

Database Design & Implementation

ICT Skills

Objectives

- Relational databases & the Impedance Mismatch
- The changing data landscape & the emergence of NoSql
- Common characteristics of NoSQL databases

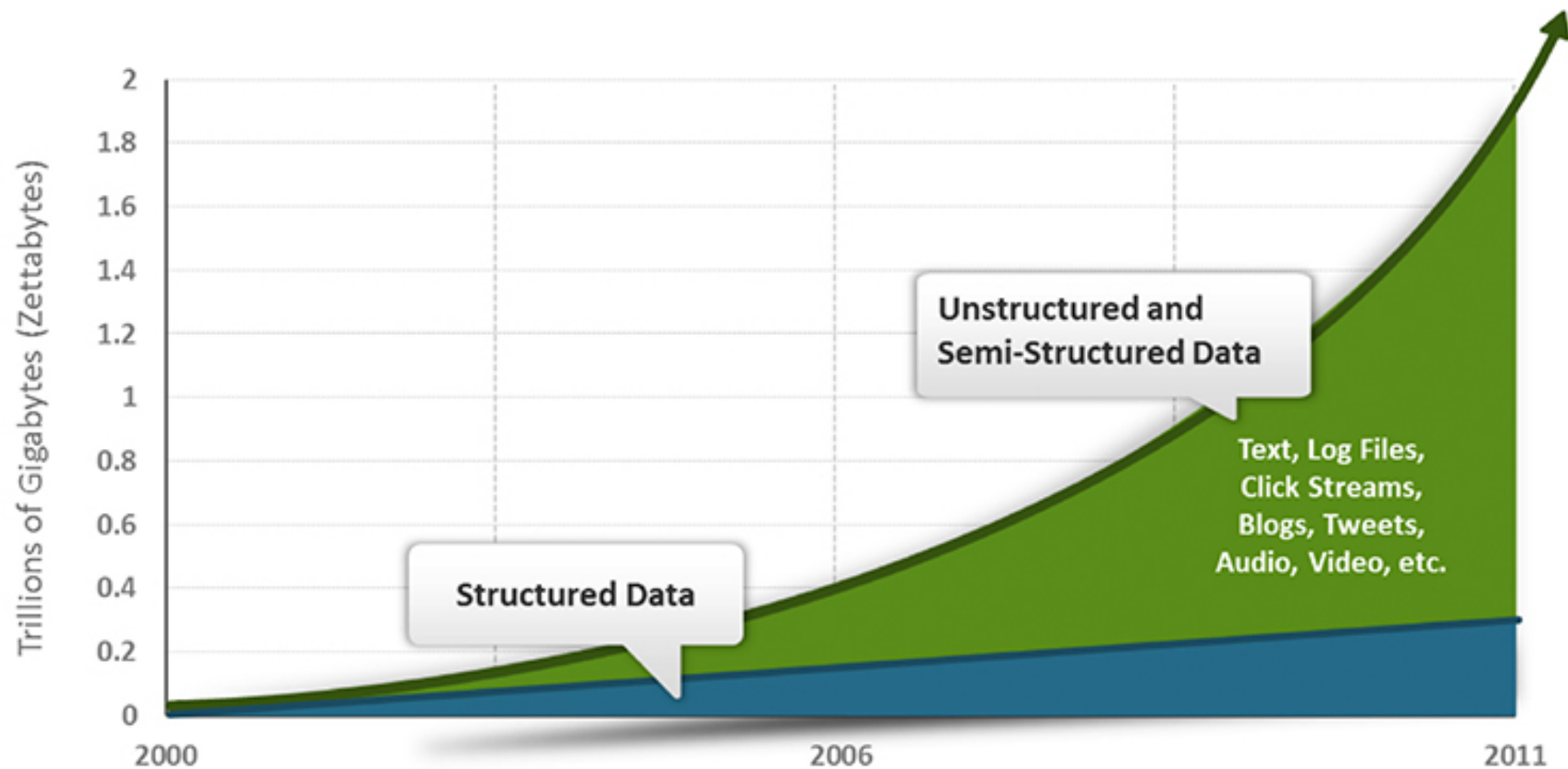
Why is NoSQL needed?

- Relational databases have been a successful technology for twenty year, delivering:
 - **Persistence:** storage of data
 - **Concurrency:** applications have many applications sharing the data at once.
 - **Integration:** multiple systems collaborating together using a single database.

Structured vs Unstructured Data

- There is a divide between the relational model and the in-memory data structures.
- Relational model
 - Organises data into a structure of tables and rows.
 - Databases produce relations of data and use relational algebra to conduct operations using SQL on those relations.
 - Has limitations where values in a relation have to be simple, they cannot contain any structure such as a nested record or a list.
- In Memory Data Structures
 - Take on much richer structures
 - Lists, Queues, Stacks, Graphs, Trees, Networks, etc...
 - Are a better match for certain types of problems

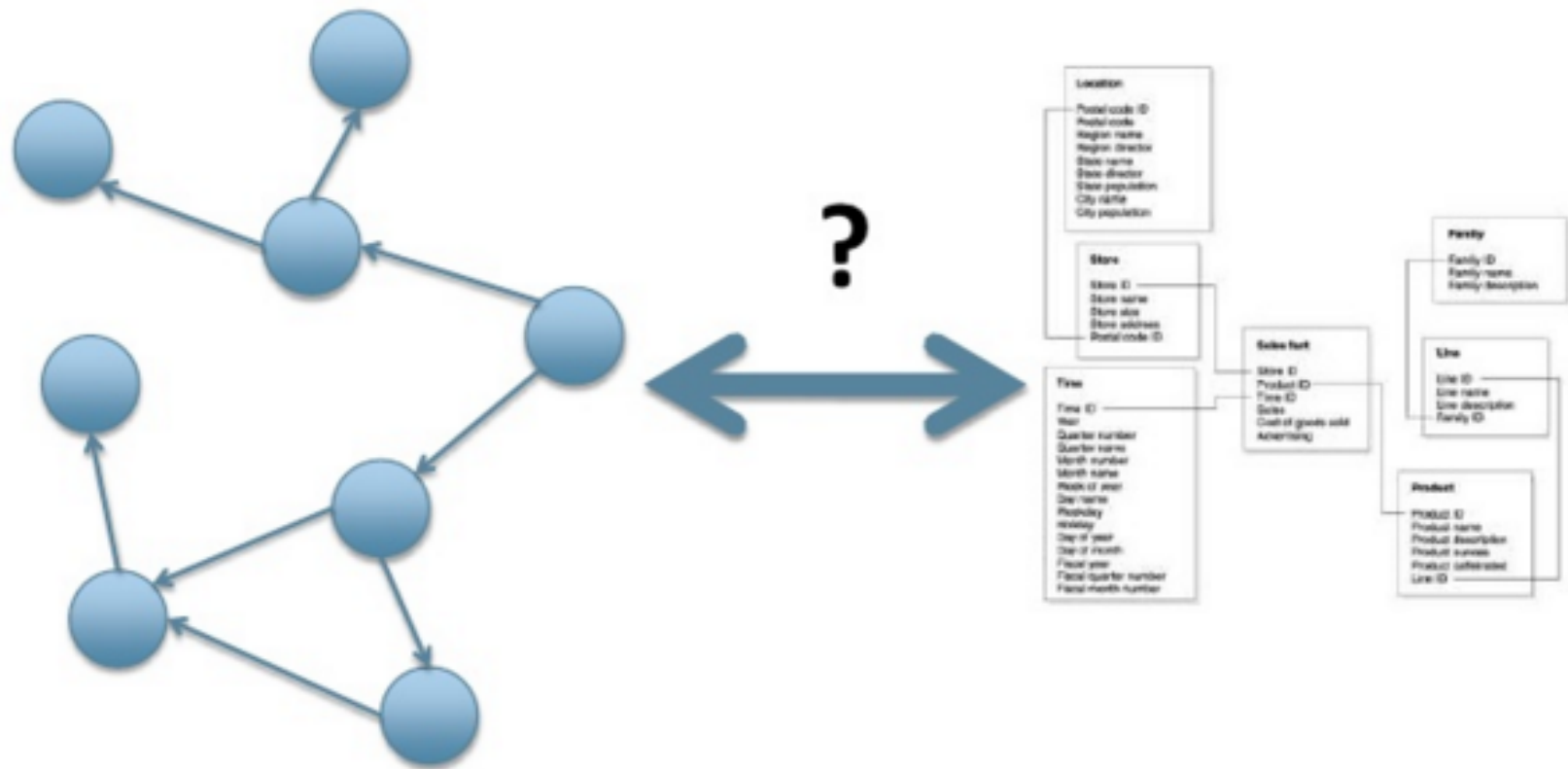
Structured vs Semistructured Data



Source: IDC 2011 Digital Universe Study (<http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011/index.htm>)

Impedance Mismatch

- Much effort spent on mapping data between in-memory data structures and a relational database.
- A NoSQL database may provide a data model that better fits the applications needs



Large Data Sets

- Organisations are finding it valuable to capture more data and process it quicker.
- They are finding it expensive to do so with relational databases.
 - Traditionally relational database is designed to run on a single machine
 - More economic to run large data and computing loads on clusters of many smaller and cheaper machines.
- *Many NoSQL databases are designed explicitly to run on clusters, so they make a better fit for big data scenarios.*

Emergence of NoSQL: Flexible Data Exchange

- During the 2000s there was a shift to web services where applications would communicate over http.
- These newer services use richer data structures with nested records and lists. - usually represented in JSON
- In general, with remote communication you want to reduce the number of round trips involved in the interaction, so it's useful to be able to put a rich structure of information into a single request or response.

Emergence of NoSQL: Diverse Data Structures

- The 2000s saw several large web properties dramatically increase in scale
 - and these started tracking activity and structure in a very detailed way:
 - Links, social networks, activity in logs, mapping data. With this growth in data came a growth in users.
- Coping with the increase in traffic required more computing resources
 - Scale Up: bigger machines, more processors, disk storage and memory.
 - Scale Out: lots of small machines in a cluster.
- As large websites moved towards clusters it revealed a new problem, relational databases are not designed to be run on clusters.

Common Characteristics of NoSQL databases

- They do not use the relational model.
- They do not use SQL
- Some have query languages that are similar to SQL.
- Generally open source.
- Most are driven by the need to run well on clusters.

Central Concept in SQL: The Aggregate

- An aggregate is a collection of data that we interact with as a unit.
- These units of data or aggregates form the boundaries for operations with the database.

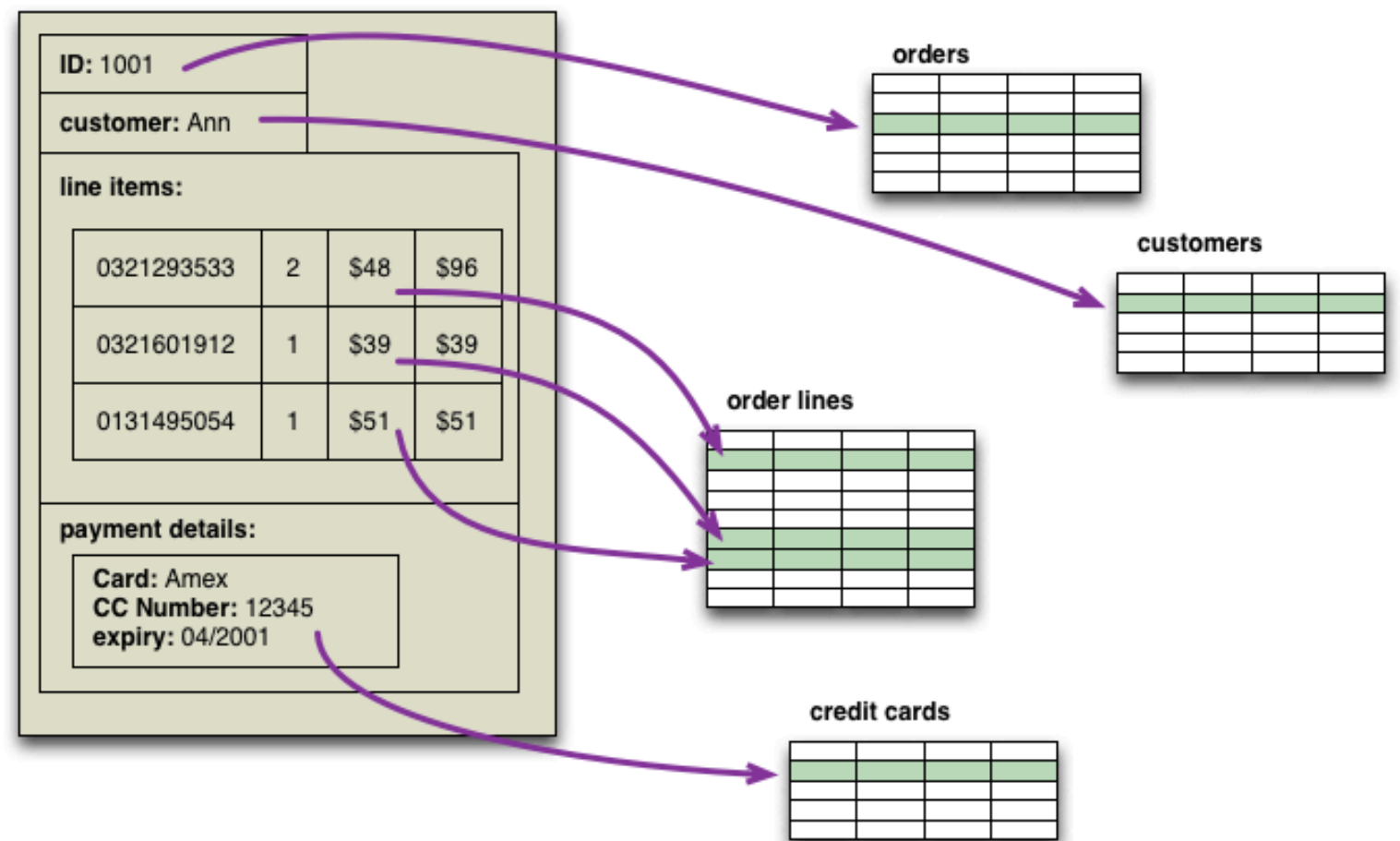
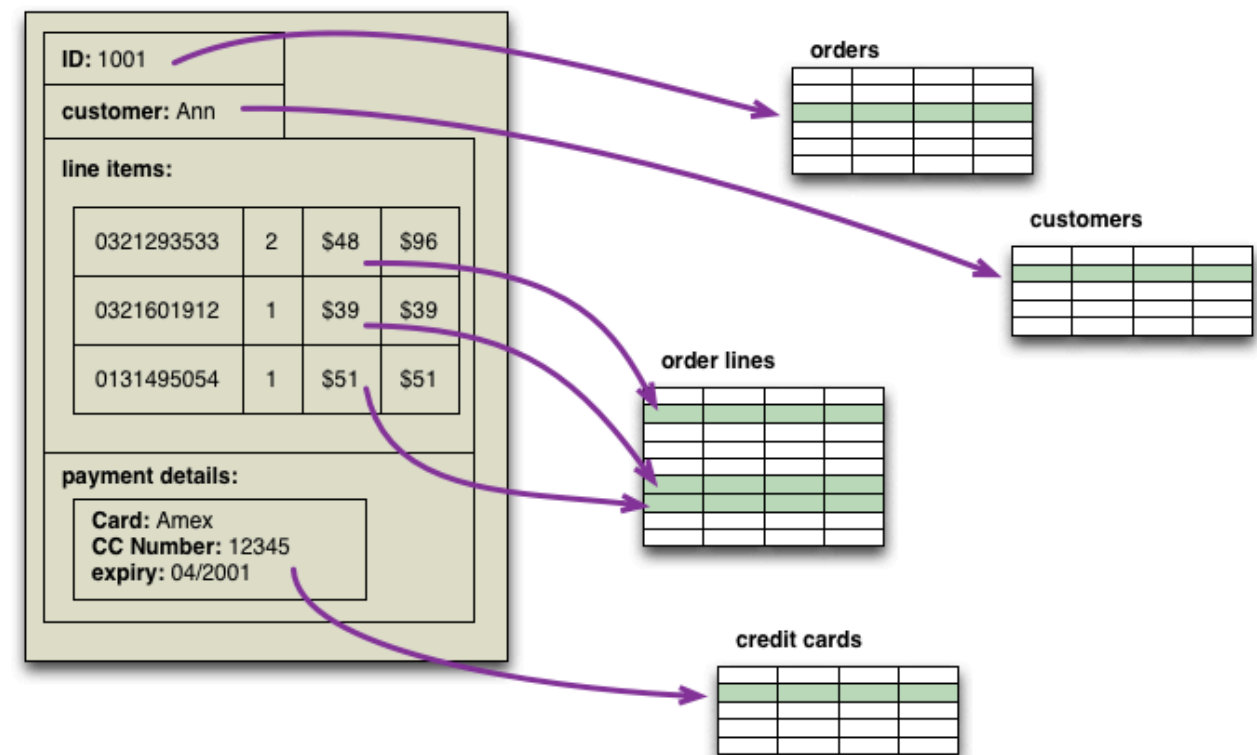
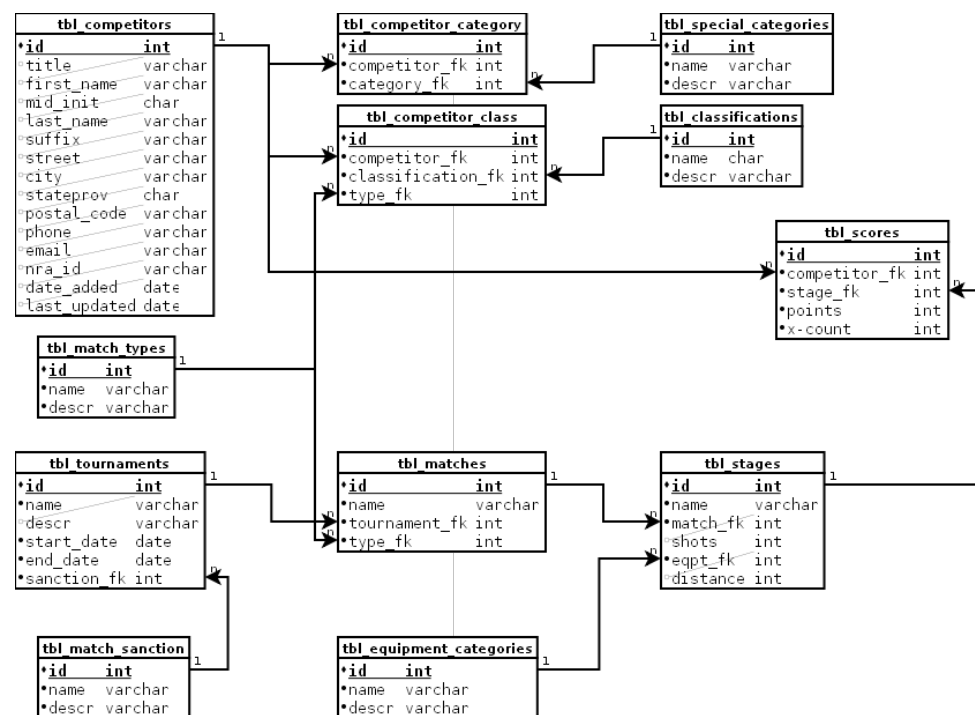


Table vs Aggregate



- Highly structured, rigid, extensive, standardised manipulation calculus (SQL)
- Loosely defined concept
- Varies depending on NoSQL DB type
- Not relevant to some DBs at all

Categories of NoSQL databases:

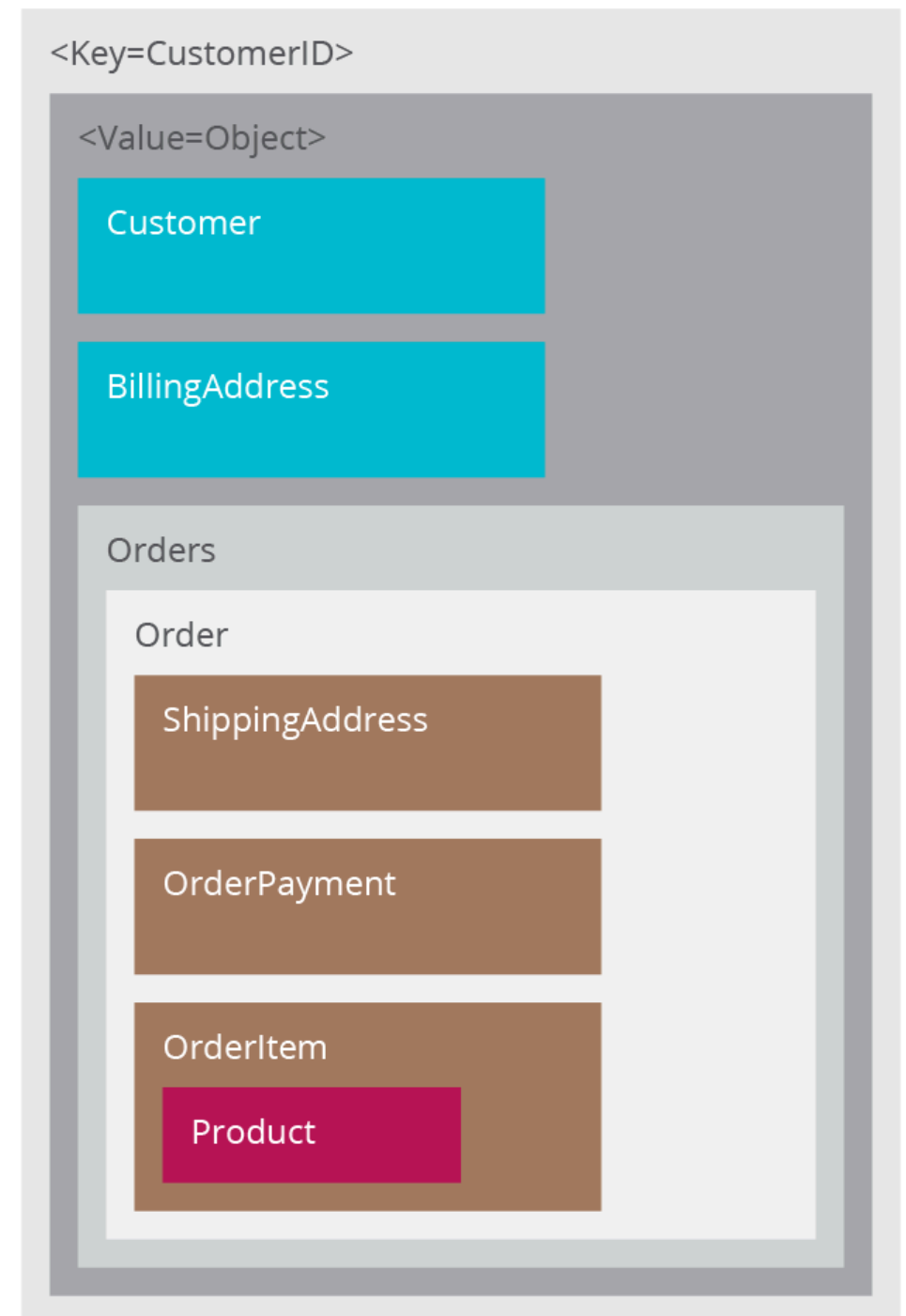
Data Model	Example Databases
Key-value	Berkely DB, LevelDB Memcached, Project Voldemort Redis, Riak
Document	CouchDB, MongoDB, RethinkDB
Column-Family	Amazon SimpleDB, Cassandra, Hbase Hypertable
Graph	FlockDB, HyperGraphDB, Infinite Graph, Neo4J,

Key Value

- Simplest NoSQL data stores to use from an API perspective.
- The client can either get the value for the key, put a value for a key, or delete a key from the data store.
- The value is a blob that the data store just stores, without caring or knowing what's inside; it's the responsibility of the application to understand what was stored.
- Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled

Key →

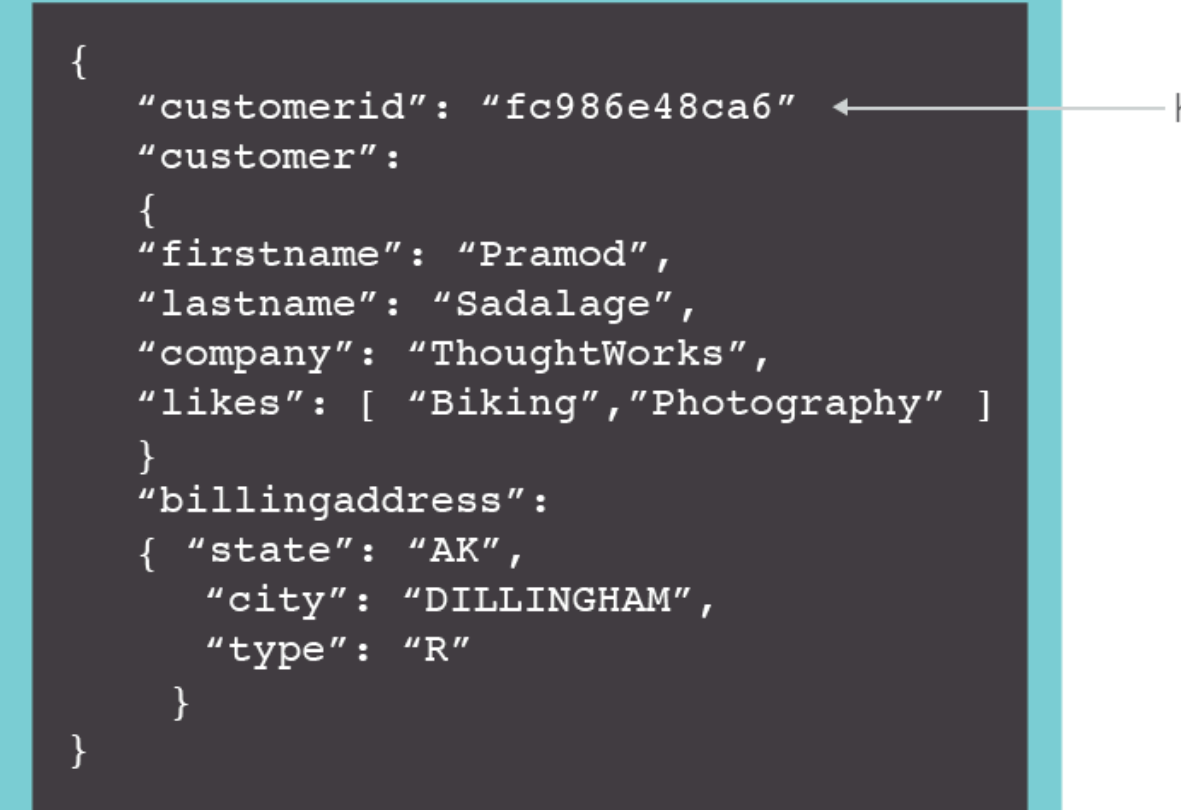
Value →



- The database stores and retrieves documents, which can be XML, JSON, BSON, and so on.
- These documents are self-describing, hierarchical tree data structures which can consist of maps, collections, and scalar values.
- The documents stored are similar to each other but do not have to be exactly the same.
- Document databases store documents in the value part of the key-value store; think about document databases as key-value stores where the value is examinable.
- Document databases such as MongoDB provide a rich query language and constructs such as database, indexes etc allowing for easier transition from relational databases.

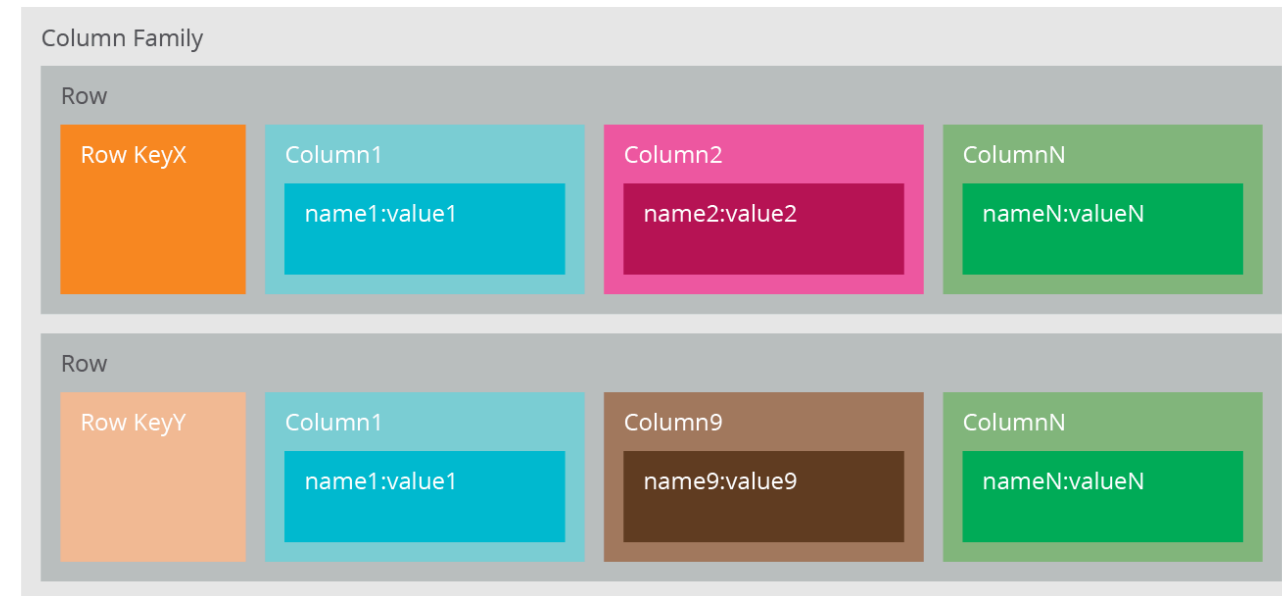
<Key=CustomerID>

```
{
  "customerid": "fc986e48ca6"
  "customer":
  {
    "firstname": "Pramod",
    "lastname": "Sadhalage",
    "company": "ThoughtWorks",
    "likes": [ "Biking", "Photography" ]
  }
  "billingaddress":
  { "state": "AK",
    "city": "DILLINGHAM",
    "type": "R"
  }
}
```



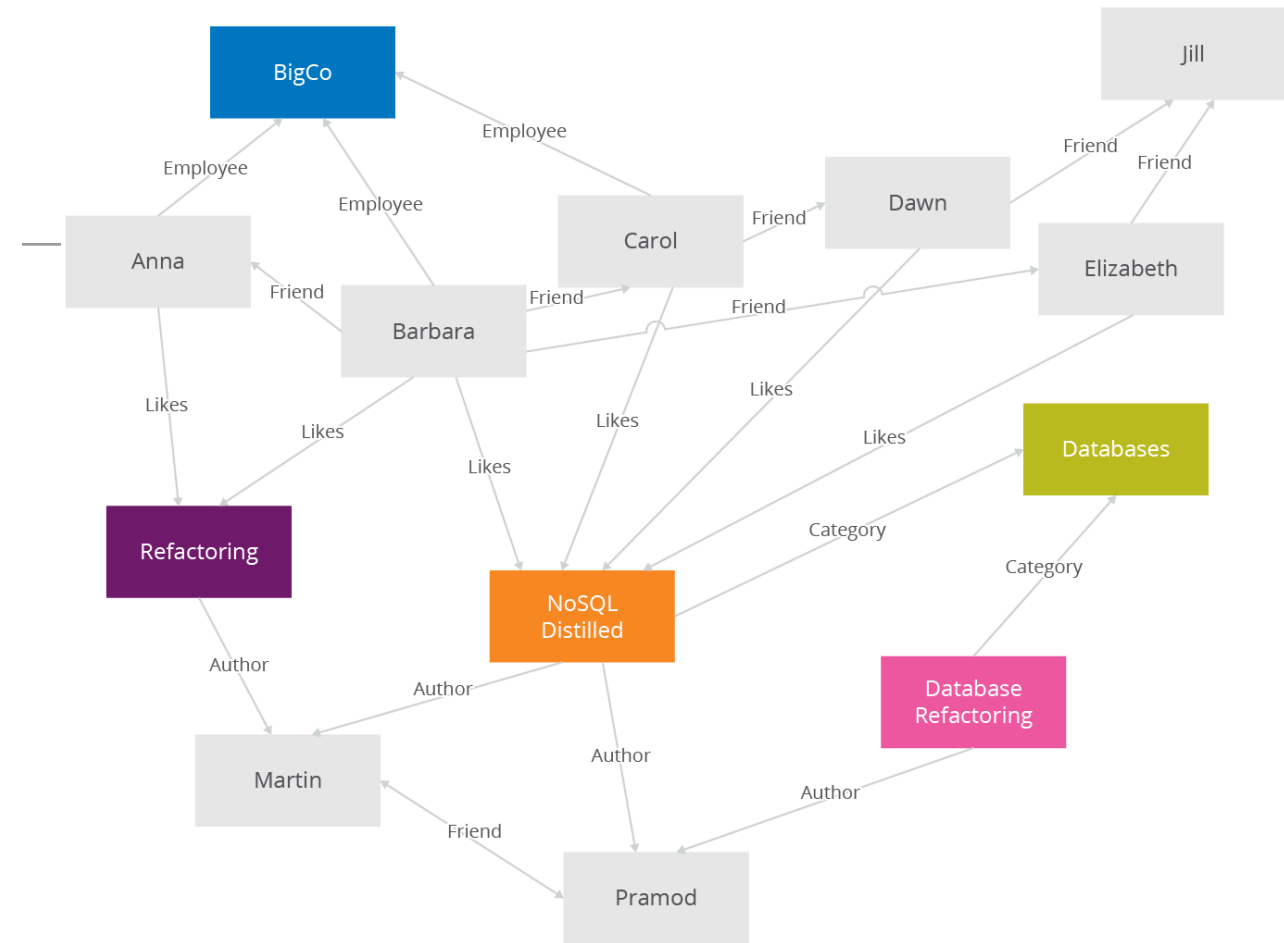
Document Databases

- Column-family databases store data in column families as rows that have many columns associated with a row key.
- Column families are groups of related data that is often accessed together.
 - For a Customer, we would often access their Profile information at the same time, but not their Orders.
- Each column family can be compared to a container of rows in an RDBMS table where the key identifies the row and the row consists of multiple columns.
- The difference is that various rows do not have to have the same columns, and columns can be added to any row at any time without having to add it to other rows.



Column Family Stores

- Graph databases allow you to store entities and relationships between these entities.
- Entities are also known as nodes, which have properties.
- Think of a node as an instance of an object in the application.
- Relations are known as edges that can have properties.
- nodes are organized by relationships which allow you to find interesting patterns between the nodes.
- The organization of the graph lets the data to be stored once and then interpreted in different ways based on relationships.



Graph Databases

MongoDB



- MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.
- A record in MongoDB is a document, which is a data structure composed of field and value pairs.
- MongoDB documents are similar to JSON objects.
- The values of fields may include other documents, arrays, and arrays of documents.

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value
← field: value
← field: value
← field: value

- Documents (i.e. objects) correspond to native data types in JavaScript
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.