DATABASE DESIGN & IMPLEMENTATION

ICT Skills

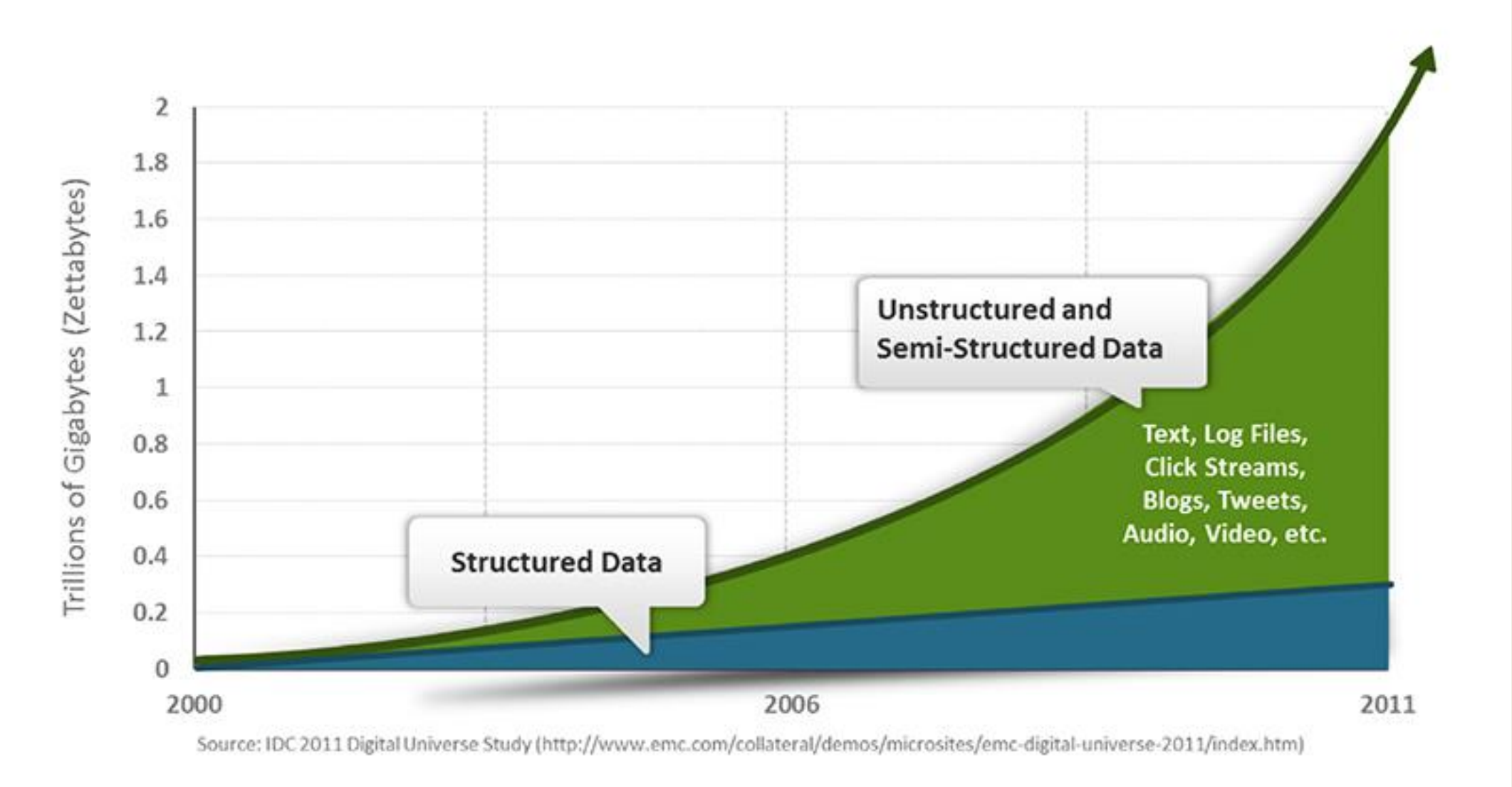
Objectives

- Relational databases and the need for NoSQL
- Common characteristics of NoSQL databases
- Types of NoSQL databases
  - *Key-value*
  - *Document*
  - *Column-family*
  - *Graph*
- NoSQL and consistency
- The CAP theorem

Why are NoSQL Databases Interesting?

- Application development productivity:
  - *A lot of application development effort is 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, thus simplifying that interaction and resulting in less code to write, debug, and evolve.*
  - *Large scale data. Organisations are finding it valuable to capture more data and process it quicker. They are finding it expensive, if even possible, to do so with relational databases. The primary reason is that a relational database is designed to run on a single machine, but it is usually 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.*

NoSQL


Trillions of Gigabytes (Zettabytes)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
2000
2006
2011
Unstructured and
Semi-Structured Data
Text, Log Files,
Click Streams,
Blogs, Tweets,
Audio, Video, etc.
Structured Data
Source: IDC 2011 Digital Universe Study (http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011/index.htm)

Why is NoSQL needed?

- Relational databases have been a successful technology for twenty years, providing persistence, concurrency control, and an integration mechanism.
- Persistence: storage of data, most commonly in a database
- Concurrency: applications have many people looking at the same data at once.
- Integration: multiple systems collaborating together using a single database.

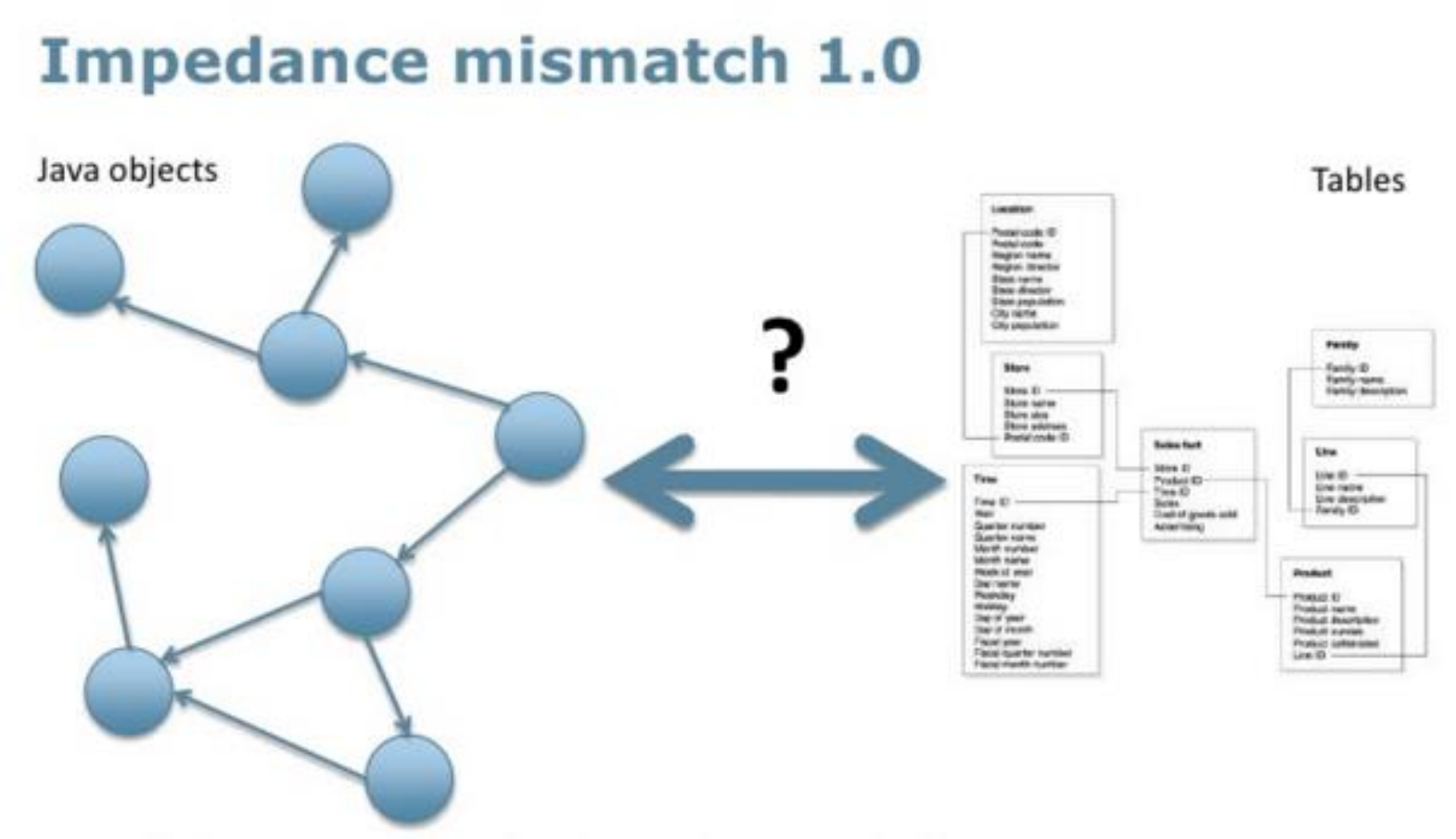
NoSQL

- Object-Relational Impedance mismatch
  - *The difference between the relational model and the in-memory data structures. The relational data 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.*
  - *But this has limitations where values in a relation have to be simple, they cannot contain any structure such as a nested record or a list.*
  - *Data structures for in-memory can take on much richer structures, if you want to use these you must translate it into relational representation.*
  - *Impedance mismatch, two different representations that require translation.*

NoSQL

- The impedance mismatch is a major source of frustration to application developers, this led in the 1990s to a growth in object oriented languages and databases.
- Object oriented database did not success and faded into distant memory.
- Impedance mismatch has been made easier by the wide availability of object-relational mapping frameworks. These frameworks remove a lot of the grunt work but can become a problem when people try to ignore the database and query performance suffers.

NoSQL



NoSQL

- Application and Integration Databases

A database can act as an integration database with multiple applications usually developed by separate teams, storing their data in a common database. This improves communication because all the applications are operating on a consistent set of persistent data.

However a structure designed to integrate many applications ends up being more complex than any single application needs. Also if an application wants to make changes to its data storage, it needs to coordinate with all the other applications using the database.

A database that acts as an application database which is only directly accessed by a single application codebase looked after by a single team.

NoSQL

- During the 2000s there was a shift to web services where applications would communicate over http.
- This resulted in more flexibility, where using SQL required you to transfer the data in the form of relations, the newer services you are ab le to use richer data structures with nested records and lists.
- These are usually represented as documents in XML or more recently, 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 requires or response.
- Once you choose an application database you have more choice as to the type of database. (many still chose relational databases but there are other options)

NoSQL

- The 2000s saw several large web properties dramatically increase in scale.
- Websites started tracking activity and structure in a very detailed way. Large sets of data appeared: 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, two choices, up or out. Up is bigger machines, more processors, disk storage and memory. Bigger machines are more expensive. The alternative is to use lots of small machines in a cluster. A cluster is cheaper and can be more resilient where individual machines fail the overall cluster will keep going.
- As large websites moved towards clusters it revealed a new problem, relational databases are not designed to be run on clusters. There are clustered relational databases but they have limitations.

NoSQL

- Emergence of NoSQL
  - *The term made its first appearance in the 1990s as the name of an open source relational database (Strozzi NoSQL), this database stores its tables as ASCII files, it doesn't use SQL it uses shell scripts.*
  - *The NoSQL we use today traces back to a meetup on June 11, 2009 in San Francisco. The example of BigTable and Dynamo had inspired a bunch of projects experimenting with alternative data storage and discussion were a feature of s/w conferences. The organiser Johan Oskarsson wanted a name for the meetup, something that would make a good Twitter hastag: short memorable and without too many Google hits so that a search on the name would quickly find the meetup. He asked for suggestions and got a few, selecting the suggestion of "NoSQL"*
  - *It had the disadvantage of being negative and not really describing these systems it did fit the hashtag criteria.*

NoSQL

- The term caught on, but it’s never been a term that’s had much in the way of a strong definition.
- There’s no generally accepted definition.
- NoSQL applied to a database refers to a set of mostly open-source databases, mostly developed in the early 21st century, and mostly not using SQL.

NoSQL

- A common statement about NoSQL databases is that since they have no schema, there is no difficulty in changing the structure of data during the life of an application.
- In fact a schemaless database still has an implicit schema that needs change discipline when you implement it.
- NoSQL is no single thing, nor will it replace relational databases.
- This area of computing is volatile, new features, new databases changes occur each year.

Common Characteristics of NoSQL databases

- They do not use the relational model.
- The 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.
- Based on the needs of the 21st century web
- Schemaless
  - *Possible to add fields to records without defining any changes in structure first.*

NoSQL

- NoSQL is more a movement than a technology.
- Relational databases aren’t going away, they are still going to be the most common form of database in use..
- The change is that now relational databases are one option for data storage.
- Polygot persistence is the idea of using different data stores in different data stores in different circumstances.
- Organisations need to also move from integration databases to application databases.
- Big Data running on clusters has driven this change in the database world, but it isn’t the only reason NoSQL databases are considered by project teams. The impedance mismatch problem has contributed too.

NoSQL

- Challenges:
  - *Maturity*
  - *Support*
  - *Analytics and BI tools*
  - *Administration*
  - *Expertise*

NoSQL

- Categories of NoSQL databases:
  - *Several data models, some databases fit more than one model.*
  - *Comprehensive lists available at: http://nosql-database.org and http://nosql.mypopescu.com/bk/nosql*
  - *The most common models are:*
    - Key-value
    - Document
    - Column family
    - Graph databases

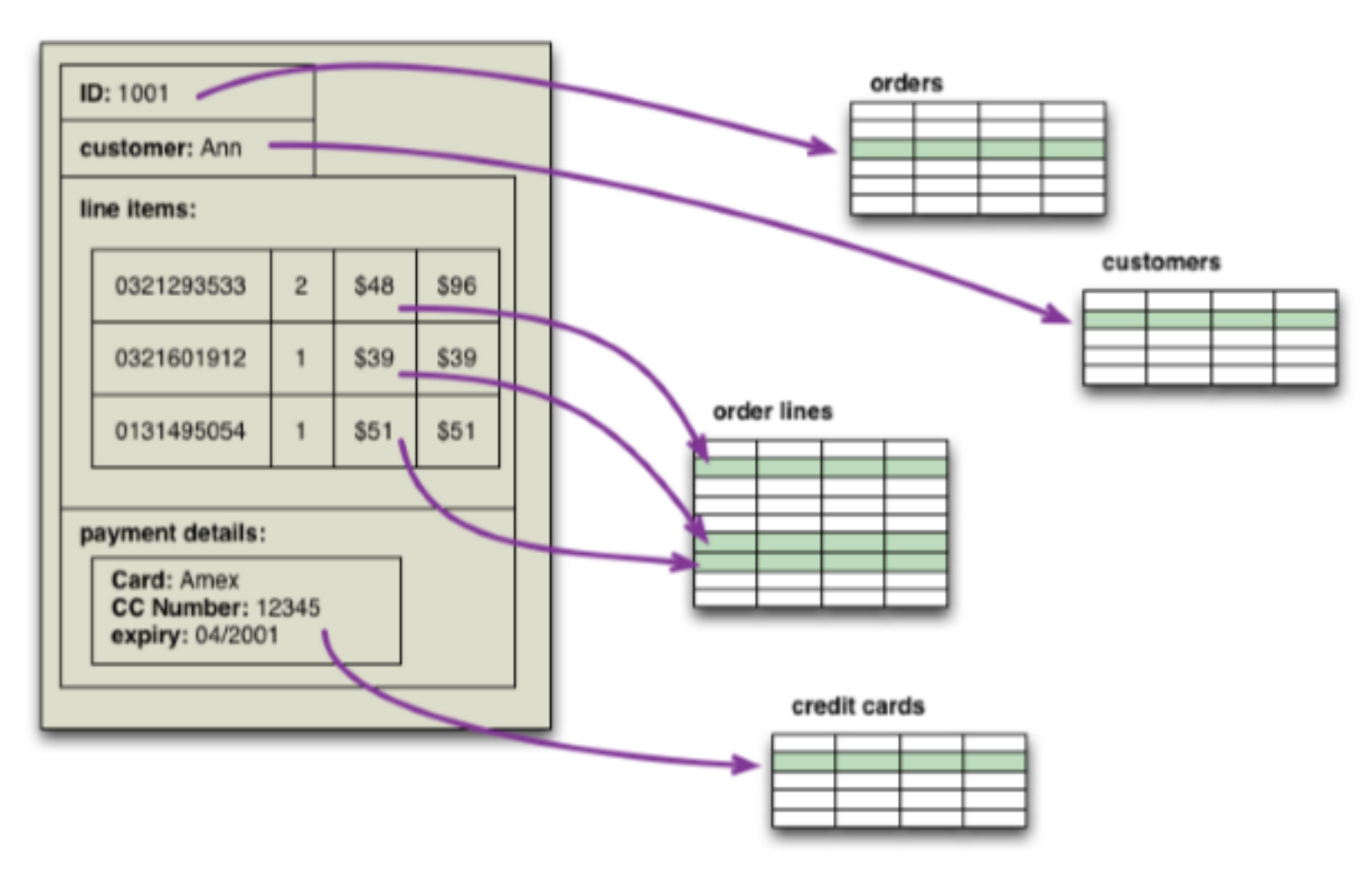
NoSQL

Data Model	Example Databases
Key-value	Berkely DB LevelDB Memcached Project Voldemort Redis Riak
Document	CouchDB MongoDB OrientDB RavenDB Terrastore
Column-Family	Amazon SimpleDB Cassandra Hbase Hypertable

NoSQL

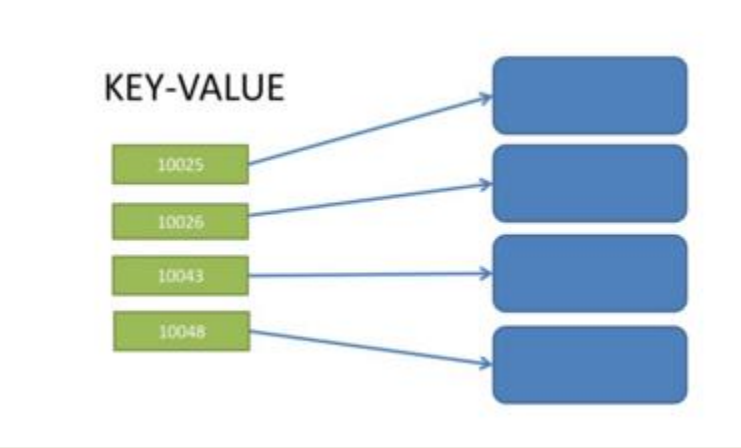
- Aggregate Data Models:
  - *A data model is the model through which we perceive and manipulate our data.*
  - *The dominant data model has been the relational data model, which is best visualised as a set of tables.*
  - *One of the shifts with NoSQL is a move away from the relational model.*
  - *Each NoSQL solution has a different model that it uses, which are in four categories widely used in the NoSQL arena: key-value, document, column-family, and graph.*
  - *The first three share a common characteristic of their data models which is called aggregate orientation.*

NoSQL


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
orders
customers
order lines
credit cards

NoSQL

- Key-Value
  - *Constructed through aggregates, each aggregate has a key or ID that used to get at the data.*
  - *In a key-value database the aggregate is opaque. This means we can store whatever we want in the aggregate*
  - *We can access an aggregate by looking up its key.*



  - *Both a key and a value are stored – in the case of the order example, everything to do with that one order is simply stored as one value*

NoSQL

- Document
  - *Constructed using aggregates, each having a key or ID that used to get to the data.*
  - *The document database is able to see a structure in the aggregate.*
  - *A document database imposes limits on what we can place in it, defining allowable structures and types.*
  - *Queries on the database are based on the fields in the aggregate, we can submit queries to the database based on the fields in the aggregate, we can retrieve part of the aggregate rather than the whole thing, and the database can create indexes based on the contents of the aggregate.*

NoSQL

- In general with key-value databases it is expected to mostly lookup aggregates using a key.
- With document databases, it is expected to submit some form of query based on the internal structure of the document; this might be a key, but it’s more likely to be something else.

NoSQL

- NoSQL Distilled (Sadalage & Fowler, 2013)
  - *Key points from this book are summarised here:*
  - *http://martinfowler.com/articles/nosqlKeyPoints.html*
- A 55 minute introduction to NoSQL by Martin Fowler is available to view here:
  - *https://www.youtube.com/watch?v=qI_g07C_Q5I&nohtml5=False*