RDBMs at scale & NoSQL

Andrei Arion, LesFurets.com, tp-bigdata@lesfurets.com

Plan

1. **Course info**
2. Scaling Relational Databases
3. NoSQL databases
4. TP: PostgreSQL

Andrei ARION

- XML databases research, INRIA & UPS
- software engineer/consultant
- data engineering team, ***LesFurets.com***

tp-bigdata@lesfurets.com, *andrei.arion@gmail.com*

Ressources (Slides/TPs/VMs) : bit.ly/bigdata-telecom ⇒ andreiarion.github.io

LesFurets.com

- Independent insurance aggregator
- *OLTP (Runtime DB)*:
  - replicated MariaDB ⇒ DRBD ⇒ Galera Cluster
  - ⇒ Cassandra / Spark On Prem ⇒ **GCP (Firestore, PubSub, CloudSQL, GCS, Dataflow)**
- *OLAP (Analytics/BI DB)*:
  - MariaDB (snowflake Schema) ⇒ QlikView dashboard
  - ⇒ Spark/Zeppelin over MySQL/Cassandra: ad-hoc analyses & ETLs ⇒ **GCP (BigQuery)**
- *Infra*: Hybrid On-PREM/AWS ⇒ **GCS**

Planning

```
10/11 : Intro NoSQL +  PostgreSQL
```

```
17/11 : Cassandra Intro + Replication
24/11 : Cassandra modeling (timeseries)
```

```
27/11 : Apache Spark 1: Intro + RDD operations
02/12 : Apache Spark 2: exploratory data analysis using Dataframes
```

```
08/12 : AWS ?
```

```
10/12 : BD Graph, Neo4J
```

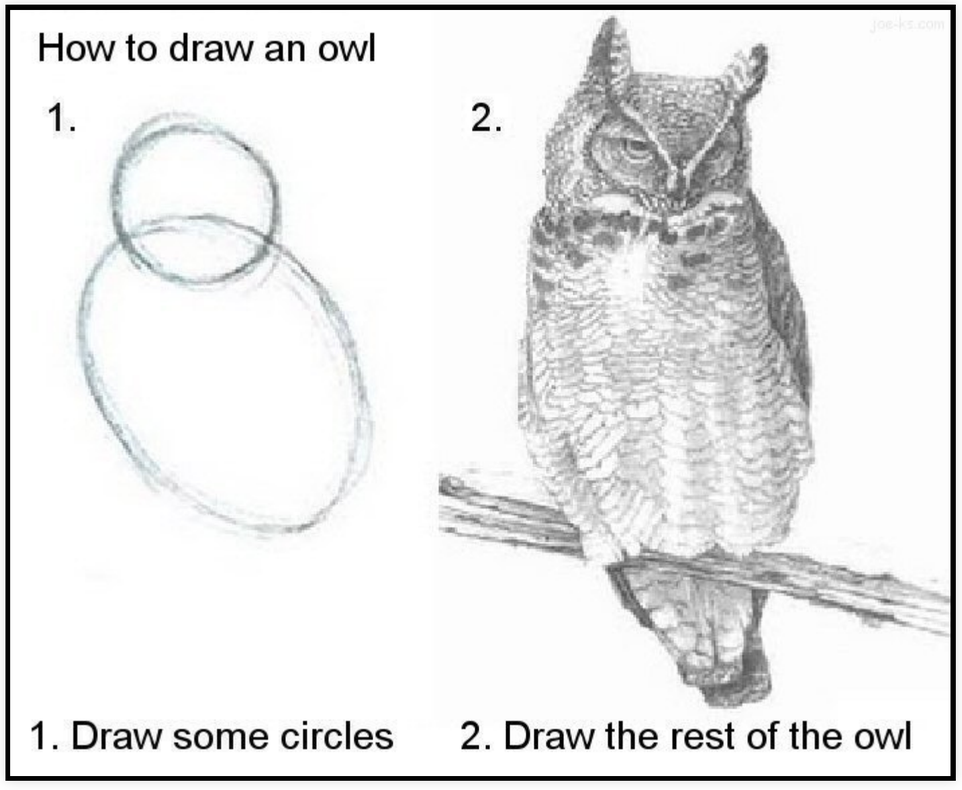
```
15/12 : MongoDB Intro + Data Modelling
07/01 : MongoDB Applications
```

```
14/01 : Projet 1
21/01 : Projet 2
28/01 : Projet 3
11/02 : Soutenances
```

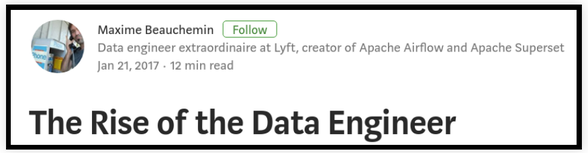
How

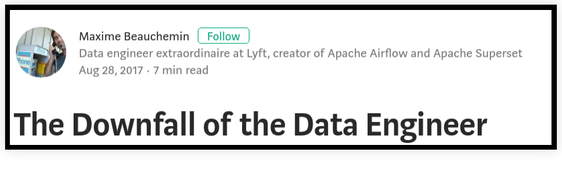
- Course: 1h30
- TP: 1h30 (pair-programming)
- Small surveys / home readings
- Projet
- Ressources ⇒ bit.ly/bigdata-telecom ⇒ andreiarion.github.io

How it may seem



Home readings

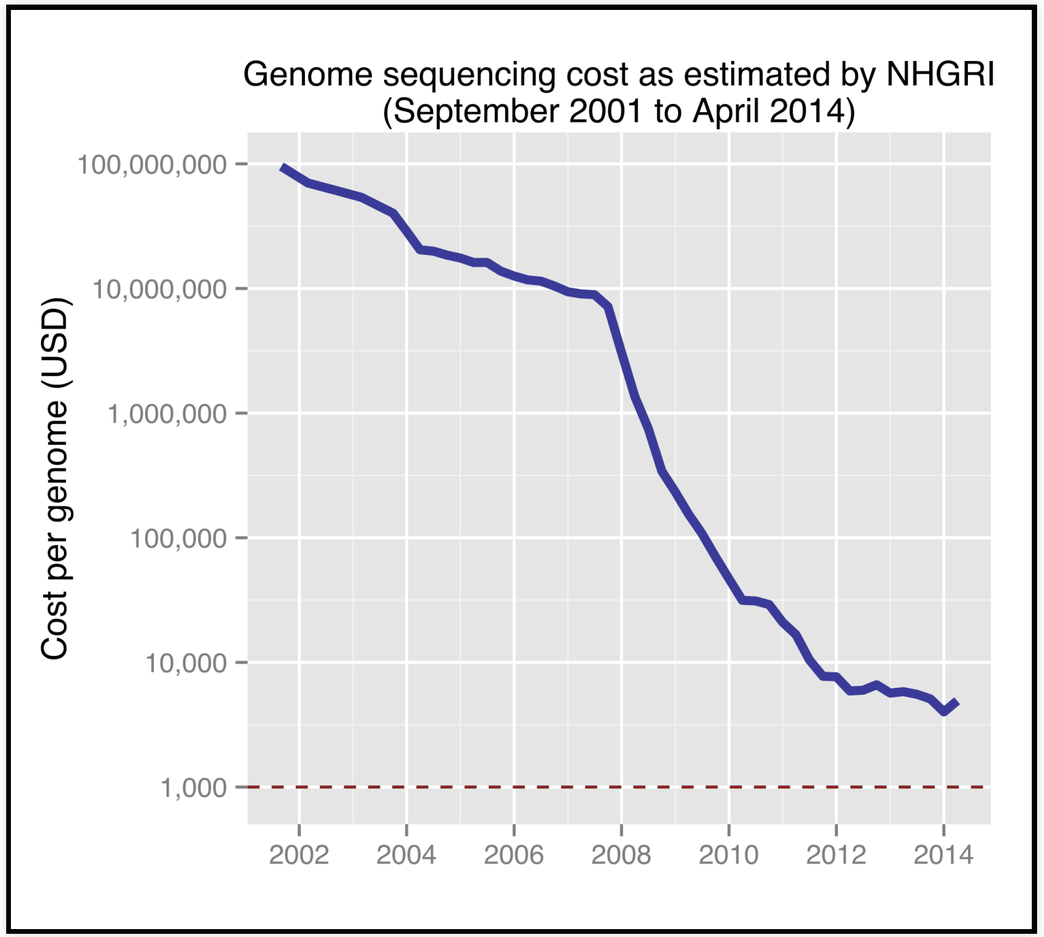




Plan

1. Course info
2. **Scaling Relational Databases**
    1. Scaling a simple application
    2. Scaling MySQL @LesFurets.com
3. NoSQL databases
4. TP PostgreSQL

Data: the new hope

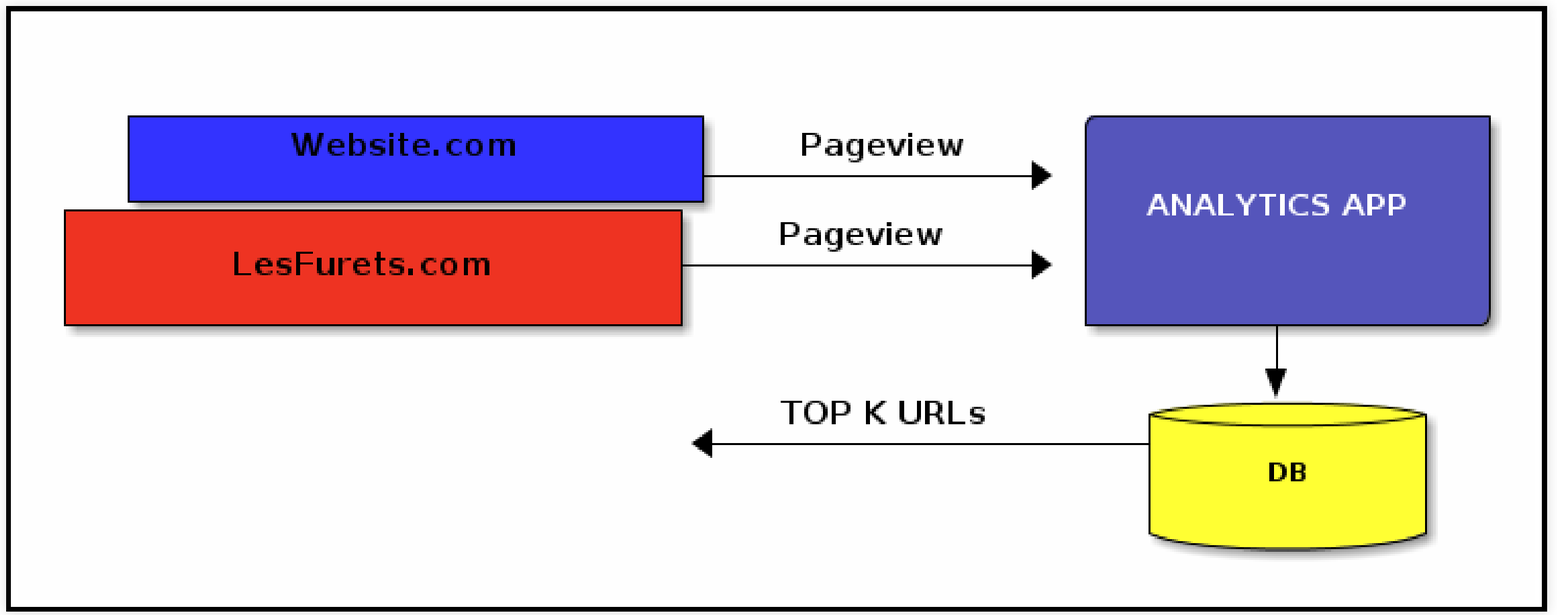


Processing the data has become the bottleneck!

Data: the new oil

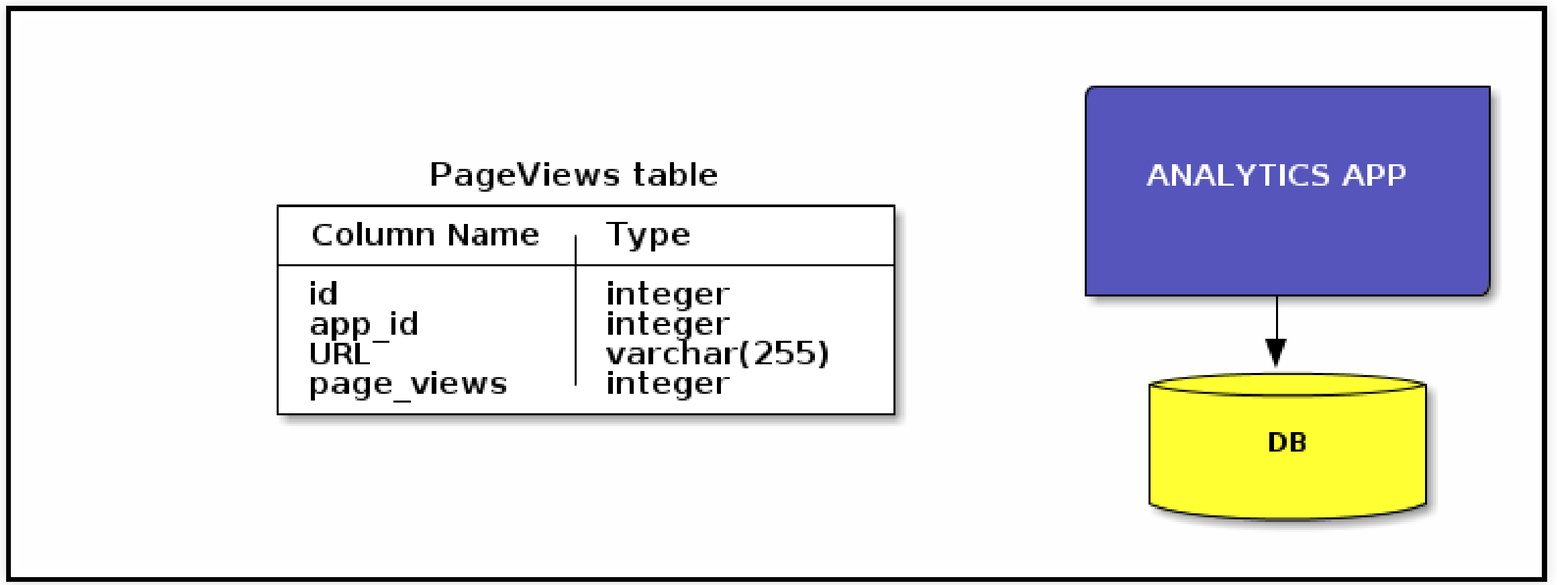
Data: most common use

- Web analytics applications
  - track the number of pageviews for each URL
  - what are the top 100 URLs



Simplest architecture

- track the number of pageviews for each URL
- what are the top 100 URLs



Queries

- insert a pageview
- update pageviews
- top 100 URLs for a client

Insert pageviews

PageViews table

Column Name	Type
id	integer
app_id	integer
URL	varchar(255)
page_views	integer

```
INSERT INTO PageViews(app_id, URL, page_views) VALUES
    (1,"http://www.lesfurets.com/index.html",1);
INSERT INTO PageViews(app_id, URL, page_views) VALUES
    (2,"http://Website.com/base.html",1);
INSERT INTO PageViews(app_id, URL, page_views) VALUES
    (1,"http://www.lesfurets.com/assurance-auto",1);
```

Update pageviews

PageViews table

Column Name	Type
id	integer
app_id	integer
URL	varchar(255)
page_views	integer

```
UPDATE PageViews SET page_views = page_views + 1
WHERE app_id="1" AND URL="http://www.lesfurets.com/index.html";
```

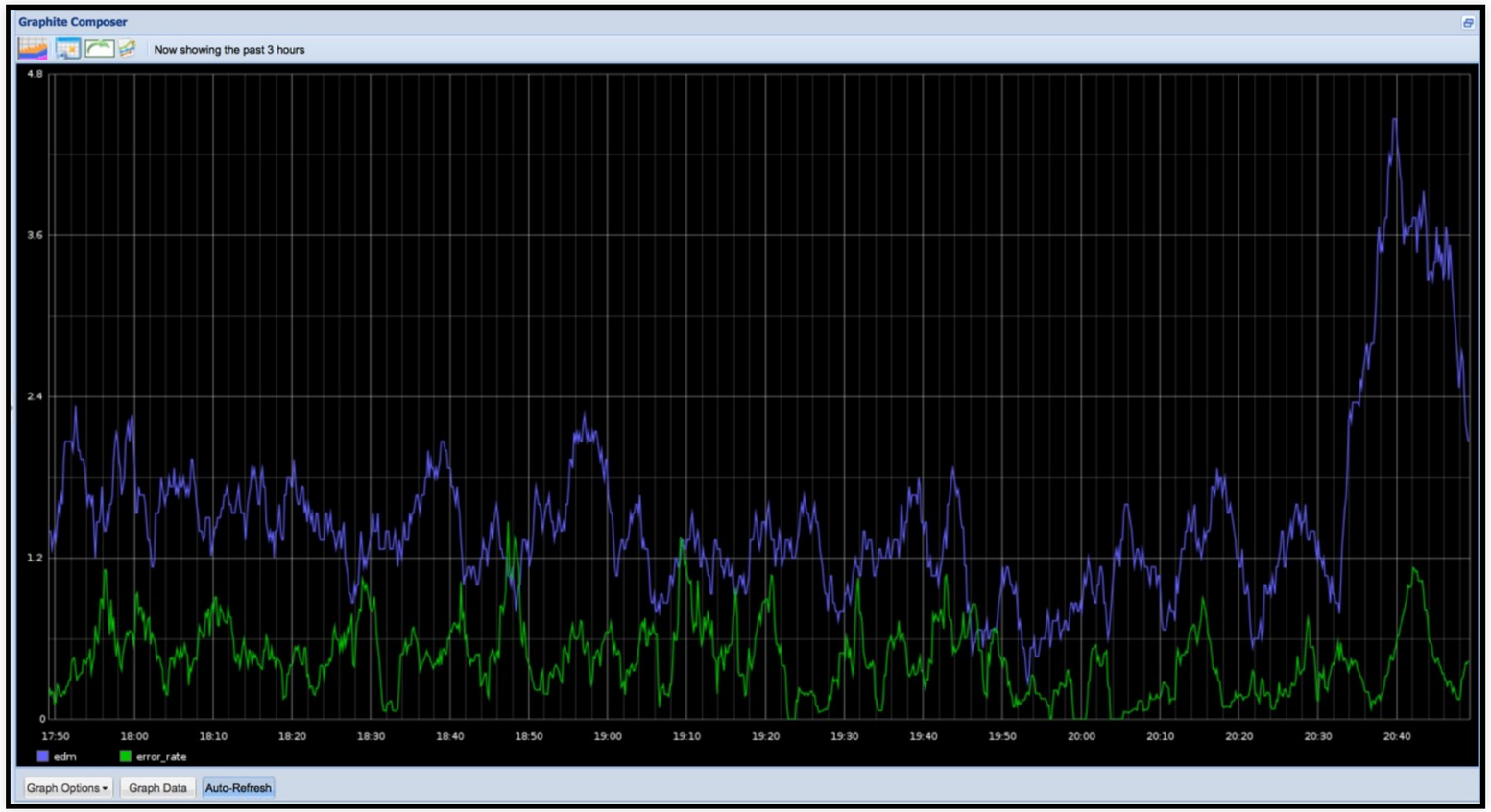
Top 100 URLs for a client

PageViews table

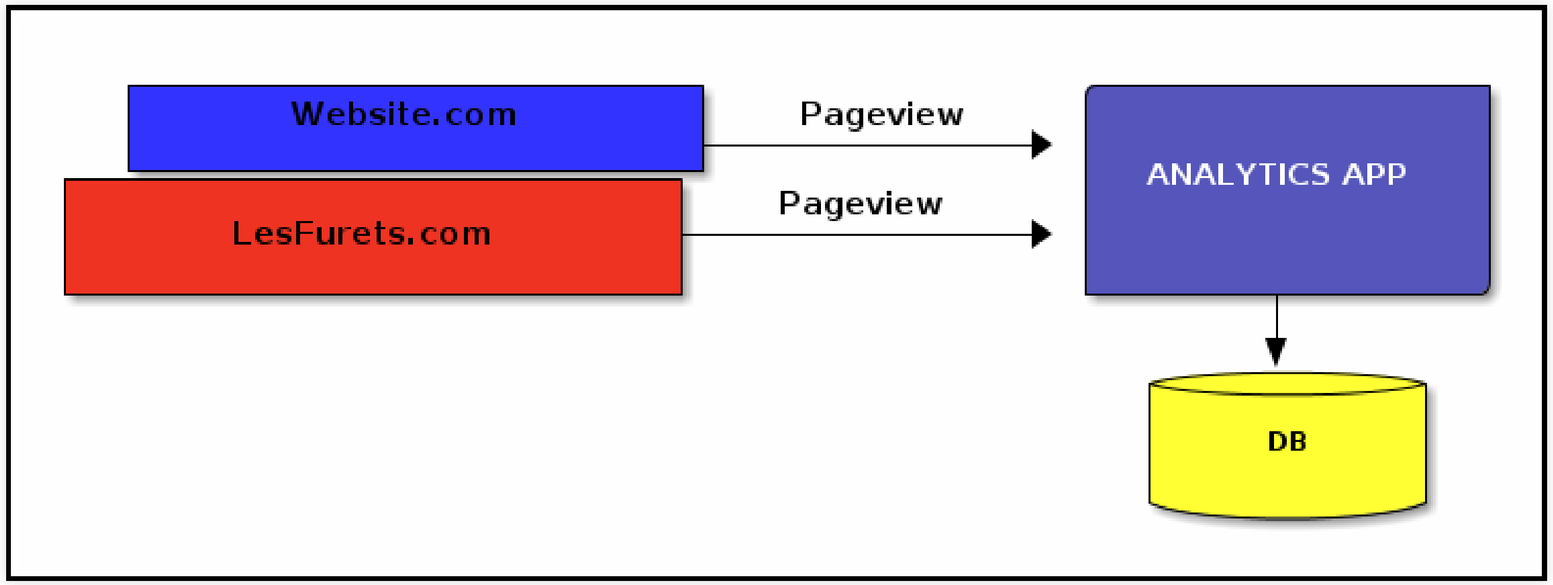
Column Name	Type
id	integer
app_id	integer
URL	varchar(255)
page_views	integer

```
SELECT URL,page_views FROM PageViews
WHERE app_id = '1'
ORDER BY page_views DESC LIMIT 100
```

Production load

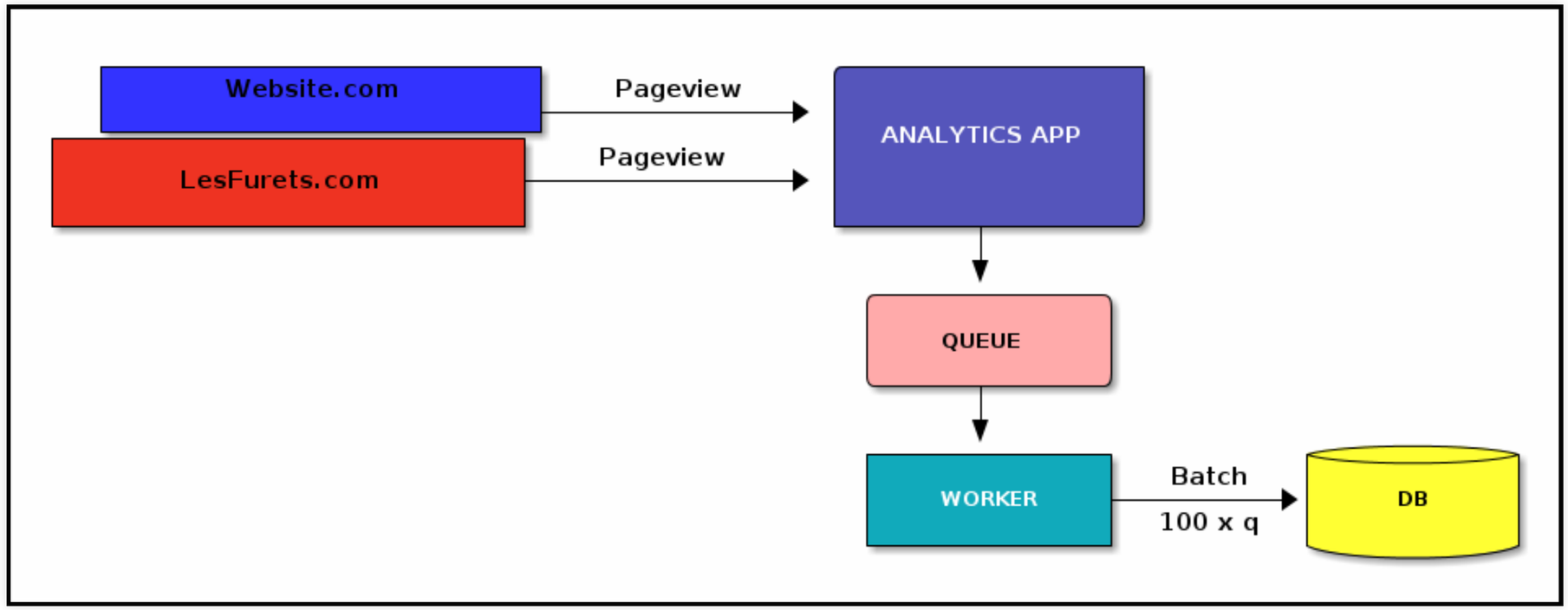


Timeouts



Timeout error on updating the database

Fix#1: queuing + batching



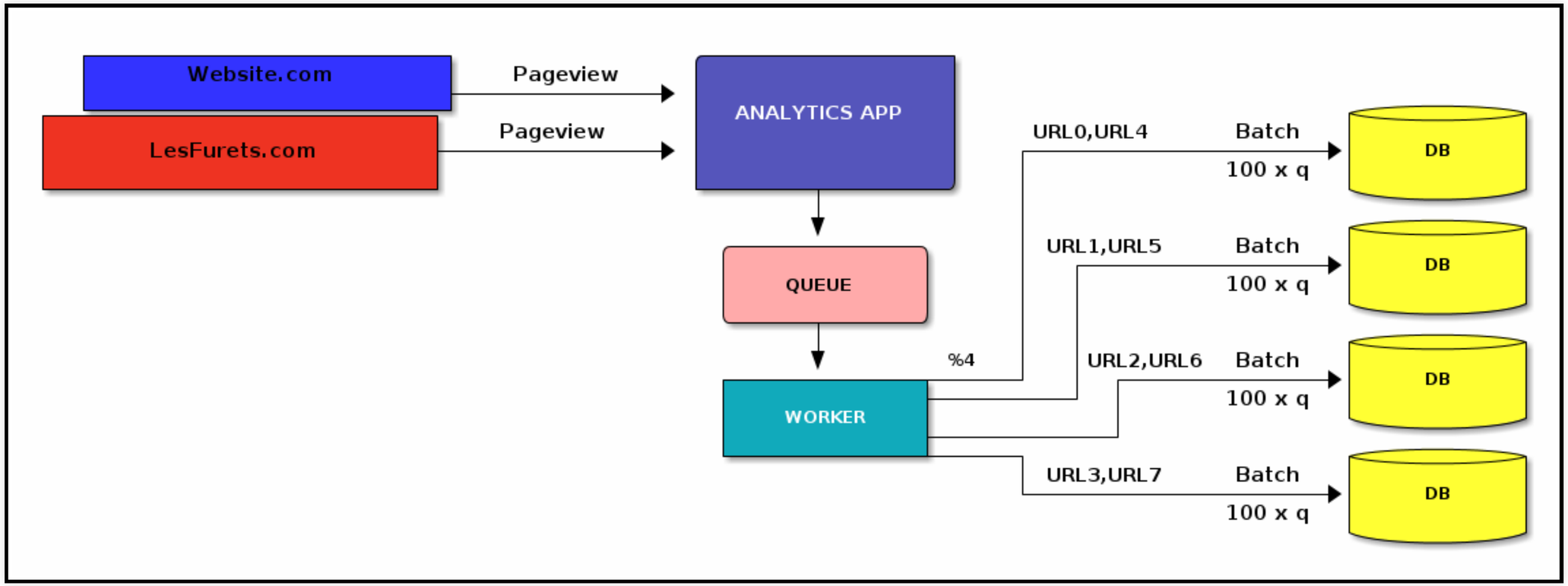
Fix#1: implications

- Modify the application ⇒ batch 100 queries
- Latency / queue size
- handle DB/queue failures ⇒ persistent queuing with event logging
- **cannot accommodate high load**

Fix#2: Sharding (vertical table partitioning)

- Spread the load
  - use multiple database servers
  - spread the PageView table across the servers
  - mapping keys to shards using a hash function

Fix#2 Sharding



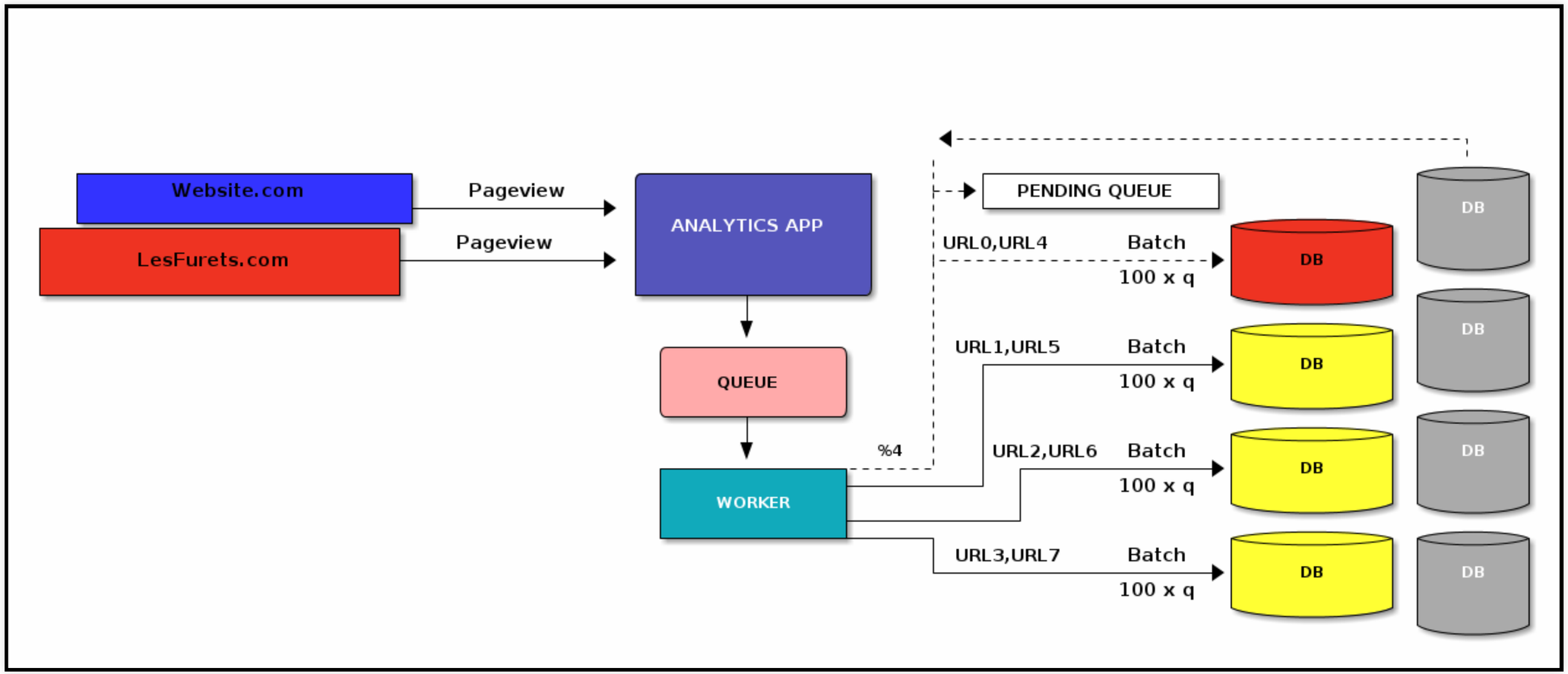
Fix#2: Sharding implications

- distribute the keys to the new servers
- write to the "right" DB instance
- aggregate data from all the shards !
- **Sharding more and more**
  - new shards to follow the load
  - repeat the last steps

Fix#2 Sharding

- **Server failures** are more likely
  - **WRITES**: use a pending queue flushed less frequently
  - **READS**: a portion of the data is unavailable
    - ⇒ **replication**

Fix#3 Replication



Human failures

Distribution (hash function) = %3

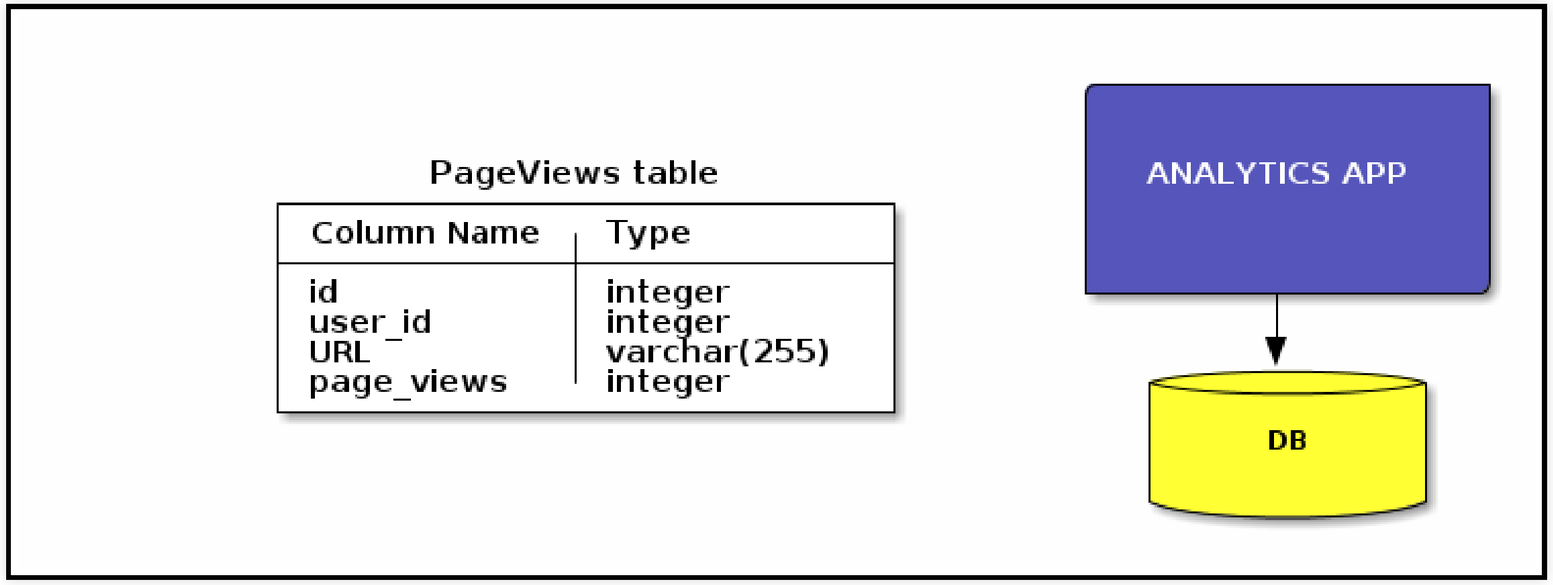
- **Data written to the wrong shards**
  - redistribute data to the right shard
  - while still accepting queries ?!

Human failures

Increments the number of pageviews + 2

```
UPDATE PageView SET page_views = page_views + 2
WHERE user_id='42' AND URL='myurl';
```

Human failures



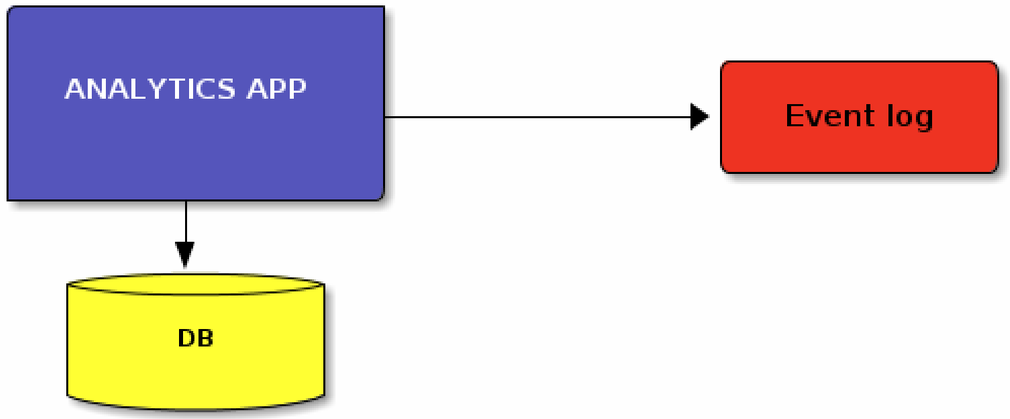

```
UPDATE PageView SET page_views = page_views + 2
WHERE user_id='42' AND URL='myurl';
```

Human failures

- **event logging**

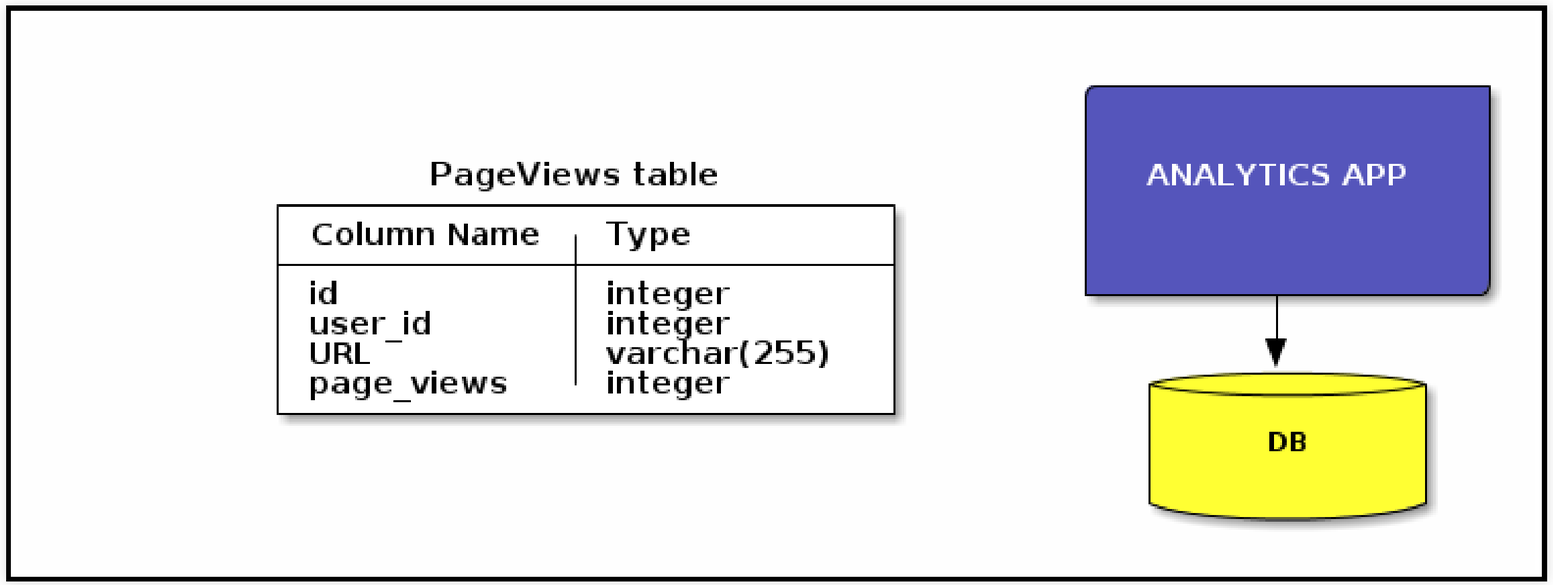
PageViews table

Column Name	Type
id	integer
user_id	integer
URL	varchar(255)
page_views	integer



Human failures

- Incremental data model

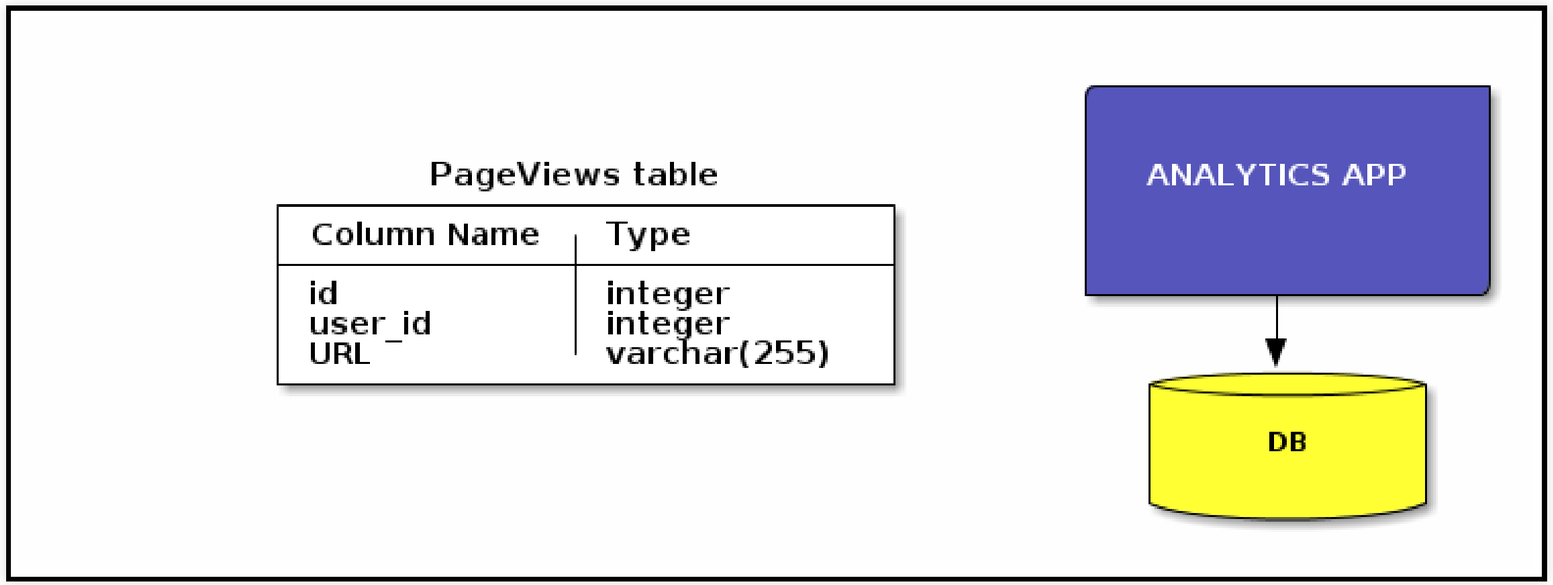


Incremental data model

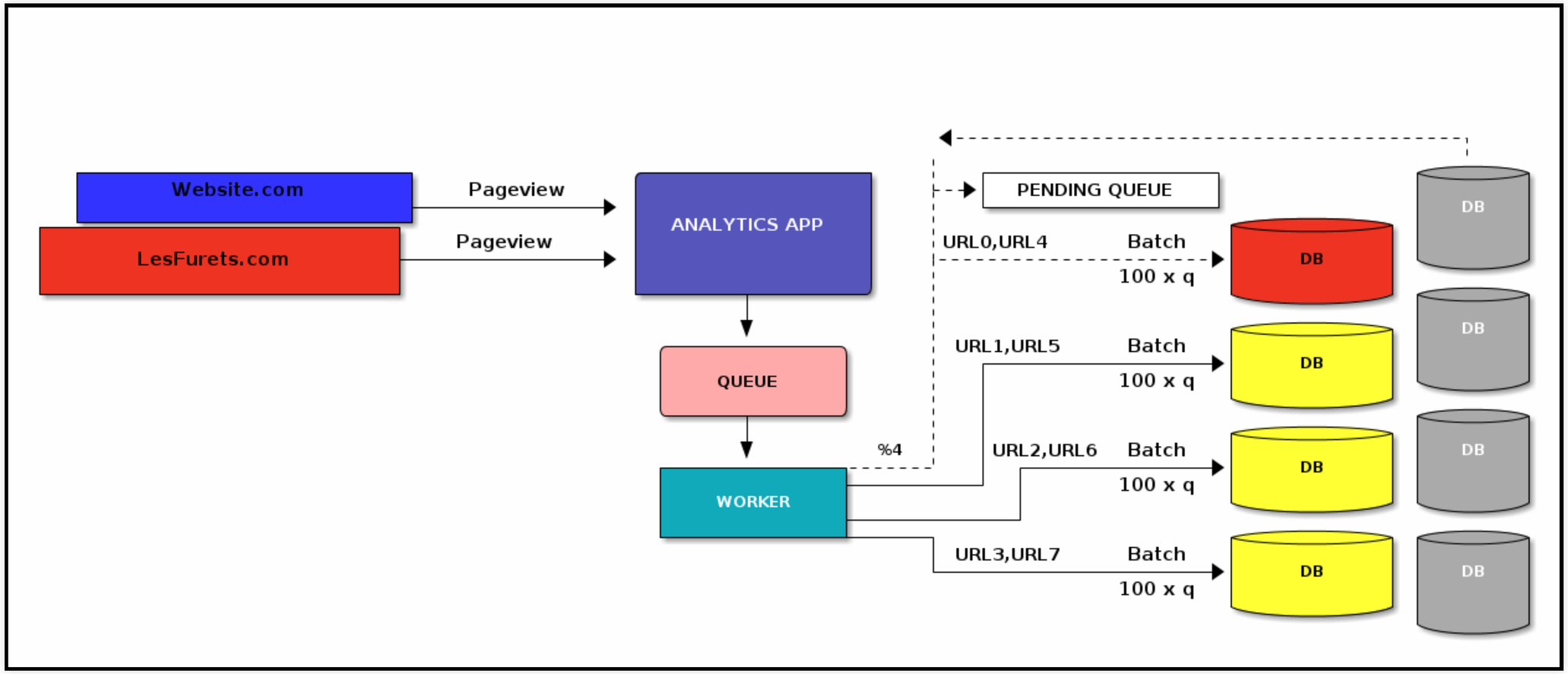
- **data corruption** ⇒ hard to correct (!)
- **contention** ⇒ locking ⇒ bad performance

Human failures

- Incremental data model ⇒ **immutable data model**



What went wrong?



- do I build new features for customers?
- or just dealing with reading/writing the data?

What went wrong?

- **A single server cannot take the load ⇒** solution / **complexity**

1. *distributed storage*
2. querying distributed data
3. built a data model that is not resilient

Wishlist 1 : Storage

- ***Better storage***:
  - easy to add/remove nodes (**scaling**)
  - transparent data distribution (**auto-sharding**)
  - handle failures (**auto-replication**)

⇒ **Distributed databases**: *Redis*, ***Cassandra***, *HBase*, ***MongoDB***, *CouchDB*, **…**

Wishlist 2 : Queries

- *General purpose (distributed) computing*:
  - distributed queries + **parallel processing**

⇒ **Distributed data processing engines** : ***MapReduce***, ***Spark***

Wishlist 3 : Data model

- *We want a resilient data model*:
  - human error is **unavoidable**
  - an **incremental data model** is not resilient

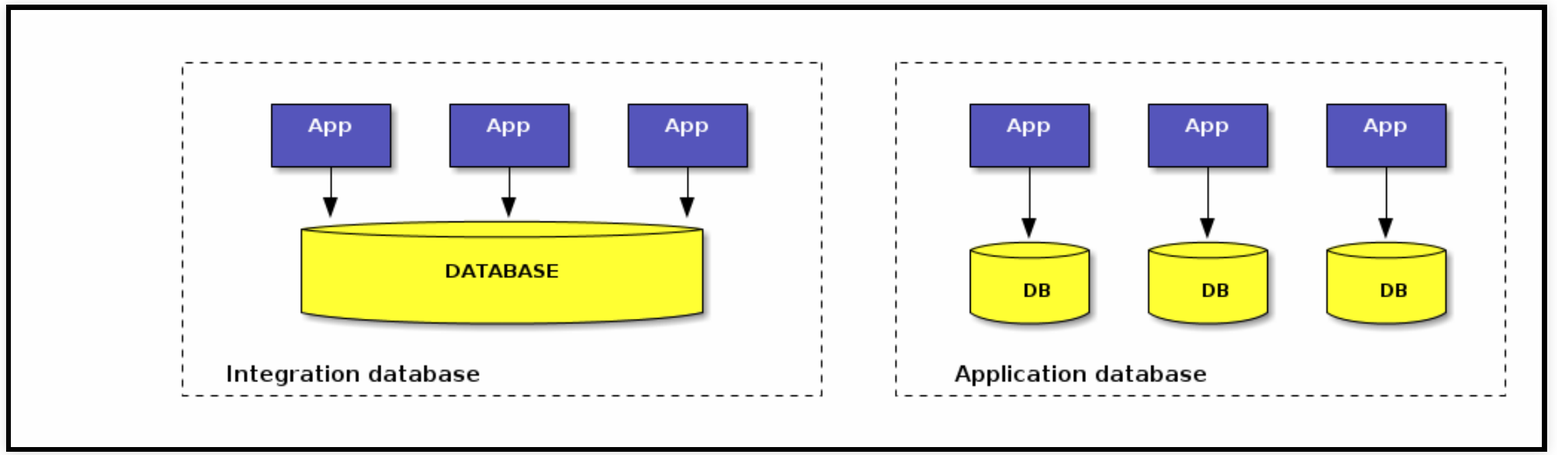
⇒ **Immutability**, *Functional Data Engineering*

Plan

1. Course info
2. From SQL to NoSQL
    1. Scaling a simple application
    2. **Scaling MySQL @LesFurets.com**
3. NoSql databases
4. TP PostgreSQL

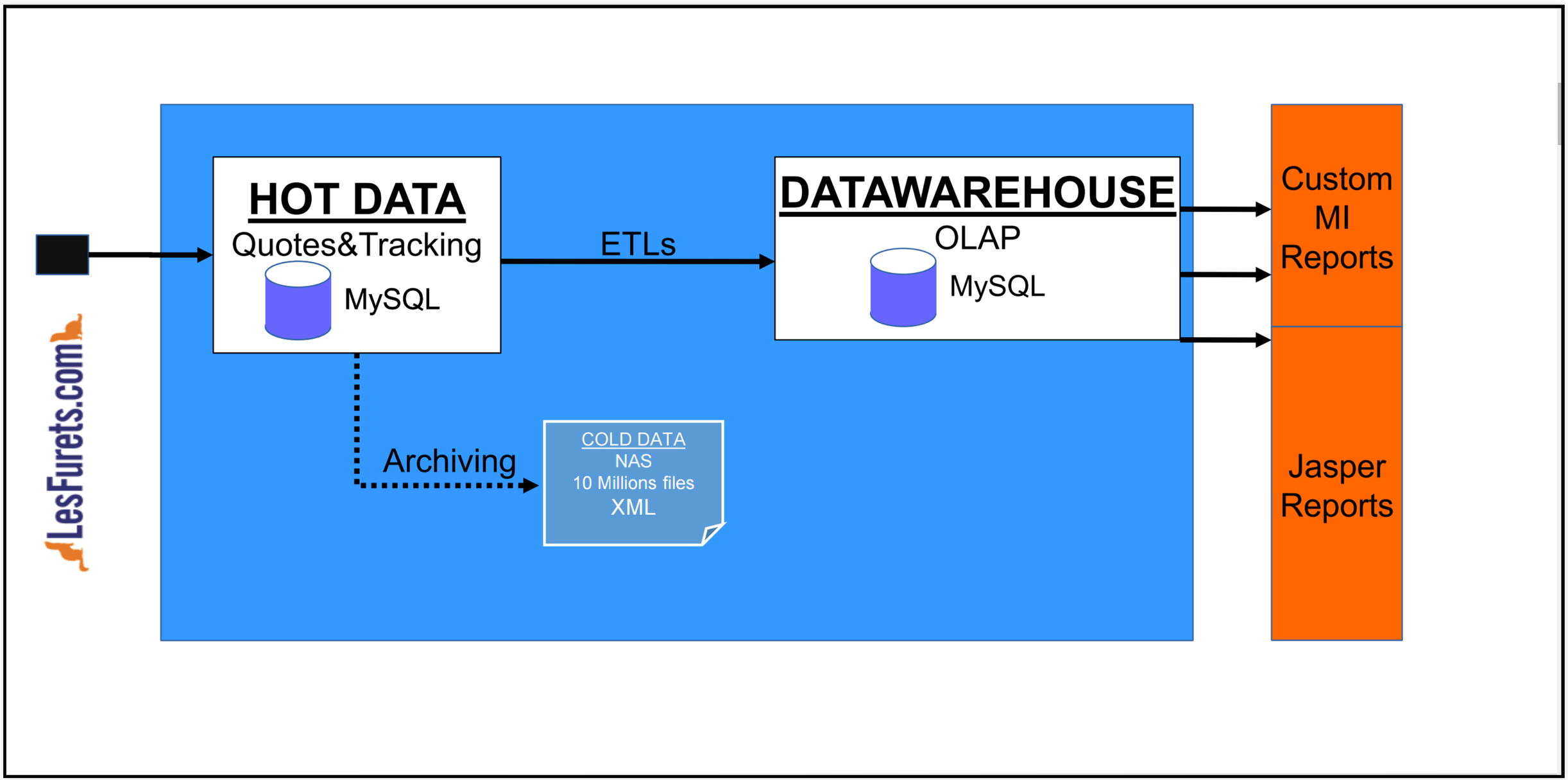
RDBMS: the good parts

- simple model with sound mathematical properties (ACID)

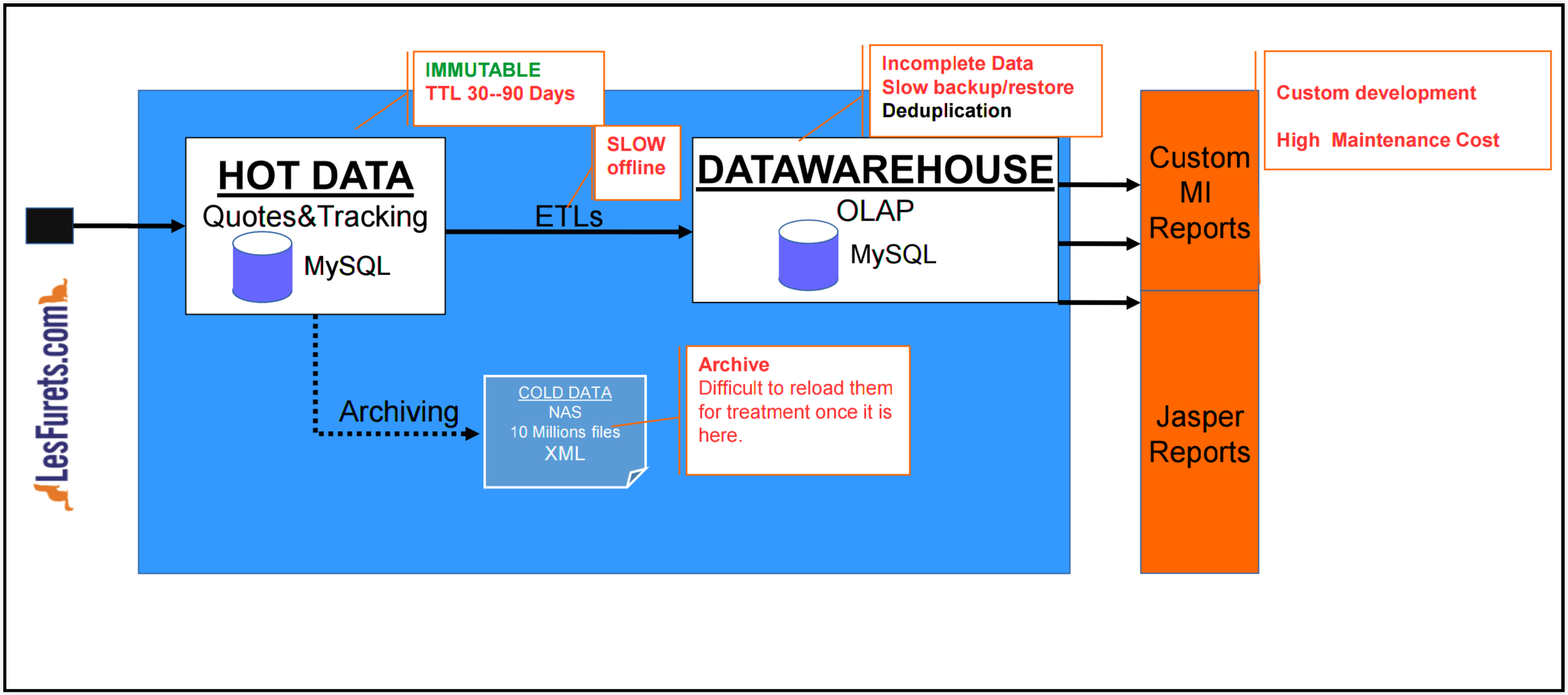


Integration database	Application database
consistent data set	easier to maintain/evolve/**scale**
changes need to be coordinated → side effects	standard interfaces between systems (SOA)

"Classical" RDBMS Architecture



"Classical" RDBMS Architecture concerns



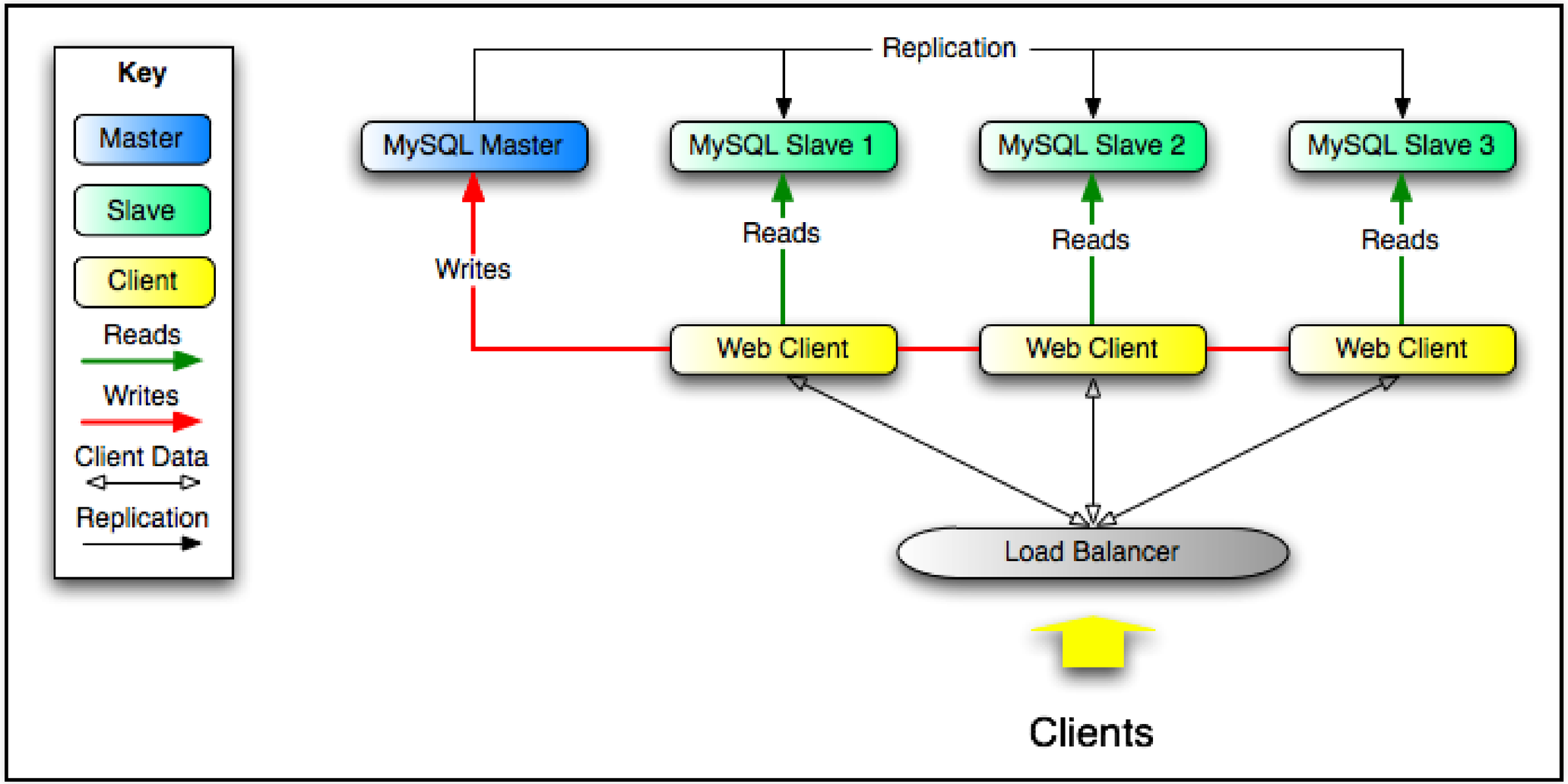
LesFurets Data characteristics

- few updates
- few synchronous queries
- applicative caches

Why replicate?

- scale out
- **data security**
- **segregate workload** (writes on master/ analytics on slaves)
- data distribution
- **backups**
- restore in a point in time (delayed replication)

MySQL scale-out



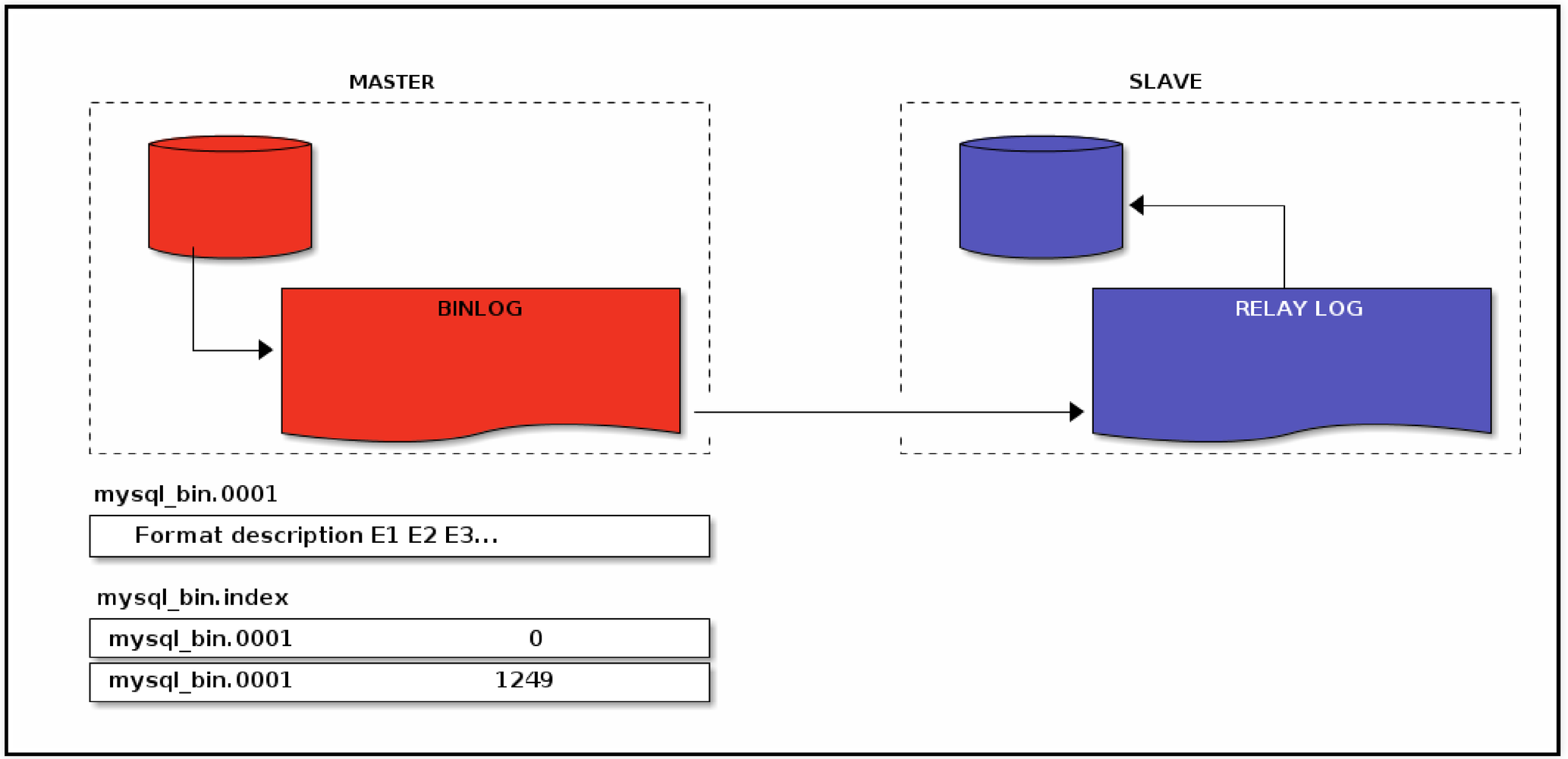
MySQL replication properties

- **1 Master ⇒ n Slaves**
- **Application transparent**
- **Full replication**: the full dataset is replicated on every node
- **LOG-based replication**

MySQL basic replication

- **Master**
  - log changes (events) on a bin-log
  - apply changes
- **Slave(s)**
  - retrieve events from the master and copy to local binlog (*relay log*)
  - replay the events

MySQL replication: binlog



MySQL replication: master configuration

```
/etc/mysql/my.cnf
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_do_db            = database_name

GRANT REPLICATION SLAVE ON *.* TO 'slave_user'@'IP' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;

-- backup: myslqdump/dumper, innobackupex/xtrabackup
```

MySQL replication: slave configuration

```
-- restore backup

/etc/mysql/my.cnf
server-id               = 2
relay-log               = /var/log/mysql/mysql-relay-bin.log
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_do_db            = database_name
```

MySQL replication: slave configuration

```
CHANGE MASTER TO
MASTER_HOST='mariadb1.vrack.courtanet.net',
MASTER_USER='slave_user',
MASTER_PASSWORD='****',
MASTER_LOG_FILE='mariadb-bin.002716',
MASTER_LOG_POS=972282938;

START SLAVE;
```

MySQL replication: SHOW SLAVE STATUS

```
+----------------------------------+-----------------------------+-------------+-------------
| Slave_IO_State                   | Master_Host                 | Master_User | Master_Port
+----------------------------------+-----------------------------+-------------+-------------
| Waiting for master to send event | mariadb1.vrack.courtanet.net | slave-data2 |        3306
+----------------------------------+-----------------------------+-------------+-------------

+---------------+--------------------+---------------------+------------------+---------------
| Connect_Retry | Master_Log_File    | Read_Master_Log_Pos | Relay_Log_File   | Relay_Log_Pos
+---------------+--------------------+---------------------+------------------+---------------
|            60 | mariadb-bin.002725 |           957457227 | relay-bin.000023 |     264745891
+---------------+--------------------+---------------------+------------------+---------------

+-----------------------+-------------------+-------------------+------------+------------+--------------
| Relay_Master_Log_File | Slave_IO_Running  | Slave_SQL_Running | Last_Errno | Last_Error | Skip_Counter
+-----------------------+-------------------+-------------------+------------+------------+--------------
| mariadb-bin.002725    | Yes               | Yes               |          0 |            |            0
+-----------------------+-------------------+-------------------+------------+------------+--------------

+--------------------+-----------------+-----------------------+---------------------------------+----------------+----------------+----------------+----------------
| Exec_Master_Log_Pos | Relay_Log_Space | Seconds_Behind_Master | Master_SSL_Verify_Server_Cert | Last_IO_Errno | Last_IO_Error | Last_SQL_Errno | Last_SQL_Error
+--------------------+-----------------+-----------------------+---------------------------------+----------------+----------------+----------------+----------------
|          928618688 |       957458099 |                   783 | No                              |              0 |                |              0 |
+--------------------+-----------------+-----------------------+---------------------------------+----------------+----------------+----------------
```

MySQL replication

- one-way M → S replication
  - single threaded/multi-threaded (5.6+ 1 worker thread per database/slave)
  - **asynchronous** : wait until change recorded (**in local binlog**)
  - **semi-synchronous** : wait until one Slave ack received and stored event!)

What is replicated?

- SBR (statement based replication)
- RBR (row base replication)
- Mixed (choose by event size for every transaction)

SBR

- send **queries** to the slave
- not all queries are safe for replication (!)
  - **SERVER STATE** : NOW(), AUTOINCREMENT, TRIGGERS, TRANSACTIONS

RBR

- send all **modified rows** to the slave
- safer but costly
- 5.6+ optimizations (ignore blobs, increment, hashing..)

Mixed

- Mixed (choose by event size for every transaction)

Semi-synchronous replication

- since 5.6+
- log SERVER_ID+TxID in a regular table
  - master blocks after the commit and waits until one semisynchronous slave acknowledges that it has received all events for the transaction or timeout
  - slave acknowledges receipt of a transaction's events only after the events have been written to its relay log
- *temporary switch to async-replication if no ACK from the slave* !
- *guarantees consistency between master and slave*
  - *as long as all transactions committed on the master have also been applied on a slave*

Crash recovery (M)

- manually promote a slave as a new master (<5.6)
- automatic slave promotion via mysqlfailover (5.6+)

Scaling Writes

- MySQL Fabric
  - set of **tools** to manage MySQL servers in a replicated GTID environment
  - automatic sharding and HA
- Multi-master replication
  - ***Galera Cluster***

Galera Cluster

- multi-master: R/W at any node
- synchronous* replication by Certification Based Replication Method
  - good latency with increased consistency (*more…*)
- Easy migration from MySQL
  - combine with MySQL binlog replication
  - automatic node provisioning (XtraBackup)
  - transparent to applications

LF MariaDB Galera Cluster

- relaxed ACID
- *wsrep_sync_wait* - ensures sync before:
  - READ (SELECT/SHOW/BEGIN/START TRANSACTION)
  - UPDATE/DELETE
  - INSERT/REPLACE
- Benefits:
  - configurable consistency
  - recover from failures (typical < 1s)
  - optimize replication lag (typical < 10ms)

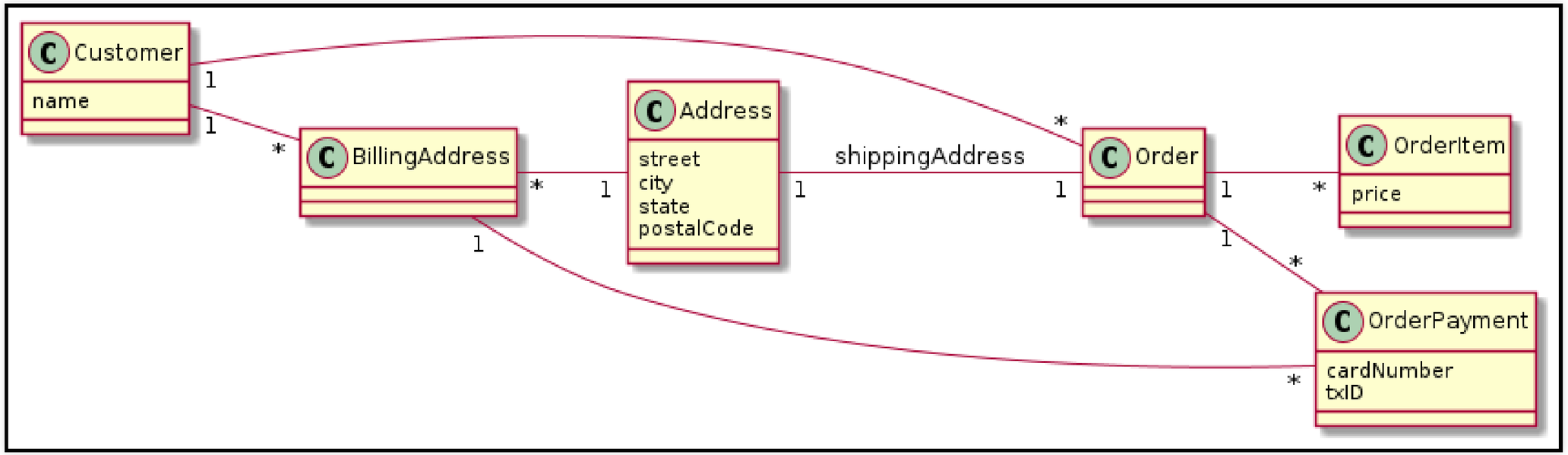
Scaling MySQL @LF

- **Scaling** is expensive:
  - scaling writes → locking + partitioning (sharding)
  - scaling reads → replication (latency, error recovery)
- *Master/Slave* replication:
  - network partitions and **split brain**
  - slowly diverging Master/Slave, not automatic check/resynchronisation
  - problematic when failover switch
- *Multi-master* (**Galera**): 6 years of incident free operation
- **Table size limits** ⇒ "ALTER TABLE" problem

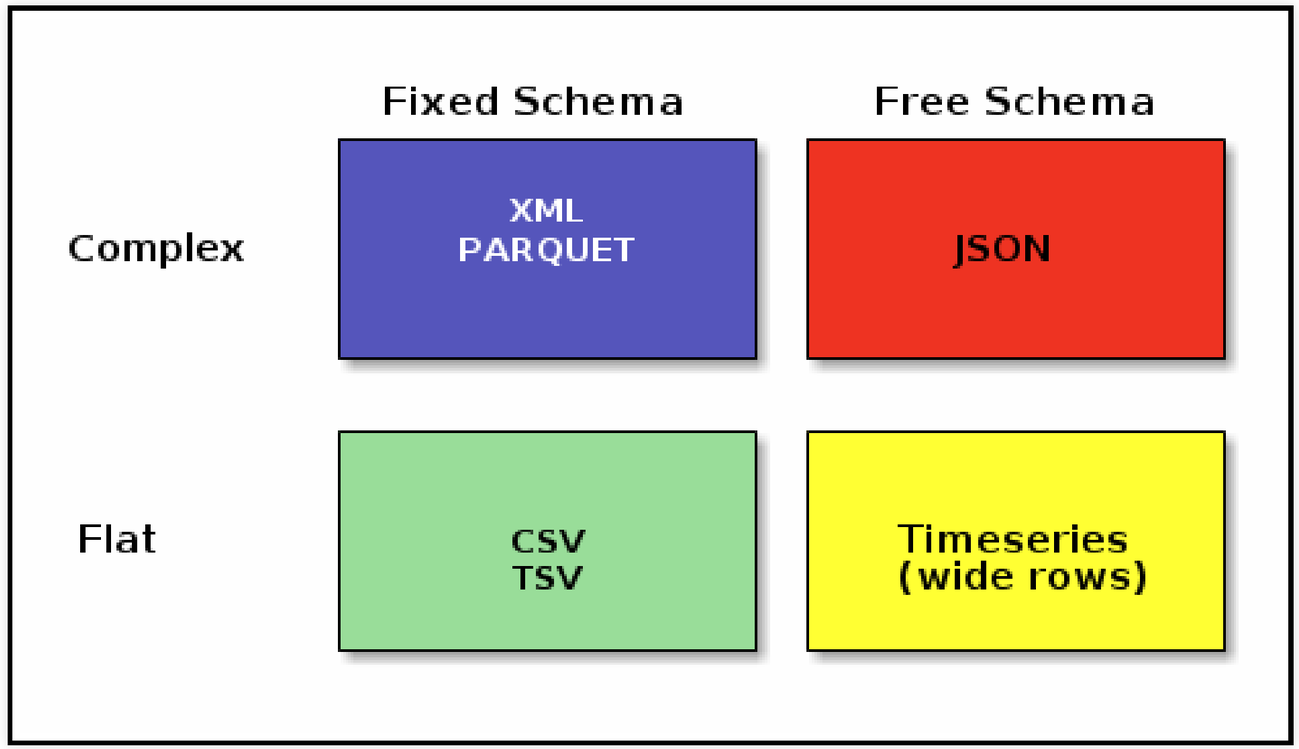
Plan

1. Course info
2. Scaling MySQL @LesFurets.com
3. **From SQL to NoSQL**
4. TP PostgreSQL

SQL models tuples and joins



Modeling complex data



Relational vs Document/Objects

- **Relational model**: relations / tuples + normalization
- **Memory**: rich data structures !

Relational vs Document/Objects

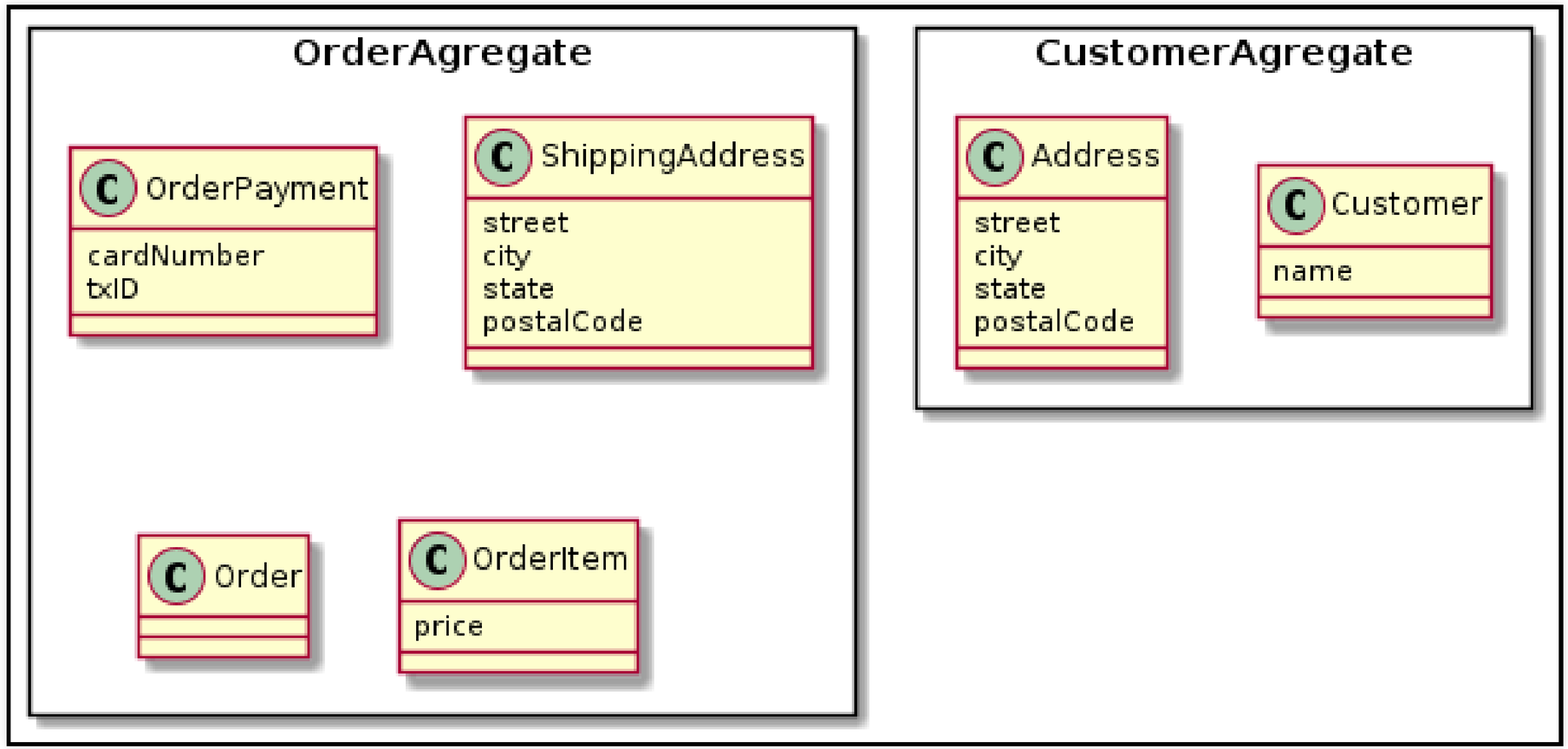
- **Memory -** object graphs
  - complex mapping from object to relations → ORMs
  - leads to performance issues
    - many rows/tables/**JOINS**
- Schema evolution:
  - adding an attribute → adding a whole column
  - expensive locks → application downtime

Not only SQL

- 2009, Johan Oskarson, #NoSQL meetup for distributed, open-source, non-relational databases
  - not using relational model SQL
  - run on clusters
  - ACID ⇒ tunable consistency
  - fixed schema ⇒ flexible schema
  - **polyglot persistance**

NoSQL models agregates

- collection of related objects that should be treated as a unit (consistency / data management)



Modelling SQL vs NoSQL

- *SQL*: **model first**
  - 1 model used for all queries
- *NoSQL*: **query first**
  - 1 data access pattern for each aggregate

NoSQL aggregate types

1. Key-value databases
2. Document-oriented databases
3. Column-oriented databases
4. Graph databases

Key-value databases

- Store and retrieve Blobs based on a primary Key
- Simplest API that matches REST : PUT/GET/DELETE

Map (K → V)

```
Use case: Session information, user profiles, ...
```

Document oriented databases

- Stores and retrieves **documents/fragments** (XML, JSON…):
  - self-describing, hierarchical tree data structures
  - maps, collections and scalar values

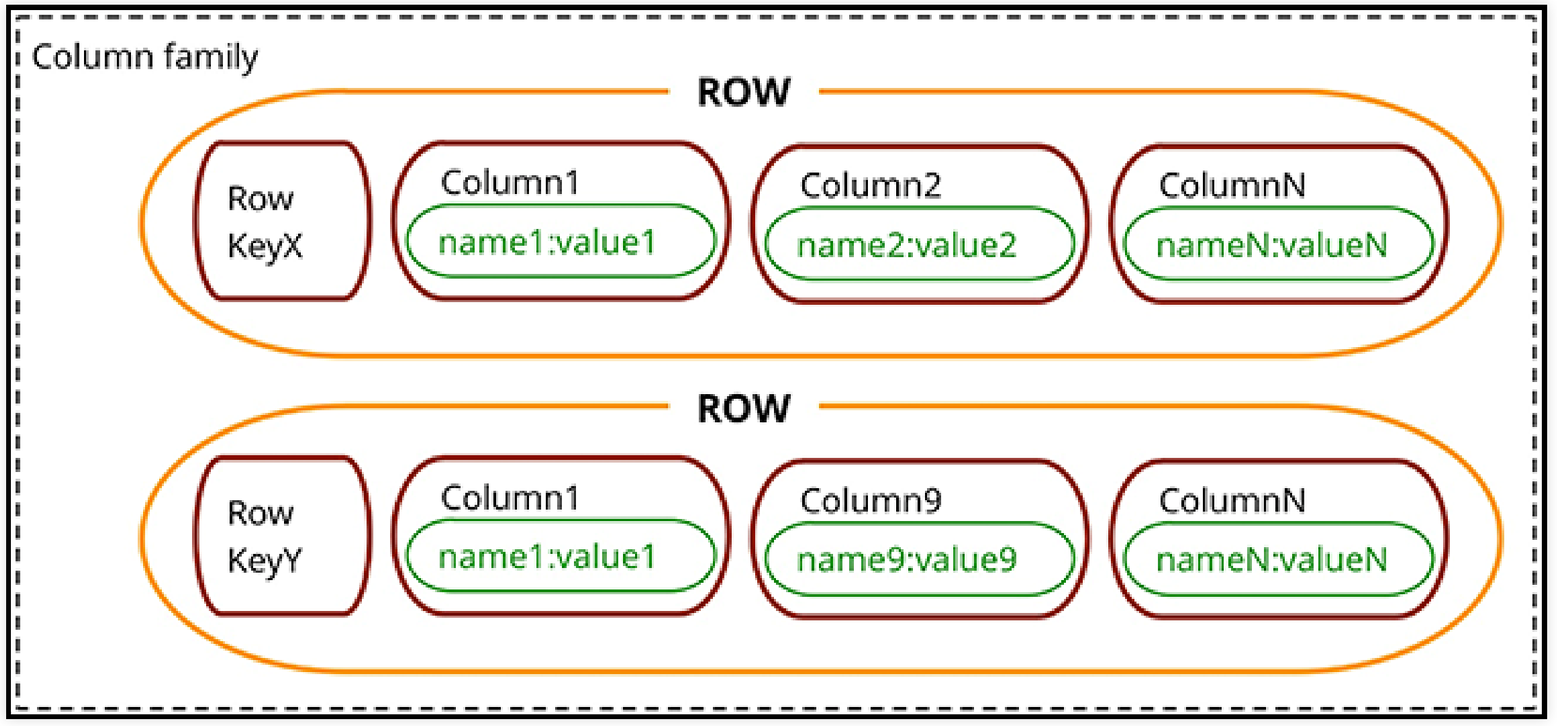
Key-value stores where the value is queryable

```
Use case:  CMS, Product catalog, ...
```

Column oriented databases

- Stores data in **tables/column families** as rows that have many columns associated to a row key

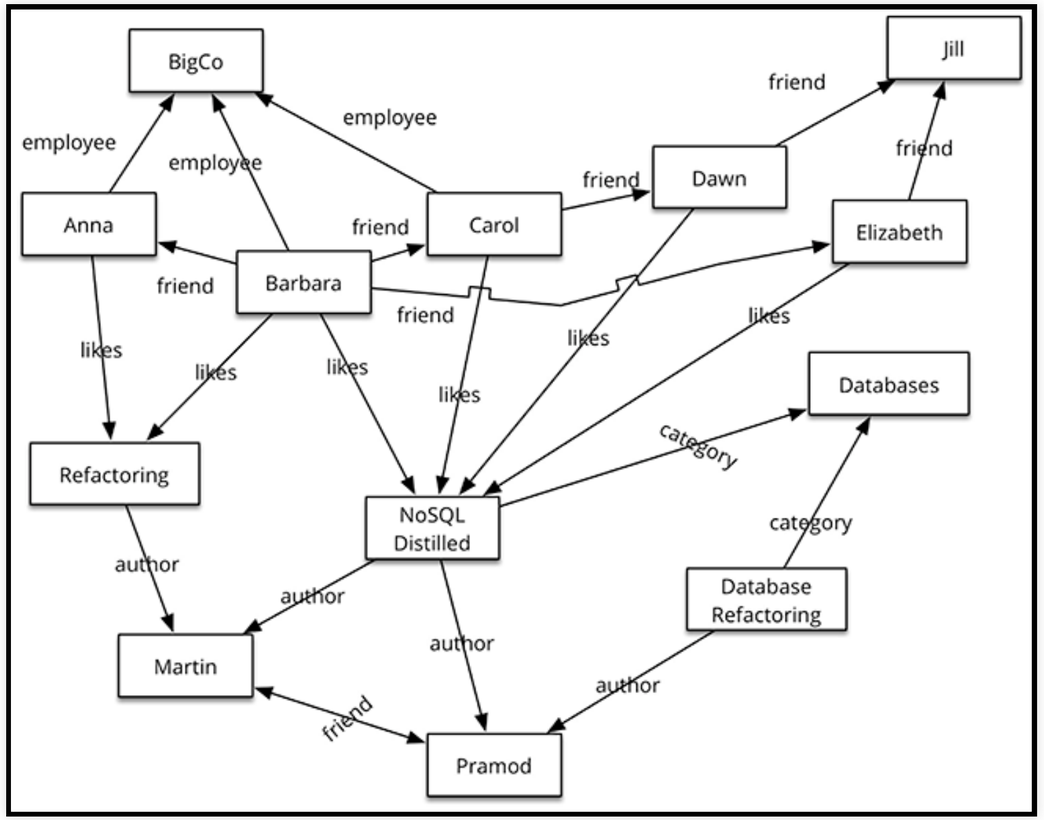
Map of maps (rowId → (columnName → columnValue))




```
Use cases: Time series, event logging, ...
```

Graph databases

- Stores **entities** and **relations** between entities
- **Query:** traversal of the graph



Use cases: Social networks, recommendation engines

Choosing a storage solution:

- business requirements
- tehnical aspects
- human / organisational
- tradeoffs

Business requirements

1. Ingest (application/batch/streaming data)
2. Store
    - how the data is accessed (file vs row vs aggregation of columns)
    - access controls (schema/database)
    - how long the data is accessed (RAM/SSD/bucket/nearline/cold)
3. Process and analyze : cleaning, normalizing,sumarysation
4. Vizualize and explore

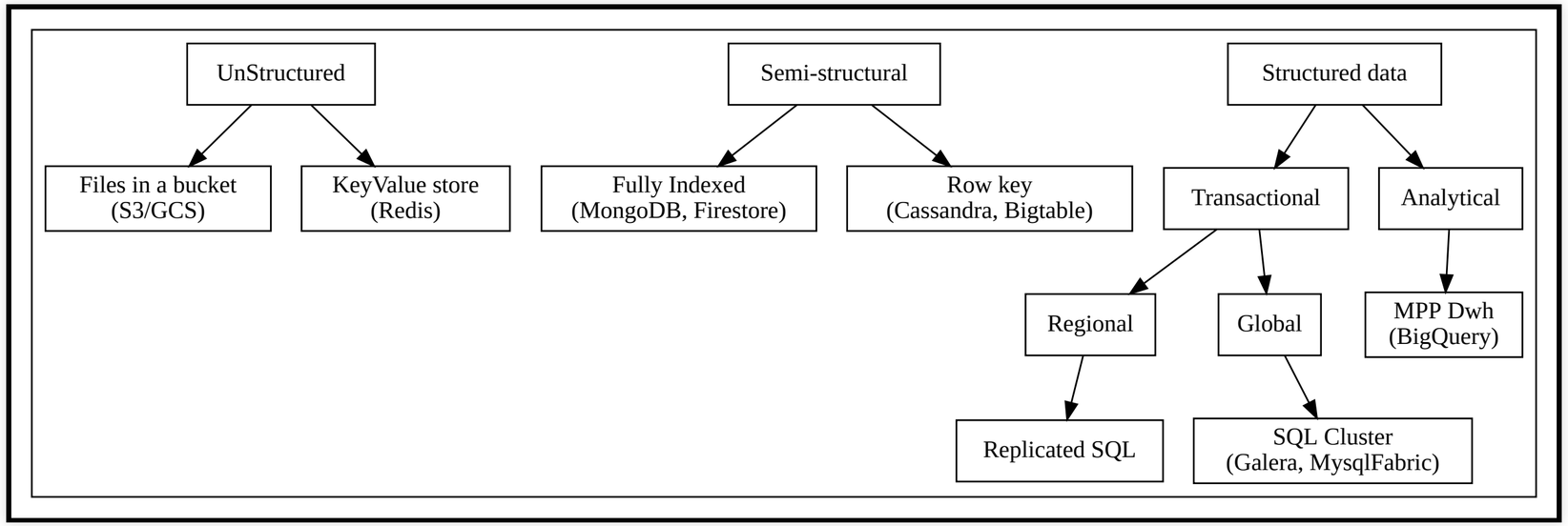
Technical aspects

- **volume** and **velocity**
- variation in **structure** (*schema vs schemaless*)
  - structured (transactional vs analytics)
  - semi-structured (fully indexed vs row key access)
  - unstructured (files/blobs)
- **data access patterns** (agregats)
- security requirements (audit, logging)

Other aspects

- *human / organisational* aspects :
  - team structure and abilities
  - entreprise culture (building software vs using software)
- cost and evolution, vendor lock-in …

Simple decision tree

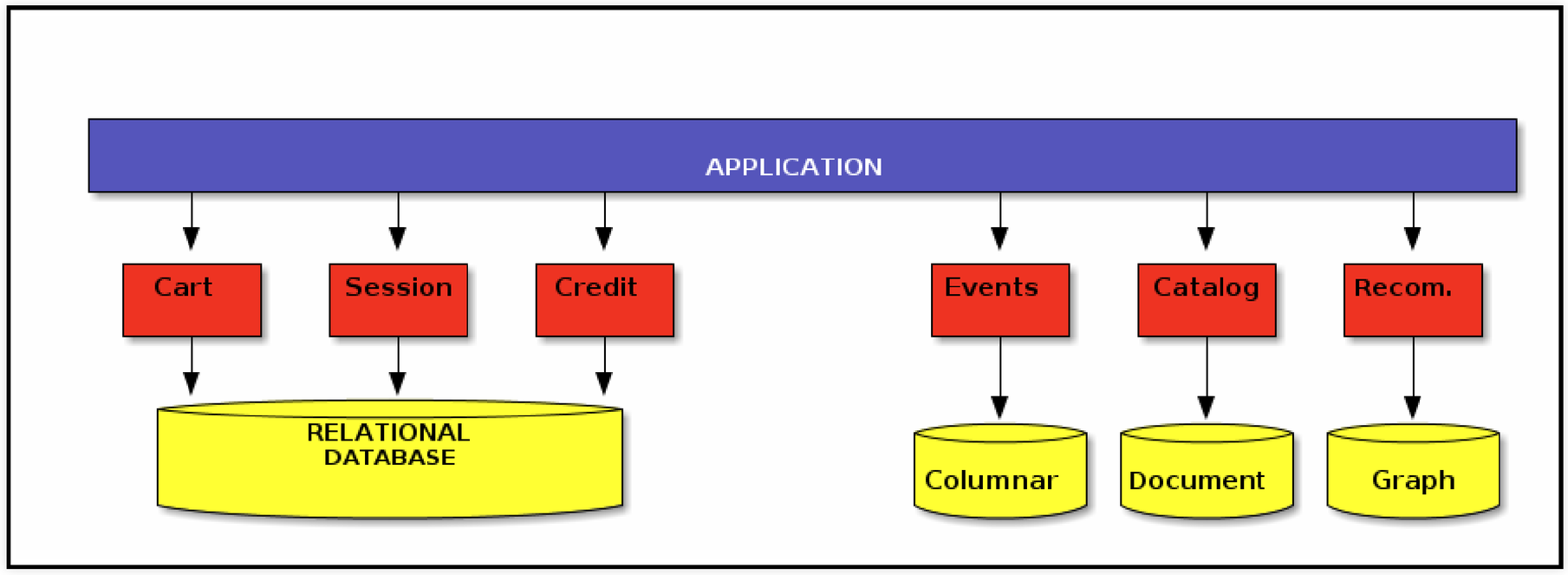


Tradeoffs

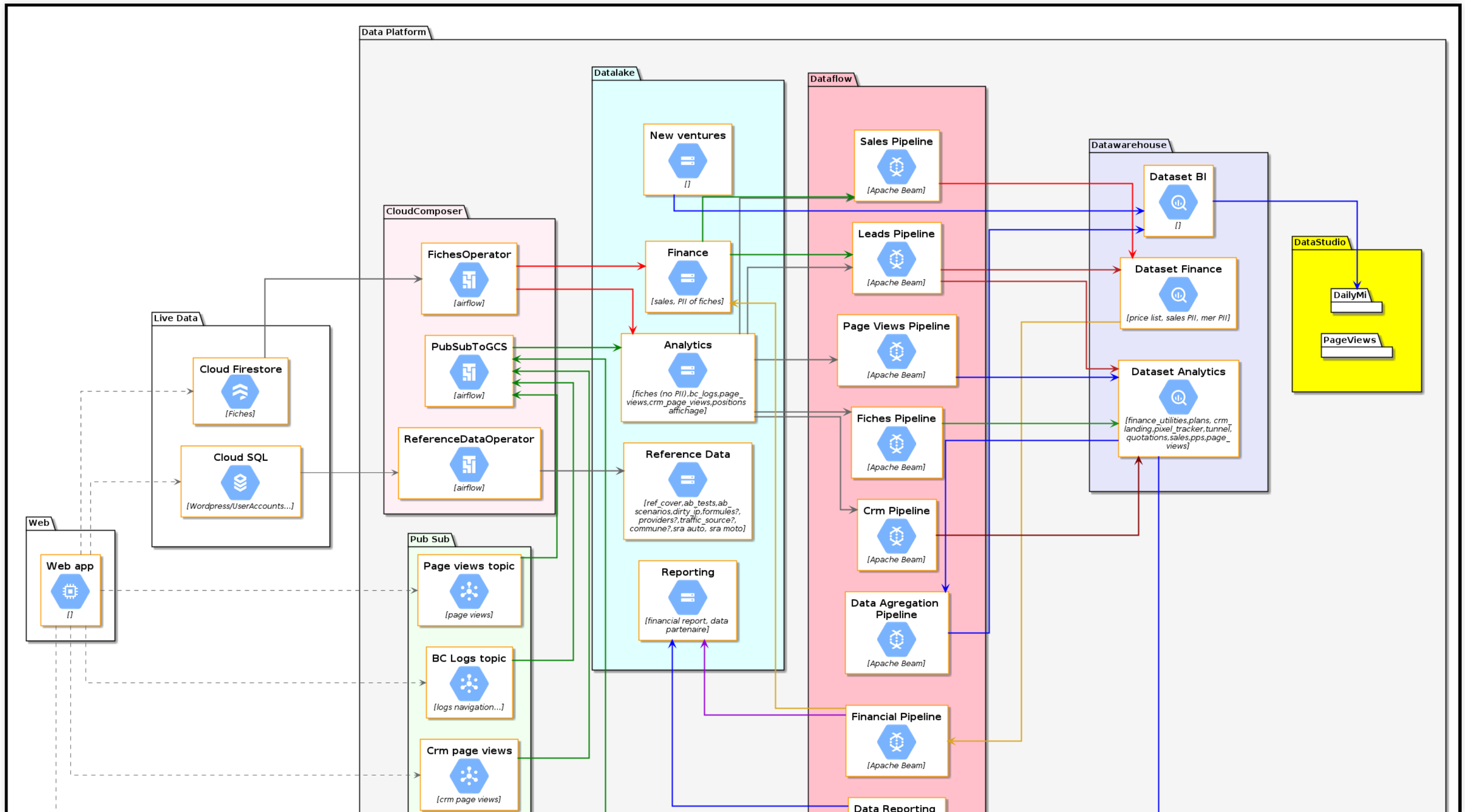
- Performance:
  - Hadoop: large scale batch computation but **high latency**
  - Cassandra: low latency, fin grain storage but **limited data model**
- Consistency: **ACID** ⇒ tunable/eventual consistency (**CAP**)
- Model:
  - Incremental architectures ⇒ human failures

Polyglot persistance

- **aggregates** have different requirements (availability/consistance/backup)
- **Mix and match relational and non-relational storage**



Cloud architecture @LesFurets



Plan

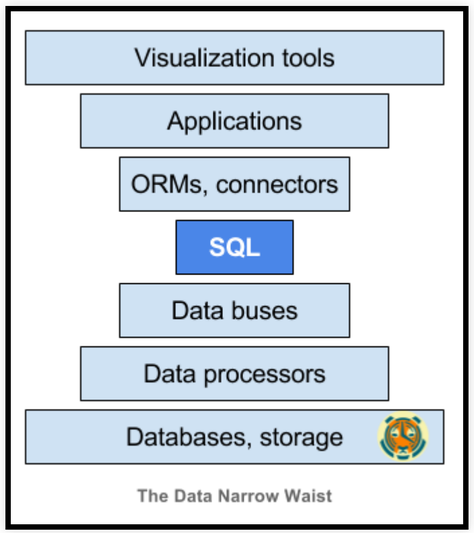
1. Course info
2. Scaling MySQL @LesFurets.com
3. From SQL to NoSQL
4. **TP PostgreSQL**

SQL in 2021 ?

- **MapReduce: A major step backwards?** (*2008*)
  - MapReduce is difficult to integrate with other DBMS tools (*BI*, *Reporting tools*..)
  - High-level, declarative language ⇒ **simpler** to use
  - **Schemas are good**
  - Separation of the **schema** from the **application** is good

SQL in 2021 ?

- **Why SQL is beating NoSQL, and what this means for the future of data** (2017)
  - SQL as ***interface / universal language for data processing***



SQL in 2021 - Massively Parallel Processing

- massive relational SQL clusters
  - separate *compute* and *storage*
- fully managed, petabyte-scale data warehouse service in the cloud
- (*Bigquery*, *Redshift*, *Snowflake*)

SQL in 2021 - ML in SQL

SQL in 2021 - SQL analytics-as-code

- *Dwh pipelines shift*: **ETL** *to* **ELT**
-  **DBT** does the T in ELT (Extract, Load, Transform) processes
  - write transformations as queries and orchestrate them
    - transform data where it lives
    - using plain SQL SELECT
  - infer dependency graphs and run transformation models in order ⇒ **pipeline automation**

Why PostgreSQL?

- PostgreSQL
  - 1974: **INtelligent Graphic RElational System**, *Michael Stonebraker (Berkeley University)*
  - 1985: POSTinGRES then **PostgreSQL** (1995)
- Features
  - easy to extend/customize : **data types** (*JSONB*, *Arrays*, *Cube*), **operators**, **UDF** (*Python*, *Ruby*, *R*, *Javascript*…)
  - **Foreign Data Wrappers**: map an external DBs to tables ⇒ strong ACID properties @scale

PostgreSQL offsprings

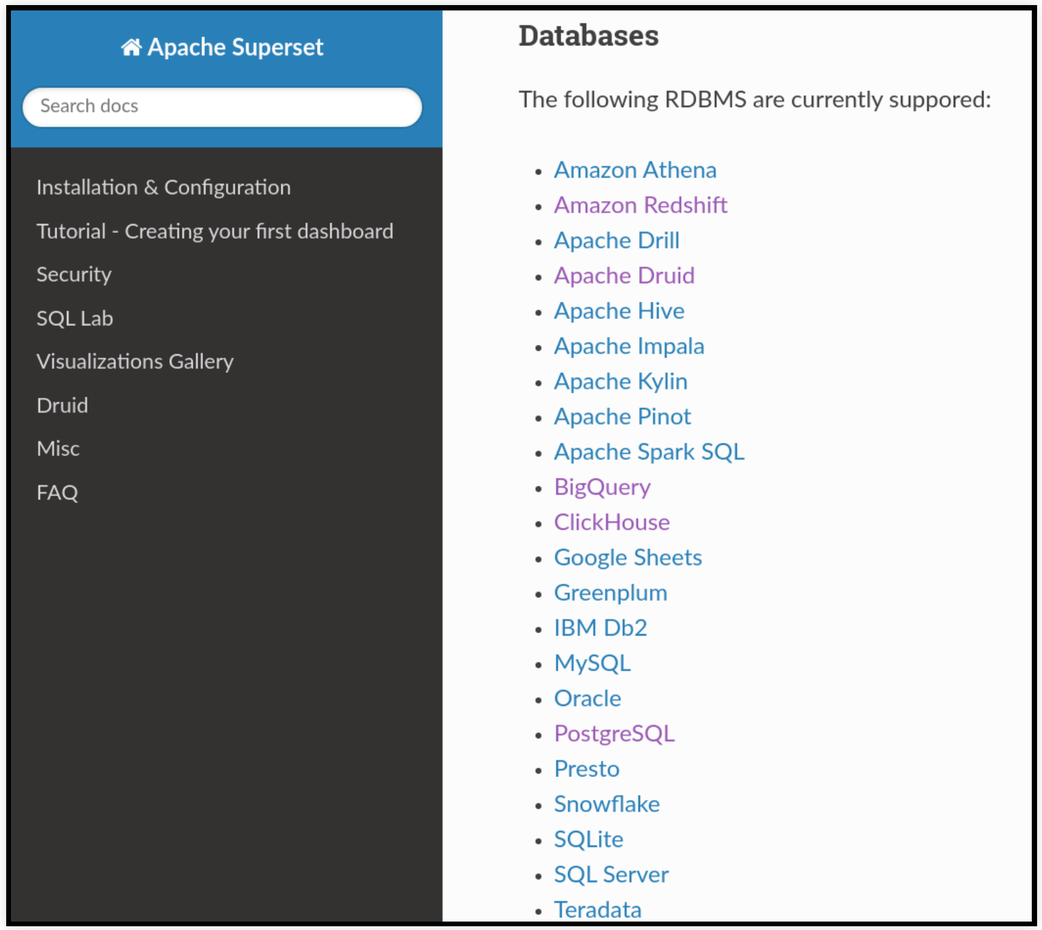
PostgreSQL features

- PostGIS - advanced geospatial database
- Rich indexes: Gin/GIST/KNN/Sp-GIST
- natural-language processing
- multidimensional indexing
- TP ⇒ PostgreSQL & extensions (*tablefunc*,*dict_xsyn*, *fuzzystrmatch*, *pg_trgm*, *cube*)

more…

Apache Superset: business intelligence web application

Apache Superset: RDBMS support



Installation de l'environnement pour le TP

Configuration d'un reseau host-only via Virtualbox

1. Démarrez l'application **Virtualbox**
2. Verifiez/creez un réseau host-only ***vboxnet0*** ("File/Host network Manager" ...)

Configuration d'un reseau host-only via VBoxManage

- Si vous avez rencontré des difficultes a l'etape precedente vous pouvez creer l'interface en ligne de commande:

```
VBoxManage hostonlyif create
VBoxManage hostonlyif ipconfig vboxnet0 --ip 192.168.56.1
VBoxManage dhcpserver add --ifname vboxnet0 --ip 192.168.56.1\
           -netmask 255.255.255.0 --lowerip 192.168.56.100\
           --upperip 192.168.56.200
VBoxManage dhcpserver modify --ifname vboxnet0 --enable
```

Telecharger la VM pour le tp

Telecharger l'image Virtualbox: http://bit.ly/telecom-postgresql-superset

Importer la VM

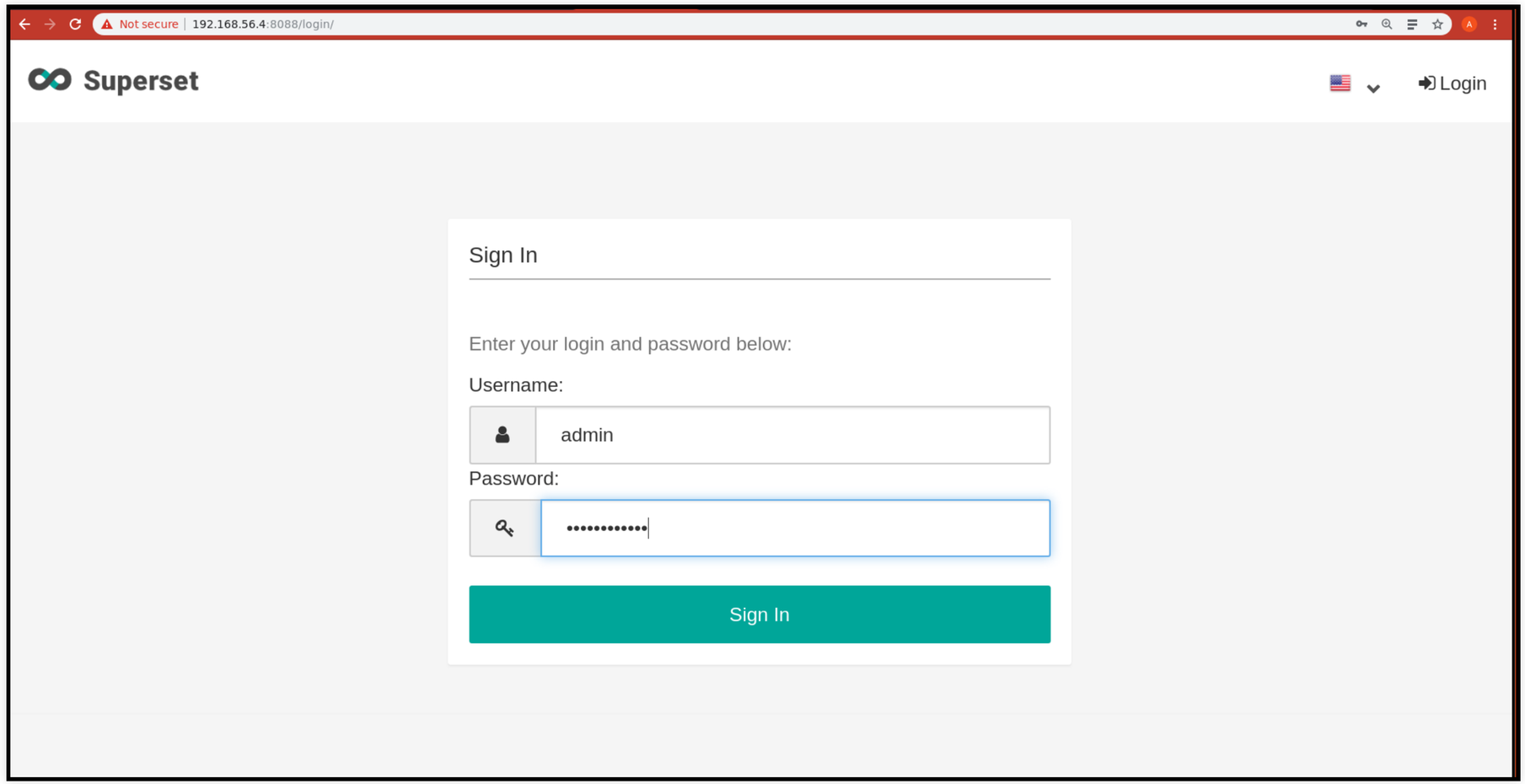
File/Import appliance ...

Demarer la VM

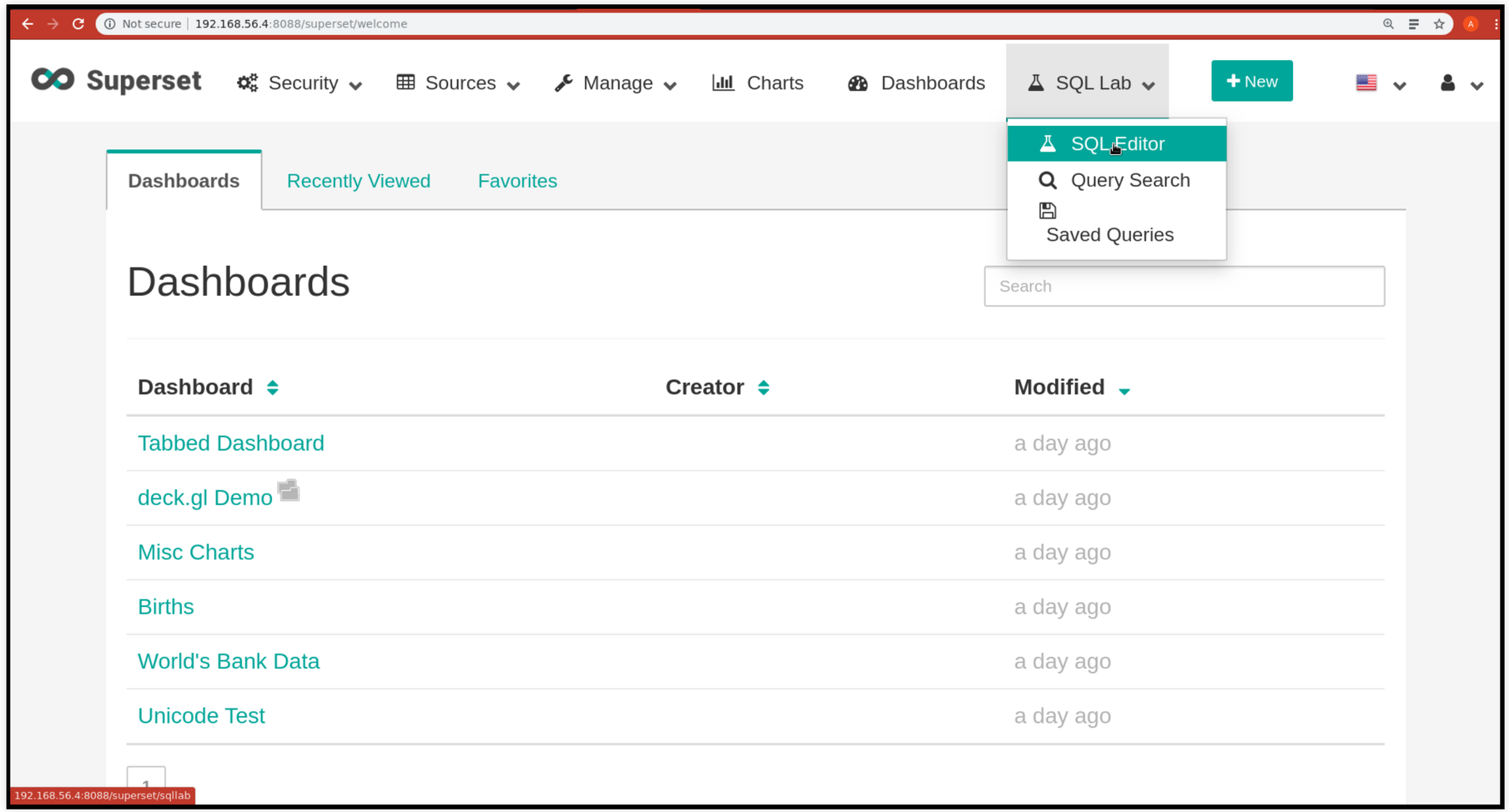
- via le button Run/Start et noter l'URL pour acceder a Superset

Connectez vous a Superset via votre navigateur

Username: admin Password: bigdatafuret



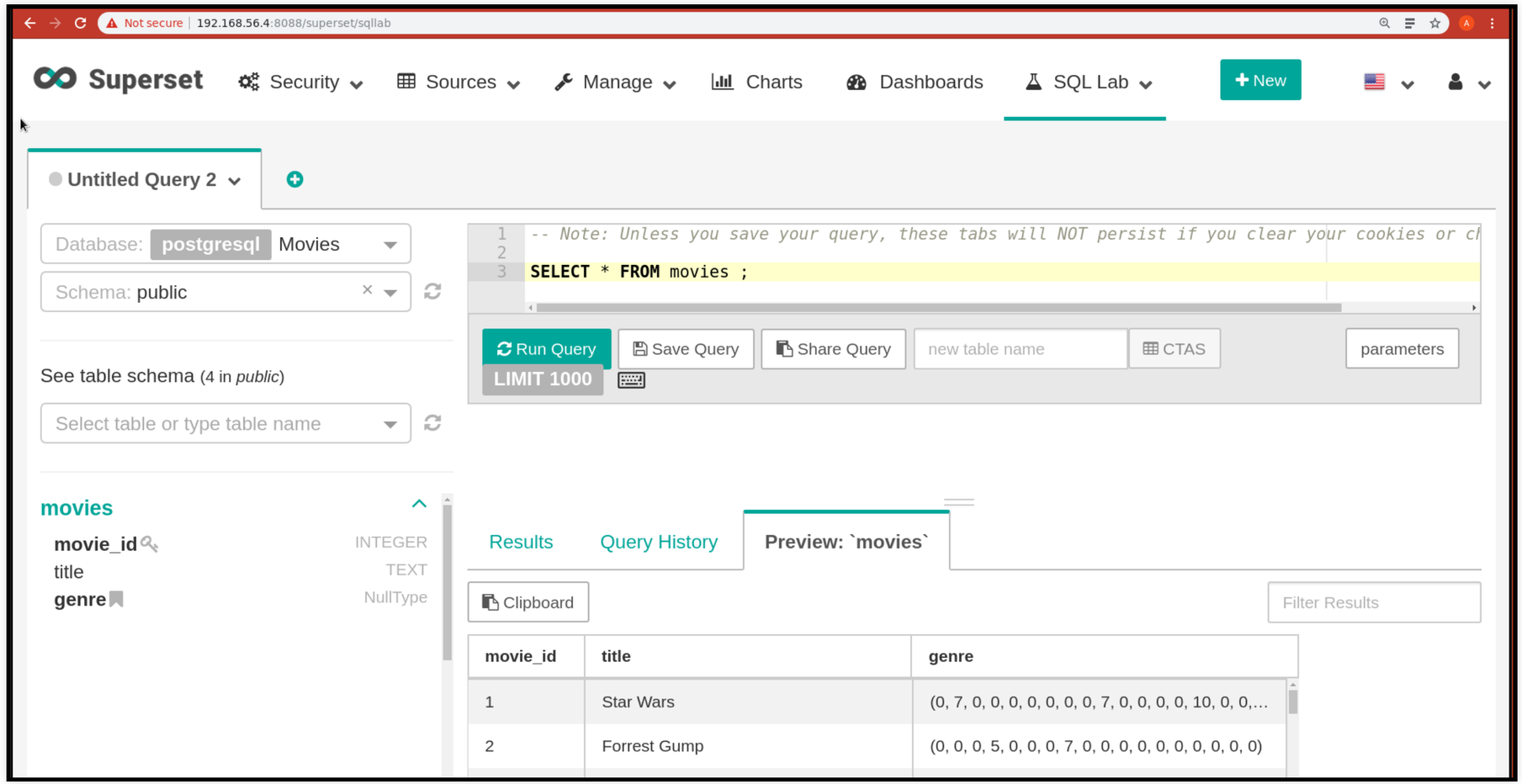
Ouvrez le SQLEditor



Dans le SQLEditor lancez une requete

Tester la requete _SELECT * FROM MOVIES_ (dans la Database Movies/ Schema: public.

Si vous avez eu des résultats, l'installation s'est bien passe, felicitations !



TP1: PostgreSQL Recherche et recommandation (1h)

- **Objectifs**:
  - prise en main de l'environnment de TP basé sur VirtualBOX
  - écrire des requêtes SQL
    - recherche: textuelle, approximative, phonétique
    - moteur de recommandation très basique

Recherche et recommandation

Moteur de recherche et recommandation des films:

- recherche: *textuelle*, *approximative*, *phonétique*
- recherche de type *graph*
- moteur de recommandation très basique

Schéma (déjà créé)

```
CREATE TABLE genres (
        name text UNIQUE,
        position integer
);
CREATE TABLE movies (
        movie_id SERIAL PRIMARY KEY,
        title text,
        genre cube
);
CREATE TABLE actors (
        actor_id SERIAL PRIMARY KEY,
        name text
);

CREATE TABLE movies_actors (
        movie_id integer REFERENCES movies NOT NULL,
```

```
CREATE INDEX movies_actors_movie_id ON movies_actors (movie_id);
CREATE INDEX movies_actors_actor_id ON movies_actors (actor_id);
CREATE INDEX movies_genres_cube ON movies USING gist (genre);
```

Create schema script Import data script

Recherche

- ***Recherche exacte / pattern matching***
- ***Distance de Levenshtein*** → typos simples
- ***N-gram/similarité*** → trouver les erreurs modérées
- ***Full text match @@*** → similarité grammaticale
- ***Métaphone*** → similarité phonétique

Recherche textuelle/patterns

Utilisez les opérateurs LIKE ou RegEX pour les requêtes suivantes:

1. Tous les films qui ont le mot ***stardust*** dans leur nom.
2. Compter tous les films dont le titre ne commence pas par le mot ***the***
3. Tous les films qui ont le mot ***war*** dans le titre mais pas en dernière position

Distance Levenshtein

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- Opérations: **S**ubstitute, **I**nsert, **D**elete
- Distance *Levenshtein* : nb minimal d'opérations

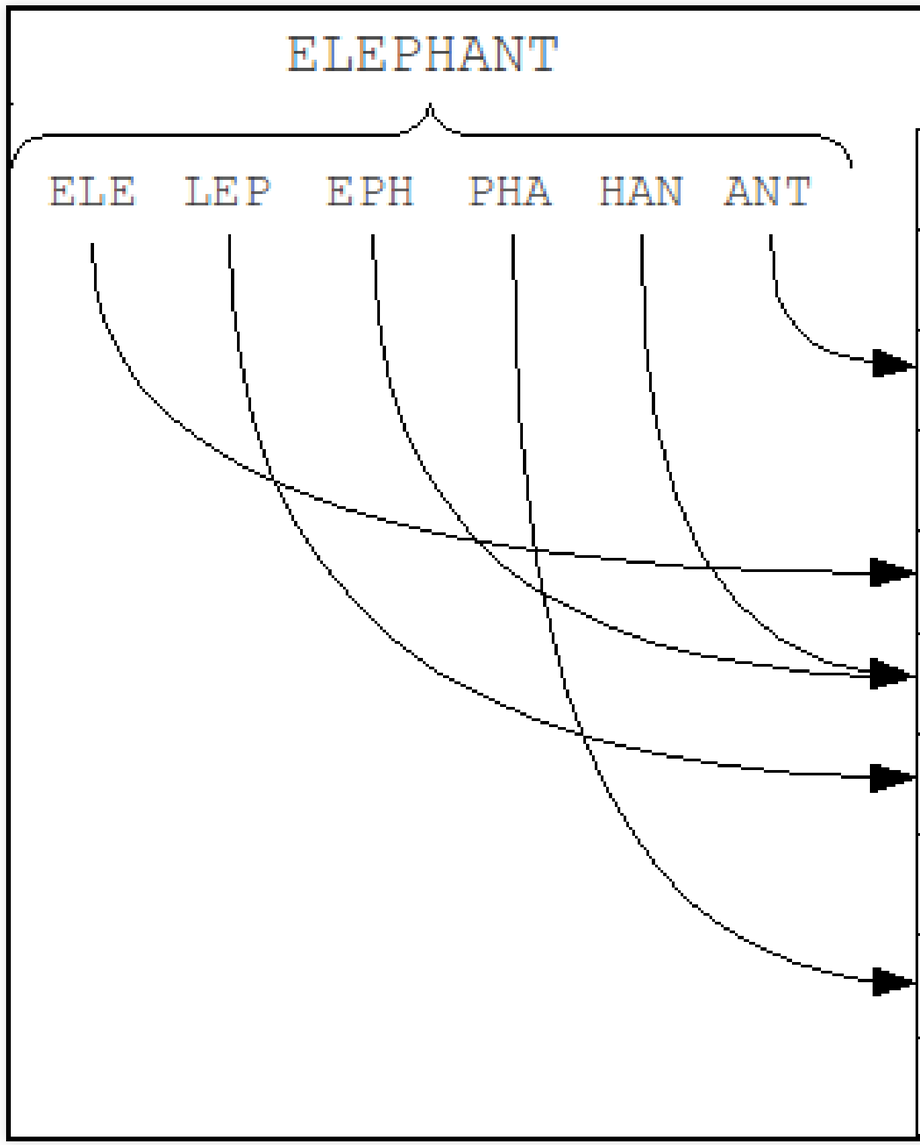
Distance Levenshtein

Utilisez les fonctions du package fuzzystrgmatch pour trouver :

1. La distance levenshtein entre les mots ***execution*** et ***intention***
2. Tous les films qui sont a une distance *levenshtein* inférieure a 9 de la chaine suivante: ***a hard day nght***

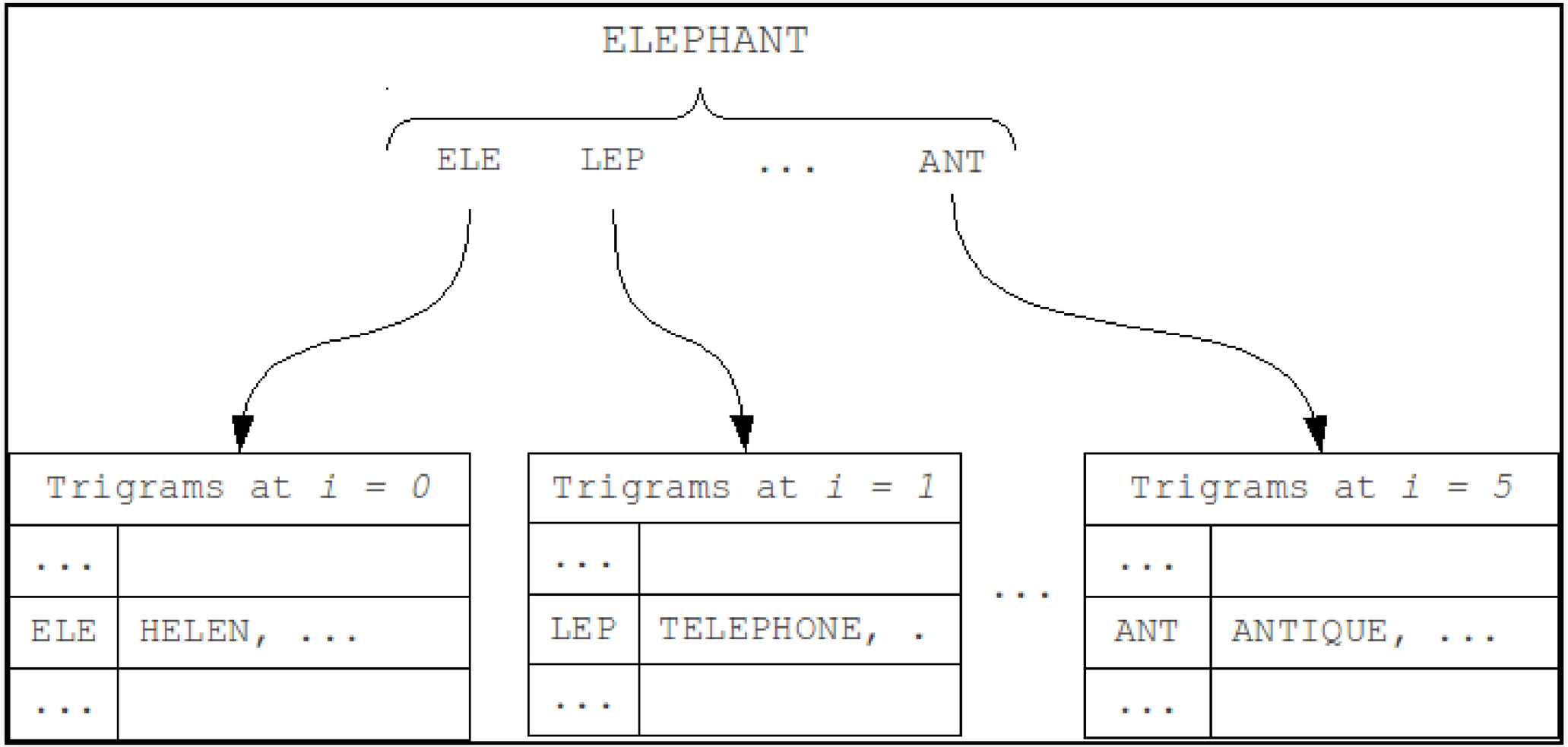
Documentation extension fuzzystrgmatch

N-gram



Trigram	List of words
...	
ANT	ANTIQUE, GIANT, ...
...	
ELE	HELEN, SELECTION, ...
...	
LEP	TELEPHONE, LEPRA, ...
...	
PHA	APHASIA, PHASE,...
...	

N-gram, similarity search



N-gram, similarity search (%)

Écrivez les requêtes pour trouver :

1. Tous les tri-grammes du mot ***Avatar***
2. La similarité entre ***VOTKA*** et ***VODKA***
3. Tous les films dont le titre est similaire a plus de 0.1% du titre ***Avatar*** .

Documentation extension trgm

Full text search

Trouver les filmes qui contiennent les formes grammaticales des mots 'night' et 'day':

(ignorer les mots de liaison/ pluriel/etc..)

Algorithme:

1. ***extraire les racine des mots (lexèmes) → spécifiques au langage !***
2. comparer les vecteurs des lexèmes

Full text search

```
SELECT to_tsvector('A Hard Day''s Night'),
     to_tsquery('english', 'night & day');

       to_tsvector         |   to_tsquery
---------------------------+-----------------
'day':3 'hard':2 'night':5 | 'night' & 'day'
```

- ***tsvector*** : lexèmes :position
- ***tsquery*** : lexèmes séparées par &
- ***spécifique au language !***

Documentation recherche plein text …

Full text search

```
SELECT title
FROM movies
WHERE to_tsvector(title) @@ to_tsquery('english', 'night & day');
```

Full text search

```
SELECT title
FROM movies
WHERE to_tsvector(title) @@ to_tsquery('english', 'night & day');
```

```
SELECT title
FROM movies
WHERE title @@ 'night & day';

A Hard Day’s Night
Six Days Seven Nights
Long Day’s Journey Into Night
```

Recherche phonétique

- plusieurs fonctions pour la codification phonétique des mots

```
SELECT name, dmetaphone(name), dmetaphone_alt(name),
  metaphone(name, 8), soundex(name)
FROM actors;

     name      | dmetaphone | dmetaphone_alt | metaphone | soundex
---------------+------------+----------------+-----------+---------
50 Cent        | SNT        | SNT            | SNT       | C530
Aaron Eckhart  | ARNK       | ARNK           | ARNKHRT   | A652
Agatha Hurle   | AK0R       | AKTR           | AK0HRL    | A236
```

Documentation fuzzystrmatch…

Recherche phonétique

1. Trouver les films qui ont des acteurs dont les noms se prononcent pareil.

2. Trouver les acteurs avec un nom similaire a ***Robin Wiliams***, triés par similarité (combiner %, metaphone et levenshtein):

```
 actor_id | name
----------+------------------
     4093 | Robin Williams
     2442 | John Williams
     4479 | Steven Williams
     4090 | Robin Shou
```

Search

- ***Recherche exacte / pattern matching***
- ***Distance de Levenstein*** → typos simples
- ***N-gram/similarite*** → trouver les erreurs modérées
- ***Full text match @@*** → similarité grammaticale
- ***Métaphone*** → similarité phonétique

Recherche ”graph”

- Trouvez le graph des acteurs connectees a Tom Hanks (ont deja joue dans un film avec l’acteur ou bien il y a un chemin films/acteurs qui mene a l’acteur)

Hint: vous pouvez utiliser les Common Table Expressions

title	name
Forrest Gump	Tom Hanks
The Green Mile	Tom Hanks
Apollo 13	Tom Hanks
Saving Private Ryan	Tom Hanks
Sleepless in Seattle	Tom Hanks
Toy Story	Tom Hanks
Toy Story 2	Tom Hanks
Big	Tom Hanks
Splash	Tom Hanks
Cast Away	Tom Hanks
You've Got Mail	Tom Hanks
The Bonfire of the Vanities	Tom Hanks
Philadelphia	Tom Hanks
Dragnet	Tom Hanks
The Money Pit	Tom Hanks
The Man with One Red Shoe	Tom Hanks
A League of Their Own	Tom Hanks
The 'Burbs	Tom Hanks
Bachelor Party	Tom Hanks
Sleeping Dogs	Tom Hanks
Forrest Gump	Robin Wright Penn
Forrest Gump	Gary Sinise
Forrest Gump	Mykelti Williamson
The Green Mile	Michael Clarke Duncan
The Green Mile	Bonnie Hunt
The Green Mile	David Morse
Apollo 13	Bill Paxton
Apollo 13	Kevin Bacon

Recherche multi-dimensionnelle

```
CREATE TABLE movies (
        movie_id SERIAL PRIMARY KEY,
        title text,
        genre cube (1)
);

INSERT INTO movies (movie_id,title,genre) VALUES
(1,'Star Wars',
'(0,7,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0)') (2)
),
```

- on utilise le type cube <1> pour mapper les notes sur un vecteur n-dimensionnel de valeurs (= score du film <2>)

Recherche multi-dimensionnelle

- les noms pour les dimensions sont définis dans la table genres

```
CREATE TABLE genres (
        name text UNIQUE,
        position integer
);

INSERT INTO genres (name,position) VALUES
('Action',1),
('Adventure',2),
('Animation',3),
...
('Sport',16),
('Thriller',17),
('Western',18);
```

Recherche multi-dimensionnelle

Utiliser le module cube pour recommander des filmes ***similaires*** (du même genre)

- Afficher les notes du film ***Star Wars***
- Quelle est la note du film ***Star Wars*** dans la catégorie 'Animation'
- Afficher les films avec les meilleurs notes dans la catégorie SciFi

Recherche multi-dimensionnelle

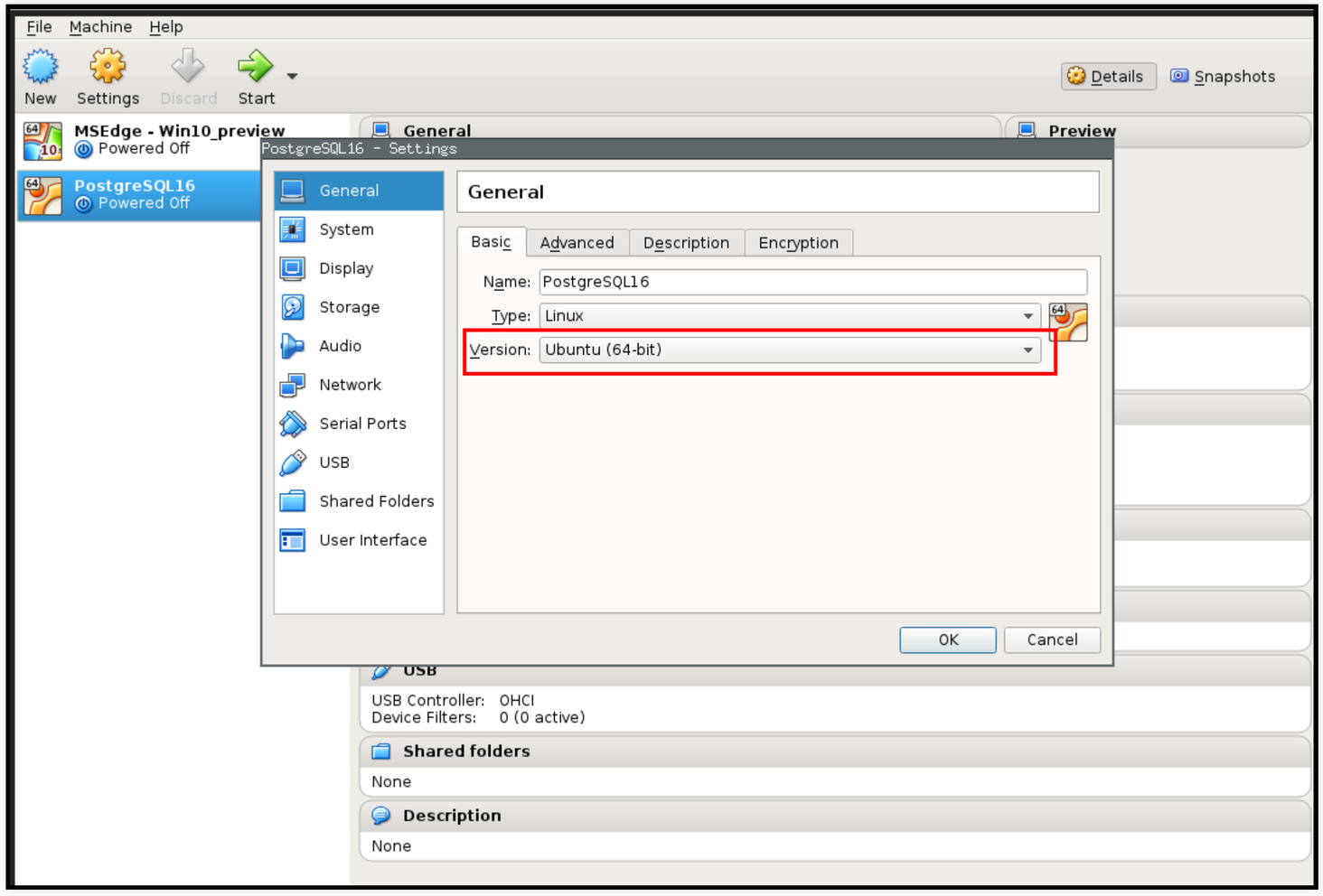
- Afficher les films similaires (**cube_distance**) a ***Star Wars*** (vecteur = (0, 7, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 10, 0, 0, 0)) du plus similaire au moins similaire

```
                    title                     |       dist
----------------------------------------------+------------------
Star Wars                                     |                0
Star Wars: Episode V - The Empire Strikes Back |                2
Avatar                                        |                5
Explorers                                     | 5.74456264653803
Krull                                         | 6.48074069840786
E.T. The Extra-Terrestrial                    | 7.61577310586391
```

- Écrivez une requête pour trouver les films qui sont a moins de 5 points de différence sur chaque dimension (utiliser *cube_enlarge* et @>).

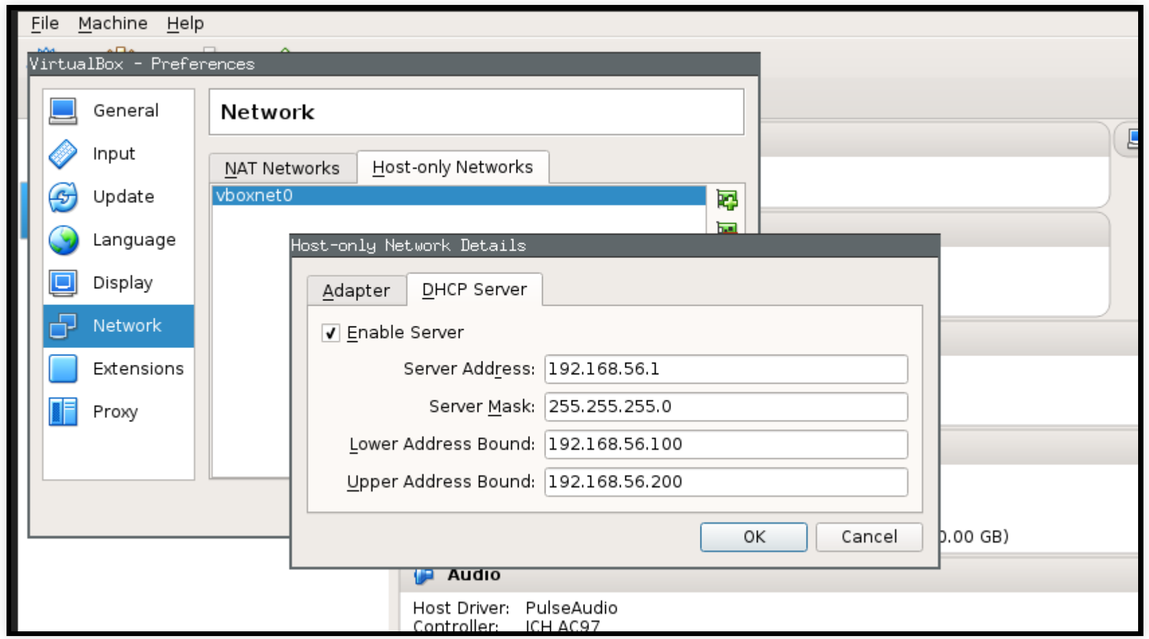
Troubleshoot VM freeze

- VM hangs at *Loading initial ramdisk ... *
- check VM type ⇒ Version Ubuntu(64 git)



Troubleshoot no IP

- check DHCP settings



Troubleshoot no IP

- try to re-create the vboxnet0 interface via cmdline
  - poweroff VM
  - re-create the network interface:

```
VBoxManage hostonlyif create remove vboxnet0
VBoxManage hostonlyif create
VBoxManage hostonlyif ipconfig vboxnet0 --ip 192.168.56.1
VBoxManage dhcpserver add --ifname vboxnet0 --ip 192.168.56.1\
          -netmask 255.255.255.0 --lowerip 192.168.56.100\
          --upperip 192.168.56.200
VBoxManage dhcpserver modify --ifname vboxnet0 --enable
```

- restart VM

Installation en détail

Fedora Ubuntu

```
dnf install postgresql postgresql-server postgresql-contrib (1)
postgresql-setup initdb (2)

systemctl start postgresql.service (3)

yum install pgadmin3   (4)

CREATE EXTENSION tablefunc; (5)
CREATE EXTENSION dict_xsyn;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION pg_trgm;
CREATE EXTENSION cube;
```

1. Installation du client/serveur/extensions suplementaires
2. Initialisation de la base
3. Démarrage du serveur
4. Front-end requetage
5. Installation des extensions Verifier les extensions installees

Create index

```
CREATE INDEX  [ nom ] ON table [ USING method ]
    ( { colonne | ( expression ) } [ classeop ]  ... )
```

- ***method***: btree/hash/gin/gist
- ***classeop*** : operator class that can use the index

Documentation

Ressources:

Bigdata - book by Nathan Marz book

NoSQL Distilled - book by Martin Fowler

7 databases in 7 days book

BigTable paper

MovieLens dataset

Ressources:

Why SQL is beating NoSQL, and what this means for the future of data

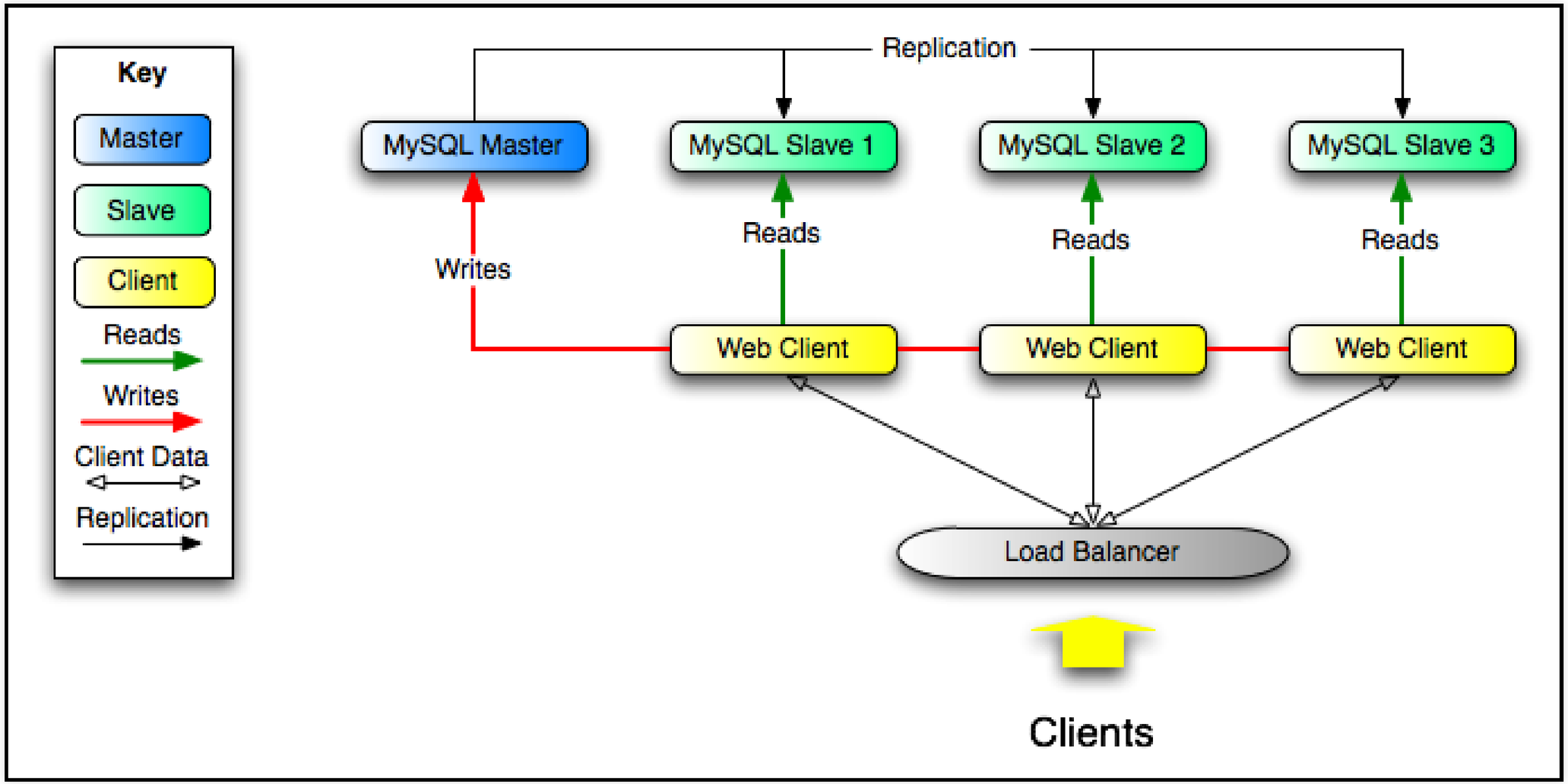
MapReduce: A major step backwards

Why PostgreSQL

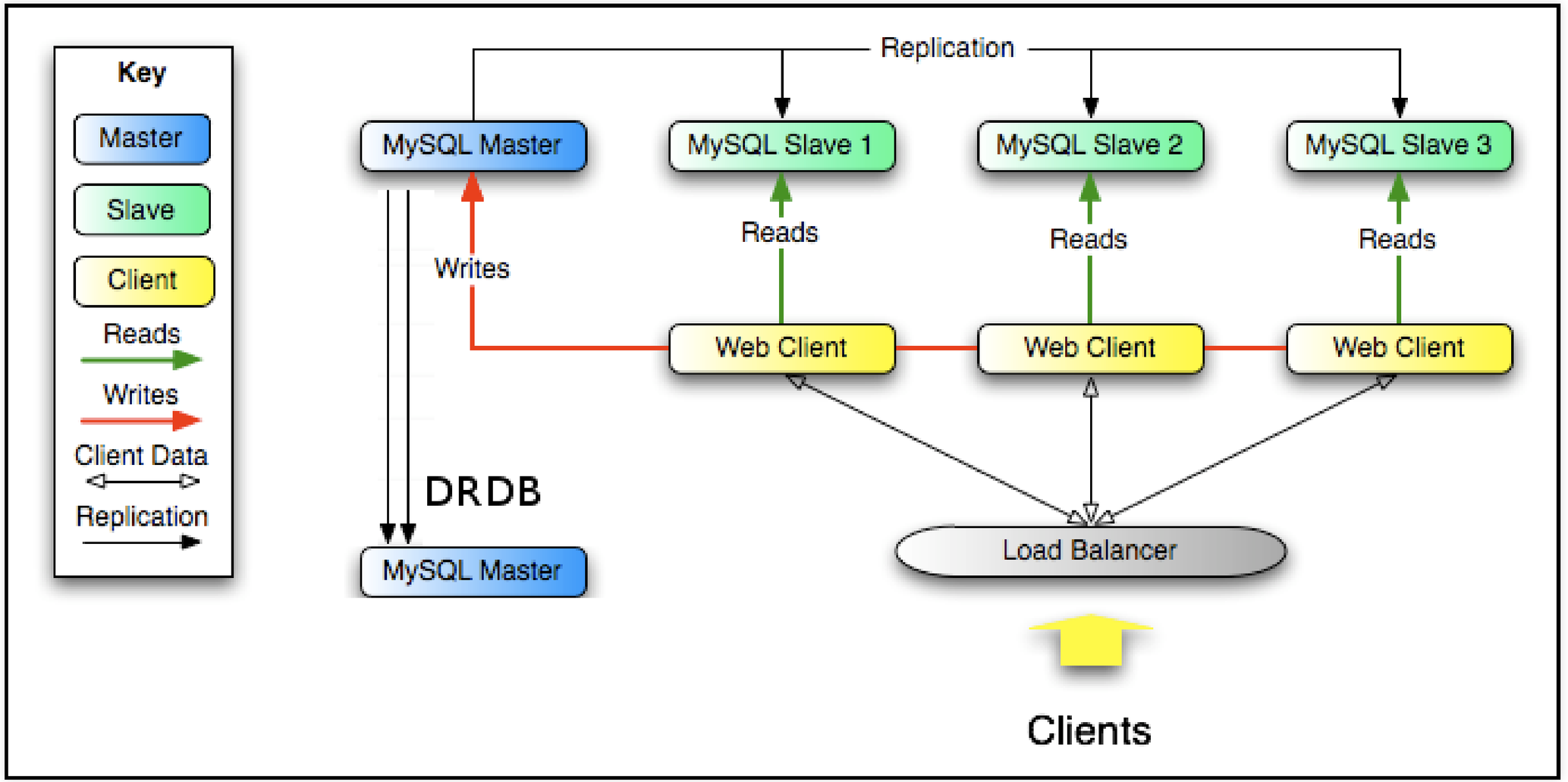
http://momjian.us/main/presentations/Postgres - Books and ressources by Bruce Momjian

Other

Master/Slave

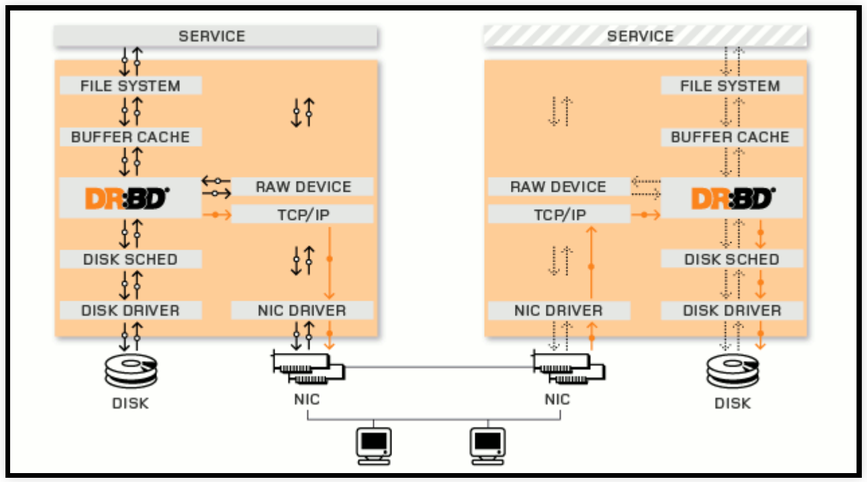


Multi - Master replication with DRBD



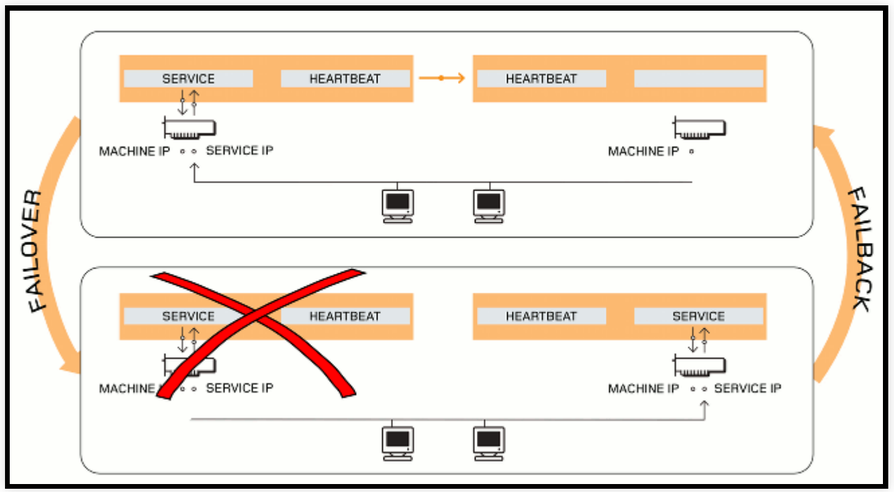
DRBD mirroring

- mirroring a linux partition over IP (sync/async)



DRBD HA

- heartbeat protocol monitors failures
- triggers service switch via IPFOs



more...

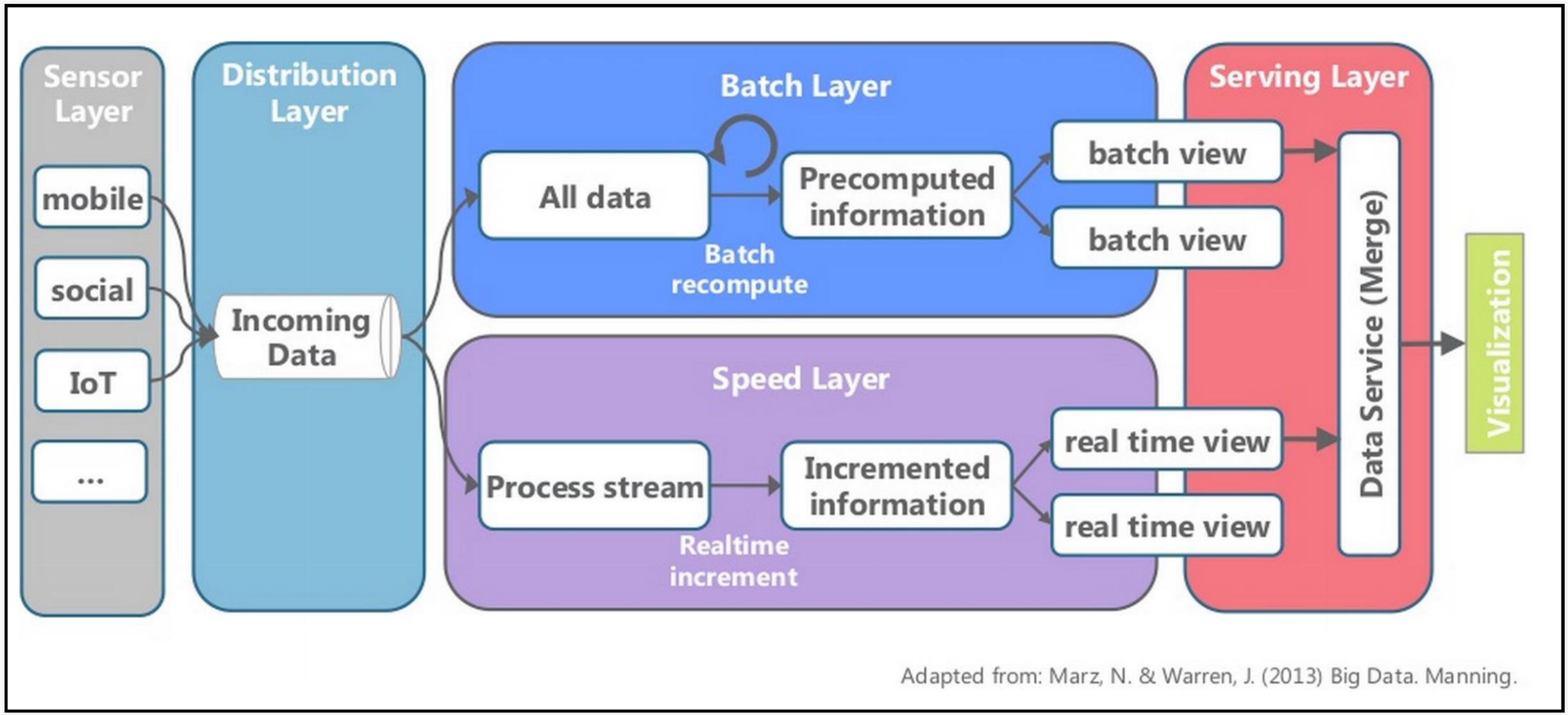
DRBD recovery

- node(s) outage
  - background sync (most up-to date node if both were down)
- replication network outage
  - automatic recovery
- storage subsystem
  - mostly transparent
- network partition
  - ***split brain!*** both nodes switched to the primary role while disconnected
  - ***Manual intervention needed***

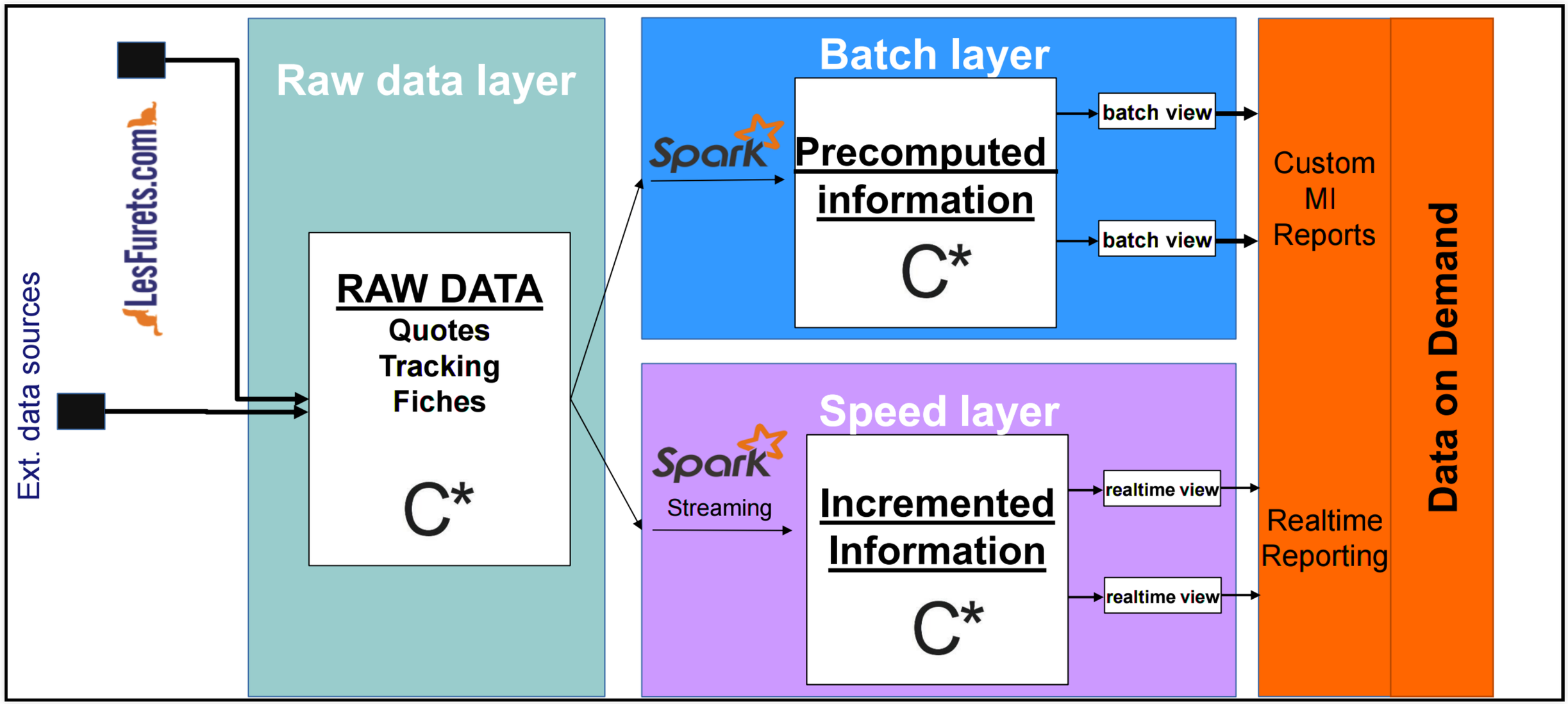
Transition to NoSQL

- NoSQL patterns: **Lambda architecture**
  - scalable systems
  - for arbitrary data problems
  - with human fault tolerance
  - and minimum complexity

Lambda architecture



Lambda architecture @LesFurets



Lambda architecture @LesFurets