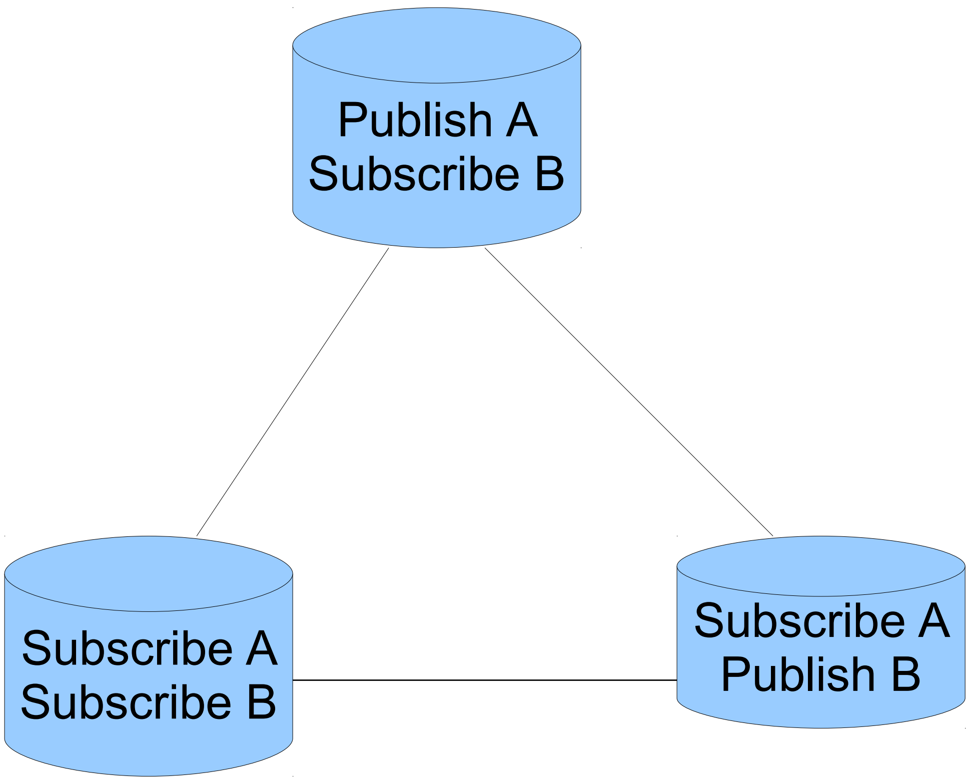
Data replication,
high availability,
load balancing,

Data replication

Publish – subscribe model



High availability – redundancy - primary server, secondary server(s)

Failover, switchover, replication, database cluster, split-brain

Load Balancing

– load balancing database servers is easy for r/o access.

– modification much more complicated – write requests must be propagated to all servers in the cluster – no single solution.

Master servers – R/W requests
Slave servers – R/O requests
Stand-by servers – not accessible until they become master

Some solutions – only one server can handle update requests

Synchronous solutions – the data modifying transaction is not committed until accepted by all servers (distributed transaction) => no data lost on failover

Asynchronous solutions – commit → delay → propagation to other servers in the cluster => some data updates may be lost on failover

If synchronous too slow => asynchronous

Granularity point of view:

per-server high availability solution
per-database
per-table

Types of solution:

Shared-disk failover

- single disk array shared by multiple servers
- if the main DB server fails, another DB server mounts the DB
- synchronization overhead avoided by having just one instance of the database
- rapid failover with no data loss
- SAN (storage area networks) lower level communication protocol than NFS
- NFS is OK if mounted synchronously (without caching)

- the stand-by server cannot never access the DB while the primary server is ON
- disk array – single point of failure, when failed/corrupted, the DB becomes unavailable

File-system Replication

- HW redundancy
- changes to a filesystem mirrored to another filesystem residing on another host
- mirroring must assure that the secondary filesystem is a consistent copy of the primary one
- the order of writes must be preserved

- LINUX world has a popular filesystem mirroring solution called DRBD

Warm stand-by through PITR (point-in-time recovery)

- a stand-by server is being kept current by reading the stream of write-ahead log (WAL) records (warm standby aka log shipping)
- when the master server fails, the stand-by server has almost all data available and can be quickly made the master

- asynchronous solution
- only the entire DB server granularity

Master-slave replication

- all data modification queries go to the master server.
- the master server asynchronously sends data changes to the slave server
  => possible data loss during fail over
- the slave can answer read-only queries while the master server is running.
- the slave server ideal for OLAP queries

Example: Slony-I (per-table granularity, support for multiple slaves)

Statement-based replication middleware

- a program intercepts every SQL query and sends it to one or all servers.
- each server operates independently.
- R/W queries are sent to all servers, while R/O queries can be sent to just one server, allowing the read workload to be distributed.

If queries are simply broadcast unmodified, functions like random(), CURRENT_TIMESTAMP, and sequences would have different values on different servers. If this is unacceptable, either the middleware or the application must query such values from a single server and then modify the write queries using those values before broadcasting them to the servers.
Also, care must be taken that all transactions either commit or abort on all servers, perhaps using two-phase commit.

Examples: Pgpool-II, Sequoia

Synchronous Multimaster Replication

- each server can accept write requests
- modified data is transmitted from the original server to every other server before each transaction commits.
- read requests can be sent to any server.
- sometimes shared disk used to reduce the communication overhead
- no problem with non-deterministic functions like random()
- advantage: any server can accept write requests. Hence, no need to distribute work-loads between masters and slaves

- Heavy write activity can cause excessive locking, leading to poor performance.

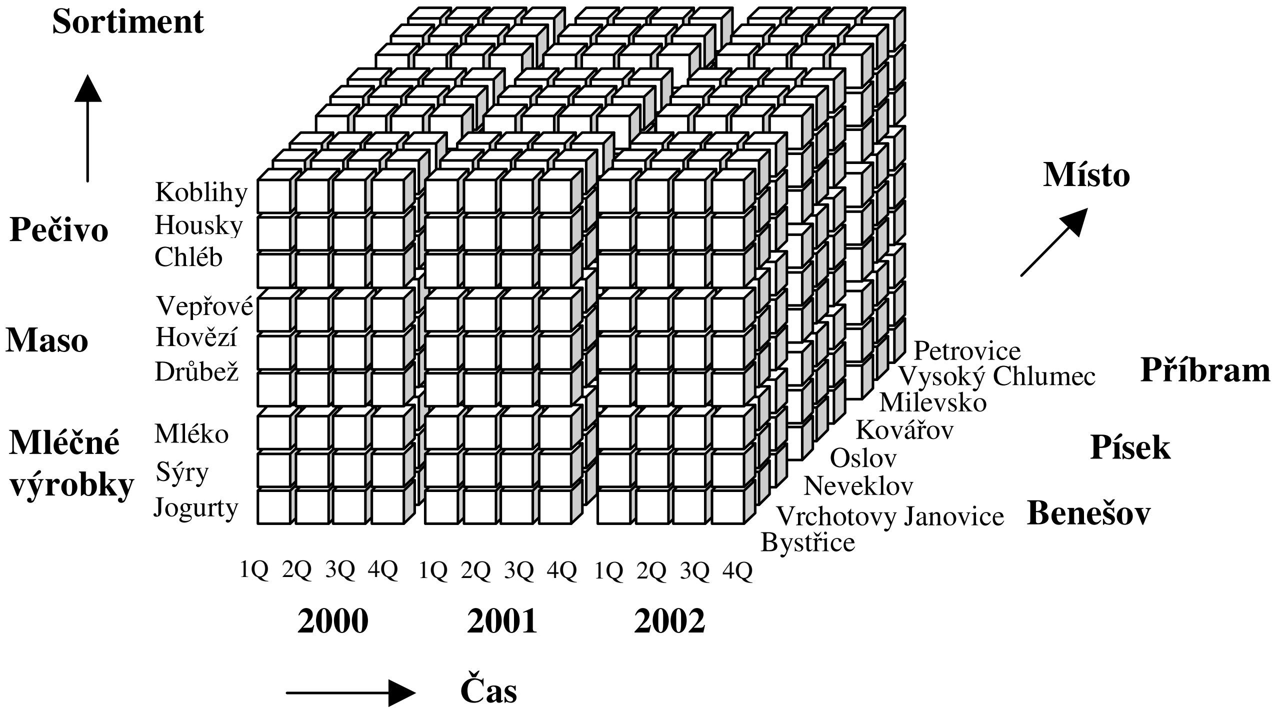
Asynchronous Multimaster Replication

- data synchronization for occasionally connected applications
- asynchronous multimaster replication
- each server works independently
- periodically communicates with the other servers to identify conflicting transactions
- the conflicts resolved by users / conflict resolution rules

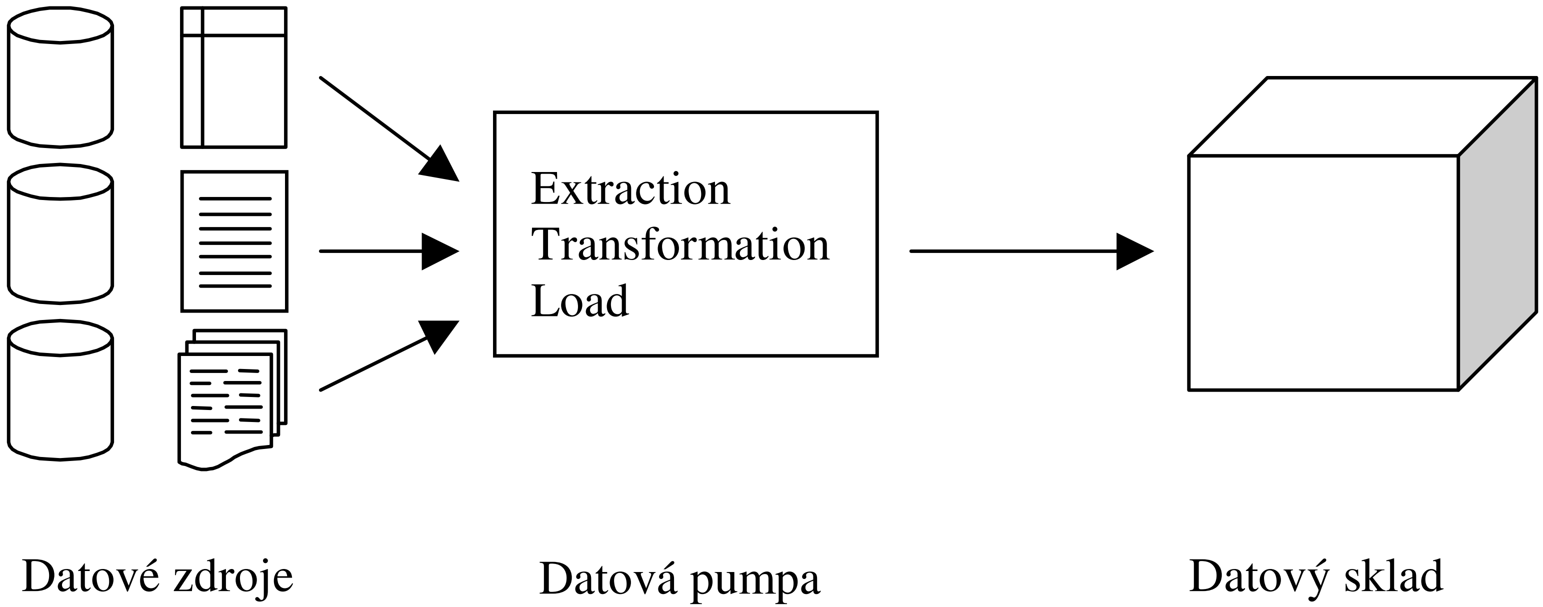
Datové sklady

Zdeněk Kouba

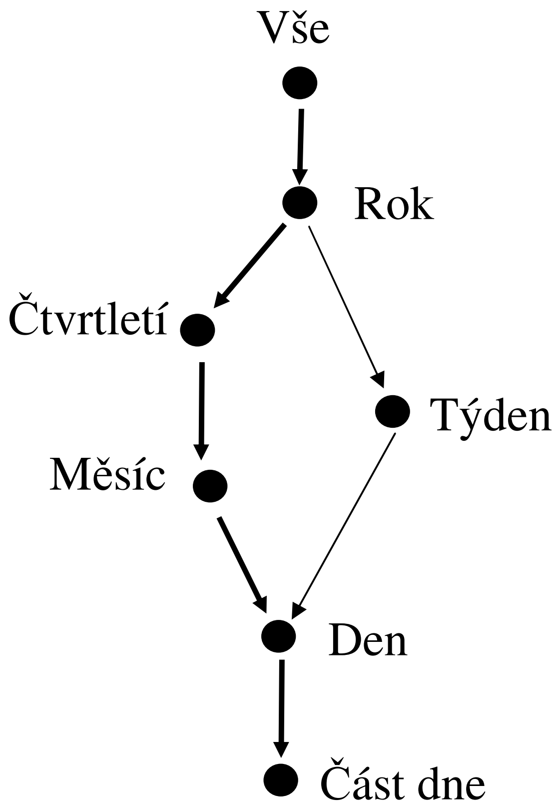
Data cube


Sortiment
Pečivo
Koblihy
Housky
Chléb
Maso
Vepřové
Hovězí
Drůbež
Mléčné výrobky
Mléko
Sýry
Jogurty
1Q 2Q 3Q 4Q 1Q 2Q 3Q 4Q 1Q 2Q 3Q 4Q
2000
2001
2002
Čas
Místo
Petrovice
Vysoký Chlumec
Milevsko
Kovářov
Oslov
Neveklov
Vrchotovy Janovice
Bystřice
Příbram
Písek
Benešov

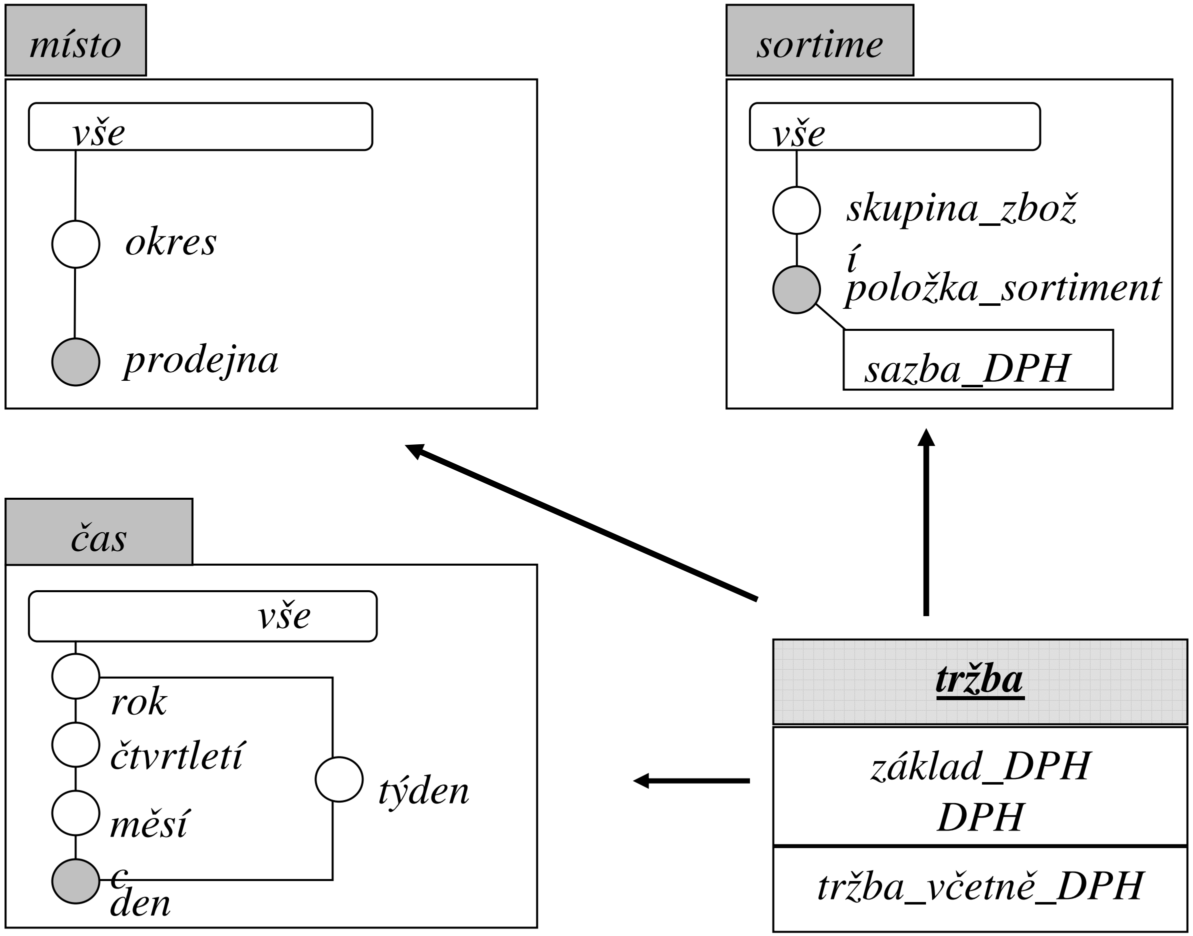
Proces ETL


Extraction
Transformation
Load
Datové zdroje
Datová pumpa
Datový sklad

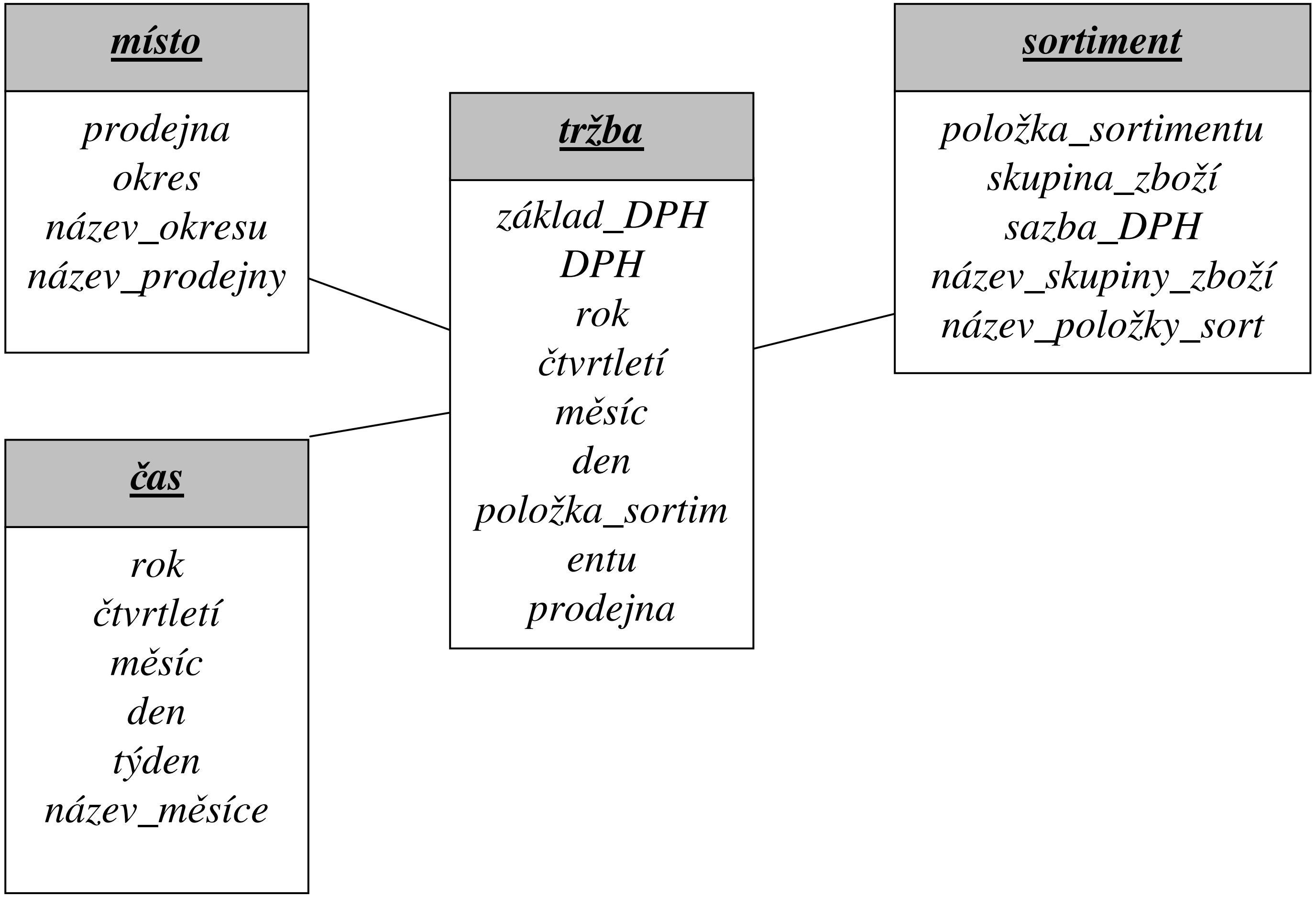
Agregační hierarchie dimenze “čas”



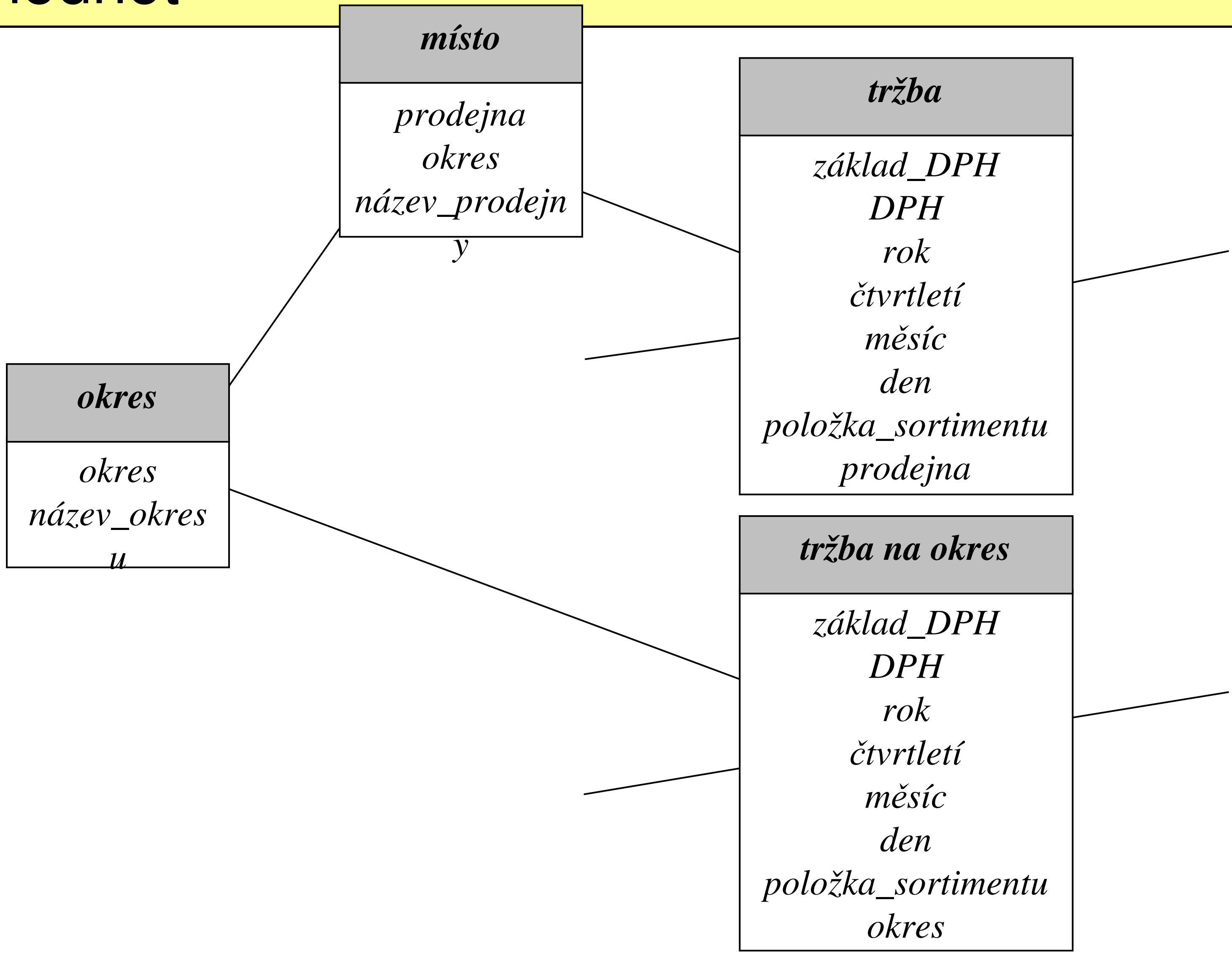
Konceptuální model motivačního příkladu



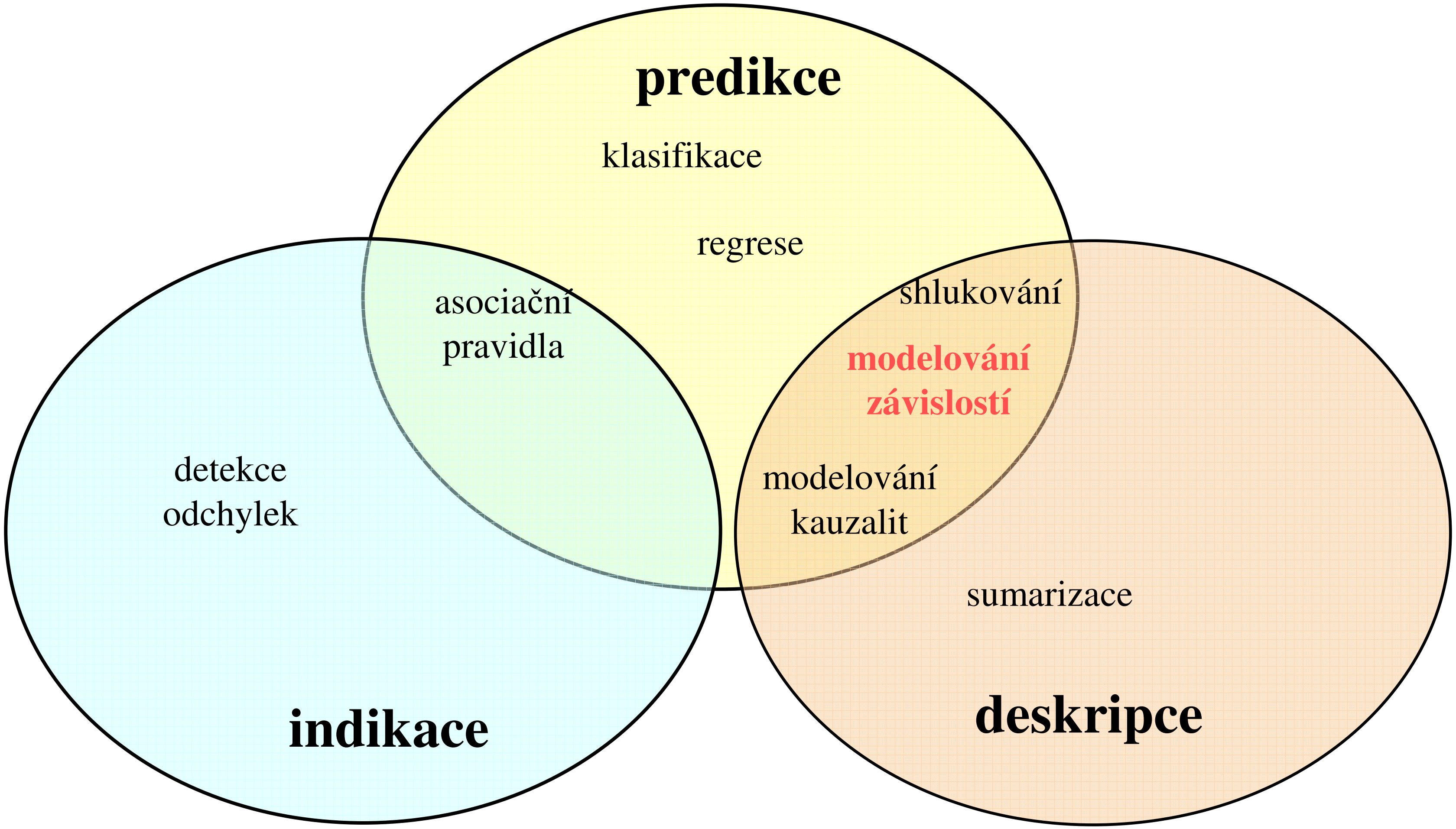
Logický model (*star-schema*) motivačního příkladu



Snowflake schema s tabulkou agregovaných hodnot



Data mining (vytěžování/dolování dat)



Využití nalezených závislosti k predikci

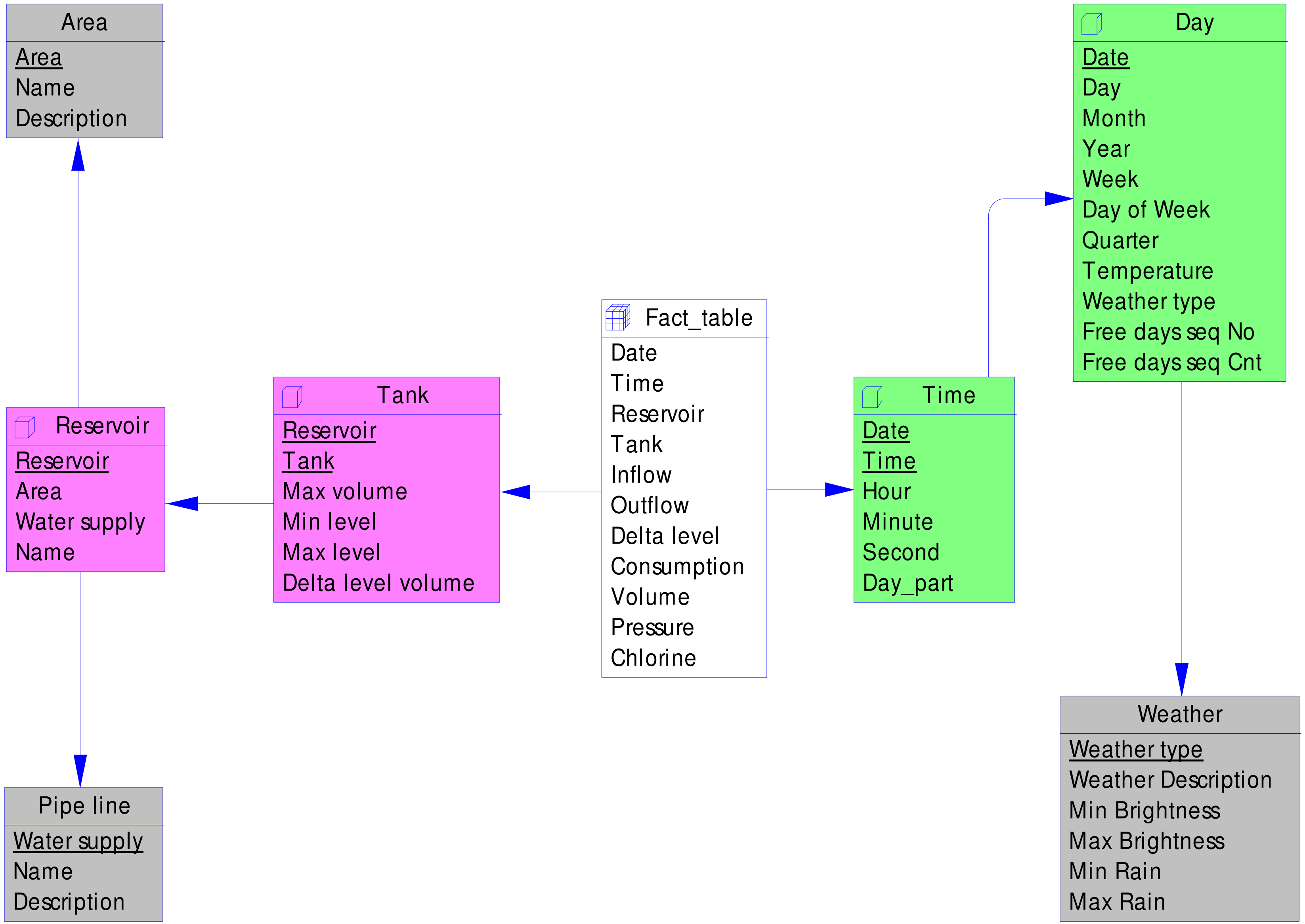
Příklad aplikace datového skladu v predikční úloze:

Predikce spotřeby pitné vody

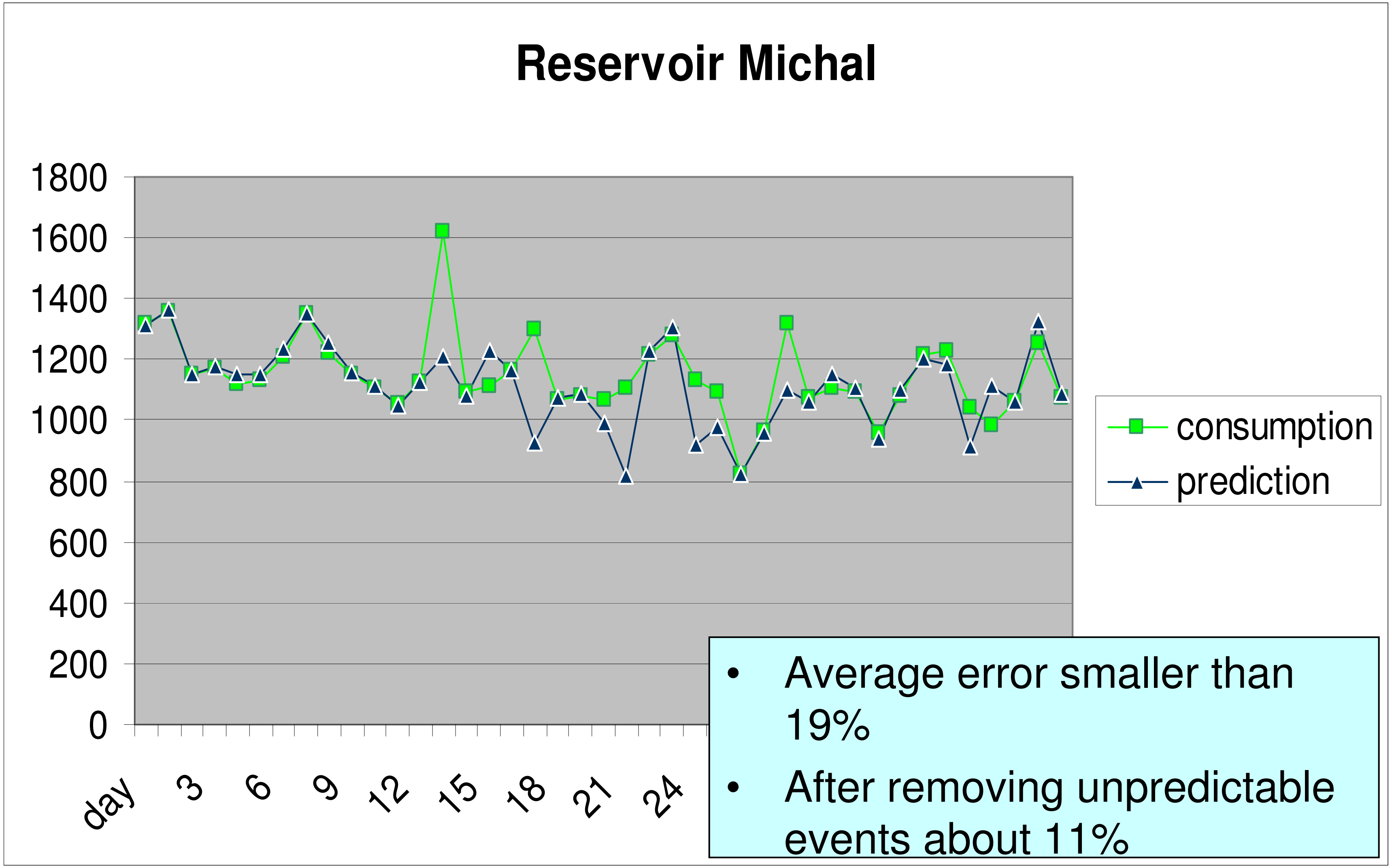
Faktory, které mohou spotřebu ovlivňovat, nalezeny metodami data mining:

- počasí
- den v týdnu
- roční období
- poloha (zahrádkářská kolonie/sídliště)

Star schema datového skladu



Výsledky



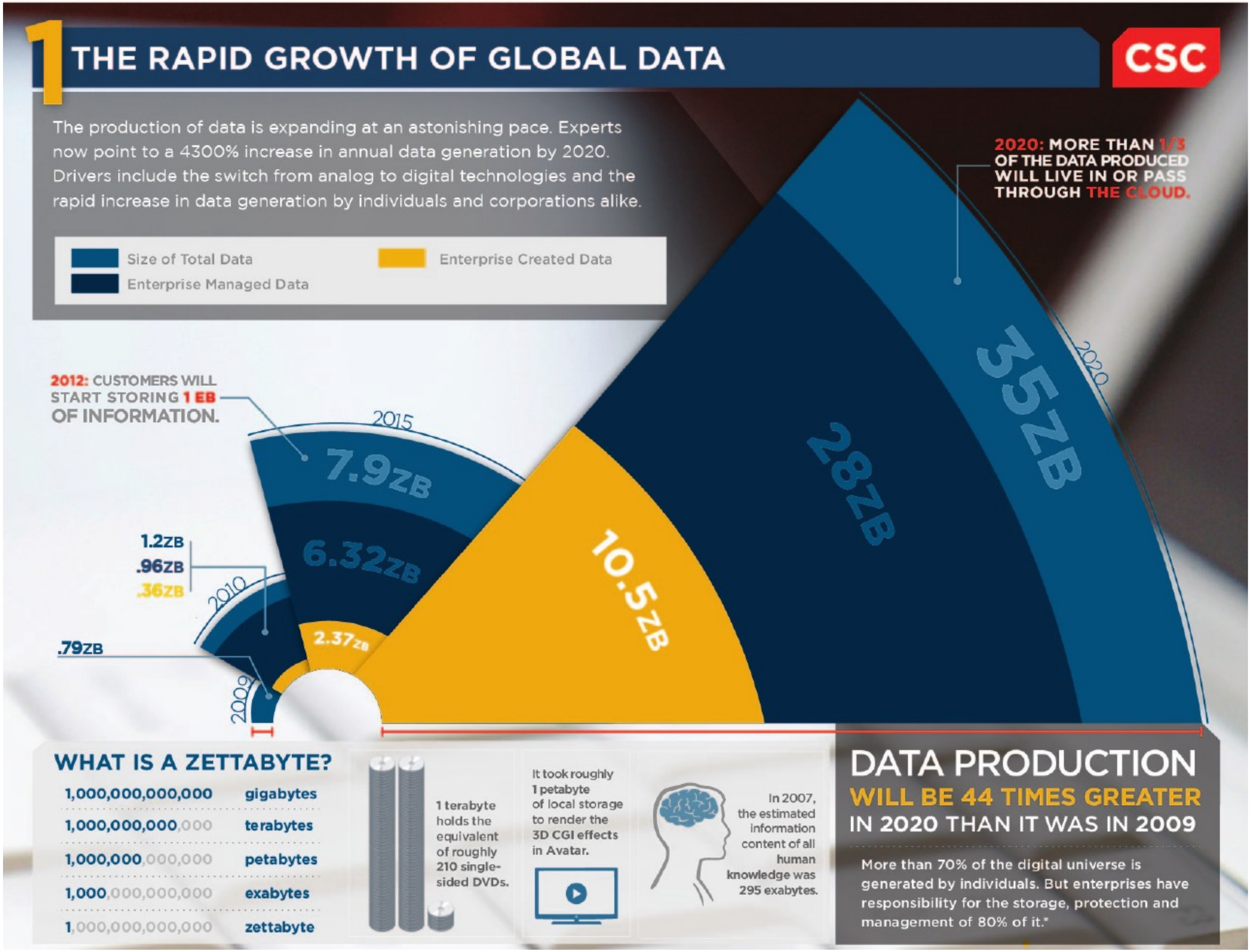
- Average error smaller than 19%
- After removing unpredictable events about 11%

Moderní trendy databázových systémů

Motivace

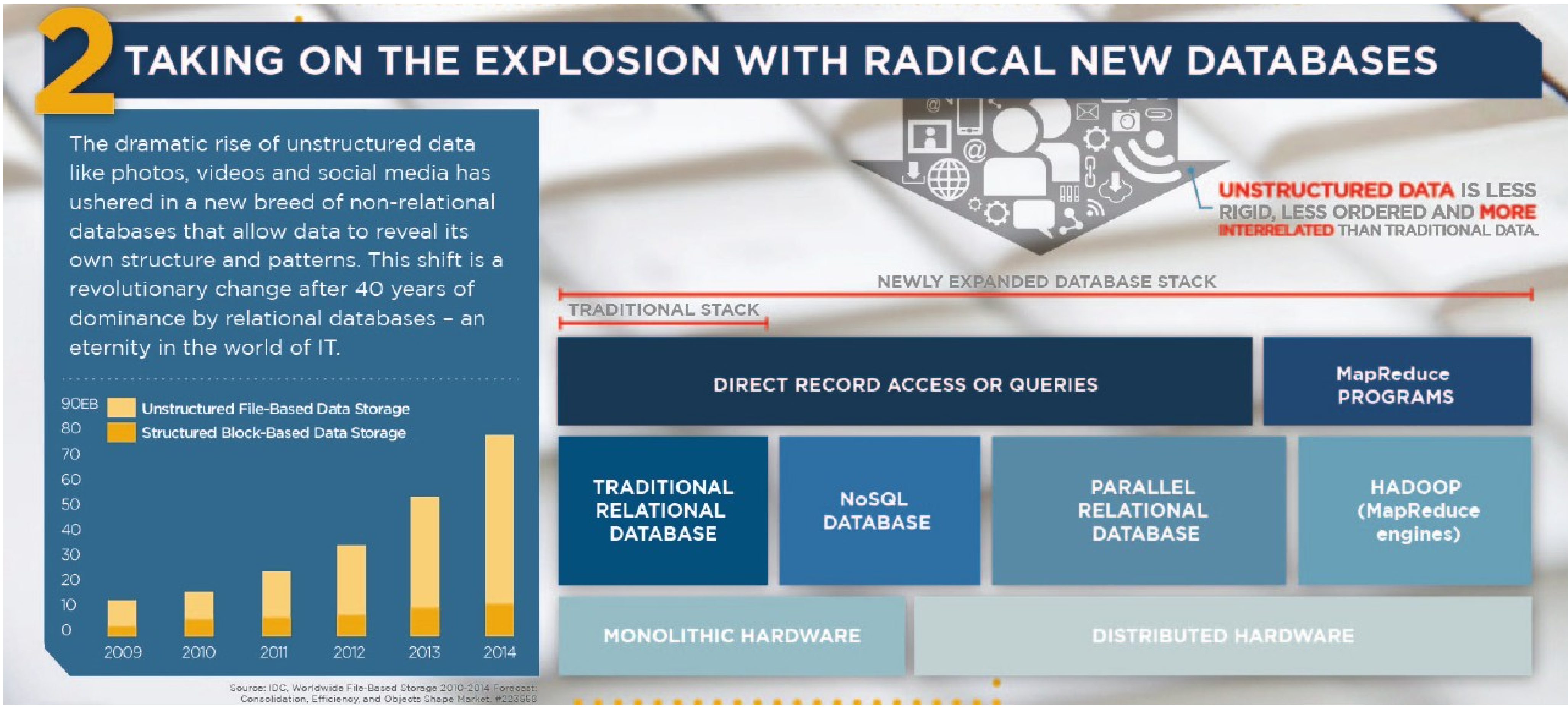
- neplatí, že “one size fits all” (RDBMS/SQL)
- Cloud Computing + Big Data boom
- rapidní růst globálních dat
- potřeba efektivně k datům přistupovat
  - data mining
  - stojové učení
  - umělá inteligence

Motivace



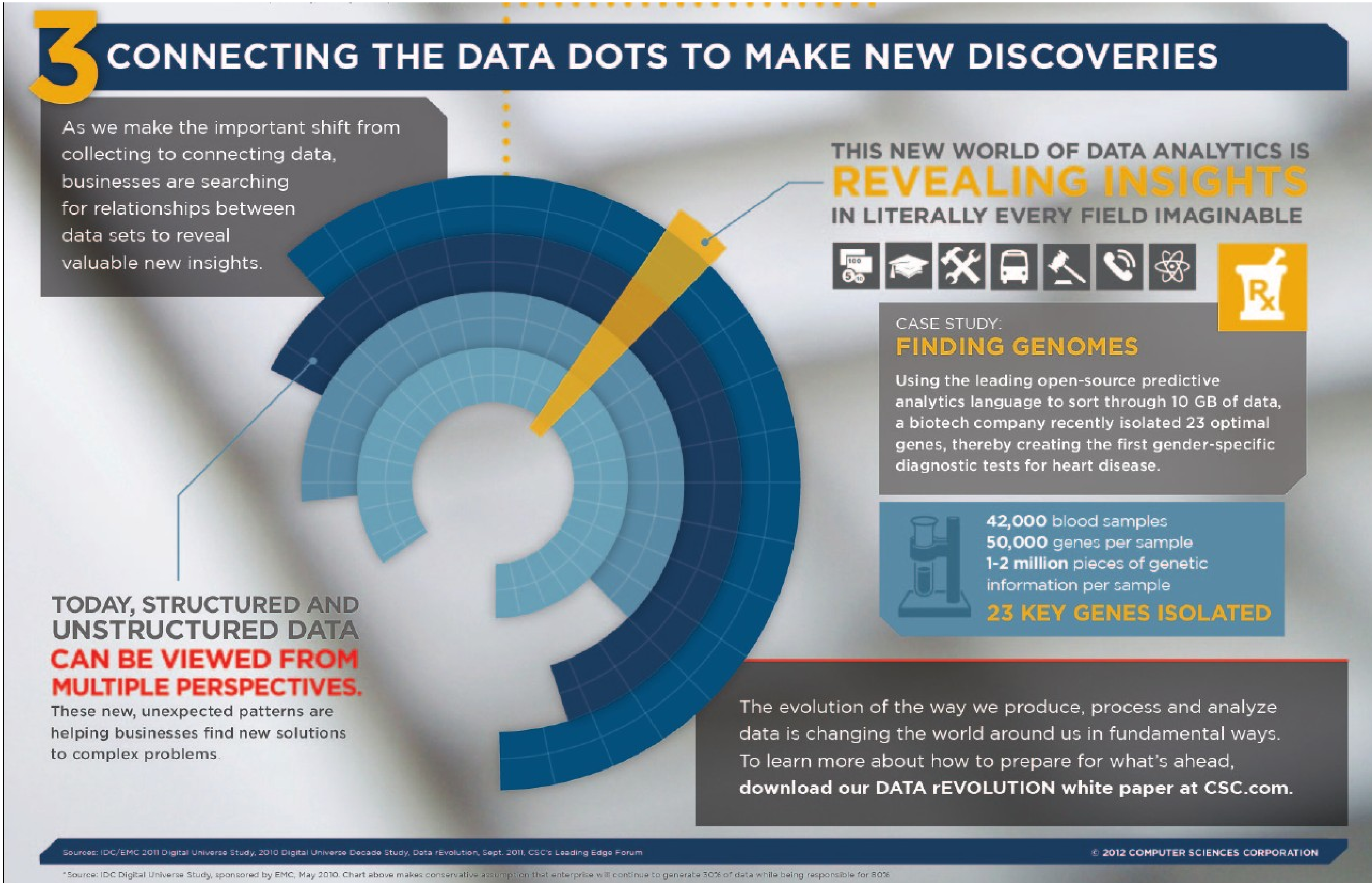
http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

Motivace



http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

Motivace



http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

Přehled

- Tradiční RDBMS [1970 →]
  - Oracle [1980 →]
  - PostgreSQL [1995 →]
  - MySQL [1997 →]
- MapReduce (Hadoop) [2005 →]
- NoSQL (not only SQL) [2007 →]
  - Databáze klíč-hodnota (Redis)
    - Dokumentové databáze (MongoDB, ale i PostgreSQL)
    - Grafové databáze (Neo4j)
    - (Wide) sloupcové databáze (Cassandra)
  - "fifth NoSQL" – částečně strukturované, MultiValue
- Paralelní/distribuované databáze (Datomic)

http://db-engines.com/en/ranking

36DB2

- připravovaný předmet 36DB2 (OI magistr)
  - Big Data a NoSQL databáze
  - Princip MapReduce
  - Databáze klíč-hodnota, dokumentové, sloupcové
  - Grafové databáze
  - NewSQL (Traditional RDBMS + NoSQL)
  - Cloud computing
  - Datové sklady a Big Data