Grid Dynamics

Scalable eCommerce Platform Solutions

Not a relational databases

by Roman Nikolaenko

Topics

I part:

- Why RDBMS is not a "silver bullet"?

- Do we have any alternatives?
- What does CAP theorem mean?

- Any use cases for noSQL?

- What was our use case?
- HBase
- MongoDB

II part:

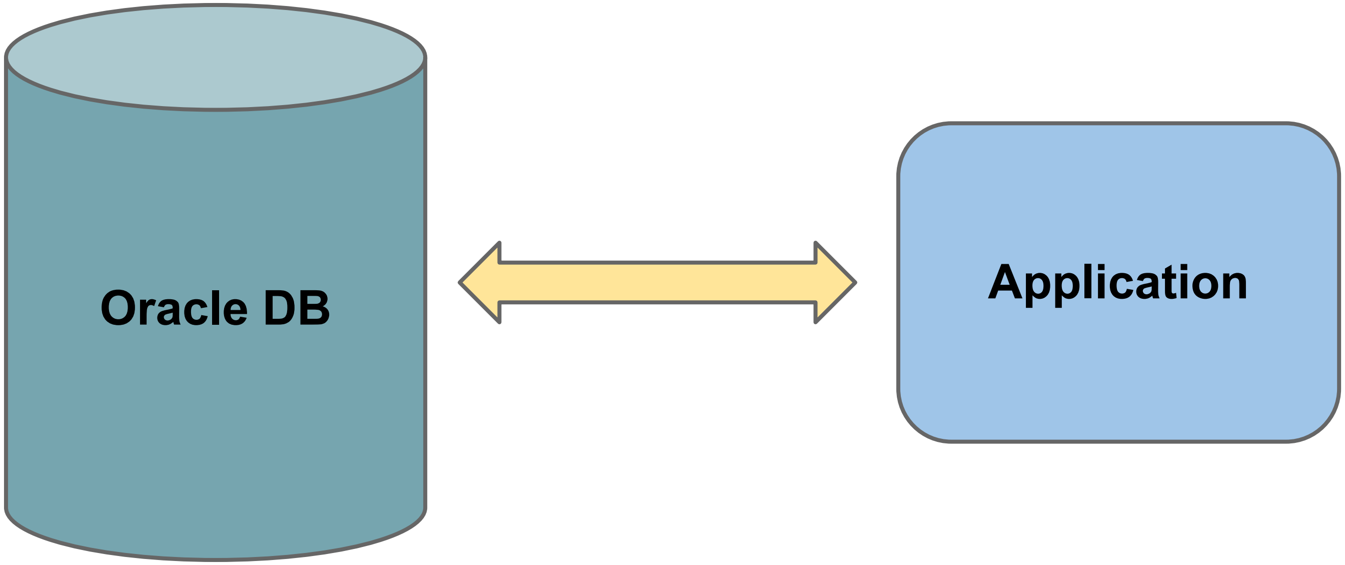
- MongoDB Demo

Why RDBMS is not a "silver bullet"?

ACID*

Transactions

Data consistency



*ACID (atomicity, consistency, isolation, durability)

Why RDBMS is not a "silver bullet"?

Amount of Data **is growing** rapidly.
Companies:

- Facebook
- Twitter
- MySpace
- Google
- Amazon

Store **petabytes** of data, make complex **analysis**

Why RDBMS is not a "silver bullet"?

Amount of Data is **growing** rapidly.
Companies:

- Facebook
- Twitter
- MySpace
- Google
- Amazon

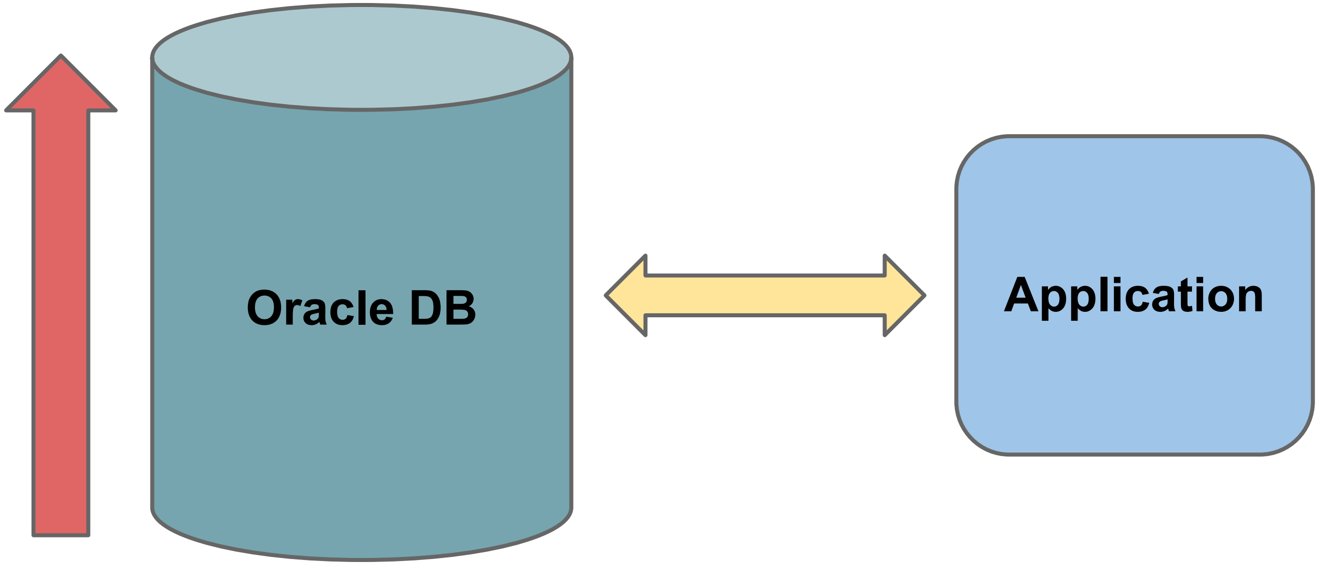
WHAT
COULD
WE DO?

Store **petabytes** of data with complex **analysis**

Why RDBMS is not a "silver bullet"?

ACID*
Transactions
Data consistency

+CPU
+RAM
+HDD
...
PRICE

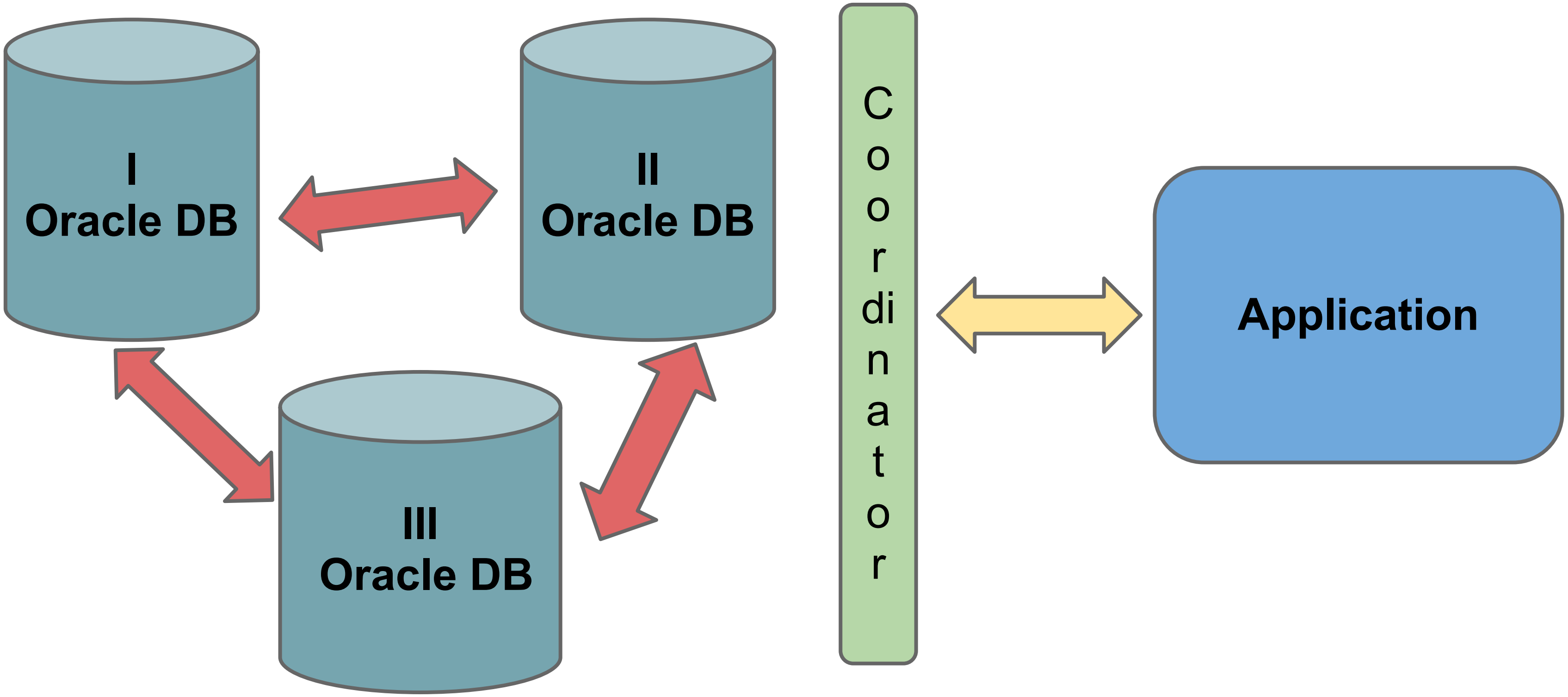


*ACID (atomicity, consistency, isolation, durability)

Why RDBMS is not a "silver bullet"?

ACID ?
Transactions ?
Data consistency ?

Fault Tolerance ?
Data replication ?
Data sharding ?



Why RDBMS is not a "silver bullet"?

Scalability issues

We need additional stuff to maintain cluster

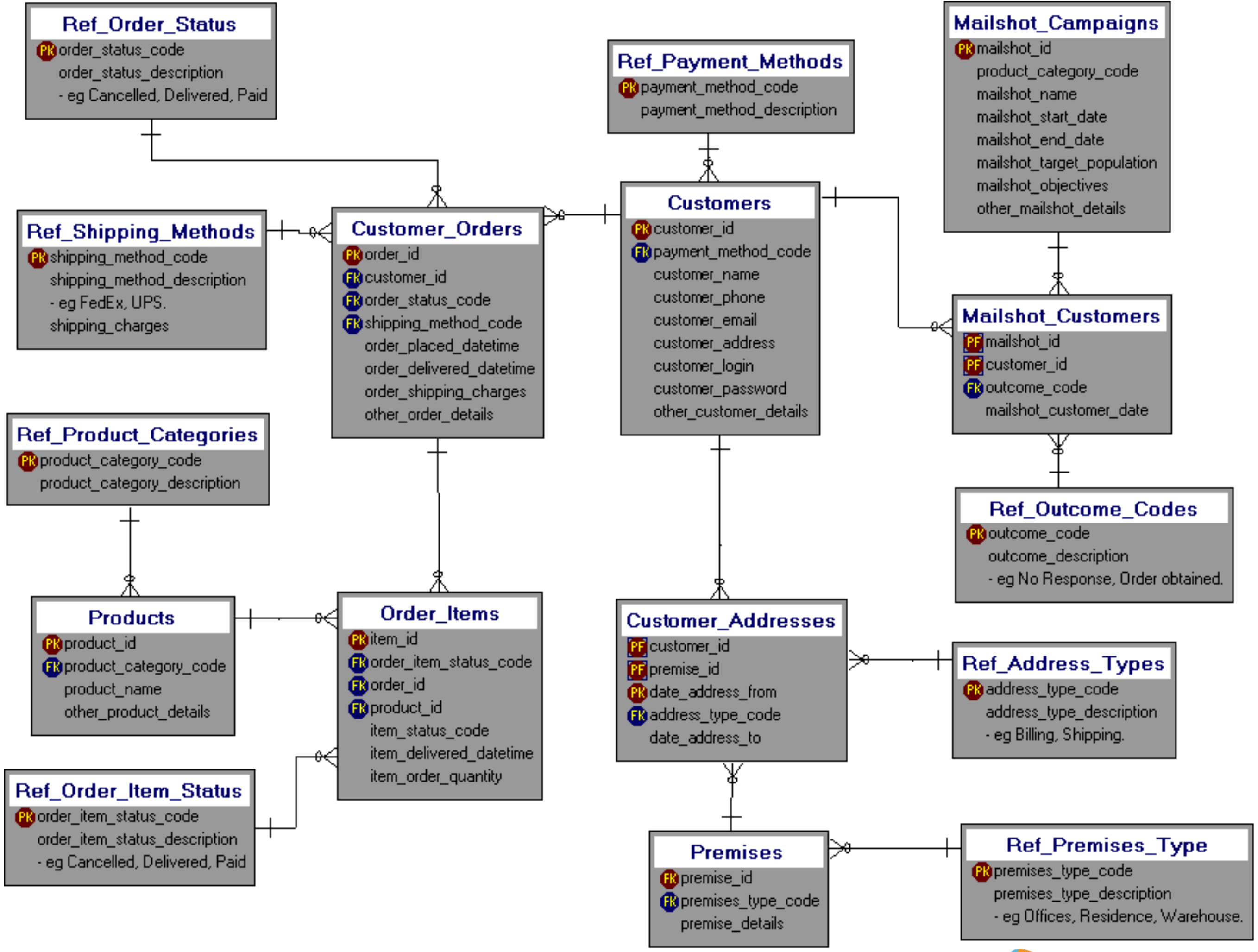
Why RDBMS is not a "silver bullet"?

Scalability issues

Complexity

Why RDBMS is not a "silver bullet"?

Complexity



Ref_Order_Status
PK order_status_code
order_status_description
- eg Cancelled, Delivered, Paid

Ref_Payment_Methods
PK payment_method_code
payment_method_description

Mailshot_Campaigns
PK mailshot_id
product_category_code
mailshot_name
mailshot_start_date
mailshot_end_date
mailshot_target_population
mailshot_objectives
other_mailshot_details

Ref_Shipping_Methods
PK shipping_method_code
shipping_method_description
- eg FedEx, UPS.
shipping_charges

Customer_Orders
PK order_id
FK customer_id
FK order_status_code
FK shipping_method_code
order_placed_datetime
order_delivered_datetime
order_shipping_charges
other_order_details

Customers
PK customer_id
FK payment_method_code
customer_name
customer_phone
customer_email
customer_address
customer_login
customer_password
other_customer_details

Mailshot_Customers
PF mailshot_id
PF customer_id
FK outcome_code
mailshot_customer_date

Ref_Product_Categories
PK product_category_code
product_category_description

Ref_Outcome_Codes
PK outcome_code
outcome_description
- eg No Response, Order obtained.

Products
PK product_id
FK product_category_code
product_name
other_product_details

Order_Items
PK item_id
FK order_item_status_code
FK order_id
FK product_id
item_status_code
item_delivered_datetime
item_order_quantity

Customer_Addresses
PF customer_id
PF premise_id
PK date_address_from
FK address_type_code
date_address_to

Ref_Address_Types
PK address_type_code
address_type_description
- eg Billing, Shipping.

Ref_Order_Item_Status
PK order_item_status_code
order_item_status_description
- eg Cancelled, Delivered, Paid

Premises
FK premise_id
FK premises_type_code
premise_details

Ref_Premises_Type
PK premises_type_code
premises_type_description
- eg Offices, Residence, Warehouse.

Why RDBMS is not a "silver bullet"?

Scalability issues

We need additional stuff to maintain cluster

Complexity

When data doesn't feet table structure schema becomes too complex

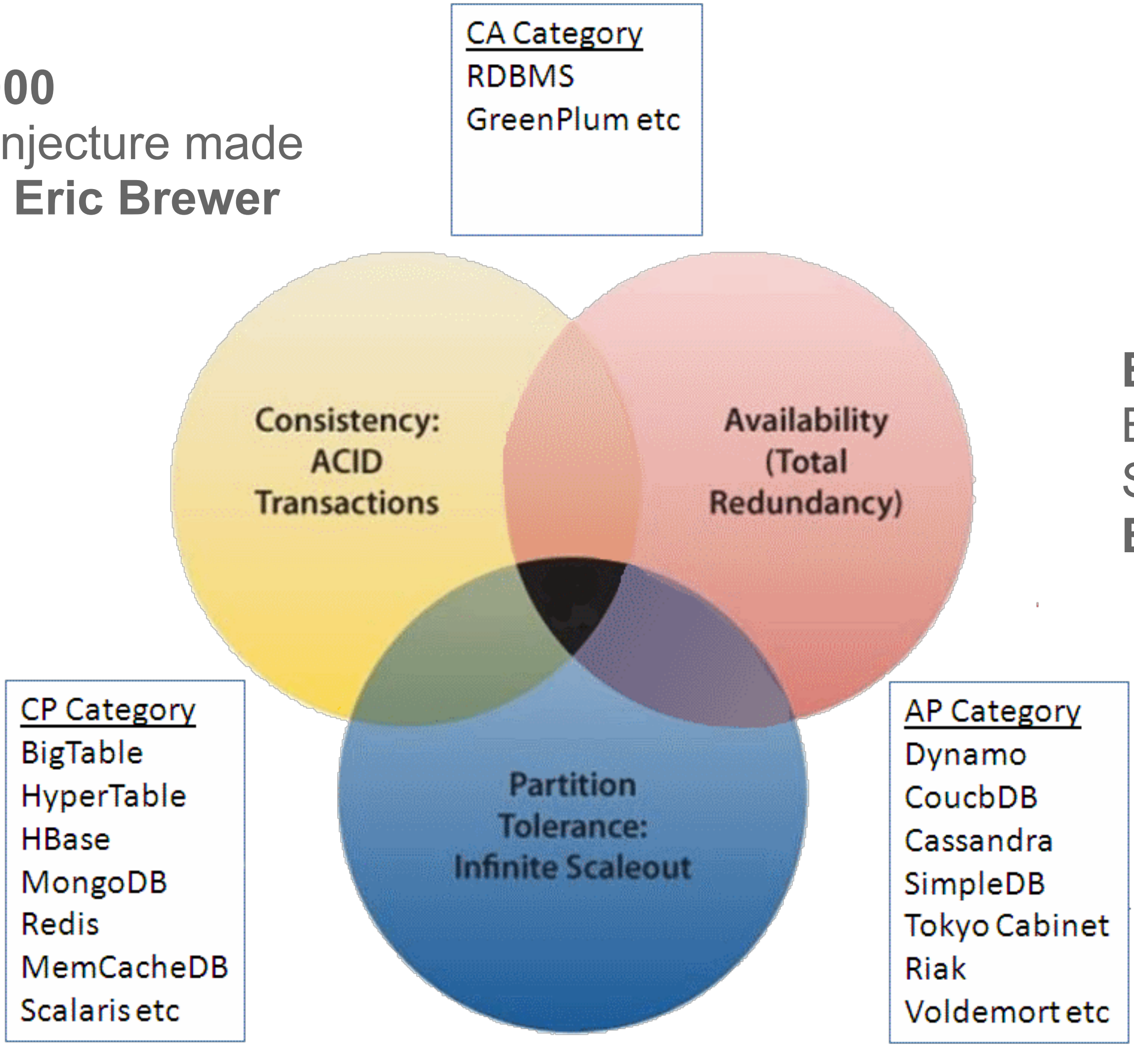
Do we have any alternatives?

Do we have any alternatives?

The answer is "YES! noSQL!"

What does CAP Theorem mean?

2000
conjecture made
by **Eric Brewer**


Consistency: ACID Transactions
Availability (Total Redundancy)
Partition Tolerance: Infinite Scaleout

CA Category
RDBMS
GreenPlum etc

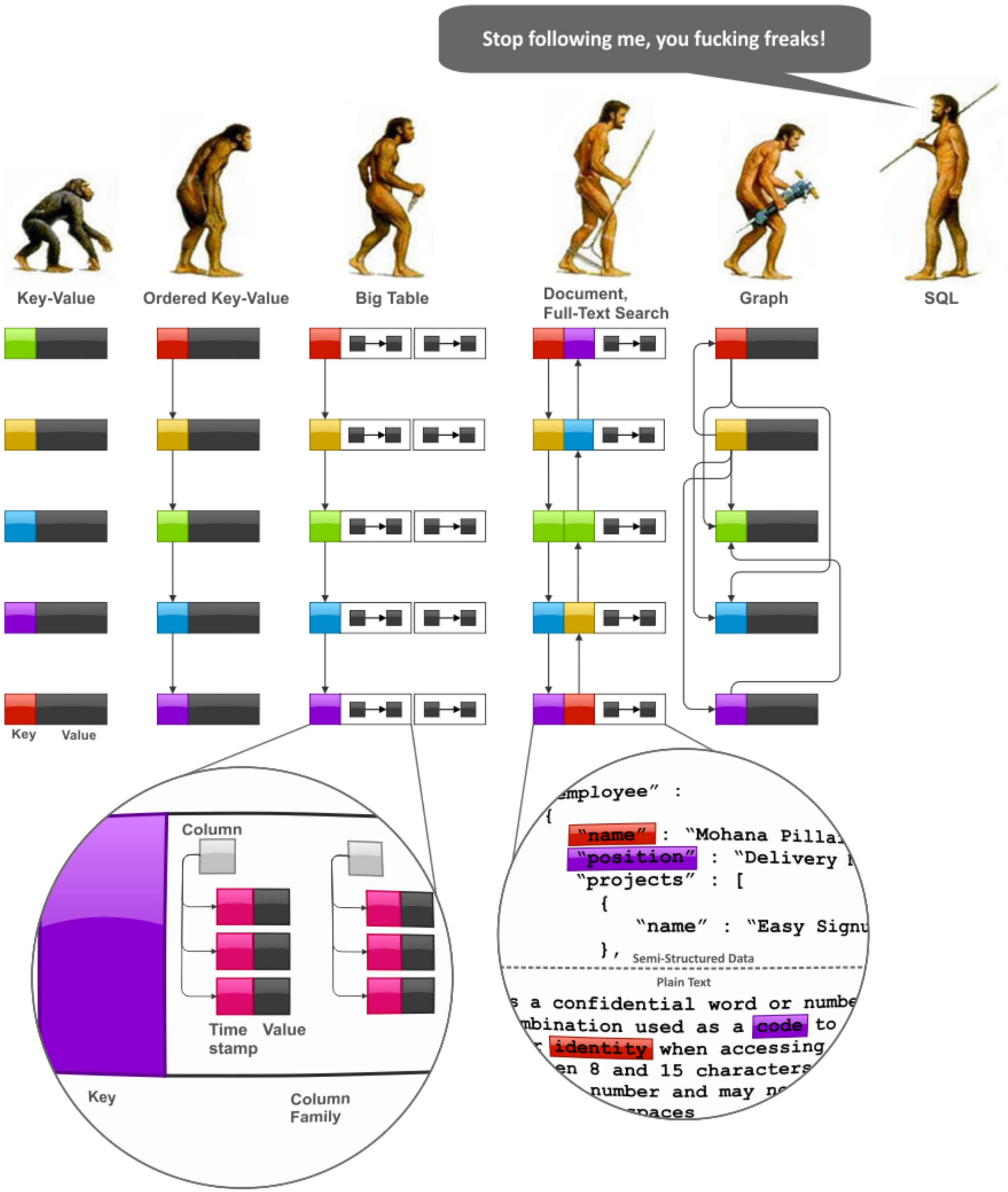
CP Category
BigTable
HyperTable
HBase
MongoDB
Redis
MemCacheDB
Scalaris etc

AP Category
Dynamo
CoucbDB
Cassandra
SimpleDB
Tokyo Cabinet
Riak
Voldemort etc

Later was proved by
Gilbert and **Lynch**

BASE, an acronym for Basically Available Soft-State services with **Eventual Consistency**

Grid Dynamics



Storages Classification:

- **Key-Value Stores**: Oracle Coherence, Redis, Kyoto Cabinet etc
- **BigTable-style Databases**: Apache HBase, Apache Cassandra etc
- **Document Databases**: MongoDB, CouchDB etc
- **Full Text Search Engines**: Apache Lucene, Apache Solr etc
- **Graph Databases**: neo4j, FlockDB etc
- **RDBMS**

Any use cases for noSQL?

- Bigness
- Massive write performance
- Fast key-value access
- Flexible schema and flexible datatypes
- Schema migration
- Write availability
- Easier maintainability, administration and operations
- No single point of failure
- Generally available parallel computing
- Programmer ease of use
- Use the right data model for the right problem
- Avoid hitting the wall
- Distributed systems support
- Tunable CAP tradeoffs

What was our use case?

In our case:

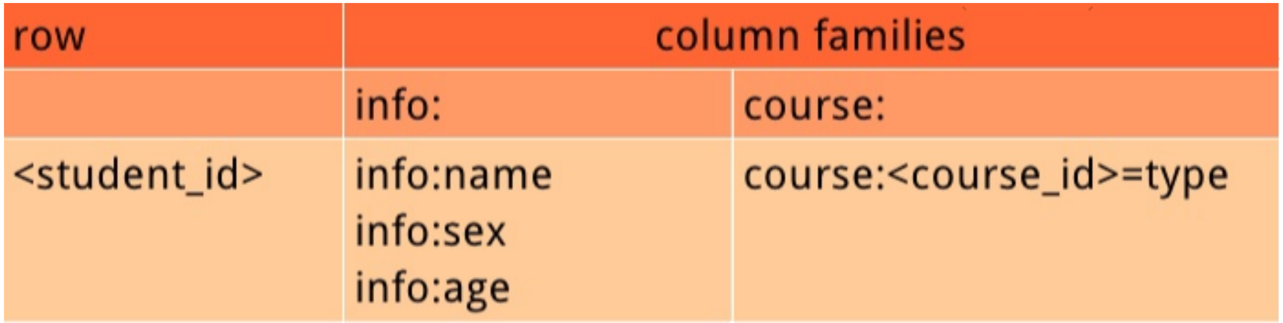
- all our data was stored in HDFS
- expected amount of data was several Tb
- application was going to build reports via Hadoop MapReduce jobs
- reports should stored in prepared state
- sorting for every column must be available
- paging also should be in place

HBase

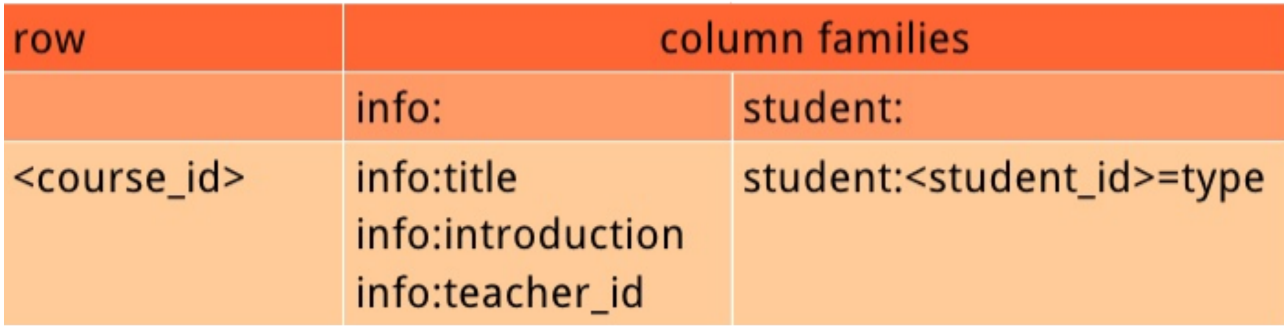
- **Written in:** Java
- **Main point:** Billions of rows X millions of columns
- **License:** Apache
- **Protocol:** HTTP/REST (also Thrift)
- Modeled after Google's BigTable
- Uses Hadoop's HDFS as storage
- Map/reduce with Hadoop
- Query predicate push down via server side scan and get filters
- Optimizations for real time queries
- A high performance **Thrift** gateway
- HTTP supports XML, Protobuf, and binary
- Cascading, hive, and pig source and sink modules
- Jruby-based (JIRB) shell
- Rolling restart for configuration changes and minor upgrades
- Random access performance is like MySQL

Best used: When you use the Hadoop/HDFS stack. When you need random, realtime read/write access to BigTable-like data.
For example: For data that's similar to a search engine's data

HBase



row	column families	
	info:	course:
<student_id>	info:name info:sex info:age	course:<course_id>=type



row	column families	
	info:	student:
<course_id>	info:title info:introduction info:teacher_id	student:<student_id>=type

HBase

CREATE new Table:

```
Configuration configuration = getHBaseConfiguration();
HBaseAdmin hBaseAdmin = new HBaseAdmin(configuration);

HTableDescriptor tableDescriptor = new HTableDescriptor
("students");

HColumnDescriptor info= new HColumnDescriptor("info");
info.setCompressionType(Compression.Algorithm.GZ);

tableDescriptor.addFamily(info);
....

hBaseAdmin.createTable(tableDescriptor);
hBaseAdmin.close();
```

HBase

```
HTable table = new HTable(getHBaseConfiguration(), "students");

long studentId = 1;
byte[] rowKey = Bytes.toBytes(studentId);
```

//Insert data:

```
Put value = new Put(rowKey);
value.add(Bytes.toBytes("info"),Bytes.toBytes("name")
                     Bytes.toBytes("John"));
....
table.put(put);
```

//Get one row by key:

```
Get get = new Get(rowKey);
table.get(get);
```

HBase

```
HTable table = new HTable(getHBaseConfiguration(), "students");

long student1Id = 1;
long student2Id = 100;
byte[] startKey= Bytes.toBytes(student1Id);
byte[] endKey= Bytes.toBytes(student2Id);

//Get data range:
Scan scan = new Scan();
scan.setStartRow(startKey);
scan.setStopRow(endKey);

ResultScanner scanner = table.getScanner(scan);
for (Result result = scanner.next();
         result != null;
         result = scanner.next())
    {
         DomainObject resultObject = processResult(result);
    }
```

Grid Dynamics

HBase

Problems in our case:

- HBase doesn't have support for indexes
- Index emulation causes data duplication
- Sorting and Paging was not efficient enough

MongoDB

- Open Source.
- High performance.
- Schema free.
- Document oriented.

Past: part of a cloud web development platform.
Now: separate database project.

- Written in C++.
- Lots of language drivers available. (*BSON*)

MongoDB

BSON:

Binary JSON -> Light weight+Efficient.
Binary **JSON** -> Language independent.

Easy manipulation.
Additional data types.
Fast scan-ability.

MongoDB

DOCUMENT EXAMPLE:

```
{
    "_id": ObjectId("4eaf9884bb6c6747b4ba5dfb"),
    "etailer_id":7,                   (this is 1 byte!)
    "cacheServerHost": "...",
     "localUrls": {
     "0": "...1.html.gz"
        },
     "pageInfo": {
        "STORE_PRICE": "356.66",
        "MFGR_NAME": "CANON",
         "AVL_DESC": "In Stock.",
         "PROD_ID": "SX30IS",
         "SKU": "B0041RSPR8",
         "PROD_NAME": "Canon SX30IS 14.1MP Digital Camera",
         "SHIPPING_COST": "0.00",
         "MFGR_NAME_ORIGINAL": "Canon"
     }
}
```

MongoDB

DEMO Time

Links

NoSQL storages (currently 122+):

http://nosql-databases.org
http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis

NoSQL modeling techniques:

http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/

CAP theorem, data integrity:

http://www.julianbrowne.com/article/viewer/brewers-cap-theorem
http://www.scribd.com/doc/50353861/NoSQL-Availability-amp-Integrity-2