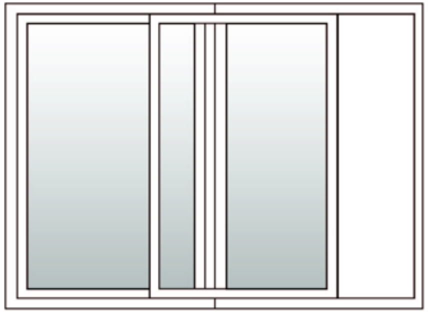
GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

RECOMMENDATION SYSTEMS

DATA SCIENCE IN THE NEWS

LAST TIME:



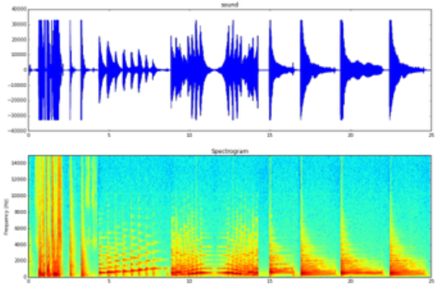
MACHINE LEARNING WITH TIME SERIES

TIME SERIES SIMILARITY

TS GLOSSARY

OTHER TOOLS AND TRICKS

REAL WORLD EXAMPLES



QUESTIONS?

WHAT WAS THE MOST INTERESTING THING YOU LEARNT?

WHAT WAS THE HARDEST TO GRASP?

LECTURE:

I. DATABASE EVOLUTION

II. THE NOSQL MOVEMENT

III. WORKING WITH STRUCTURED DATA (MYSQL, SQLITE)

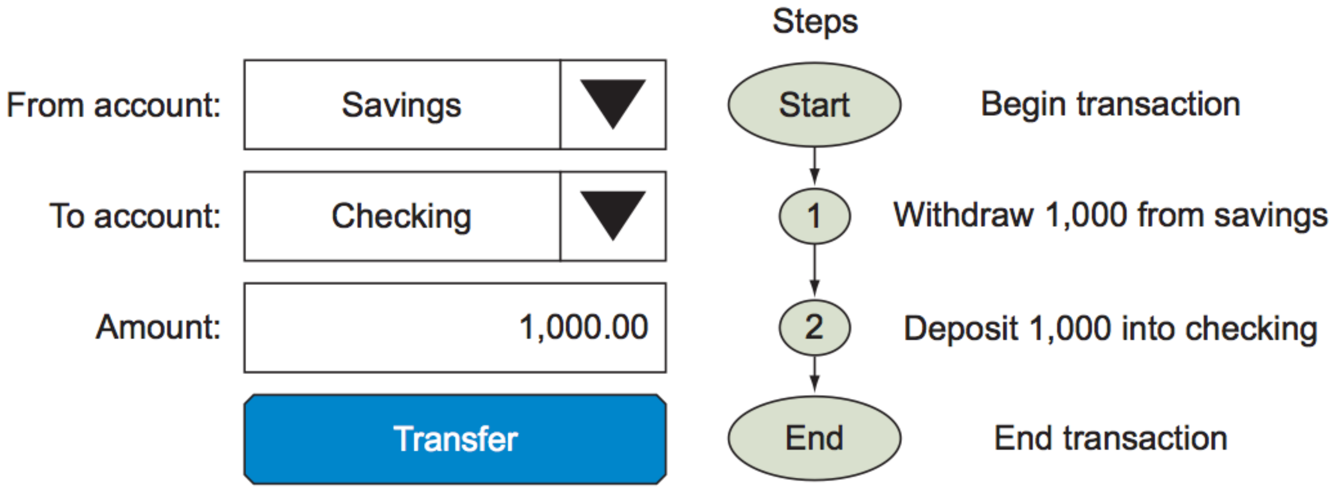
LAB: SQL (SQLITE)

KEY OBJECTIVES

- KNOW HISTORY OF DB TECHNOLOGY
- UNDERSTAND ACID AND BASE CONCEPTS
- BE ABLE TO EXPLAIN DIFFERENCE BETWEEN RELATIONAL AND NON-RELATIONAL
- BE ABLE TO RETRIEVE DATA FROM DATABASES

I. THE EVOLUTION OF DATABASE TECHNOLOGY

What is transactional integrity? A motivating example:



From account: Savings

To account: Checking

Amount: 1,000.00

Steps

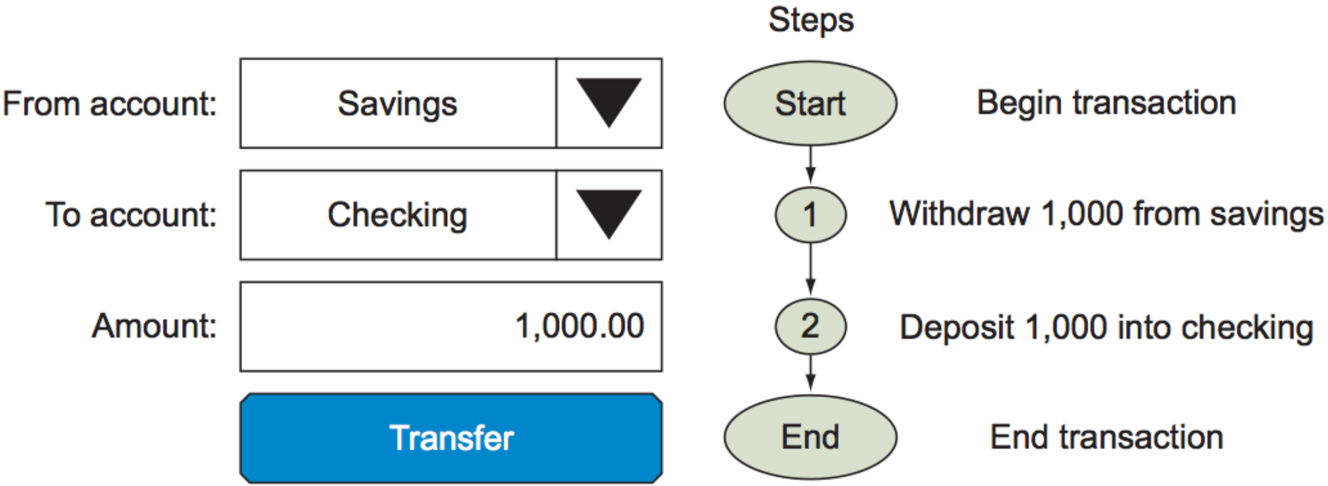
Start — Begin transaction

1 — Withdraw 1,000 from savings

2 — Deposit 1,000 into checking

End — End transaction

What happens if step 1 succeeds and step 2 fails?



From account: Savings ▼

To account: Checking ▼

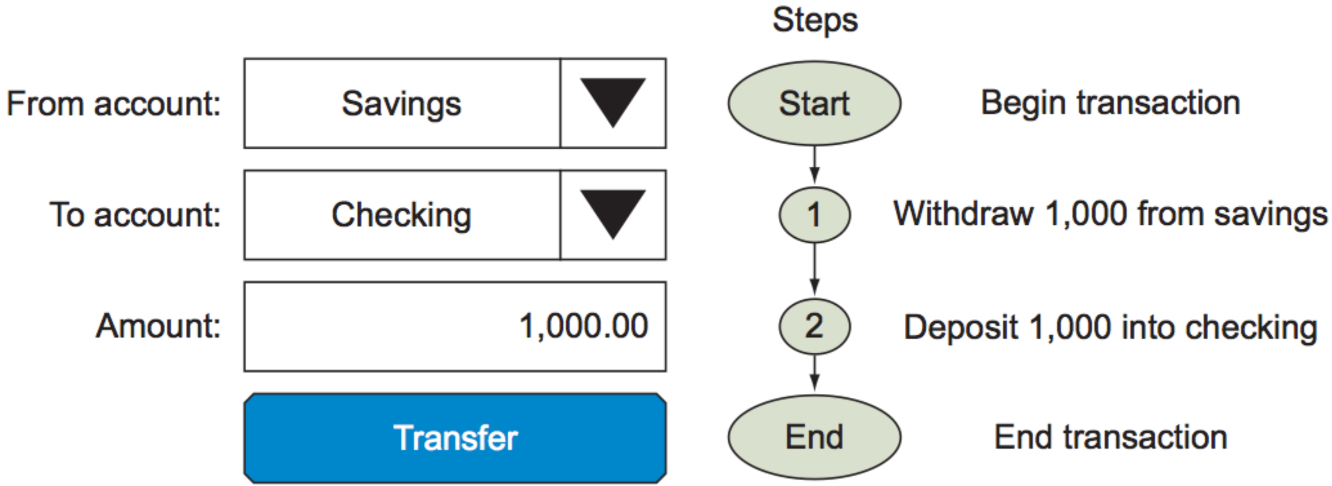
Amount: 1,000.00

Transfer

Steps

- Start: Begin transaction
- 1: Withdraw 1,000 from savings
- 2: Deposit 1,000 into checking
- End: End transaction

What if you request your balance between step 1 and step2?



From account: Savings

To account: Checking

Amount: 1,000.00

Steps

Start — Begin transaction

1 — Withdraw 1,000 from savings

2 — Deposit 1,000 into checking

End — End transaction

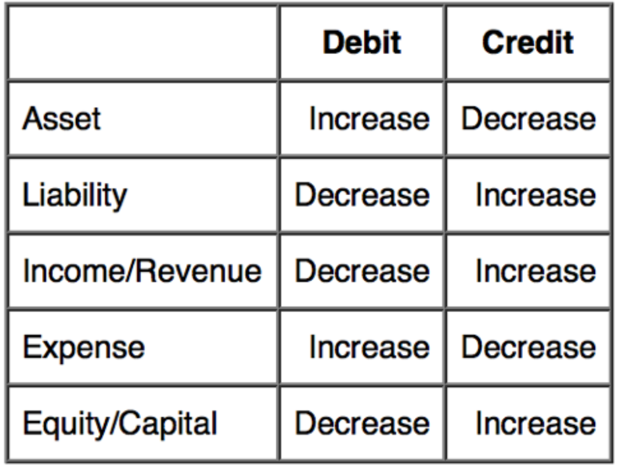
Transaction concepts:

- *Transaction* start a transaction - whatever intended action. If don't succeed until the end, rollback to the beginning, so no harm is done.
- *Begin / end transaction*
- *Rollback*

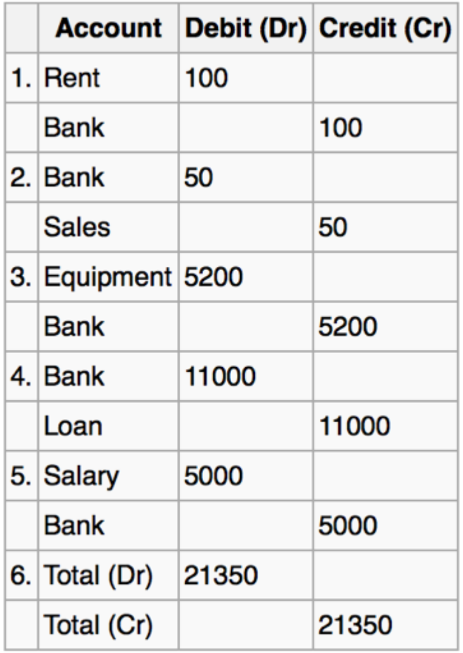
TRANSACTIONAL INTEGRITY

What other types of business activities can you think of that would be “transactions” as defined here…?

What other types of business activities can you think of that would be “transactions” as defined here…?

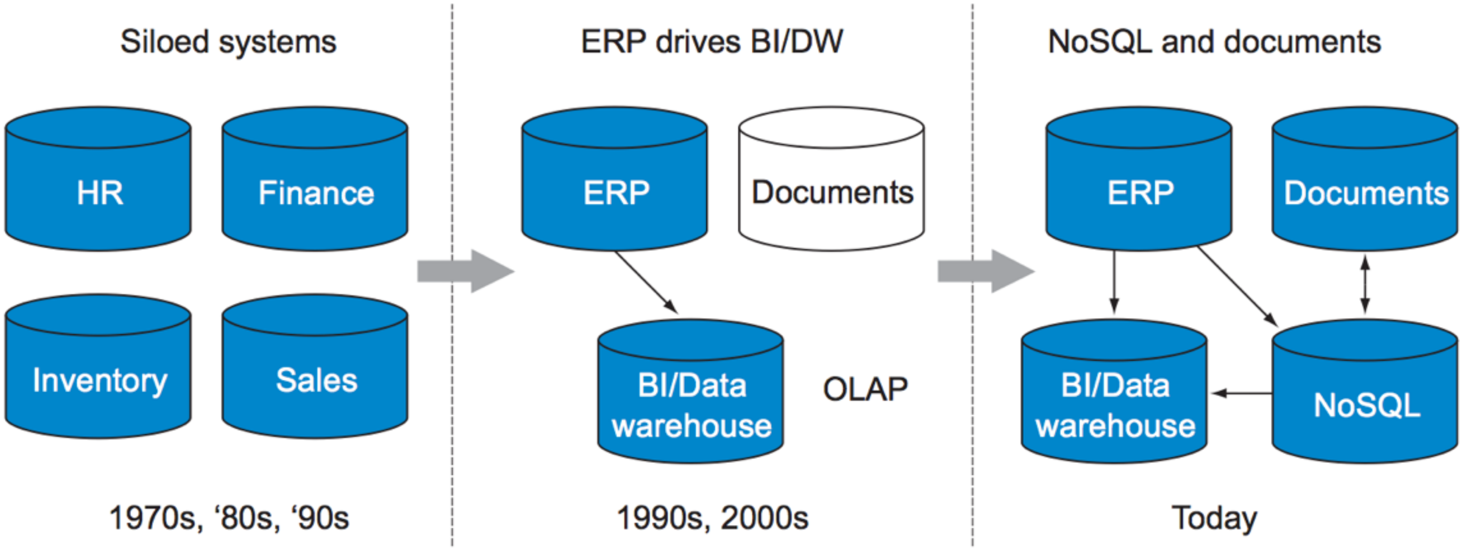


	Debit	Credit
Asset	Increase	Decrease
Liability	Decrease	Increase
Income/Revenue	Decrease	Increase
Expense	Increase	Decrease
Equity/Capital	Decrease	Increase



	Account	Debit (Dr)	Credit (Cr)
1.	Rent	100	
	Bank		100
2.	Bank	50	
	Sales		50
3.	Equipment	5200	
	Bank		5200
4.	Bank	11000	
	Loan		11000
5.	Salary	5000	
	Bank		5000
6.	Total (Dr)	21350	
	Total (Cr)		21350

That's why enterprise resource planning (ERP) systems and relational database management systems (RDBMS) grew up together.



- 1960s
  - Hierarchical data structure (IBM IMS)
  - Network data structure (CODASYL)
- 1970s
  - Relational data model
    - *A Relational Model of Data for Large Shared Data Banks* – E. F. Codd [1970]
  - System R (IBM), Ingres (Berkeley)

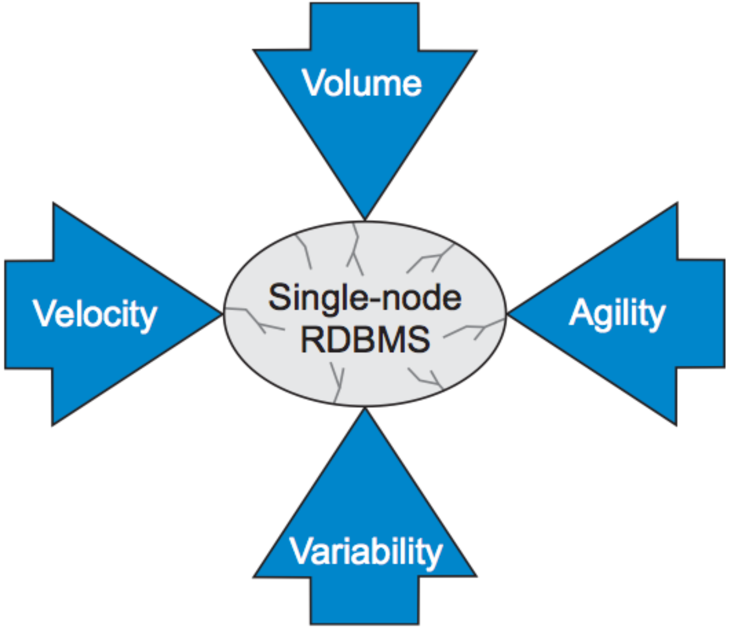
DATABASE EVOLUTION

- 1980s
  - Commercialization of RDBMS
    - Oracle, Sybase, IBM DB2, Informix
  - SQL
  - ACID (**A**tomic, **C**onsistent, **I**solated, **D**urable)
- 1990s
  - PC RDBMS
    - Paradox, Microsoft SQL Server & Access
  - Larger DBs, driven by internet
  - Consolidation among commercial DB vendors

- 2000s
  - Commercialization of Open Source RDBMS
    - MySQL, Postgres
  - Evolving requirements expose RDBMS limitations
    - Storing complex and dynamic objects
    - Processing increasing data volumes
    - Analyzing massive amounts of data

Business drivers for NoSQL include:

- Volume
- Velocity
- Variability
- Agility



SQL doesn't do well with data with these characteristics

Business drivers for NoSQL include:

- Volume - the ability to query big data using clusters of commodity processors (horizontal scaling, parallel processing)
- Velocity - the ability to maintain performance in the face of traffic bursts from public-facing websites
- Variability - the ease of capturing & reporting on exception data
- Agility - object-relational mapping is complicated; even small changes can substantially slow development projects

II. THE NOSQL MOVEMENT

Mathematical theorem - given CAP - you can only have 2 for a distributed system. Mathematically, you can't have all 3 for distributed system.

Eric Brewer's CAP (**C**onsistency, **A**vailability, **P**artition Tolerance) Theorem [2000]

- For a distributed system -> Pick 2!

Research

- MapReduce: Simplified Data Processing on Large Clusters – Google [2004]
- Bigtable: A Distributed Storage System for Structured Data – Google [2006]
- Dynamo: Amazon's Highly Available Key-value Store – Werner Vogels, et. al. [2007]
- Pregel: A System for Large-Scale Graph Processing – Google [2010]

BASE (**B**asic **A**vailability, **S**oft-state, **E**ventually Consistent)

THE NOSQL MOVEMENT

Vs.

Acid

Defensive

- Get transaction details right
- Block any reports while you are working

  e.g., asking for balance while transferring $
- Be pessimistic: anything might go wrong!

  Rollback if there's failure
- Detailed testing and failure mode analysis
- Lots of locks and unlocks

Whenever DB is being written to, can't have something else happening.
Do one operation, release lock, then do next operation.

Base

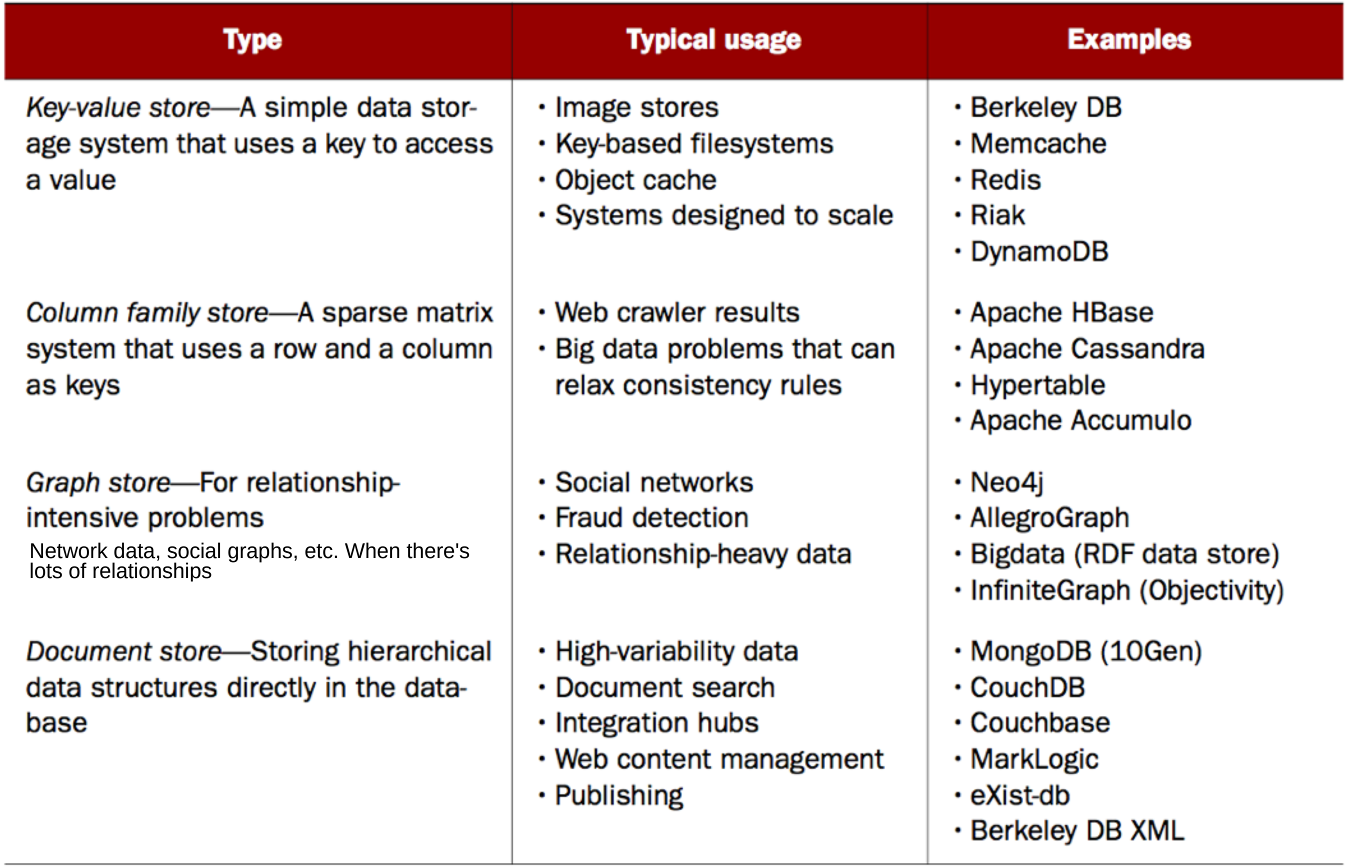
- Never block a write

  Everyone can write to DB
- Focus on throughput, not consistency

  Be fast!
- Be optimistic: if one service fails it will eventually get caught up

  Maybe a temp error, but consistency will be restored
- Some reports may be inconsistent for a while, but don't worry
- Keep things simple and avoid locks

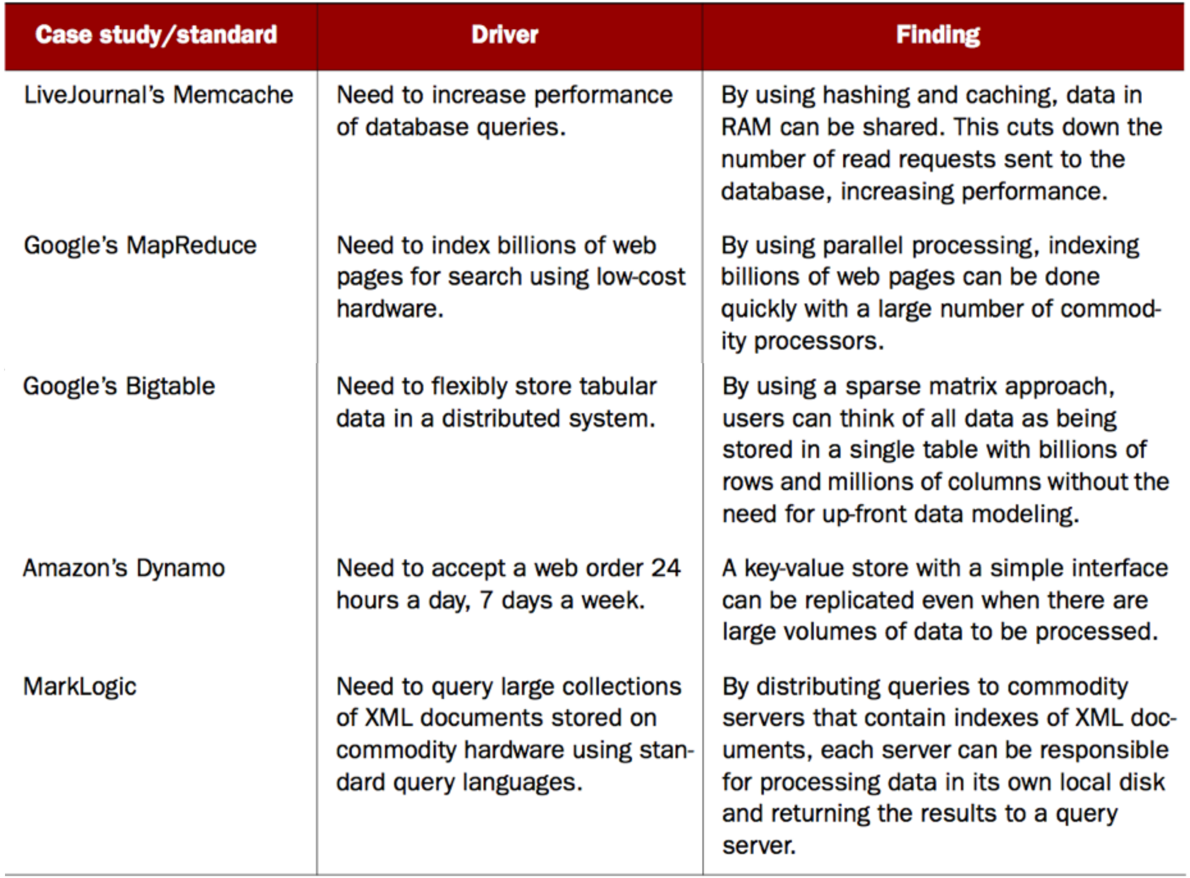
THE NOSQL MOVEMENT



Type	Typical usage	Examples
Key-value store—A simple data storage system that uses a key to access a value	• Image stores • Key-based filesystems • Object cache • Systems designed to scale	• Berkeley DB • Memcache • Redis • Riak • DynamoDB
Column family store—A sparse matrix system that uses a row and a column as keys	• Web crawler results • Big data problems that can relax consistency rules	• Apache HBase • Apache Cassandra • Hypertable • Apache Accumulo
Graph store—For relationship-intensive problems Network data, social graphs, etc. When there's lots of relationships	• Social networks • Fraud detection • Relationship-heavy data	• Neo4j • AllegroGraph • Bigdata (RDF data store) • InfiniteGraph (Objectivity)
Document store—Storing hierarchical data structures directly in the database	• High-variability data • Document search • Integration hubs • Web content management • Publishing	• MongoDB (10Gen) • CouchDB • Couchbase • MarkLogic • eXist-db • Berkeley DB XML

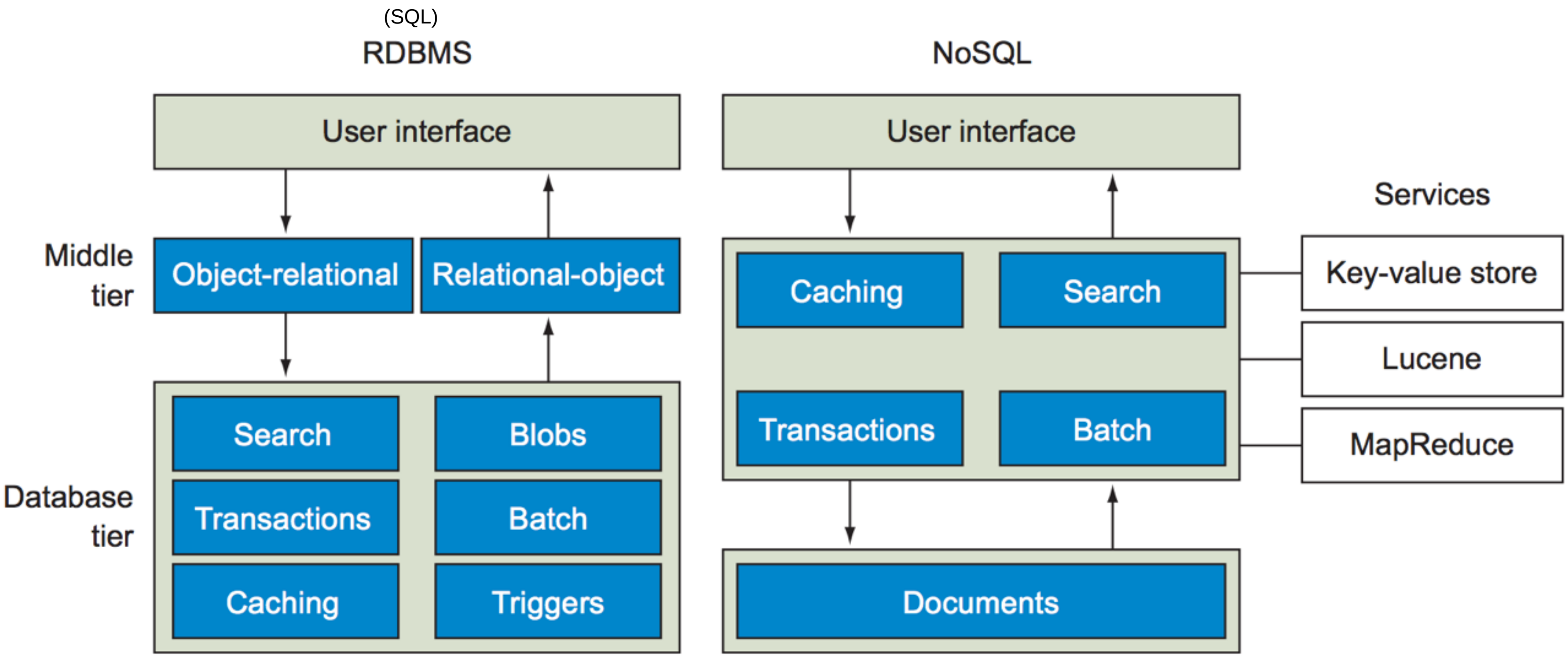
- Key-value
    - memcached, Redis, Riak, Tokyo Cabinet, Voldemort, Amazon SimpleDB
- Column-oriented (Bigtable clones)
    - Cassandra, HBase
- Document-oriented
    - MongoDB, CouchDB
- Graph
    - Neo4J, FlockDB, OrientDB, Pregel (Google)

THE NOSQL MOVEMENT

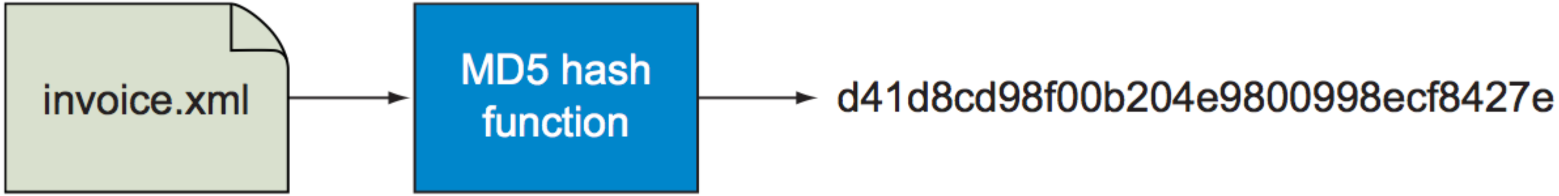


Case study/standard	Driver	Finding
LiveJournal's Memcache	Need to increase performance of database queries.	By using hashing and caching, data in RAM can be shared. This cuts down the number of read requests sent to the database, increasing performance.
Google's MapReduce	Need to index billions of web pages for search using low-cost hardware.	By using parallel processing, indexing billions of web pages can be done quickly with a large number of commodity processors.
Google's Bigtable	Need to flexibly store tabular data in a distributed system.	By using a sparse matrix approach, users can think of all data as being stored in a single table with billions of rows and millions of columns without the need for up-front data modeling.
Amazon's Dynamo	Need to accept a web order 24 hours a day, 7 days a week.	A key-value store with a simple interface can be replicated even when there are large volumes of data to be processed.
MarkLogic	Need to query large collections of XML documents stored on commodity hardware using standard query languages.	By distributing queries to commodity servers that contain indexes of XML documents, each server can be responsible for processing data in its own local disk and returning the results to a query server.

THE NOSQL MOVEMENT - APPLICATION TIERS

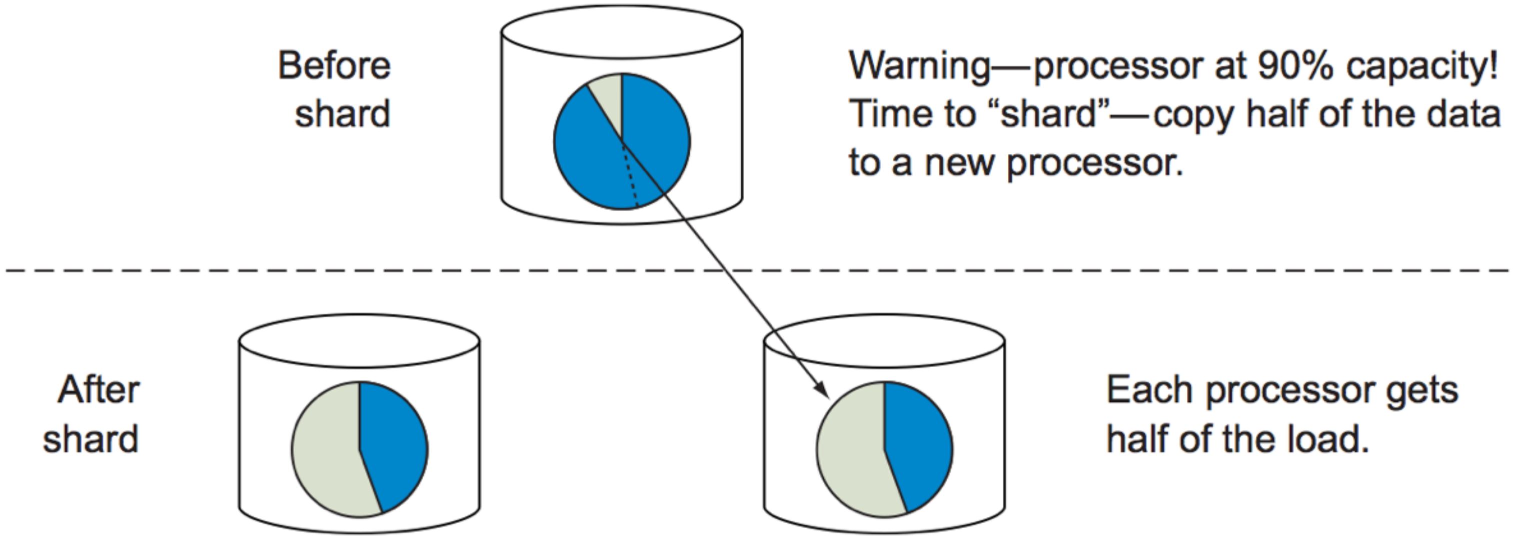


CONSISTENT HASHING



```
let $hash := hash($invoice, 'md5')
```

SHARDING


Before shard

Warning—processor at 90% capacity! Time to “shard”—copy half of the data to a new processor.

After shard

Each processor gets half of the load.

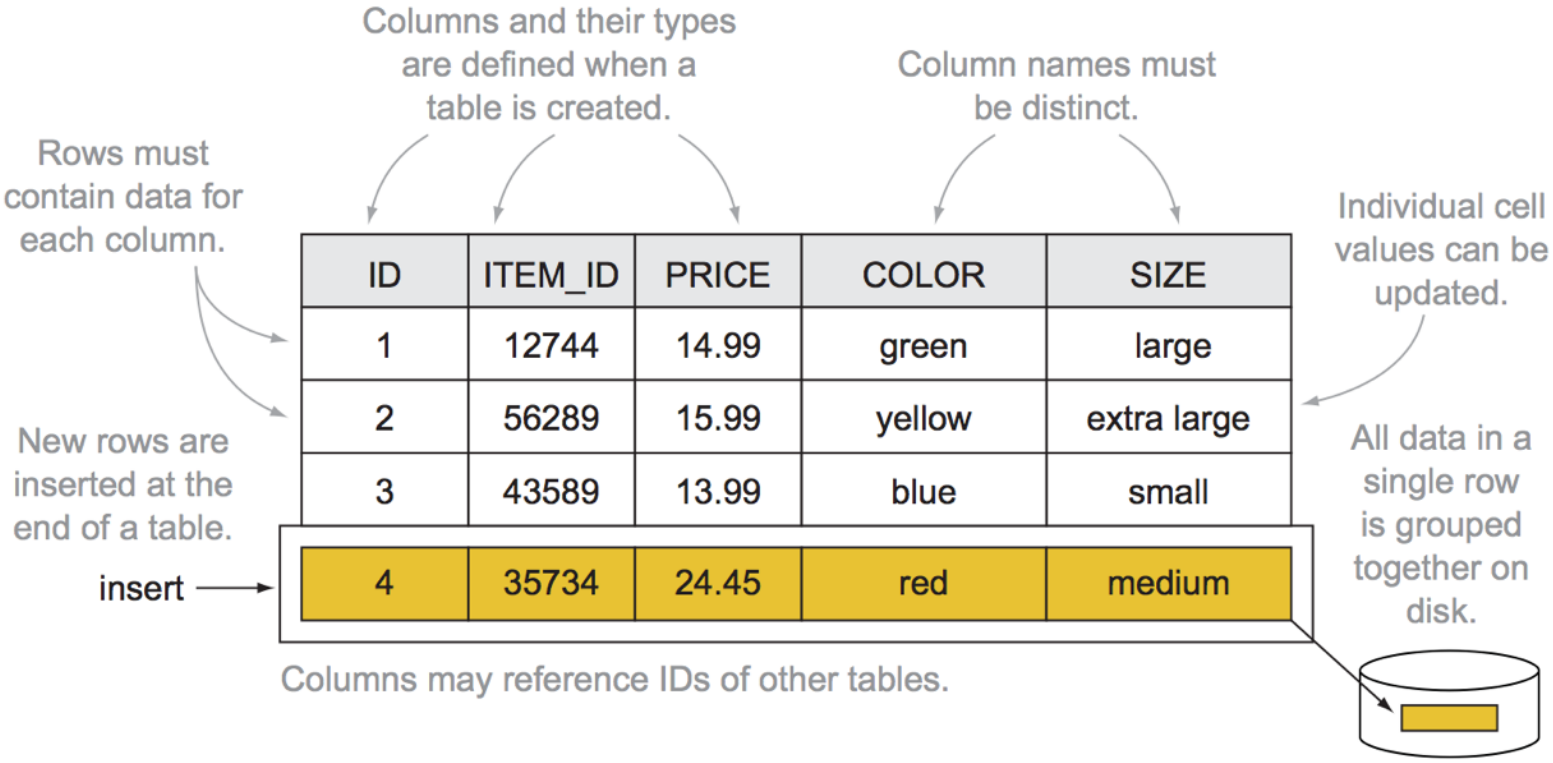
Original processor

New processor

Amazon - SQL
Yelp - SQL
Reddit - noSQL
LinkedIn, FB - noSQL
Some companies use both SQL, noSQL
Twitter used to use key-value stored front end, then diff databases.
As it gets more complicated - can use diff systems.

III. WORKING WITH STRUCTURED DATA (MYSQL, SQLITE)

ROW STORES



Columns and their types are defined when a table is created.

Column names must be distinct.

Rows must contain data for each column.

Individual cell values can be updated.

ID	ITEM_ID	PRICE	COLOR	SIZE
1	12744	14.99	green	large
2	56289	15.99	yellow	extra large
3	43589	13.99	blue	small
4	35734	24.45	red	medium

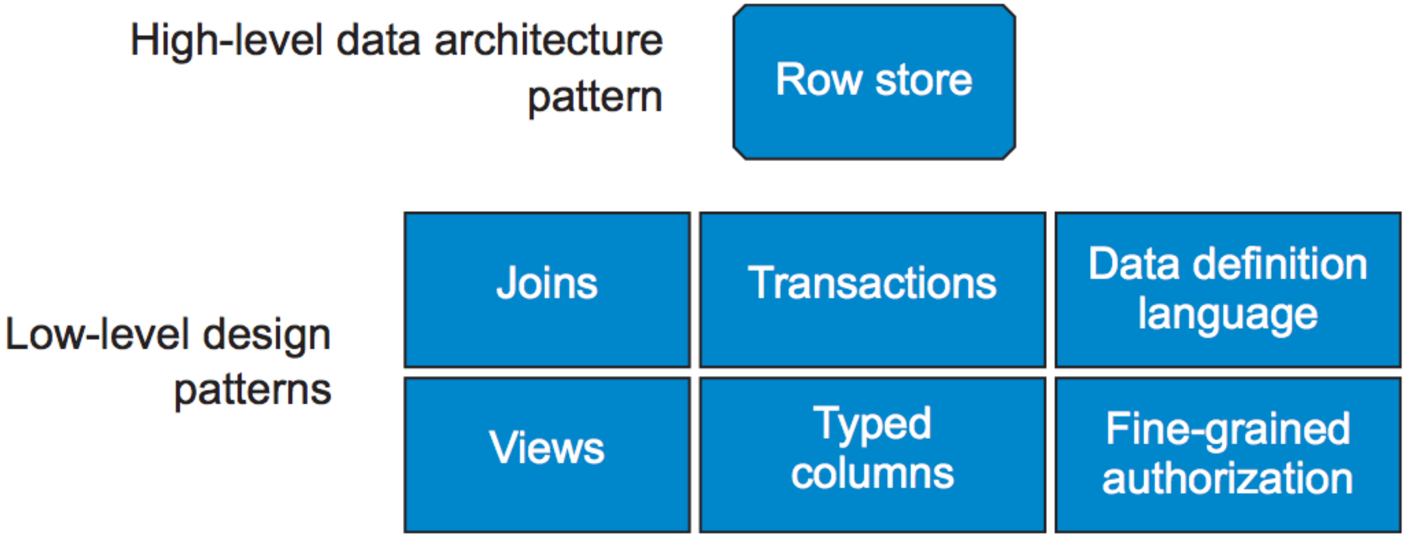
New rows are inserted at the end of a table.

insert →

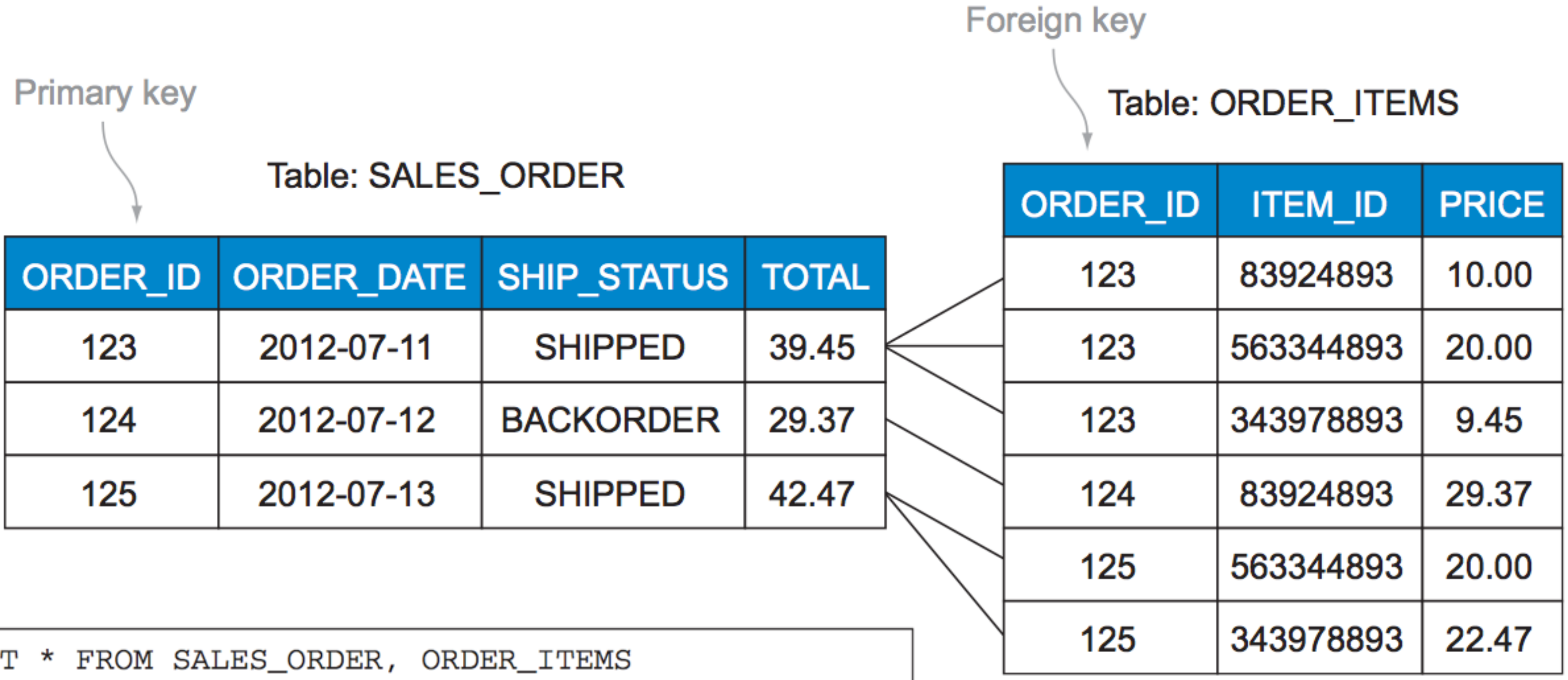
All data in a single row is grouped together on disk.

Columns may reference IDs of other tables.

ROW STORES - ARCHITECTURE PATTERN


High-level data architecture pattern
Row store
Low-level design patterns
Joins
Transactions
Data definition language
Views
Typed columns
Fine-grained authorization

JOINS – EXAMPLE OF SALES ORDER



Primary key

Table: SALES_ORDER

ORDER_ID	ORDER_DATE	SHIP_STATUS	TOTAL
123	2012-07-11	SHIPPED	39.45
124	2012-07-12	BACKORDER	29.37
125	2012-07-13	SHIPPED	42.47

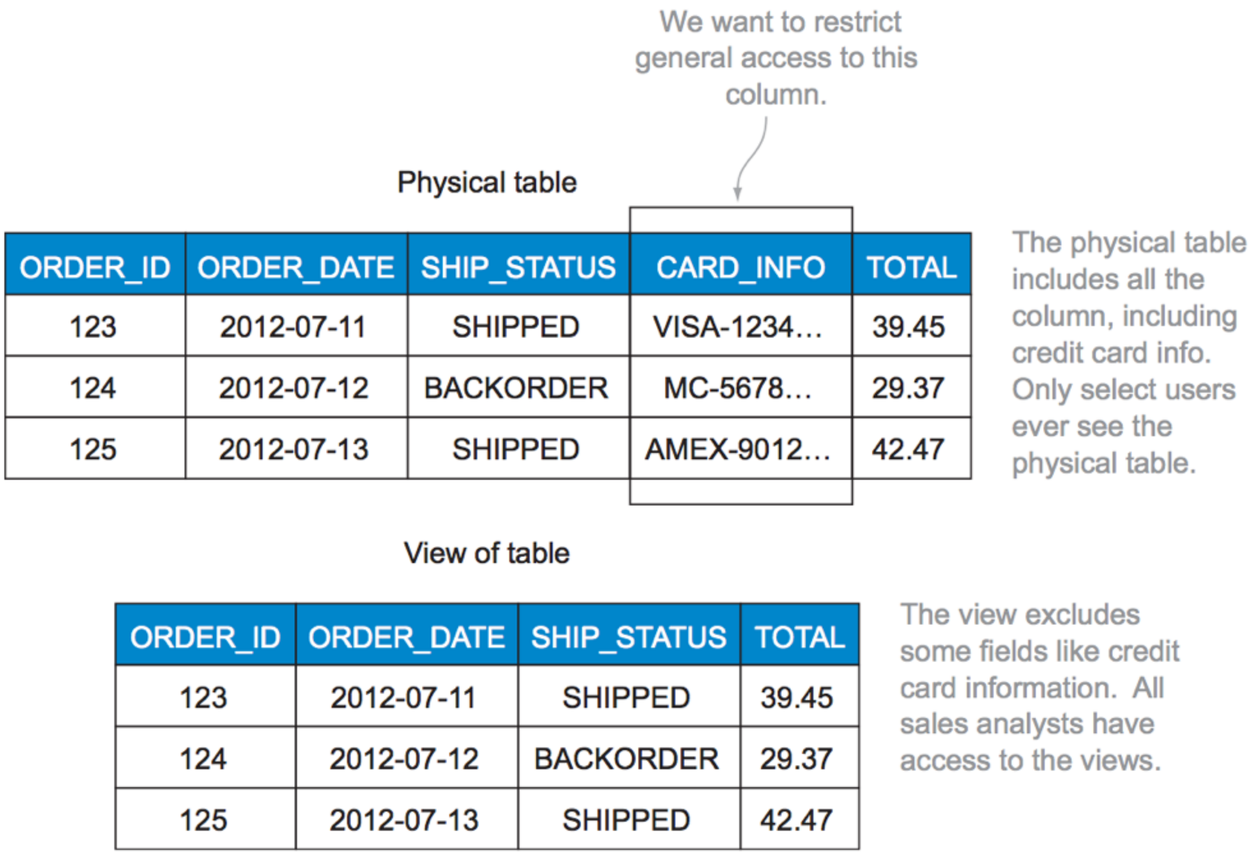
Foreign key

Table: ORDER_ITEMS

ORDER_ID	ITEM_ID	PRICE
123	83924893	10.00
123	563344893	20.00
123	343978893	9.45
124	83924893	29.37
125	563344893	20.00
125	343978893	22.47

```
SELECT * FROM SALES_ORDER, ORDER_ITEMS
WHERE SALES_ORDER.ORDER_ID = ORDER_ITEMS.ORDER_ID
```

VIEWS



We want to restrict general access to this column.

Physical table

ORDER_ID	ORDER_DATE	SHIP_STATUS	CARD_INFO	TOTAL
123	2012-07-11	SHIPPED	VISA-1234…	39.45
124	2012-07-12	BACKORDER	MC-5678…	29.37
125	2012-07-13	SHIPPED	AMEX-9012…	42.47

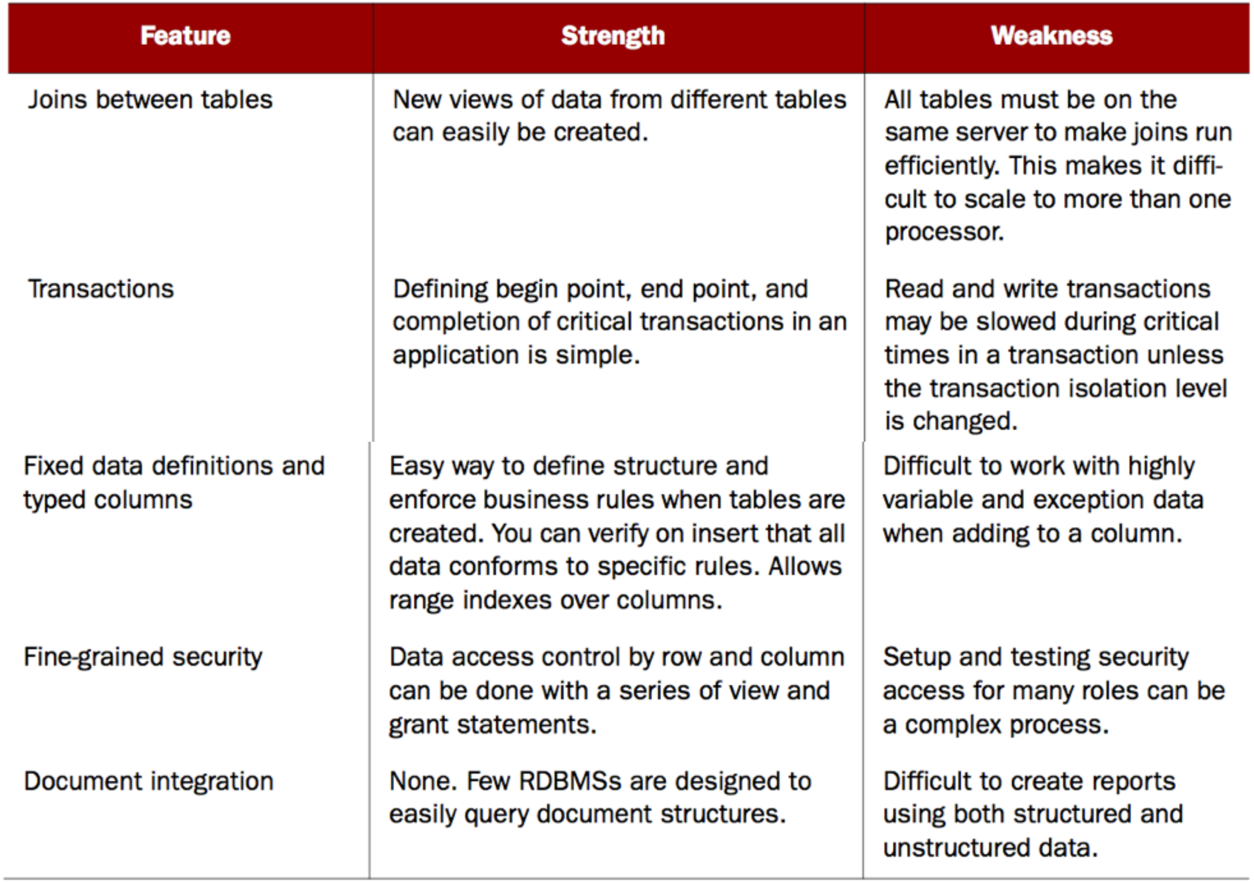
The physical table includes all the column, including credit card info. Only select users ever see the physical table.

View of table

ORDER_ID	ORDER_DATE	SHIP_STATUS	TOTAL
123	2012-07-11	SHIPPED	39.45
124	2012-07-12	BACKORDER	29.37
125	2012-07-13	SHIPPED	42.47

The view excludes some fields like credit card information. All sales analysts have access to the views.

RDBMS STRENGTHS & WEAKNESSES



Feature	Strength	Weakness
Joins between tables	New views of data from different tables can easily be created.	All tables must be on the same server to make joins run efficiently. This makes it difficult to scale to more than one processor.
Transactions	Defining begin point, end point, and completion of critical transactions in an application is simple.	Read and write transactions may be slowed during critical times in a transaction unless the transaction isolation level is changed.
Fixed data definitions and typed columns	Easy way to define structure and enforce business rules when tables are created. You can verify on insert that all data conforms to specific rules. Allows range indexes over columns.	Difficult to work with highly variable and exception data when adding to a column.
Fine-grained security	Data access control by row and column can be done with a series of view and grant statements.	Setup and testing security access for many roles can be a complex process.
Document integration	None. Few RDBMSs are designed to easily query document structures.	Difficult to create reports using both structured and unstructured data.