Introduction to NoSQL Databases

Chapter 2

HO
GENT

See Chapter 1:

Big Data Challenges:

- **Volume**: BIG data → horizontal scaling/distributed data
- **Variety**: varied data → schemaless databases
- **Velocity**: fast data → NoACID databases

Classical relational database follow the ACID Rules

- A database transaction must be
  - Atomic: A transaction is a logical unit of work which must be either completed with all of its data modifications or nothing at all
  - Consistent: At the end of the transaction, all data must be left in a consistent state
  - Isolated: Modifications of data performed by a transaction must be independent of another transaction. Otherwise the outcome of a transaction may be erroneous
  - Durable: When the transaction is completed, effects of the modifications performed by the transaction must be permanent in the system

The NoSQL movement

- RDBMSs put a lot of emphasis on keeping data consistent.
  - Entire database is consistent at all times (ACID)
- Focus on consistency may hamper flexibility and scalability
- As the data volume or the number of parallel transactions increases, capacity can be increased by
  - Vertical scaling: extending storage capacity and/or CPU power of the database server
  - Horizontal scaling: multiple DBMS servers being arranged in a cluster

Vertical vs Horizontal scaling

- As you get large amounts, you need to scale things.
  - (1) You can scale things up using bigger boxes
    - It costs a lot and there are real limits as to how far you can go

Lots of Traffic

  - (2) You can use lots and lots of little boxes, just commodity hardware, all thrown into massive grids
    - Relational databases were not designed to run efficiently on clusters. It’s very hard to spread relational databases and run them on clusters

Vertical vs Horizontal scaling

- RDBMSs are not good at extensive horizontal scaling
  - Coordination overhead because of focus on consistency
  - Rigid database schemas
- Other types of DBMSs needed for situations with massive volumes, flexible data structures and where scalability and availability are more important ➔ NoSQL databases

The NoSQL movement

- NoSQL databases
  - Describes databases that store and manipulate data in other formats than tabular relations, i.e. non-relational databases
(*NoREL would be a better name than NoSQL*)
- NoSQL databases aim at near linear horizontal scalability, by distributing data over a cluster of database nodes for the sake of performance as well as availability
- Eventual consistency: the data (and its replicas) will become consistent at some point in time after each transaction

The NoSQL movement

	Relational Databases	**NoSQL Databases**
Data paradigm	Relational tables	Key-value (tuple) based Document based Column based Graph based XML, object based Others: time series, probabilistic, etc.
Distribution	Single-node and distributed	Mainly distributed
Scalability	Vertical scaling, harder to scale horizontally	Easy to scale horizontally, easy data replication
Openness	Closed and open source	Mainly open source
Schema role	Schema-driven	Mainly schema-free or flexible schema
Query language	SQL as query language	No or simple querying facilities, or special-purpose languages
Transaction mechanism	ACID: Atomicity, Consistency, Isolation, Durability	BASE: Basically available, Soft state, Eventual consistency (see further)
Feature set	Many features (triggers, views, stored procedures, etc.)	Simple API
Data volume	Capable of handling normal-sized data sets	Capable of handling huge amounts of data and/or very high frequencies of read/write requests

NoSQL what does it mean

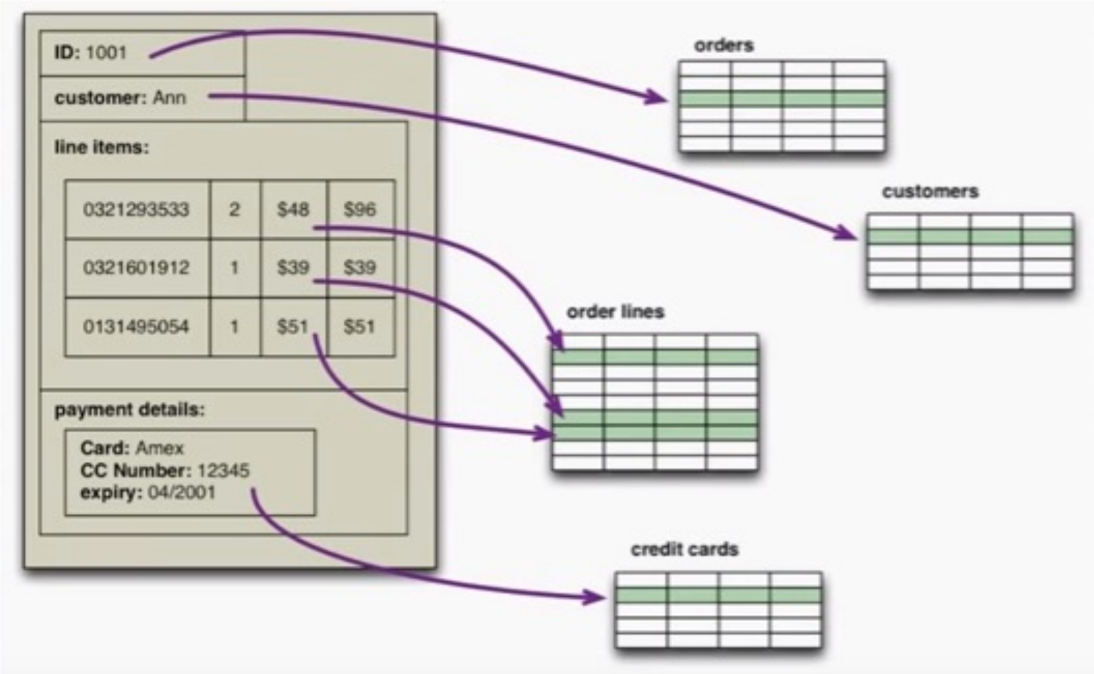
- NoSQL = Not Only SQL = There is more than one storage mechanism that could be used when designing a software solution
- 1998: Carlo Strozzi used the term to name his Open Source, Light Weight database which did not have an SQL interface
- 2009: Eric Evans reused the term as a twitter hashtag (#nosql) for a conference in Atlanta about databases which are non-relational, distributed, and do not conform to atomicity, consistency, isolation, durability

Limitations of NoSQL

- SQL
  - over 40 years old => very mature
  - Switching from 1 relational database to another is much easier than switching between 2 NoSQL databases
- Each NoSQL database has unique aspects
  - The developer must invest time and effort to learn the new query language and the consistency semantics

Impedance mismatch

- Impedance mismatch
  - In software: cohesive structures of objects in memory
  - In databases: you have to stripe the object over multiple tables
- NoSQL databases allow developers to develop without having to convert in-memory structures to relational structures



ID: 1001

customer: Ann

line items:

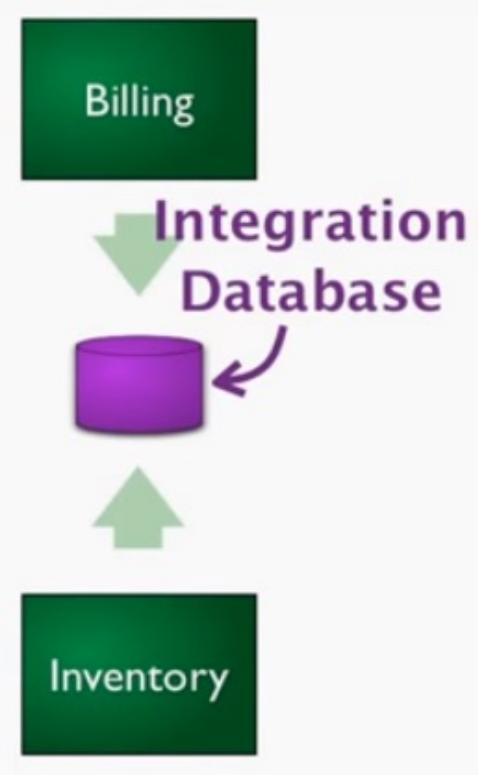
0321293533	2	$48	$96
0321601912	1	$39	$39
0131495054	1	$51	$51

payment details:

Card: Amex
CC Number: 12345
expiry: 04/2001

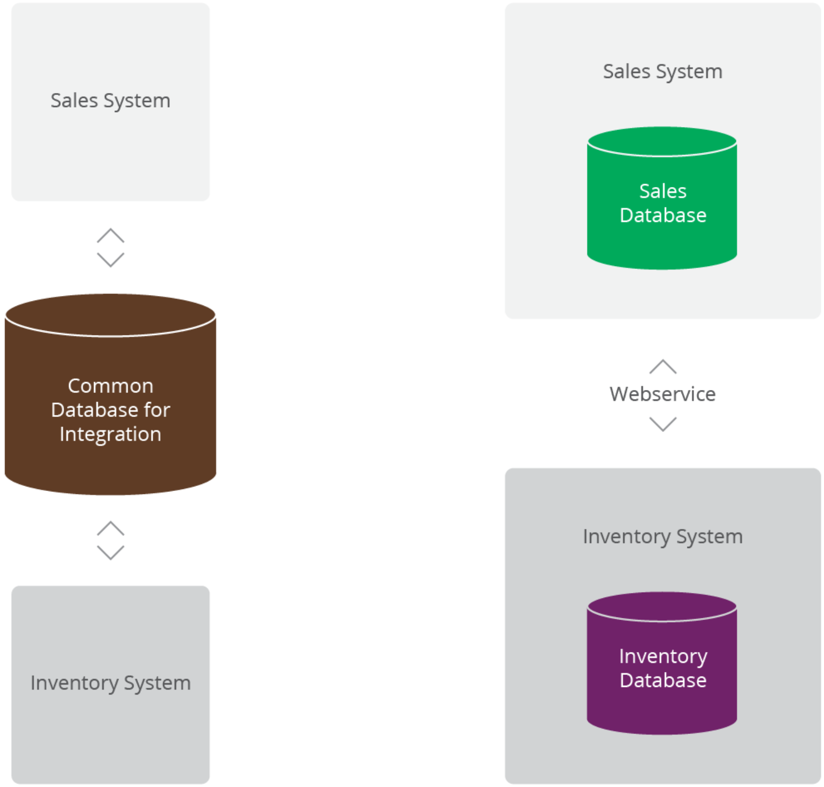
Impedance mismatch

- This impedance mismatch problem led to the fact that in the mid-nineties people said: "We think relational databases are going to go away and object databases will be replacing them. In that way we can take care of memory structures and save them directly to disk without any of this mapping between the two.“
- But this didn’t happen. Why not?
- SQL databases had become an integration mechanism through which people integrated different applications



Impedance mismatch

- Nowadays there is a movement away from using databases as integration points in favor of encapsulating databases using services.


Sales System
Common Database for Integration
Inventory System
Sales System
Sales Database
Webservice
Inventory System
Inventory Database