ISAD353: Advanced Databases and Data Management

Seminar 3: Choosing your database

Marco Palomino

Module Aims

To expose students to the challenges of solutions for managing, processing, analysing and interpreting large amounts of unstructured data within relational and **non-relational database** environments.

Assessed Learning Outcomes

Critically evaluate current and emerging approaches for analysing and interpreting data using data mining and business intelligence techniques.

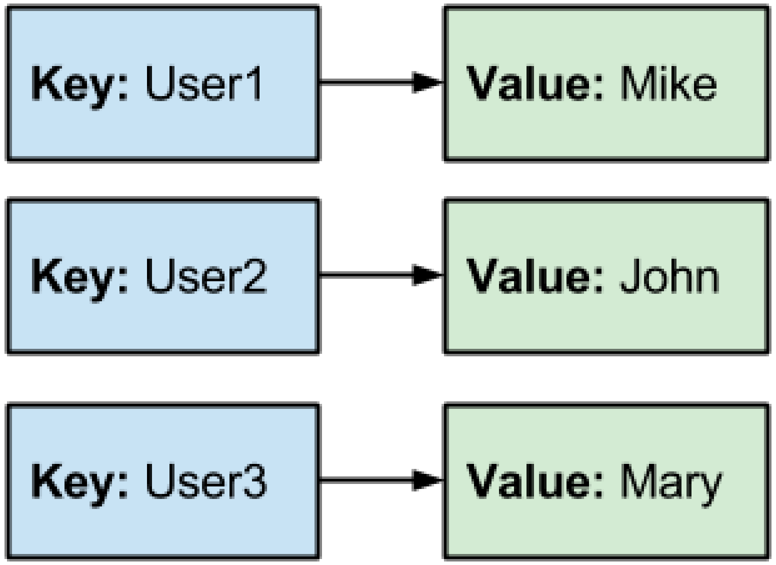
Outline

- NoSQL models
  - **Key-value store, or key-value database:** Redis, MemcacheDB, Berkeley DB (BDB), HamsterDB...
  - **Document-oriented database, or document store:** MongoDB, CouchDB, OrientDB, RavenDB, Lotus Notes....
  - **Column-family stores:** Cassandra, Hbase, Hypertable, Amazon DynamoDB...
  - **Graph database:** Neo4j, HyperGraphDB...

Google Bigtable

amazon DynamoDB

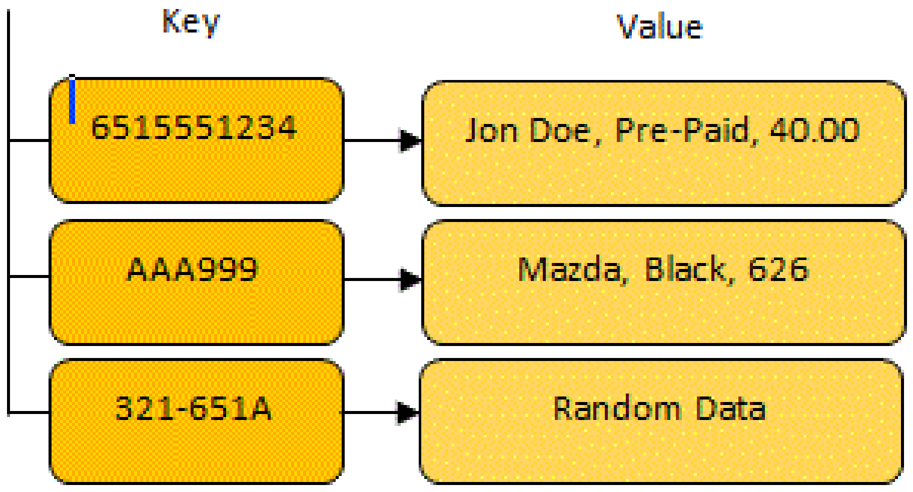
Key-value store



- Key-value pairs are similar to **hash tables**, where the key serves as an index that is used to find an associated value.
- Key-value pairs are similar to accessing data in memory, where the key is a **memory location** and the value is the **data stored at the location**, making key-value pairs a good data model for **in-memory databases**.

Features

- The data store can store up a value without caring or knowing what's inside.
- It is the responsibility of the application to understand what the value stores.
- In some key-value stores, such as **Redis**, the value can store lists, sets, hashes...



Relational databases

Employee

Employee ID	First Name	Last Name	Email	Computer ID
101	Andrew	Adams	a.adams@acme	LAP34278
102	Ben	Baker	b.baker@acme	LAP03671
103	Steve	Andrews	s.andrews@acme	DSK43901

Relational databases

Employee

Employee ID	First Name	Last Name	Email	Computer ID
101	Andrew	Adams	a.adams@acme	LAP34278, MAC10756
102	Ben	Baker	b.baker@acme	LAP03671
103	Steve	Andrews	s.andrews@acme	DSK43901, MAC11786, TBT33612

Relational databases

Employee

Employee ID	First Name	Last Name	Email	Computer ID 1	Computer ID 2	Computer ID 3
101	Andrew	Adams	a.adams@acme	LAP34278	MAC10756	
102	Ben	Baker	b.baker@acme	LAP03671		
103	Steve	Andrews	s.andrews@acme	DSK43901	MAC11786	TBT33612

Repeating group!

Relational databases

Employee

Employee ID	First Name	Last Name	Email	Computer ID 1	Computer ID 2	Computer ID 3
101	Andrew	Adams	a.adams@acme	LAP34278	MAC10756	
102	Ben	Baker	b.baker@acme	LAP03671		
103	Steve	Andrews	s.andrews@acme	DSK43901	MAC11786	TBT33612

Relational databases

Employee

Employee ID	First Name	Last Name	Email
101	Andrew	Adams	a.adams@acme
102	Ben	Baker	b.baker@acme
103	Steve	Andrews	s.andrews@acme

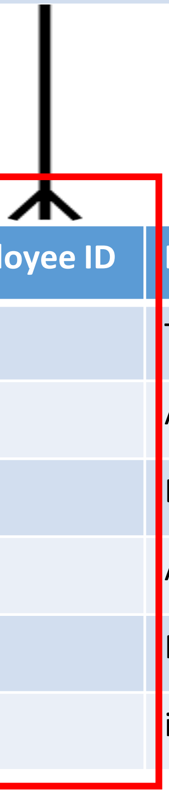
Computer

Computer ID	Employee ID	Description
LAP34278	101	Toshiba Laptop
MAC10756	101	Apple MacBook
LAP03671	102	Dell Laptop
DSK43901	103	Acer Desktop
MAC11786	103	MacBook Pro
TBT33612	103	iPad Air

Relational databases

Employee

Employee ID	First Name	Last Name	Email
101	Andrew	Adams	a.adams@acme
102	Ben	Baker	b.baker@acme
103	Steve	Andrews	s.andrews@acme

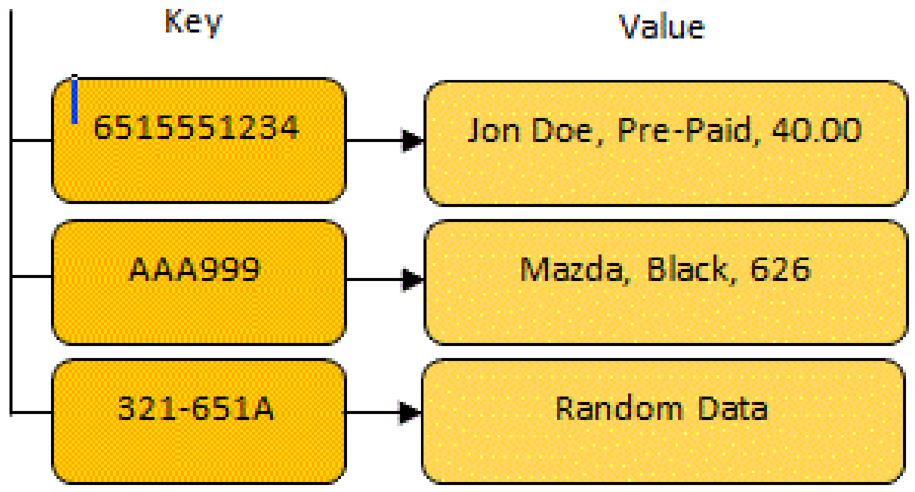


Computer

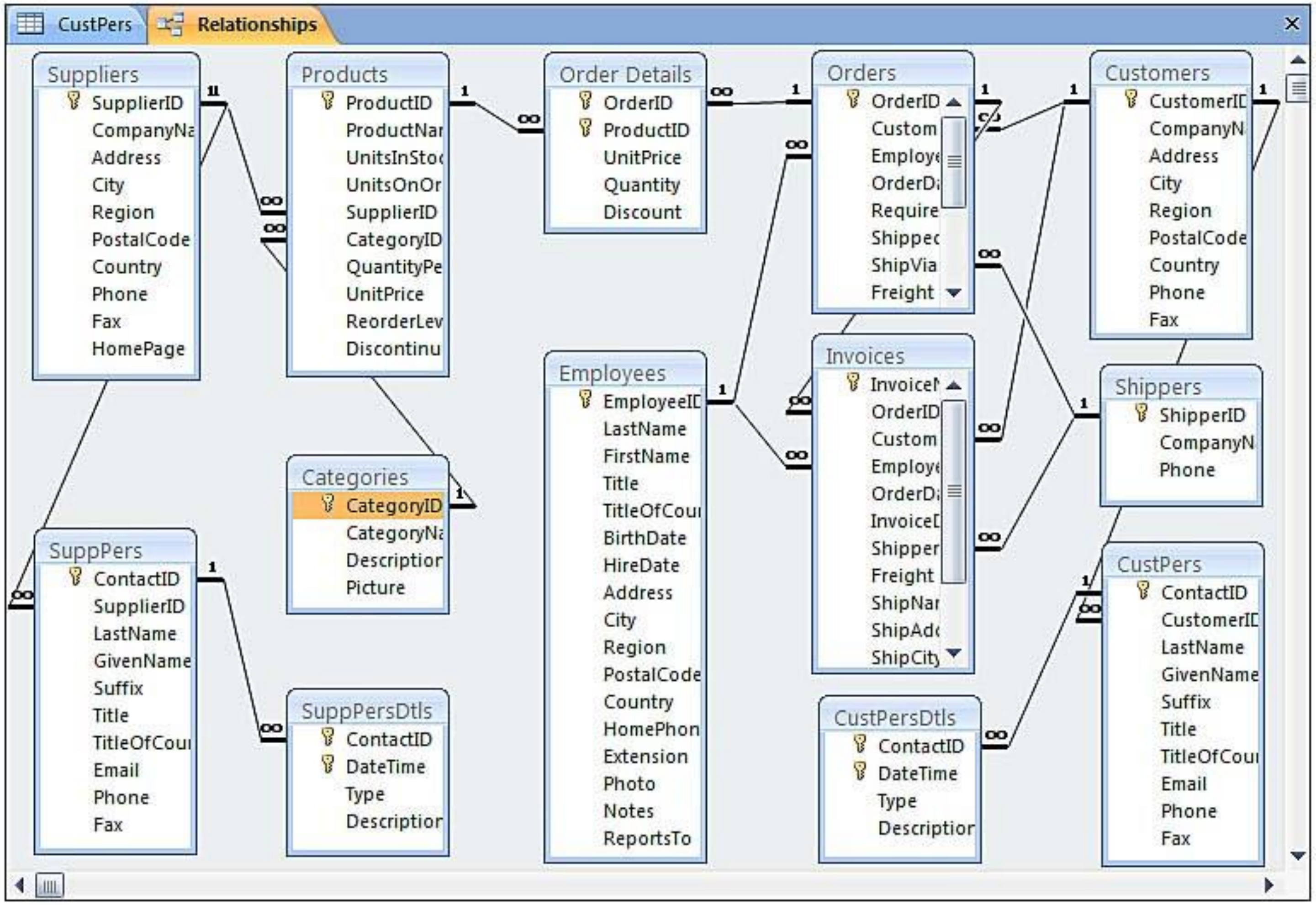
Computer ID	Employee ID	Description
LAP34278	101	Toshiba Laptop
MAC10756	101	Apple MacBook
LAP03671	102	Dell Laptop
DSK43901	103	Acer Desktop
MAC11786	103	MacBook Pro
TBT33612	103	iPad Air

Features

- The data store can store up a value without caring or knowing what's inside.
- It is the responsibility of the application to understand what the value stores.
- In some key-value stores, such as **Redis**, the value can store lists, sets, hashes...



Impedance mismatch


CustPers
Relationships
Suppliers: SupplierID, CompanyNa, Address, City, Region, PostalCode, Country, Phone, Fax, HomePage
Products: ProductID, ProductNar, UnitsInStoc, UnitsOnOr, SupplierID, CategoryID, QuantityPe, UnitPrice, ReorderLev, Discontinu
Order Details: OrderID, ProductID, UnitPrice, Quantity, Discount
Orders: OrderID, Custom, Employe, OrderDa, Require, Shippec, ShipVia, Freight
Customers: CustomerID, CompanyN, Address, City, Region, PostalCode, Country, Phone, Fax
Categories: CategoryID, CategoryNa, Descriptior, Picture
Employees: EmployeeID, LastName, FirstName, Title, TitleOfCou, BirthDate, HireDate, Address, City, Region, PostalCode, Country, HomePhon, Extension, Photo, Notes, ReportsTo
Invoices: InvoiceI, OrderID, Custom, Employe, OrderDa, InvoiceD, Shipper, Freight, ShipNar, ShipAdc, ShipCity
Shippers: ShipperID, CompanyN, Phone
SuppPers: ContactID, SupplierID, LastName, GivenName, Suffix, Title, TitleOfCou, Email, Phone, Fax
SuppPersDtls: ContactID, DateTime, Type, Descriptior
CustPersDtls: ContactID, DateTime, Type, Descriptior
CustPers: ContactID, CustomerID, LastName, GivenName, Suffix, Title, TitleOfCou, Email, Phone, Fax
1
∞

Examples of key-value stores

- Redis
- MemcacheDB
- Berkeley DB (BDB)
- HamsterDB (especially suited for **embedded devices**)
- Amazon DynamoDB (**not open source!**)
- Project Voldemort (an open source implementation of Amazon DynamoDB)

HamsterDB
embedded database

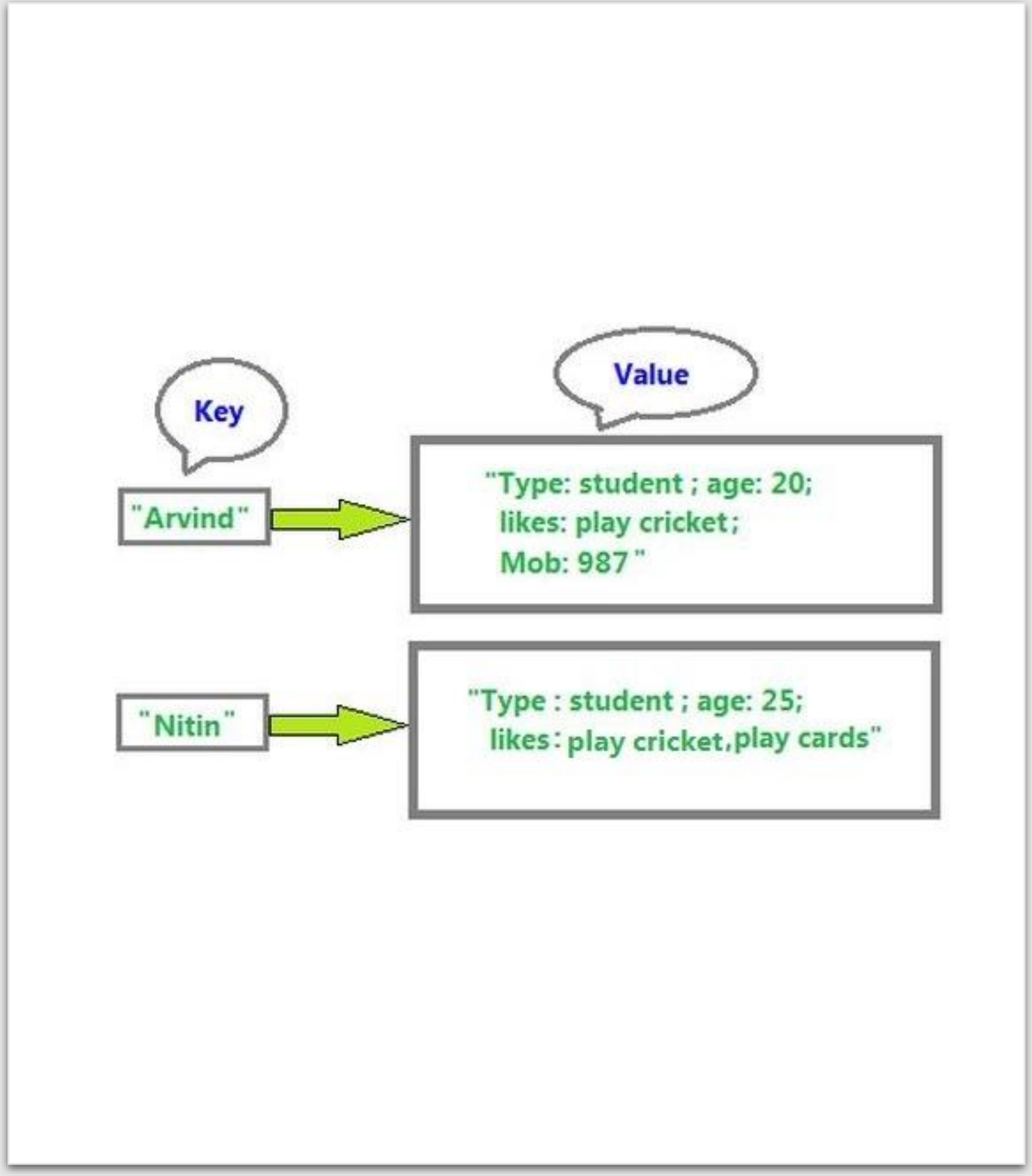
When to use...

- **User profiles:** Users have a unique ID, as well as preferences, such as language, time zone, which products the user has access to, etc... This can all be put into an object, so **getting preferences of a user takes a single GET operation**.

- **Shopping carts:** E-commerce websites have shopping carts tied to the user. As we want the shopping carts to be available all the time across browsers, machines, and sessions, **all the shopping information can be put into a single value**, where the key is the user ID.

Not advisable to use when...

Typically, key-value stores can **only be queried by the key**. It's not possible to query by some attribute of the value. In such a case, your application (client) would need to read the values from the database to figure out if the attributes meet the conditions.

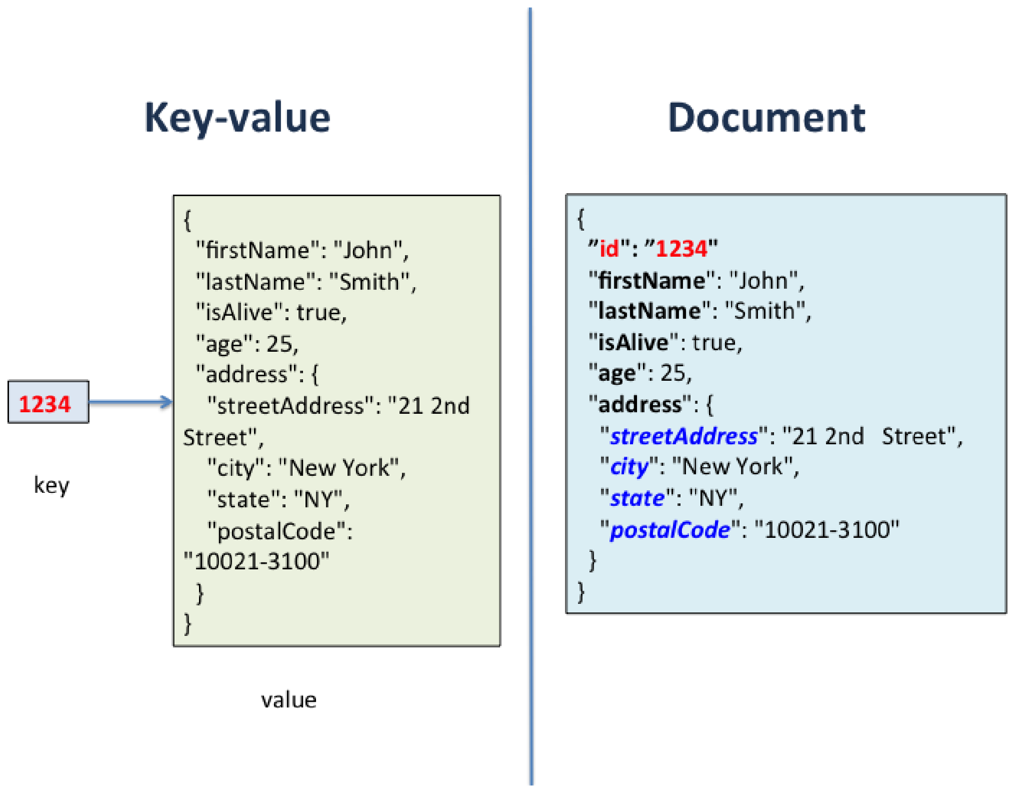

Key
Value
"Arvind"
"Type: student ; age: 20;
likes: play cricket;
Mob: 987 "
"Nitin"
"Type : student ; age: 25;
likes: play cricket, play cards"

Document-oriented databases

- In a **document database**, all of the data associated with an individual record is encapsulated in what is referred to as a **document**.

- Essentially, it is like a key-value store, except that the value is a document.

- A document is usually encoded in some form of standard format such as **XML** or **JSON**.

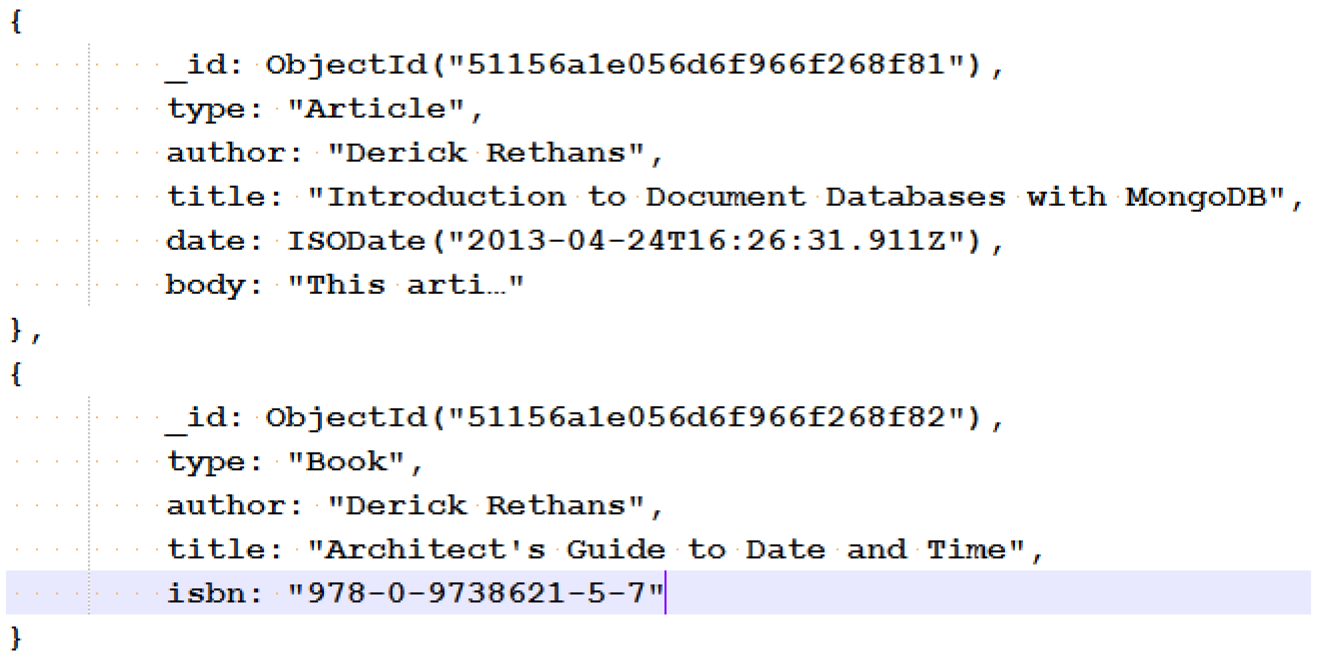


```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd
Street",
    "city": "New York",
    "state": "NY",
    "postalCode":
"10021-3100"
  }
}
```

```
{
 "id": "1234"
 "firstName": "John",
 "lastName": "Smith",
 "isAlive": true,
 "age": 25,
 "address": {
  "streetAddress": "21 2nd  Street",
  "city": "New York",
  "state": "NY",
  "postalCode": "10021-3100"
 }
}
```

Features

- Each document can share a common structure but can also **vary in the types of fields that they contain**. This approach allows new information to be added to some documents without requiring that all documents have the same structure.



```
{
        _id: ObjectId("51156a1e056d6f966f268f81"),
        type: "Article",
        author: "Derick Rethans",
        title: "Introduction to Document Databases with MongoDB",
        date: ISODate("2013-04-24T16:26:31.911Z"),
        body: "This arti…"
},
{
        _id: ObjectId("51156a1e056d6f966f268f82"),
        type: "Book",
        author: "Derick Rethans",
        title: "Architect's Guide to Date and Time",
        isbn: "978-0-9738621-5-7"
}
```

Examples of document databases

- MongoDB
- CouchDB
- Terrastore
- OrientDB
- RavenDB
- Lotus Notes

Useful for event logging

- Servers generate a large number of events that contain useful information about their operation, including errors, warnings, and user behaviour.
- Most servers, store log data in plain text files. Although plain-text logs are accessible and human-readable, they are difficult to use, reference, and analyse without other systems for aggregating and storing these data.
- Document databases can store all these different types of events and can act as a central data store for event storage. This is especially true when **the type of data** being captured by the events **keeps changing**.

Useful for content management systems

- Back in 2000, websites comprised mostly static text: **Brochureware**. Today, though, an array of text, audio, video, images and social media is needed to get users' attention.

- New attributes cannot be easily added to a relational database – not without impacting performance or taking the database offline.

- Since document databases have no predefined schemas and usually understand JSON documents, they work well in **content management systems** or applications for publishing websites, managing user comments, user registrations, profiles, etc...

Usage examples

- **Orange Digital**, a subsidiary of France Telecom, maintains the websites **Orange** (www.orange.co.uk), **Orange World**, and the **Orange Business** site, as well as a number of EE’s digital assets. It is no longer hosting the data and has moved to Amazon's cloud-based web service.

- Orange Digital decided to use **MongoDB to store content and metadata**.

- *“Moving to MongoDB has required a mindset shift... you have to think about how data is used, rather than how it's stored,”* Phil Butterworth – Principal Database Engineer.

Disadvantages

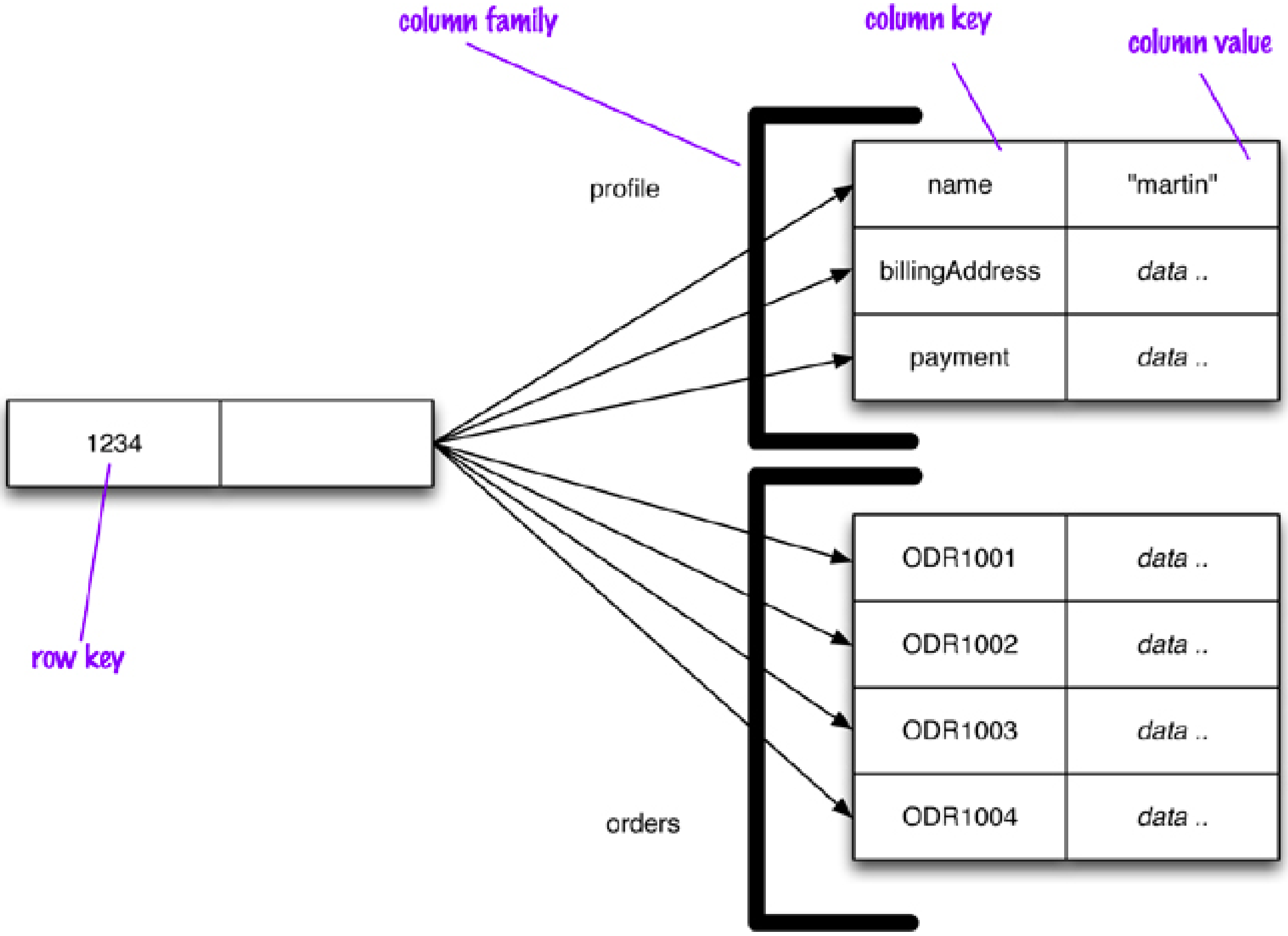
- Queries against a varying structure.

Column-family databases

- Column oriented databases build on the concepts introduced by **Google's BigTable** approach to provide a flexible way of managing data with variable structure.

- Column-oriented databases provide a way to **group columns** into **column families** and to dynamically add new columns as needed to a column family.

- **Column families** are groups of **related data** that is often **accessed together**.


column family
column key
column value
profile
name
"martin"
billingAddress
data ..
payment
data ..
1234
row key
ODR1001
data ..
ODR1002
data ..
ODR1003
data ..
ODR1004
data ..
orders

Examples

- Cassandra
- HBase
- Hypertable
- Amazon SimpleDB
- Amazon DynamoDB

Useful when: Expiring usage

- **Expiring columns**: Some databases, such as **Cassandra**, allows you to have columns which are deleted automatically after a given time. This time is known as **TTL (Time to Live)**. The column is deleted after the TTL has elapsed.

- Examples: Expiring columns can be very useful for applications providing **demo access** to users, or applications **displaying ads** on a website for a **specific time**. When the TTL has elapsed and the column does not exist anymore, the access can be revoked or the banner can be removed.

Useful for content management systems

- Using column families, we can store **blog entries** with tags, categories and links in different columns.

- Blog users and the actual **blogs** can be put into different column families.

Usage examples

- Cassandra was developed at **Facebook** to power the **Facebook inbox search feature**.
- Facebook released Cassandra as an open-source project on Google code in July 2008.
- Currently, **Globo.com's** live streaming platform is implemented using Cassandra. Globo.com is the Internet branch for Grupo Globo, one of the 5 largest media conglomerates in the world, producing content such as TV series, TV shows, news shows, etc. exporting them worldwide.

Not advisable to use when...

- If you need to aggregate data using queries involving operations such as SUM or AVG, you will have to do this on the client side, using the data retrieved by the client from the database.

- Column-oriented databases are **not great for prototyping**: during the early stages, you may not be sure about how the query patterns will change, and as these change, you have to modify the column family design.