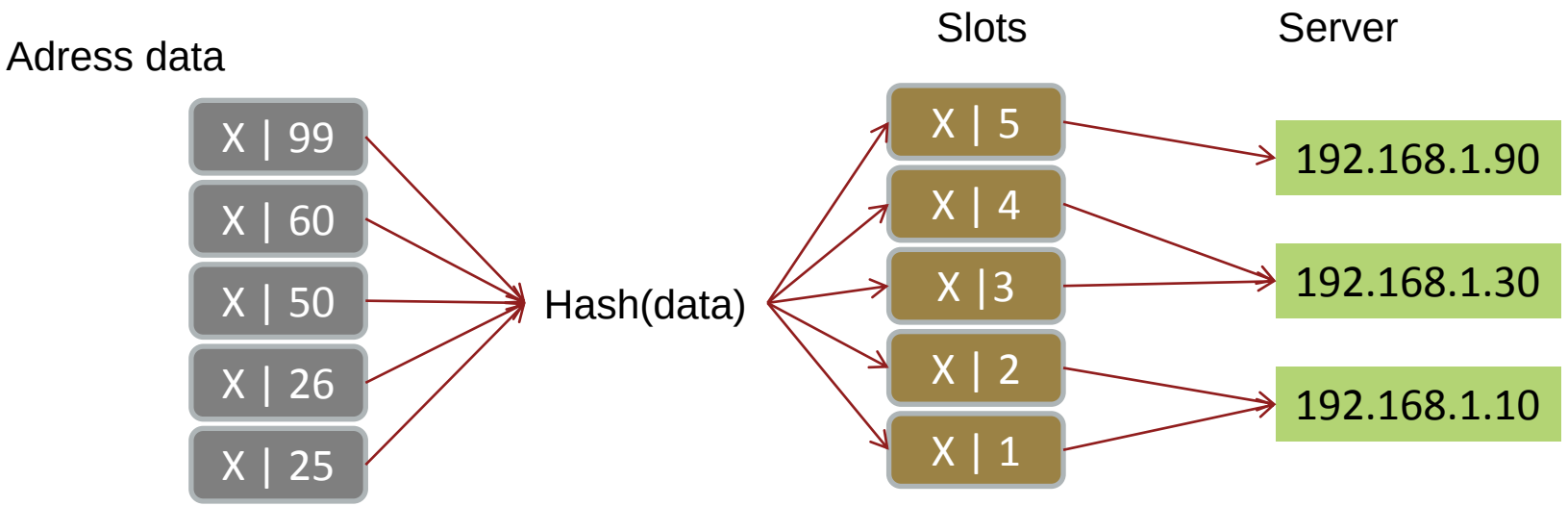
Map and Reduce

- Very easy to implement in parallel



Consistent Hashing

- Consistent Hashing
  - A hash function to find the data in the right slot
  - The addition or removal of one **slot does not significantly change the mapping** of keys to slots.

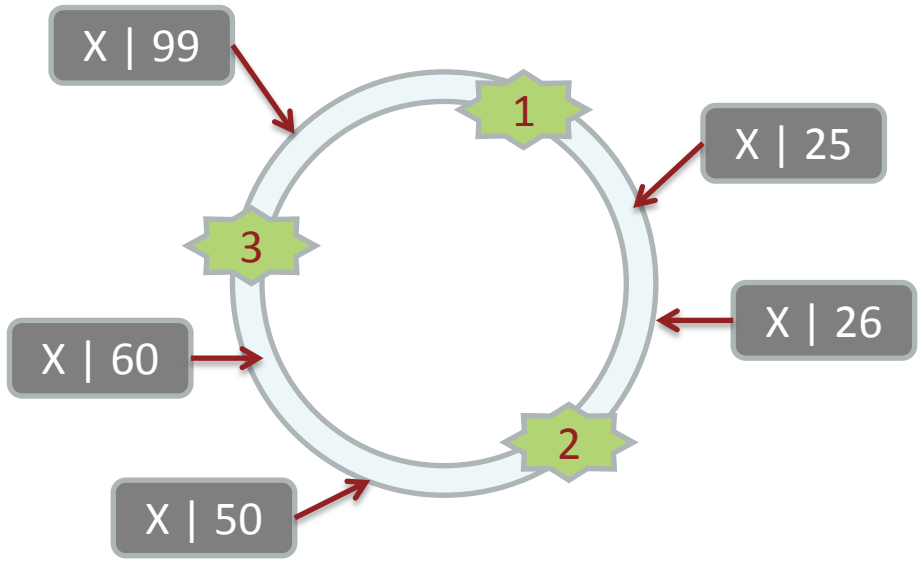


See: http://en.wikipedia.org/wiki/Consistent_hashing

Consistent Hashing (1)

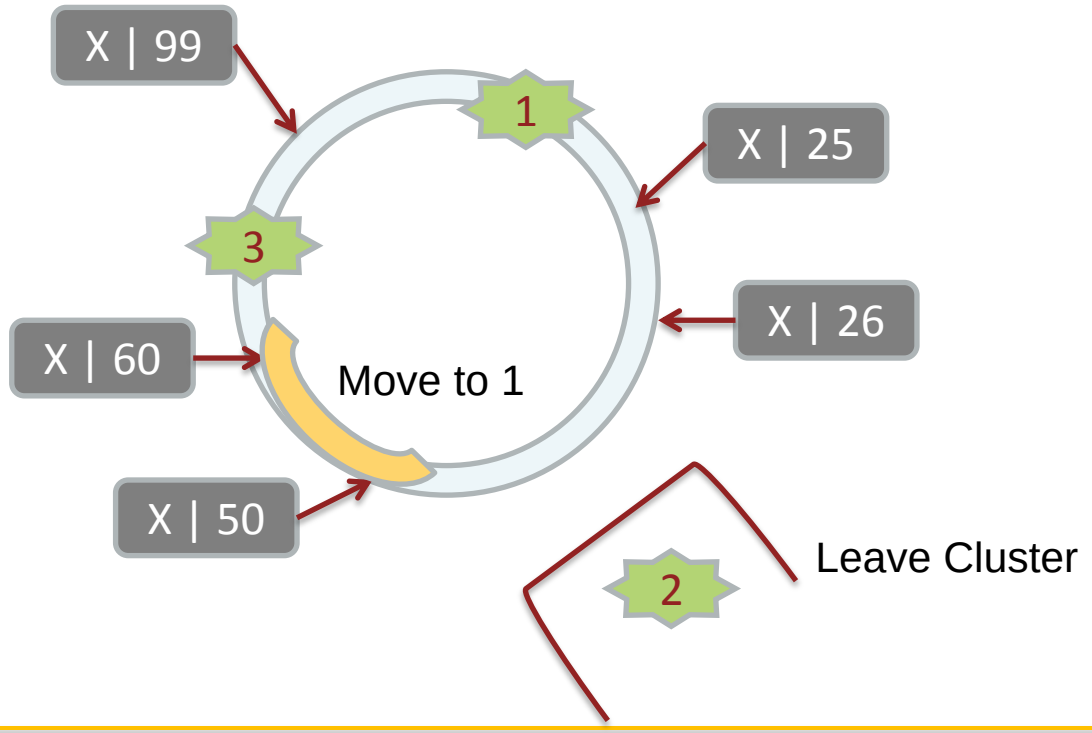
- Distributed Servers hold the data
  - Hash(server) => Range of the storage location
  - hash(data) => storage location
  - Example:

1. hash(192.168.1.10) = 15
2. hash(192.168.1.30) = 30
3. hash(192.168.1.90) = 80



Consistent Hashing (2)

- Distributed servers hold the data
  - If one server leave the cluster, only his data moved to the next server
  - The addressing at self is not changing!



MVCC-Protocol

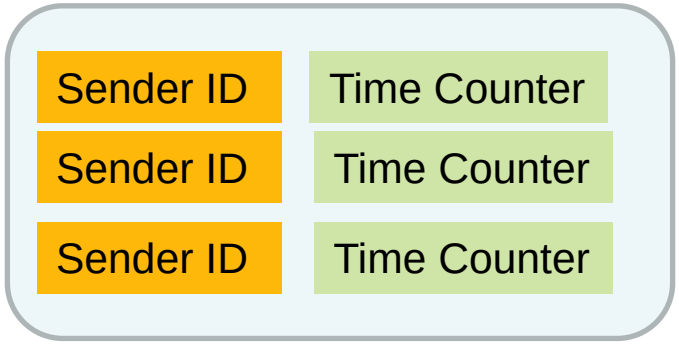
- Multiversion concurrency control (abbreviated MCC or MVCC)
  - provide concurrent access to the database
  - a database will implement updates not by deleting an old piece of data and overwriting it with a new one
  - instead marking the old data as obsolete and adding the newer "version."
  - multiple versions where stored, but only one is the latest.

Oracle DB since V3!

See: http://en.wikipedia.org/wiki/Multiversion_concurrency_control

Vector clock

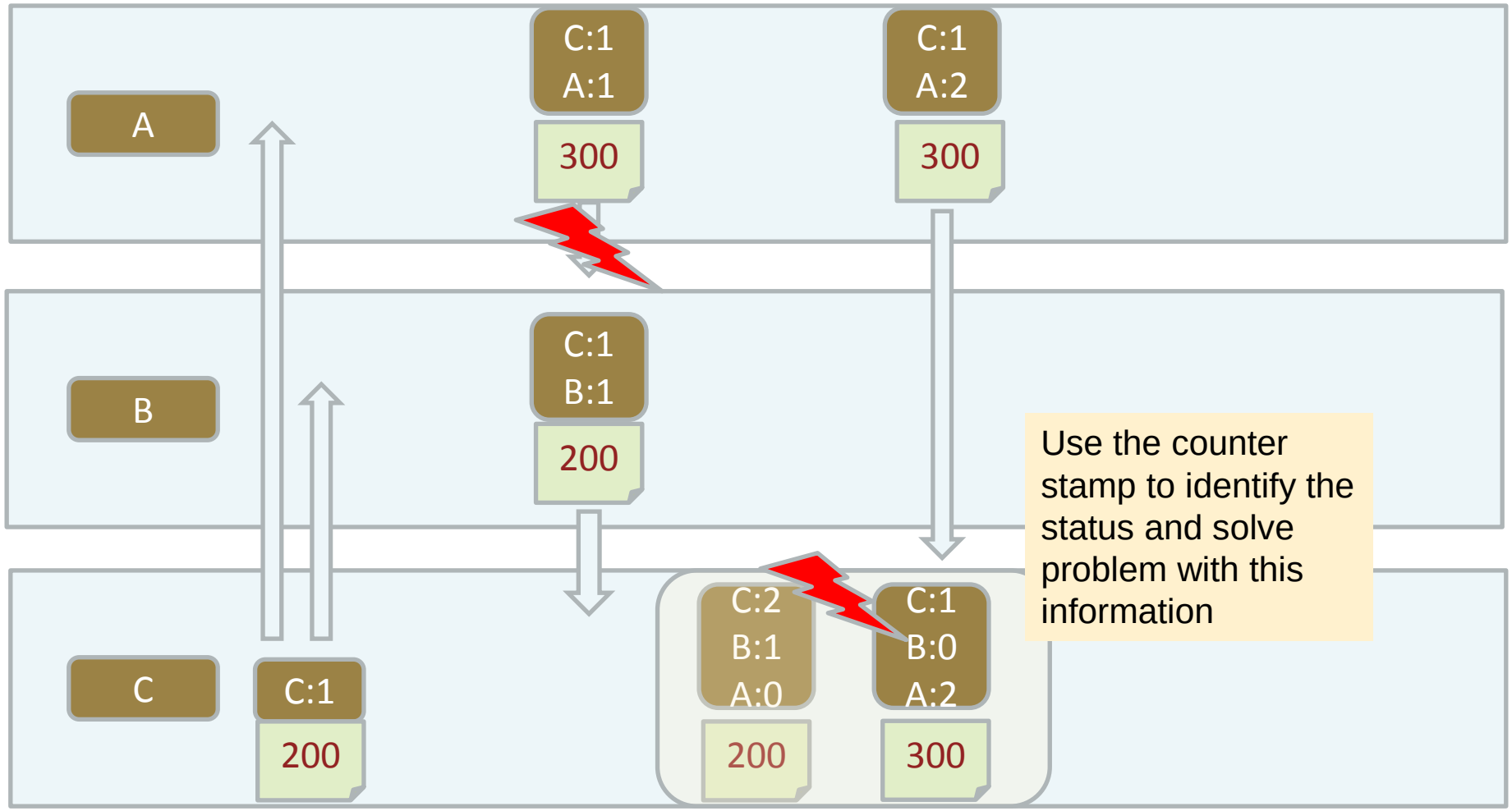
- algorithm for generating a partial ordering of events in a distributed system and detecting causality violations.
- Contain the state of the sending process's logical clock.



See: http://en.wikipedia.org/wiki/Vector_clock

Vector clock

- Example



Paxos

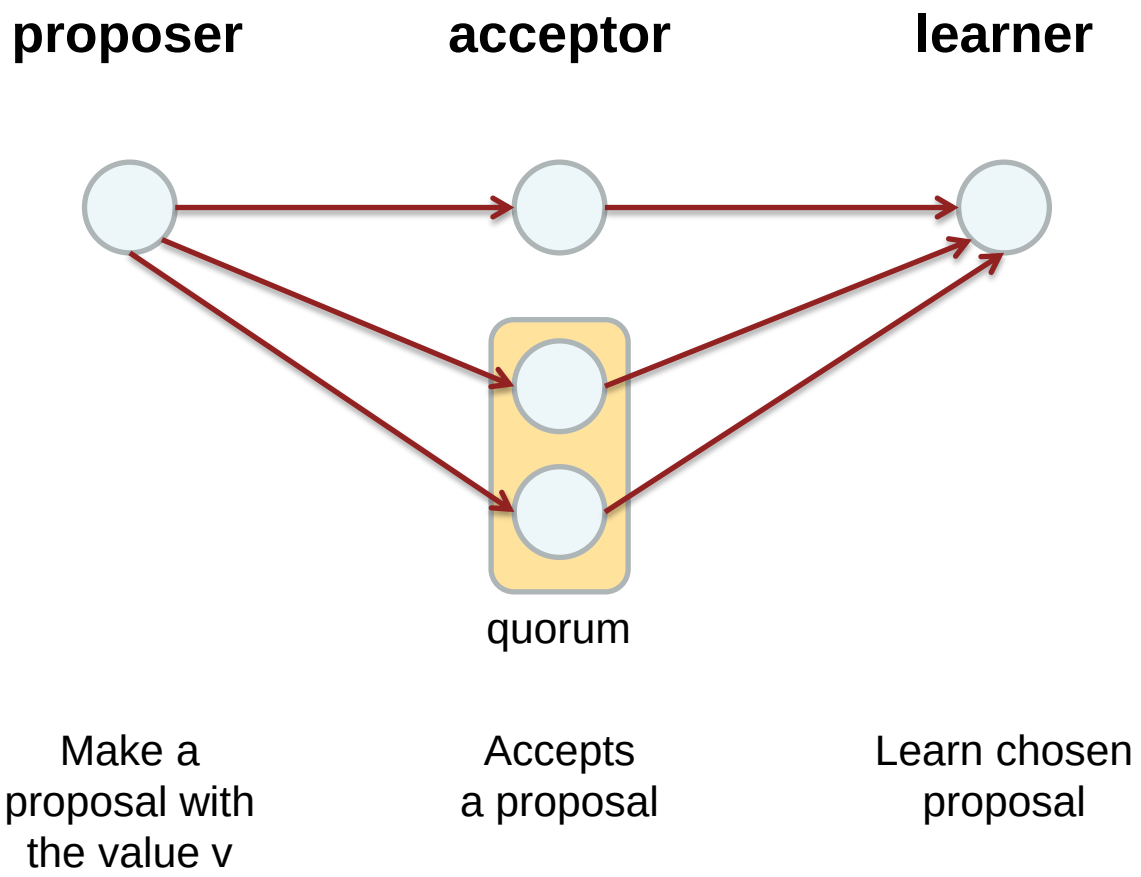
- **Paxos** is a family of protocols for solving consensus in a network of unreliable processors.
  - Consensus is the process of agreeing on one result among a group of participants.

See: http://en.wikipedia.org/wiki/Paxos_algorithm

Opposite to the ACID two phase Commit (2PC => all processes can hang until all commit)

Paxos

- Roles in a consensus Model



Wide Column Stores

- Opposite of the row oriented table structure (for example in a oracle database)

- Column oriented

Example: Wide Column stores
ID:NAME:CITY:SALARY

“1,2,3” : “Scott,hugo,tiger”: “Huston,Munic,New York” : “1000,4500,6000”

Examples NoSQL

Some Examples

NOSQL Categories – for each problem a solution

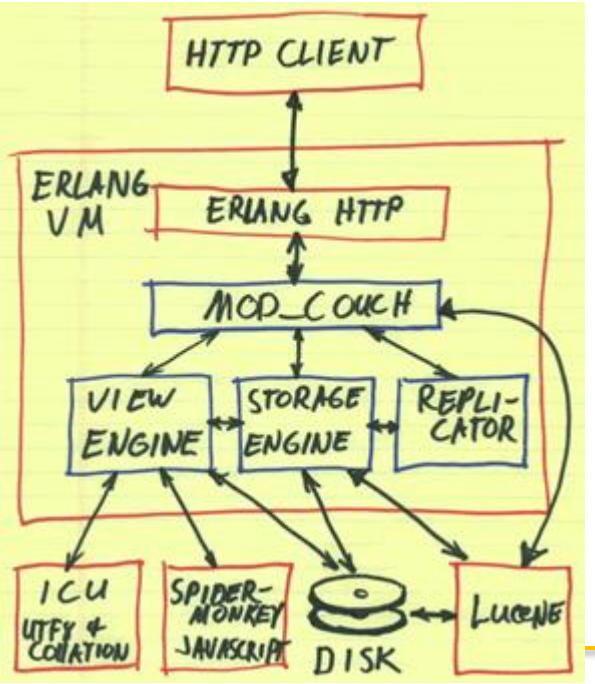
- Document store
- Graph
- Key Store
  - Key/value store on disk
  - Key/value cache in RAM
  - Eventually-consistent key-value store
  - Key-value stores implementing the Paxos algorithm
  - Hierarchical key-value store
  - Ordered key-value store
- Multivalue databases
- Object database
- Tabular
- Tuple store

Source: http://en.wikipedia.org/wiki/NoSQL

Document store -CouchDB

- Apache CouchDB
  "cluster of unreliable commodity hardware Data Base"
  - document-oriented database
  - queried and indexed in a MapReduce fashion using JavaScript.
  - incremental replication
    with bi-directional conflict detection and resolution.

http://couchdb.apache.org/

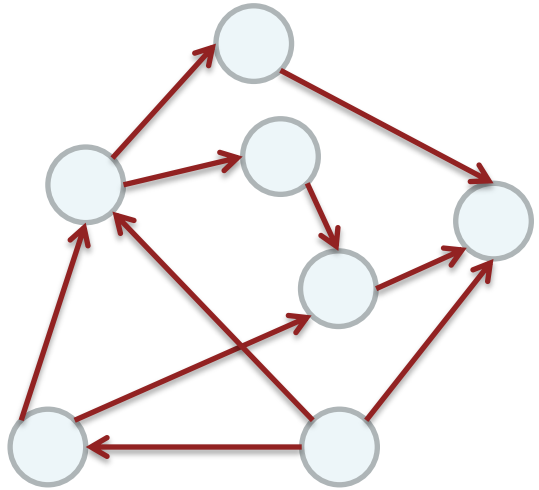


- Store documents
  - Example of a document:

```
Surname: Gunther
Adress : Munich
Hobbies: Oracle DB
```

Graph database

- Neo4J
  - Neo4j is a graph database
  - fully transactional database
  - stores data structured as graphs.



Neo4j
the graph database

http://neo4j.org/

Key-Value Database

- Redis
  - open source, advanced key-value store.
  - It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets.

Key	Value
Key	Value
Key	Value
Key	Value

In Memory Database

http://redis.io/

ColumnFamily-based data model

- Cassandra
  - The Apache Cassandra Project develops a highly scalable second-generation distributed database, bringing together Dynamo's fully distributed design and Bigtable's ColumnFamily-based data model.
  - Implemented in Java
  - Example: Facebook , Twitter

http://cassandra.apache.org/

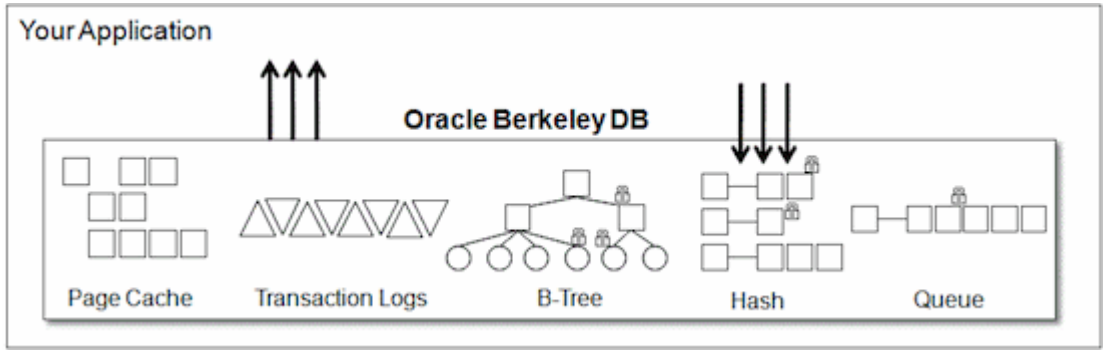
  - Schema less
    - Only entities (column families are defined at startup)
    - Add column at runtime

Oracle

?

Oracle Berkeley DB 11g

- Oracle Berkeley DB 11g
  - designed to store data as opaque byte arrays of data in key/value pairs indexed in one of the available access methods.
  - Create, read, update and delete (CRUD) operations on these key/value pairs is done using the BDB get/put or cursor position-based APIs.



ORACLE 11g
BERKELEY DB

http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html

Where we find this ideas in an Oracle RDBMS?

- Map and Reduce
  - ???
- Consistent Hashing
  - Hash partioning
- MVCC-Protocol
  - Default since V3
- Vector Clocks
  - May be in the inter process mechanism
- Paxos
  - ???

Oracle NoSQL Feature

- Mostly some Storage Feature in the database
  - Paritioning
  - Clustering
  - Bitmap Join Index

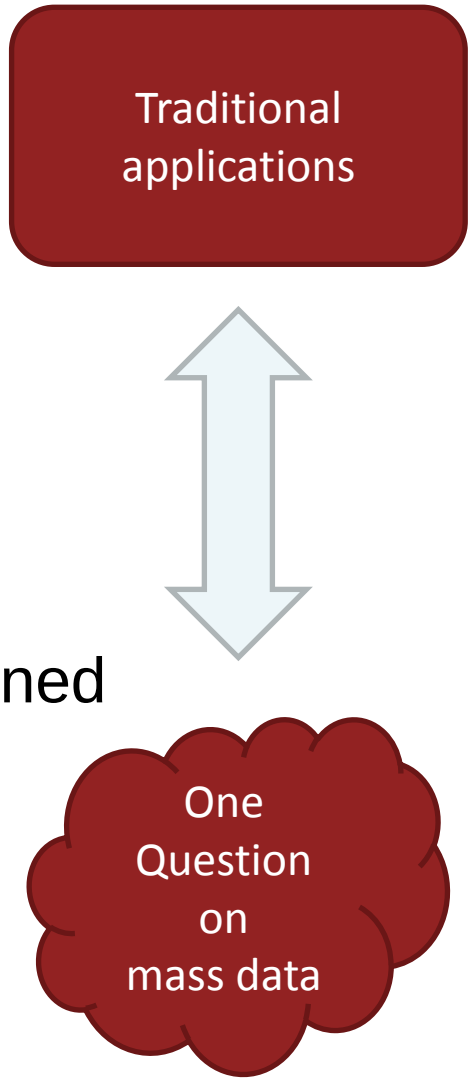
- But where is the scalability?
  - Cluster? Only Horizontal……

See: http://asktom.oracle.com/pls/apex/f?p=100:11:0::::P11_QUESTION_ID:2664632900346253817

Summary

- Relational databases
  => ALL Features First Solutions
  - Best for feature-rich solutions
  - Proven technologies

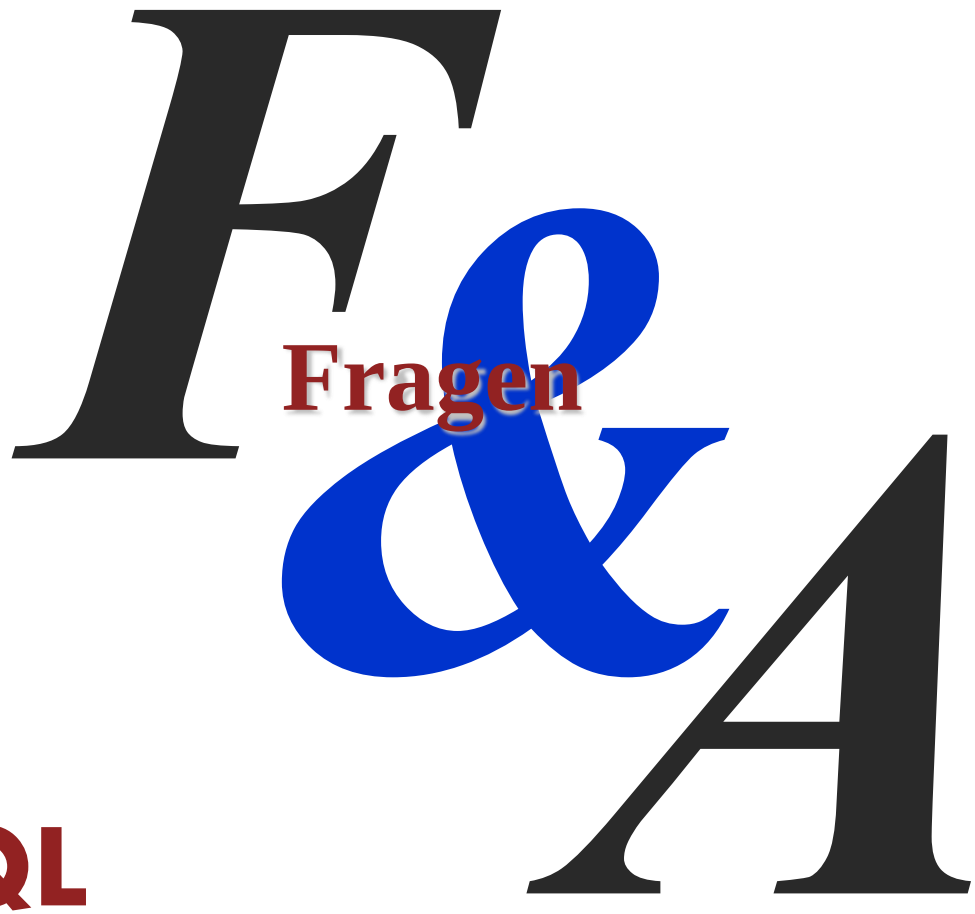
- NoSQL
  => Solve one problem First solution
  - Solves the problem, for that the DB is designed
  - In some situations=>
    the best solution to solve the problem



Conclusion

- From the technical view:
  - If we use the technologies to solve real problems both, strong ACID and BASE DB's will live together
  - Trend of the future!
- From the business view:
  - Save education for Java developers
  - Save license costs
  - Fight against the arrogance of the established companies
  - More a hype then a advance

Discussion



NO:SQL

More Information and Sources

- The NoSQL Websites:
  - http://nosql-database.org/
- CAP
  - http://en.wikipedia.org/wiki/CAP_theorem
  - http://www.julianbrowne.com/article/viewer/brewers-cap-theorem
- ACID versus Base
  - http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf
- Map And Reduce
  - http://labs.google.com/papers/mapreduce-osdi04-slides

More Information

- Paxos
  - http://informatik.unibas.ch/lehre/hs10/cs341/_Downloads/Workshop/Reports/2010-HS-DIS-P_Fath-PAXOS-Report.pdf

Books

- NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken
  - Stefan Edlich, Achim Friedland, Jens Hampe , Benjamin Brauer
    - Verlag: Hanser Fachbuchverlag (7. Oktober 2010)
    - Sprache: Deutsch
    - ISBN-10: 3446423559

Referent

Technology Consultant

Gunther Pippèrr
Dipl. Ing. Technische Informatik (FH)

>10 Jahre IT Beratungserfahrung
Freiberuflich tätig

Schwerpunkte der Beratungstätigkeit

- IT System Architekt und technische Projektleitung
- Web-Technologien
- Design und Implementierung von Datenbank-basierten Internet-/Intranet-Anwendungen
- Entwurf und Umsetzung von IT Infrastrukturen zum Datenmanagement mit Oracle Basis Technologien
- Training rund um die Oracle Datenbank

Beruflicher Hintergrund

- Dipl. Ing. Technische Informatik (FH) Weingarten
- Mehr als10 Jahre Erfahrung in komplexen IT Projekten zum Thema Datenhaltung/Datenmanagement
- Freiberufliche Projektarbeit
- 8 Jahre Geschäftsführer
- Consultant bei großen Datenbank Hersteller

Projekterfahrung

- Architekt für ein System zur Massendaten-Erfassung für Monitoring in der Telekommunikation
- Architekt und technische Projektverantwortung für ein Smart Metering Portal für das Erfassen von Energiezählerdaten und Asset Management
- Architekt und Projektleitung , Datenbank Design und Umsetzung für die Auftragsverwaltung mit Steuerung von externen Mitarbeitern für den Sprachdienstleister von deutschen Technologiekonzern
- Architekt und technische Projektverantwortung für IT Infrastrukturprojekte, z.B.:
  - Zentrale Datenhaltung für Münchner Hotelgruppe mit über 25 Hotels weltweit,
  - Messdaten Erfassung für russischen Kabelnetzbetreiber
  - Redundante Cluster Datenbank Infrastrukturen für diverse größere Web Anwendungen wie Fondplattform und Versicherungsportale
- CRM- und Ausschreibungsportal für großen Münchner Bauträger

Kontaktdaten: E-Mail: gunther@pipperr.de – Mobil:+49- (0)17180656113 – Internet : www.pipperr.de