TASK

Introduction to Databases

Visit our website

Introduction

WELCOME TO THE INTRODUCTION TO DATABASES TASK!

Dynamic web applications rely on data. If you want to make a web application truly dynamic, storing data about your users is a good starting point. In this task, you will learn some theory about the various types of databases. You will also be introduced to MongoDB, a very popular database that you will be using in this bootcamp.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

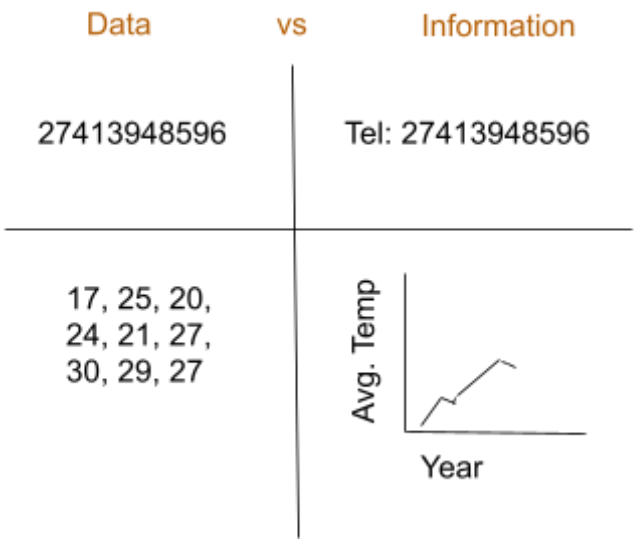
DATABASES

Data is core to software development. Programs are designed to manipulate, create and visualise data. Therefore, it is important for all software developers to be able to access, manipulate and store data.

Databases are used to store data. A database is simply a large container of data, with the ability to order the data in multiple ways while providing easy access to the data itself. Web developers often have to manipulate the data stored in databases. For example, you may need to store your users' usernames, passwords, names, addresses and telephone numbers, etc. Therefore, full stack web developers need to be able to work proficiently with databases.

DATA VS. INFORMATION

In order to properly understand databases, you must first understand the difference between data and information. Simply put, data are raw facts. The word raw indicates that the facts have not yet been processed to reveal their meaning. Information, on the other hand, is the result of processing the raw data to reveal its meaning. Data processing can be as simple as organising data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modelling.

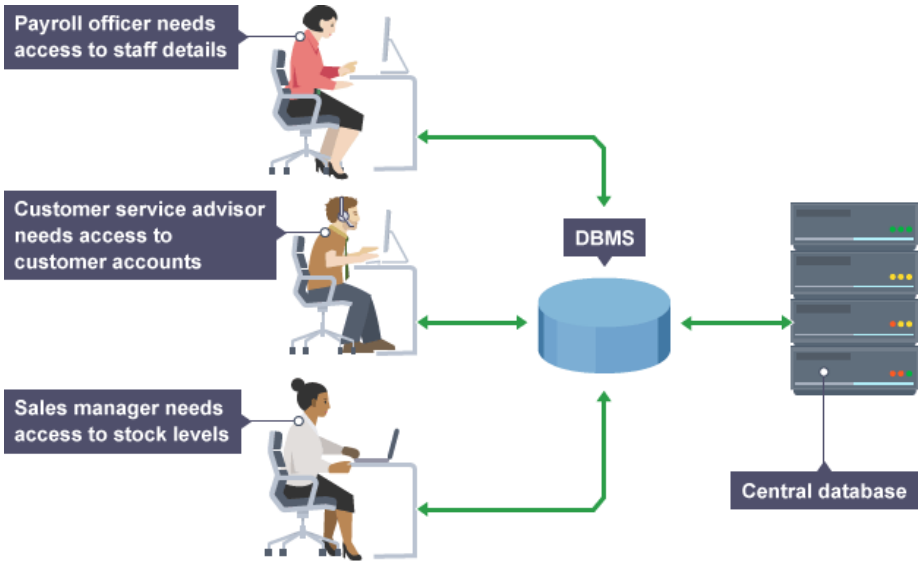


The production of accurate, relevant and timely information is the key to good decision-making and, in turn, good decision-making is the key to a business' survival in a global environment. Timely and useful information requires accurate

data which must be generated properly and stored in a format that is easy to access and process. The data environment should be carefully managed.

DBMS

A database can be thought of as a well-organised electronic filing cabinet where powerful software, known as a database management system (DBMS), helps manage the contents of the cabinet. A database management system is a collection of programs that manage the database structure and control access to the data stored in the database.



The DBMS manages the interaction between the end user and the database (bbc.co.uk)

The illustration above shows how the DBMS serves as an intermediary between the user and the database. The DBMS receives all application requests and translates them into the complex operations required to fulfil those requests.

Much of the database's internal complexity is hidden from the application programs and end users by the DBMS. There are some very important advantages to having a DBMS between the end user's application and the database. Firstly, the DBMS allows the data in the database to be shared among multiple applications or users. Secondly, the DBMS integrates many different users' views of the data into a single data repository.

The DBMS helps make data management much more efficient and effective and provides advantages such as:

- **Improved data sharing**: The DBMS helps create an environment in which end users have better access to more and better-managed data.

- **Better data integration**: An integrated view of the organisation's operations and a clearer view of the big picture is promoted by wider access to well-managed data.

- **Minimised data inconsistency**: Data inconsistency occurs when different versions of the same data appear in different places. A properly designed database greatly reduces the probability of data inconsistency.

- **Improved data access**: A query is a specific request for data manipulation (e.g. to read or update the data) sent to the DBMS. The DBMS makes it possible to produce quick answers to spur-of-the-moment queries.

- **Improved decision-making**: Better-quality information (on which decisions are made) is generated, due to better-managed data and improved data access.

- **Increased end user productivity**: The availability of data and the tools that transform data into usable information encourages end users to make quick, informed decisions.

SPOT CHECK 1

Let's see what you can remember from this section.

1. What is the difference between data and information?
2. What are 3 advantages of using a database management system?

TYPES OF DATABASES

There are many different types of databases. These databases can be classified according to the number of users supported, where the data are located, the type of data stored, the intended data usage and the degree to which the data are structured.

A database can be classified as either **single-user** or **multi-user**. A database that only supports one user at a time is known as a single-user database. With a single user database, if user A is using the database, users B and C must wait until user A

is done. A desktop database is a single-user database that runs on a personal computer. A multi-user database, on the other hand, supports multiple users at the same time. A workgroup database is a multi-user database that supports a relatively small number of users (usually less than 50) or a specific department within an organisation. When a multi-user database supports many users (more than 50) and is used by the entire organisation, across many departments, it is known as an enterprise database.

A database can also be classified based on location. A **centralised database** is a database that supports data located at a single site, while a **distributed database** supports data distributed across several different sites.

A popular way of classifying databases is based on how they will be used and on the time sensitivity of the information gathered from them. An example of this is an **operational database**, which is a database that is designed to primarily support a company's day-to-day operations. Operational databases are also known as online transaction processing (OLTP), transactional, or production databases.

The degree to which data is structured is another way of classifying databases. Data that exist in their original, or raw, state are known as **unstructured data**. In other words, they are in the format in which they were collected. **Structured data** are the result of formatting unstructured data to facilitate storage, use, and the generation of information. You apply structure based on the type of processing that you intend to perform on the data. For example, imagine that you have a stack of printed invoices. If you just want to store these invoices so that you are able to retrieve them or display them later, you can scan them and save them in a graphical format. However, if you want to derive information from them, such as monthly totals or average sales, having the invoices in a graphical format will not be useful. You could instead store the invoice data in a structured spreadsheet format so that you can perform the desired computations.

Analytical databases focus on storing historical data and business metrics used exclusively for tactical or strategic decision making. They typically comprise two components; a data warehouse and online analytical processing (OLAP) front end. Analytical databases allow the end-user to perform advanced data analysis of business data using sophisticated tools. A data warehouse, on the other hand, focuses on storing data used to generate the information required to make tactical or strategic decisions.

A type of database you may have heard of is the **relational database**. A relational database is a database structured to recognize relations between stored items of information. To put it more simply, a relational database organises data into tables

and links them based on defined relationships. These relationships then enable you to retrieve and combine data from one or more tables with a single query.

Clearly there are many ways of storing data, each with its own benefits and limitations. Therefore, databases can be designed differently to meet different needs. We are going to consider two of the most popular types of databases in this task: relational databases and NoSQL databases.

Relational databases

Relational databases store information about different entities and the relationships between them. The image below is an example of an entity-relationship diagram (ERD) that is used to describe the relationships between certain entities.

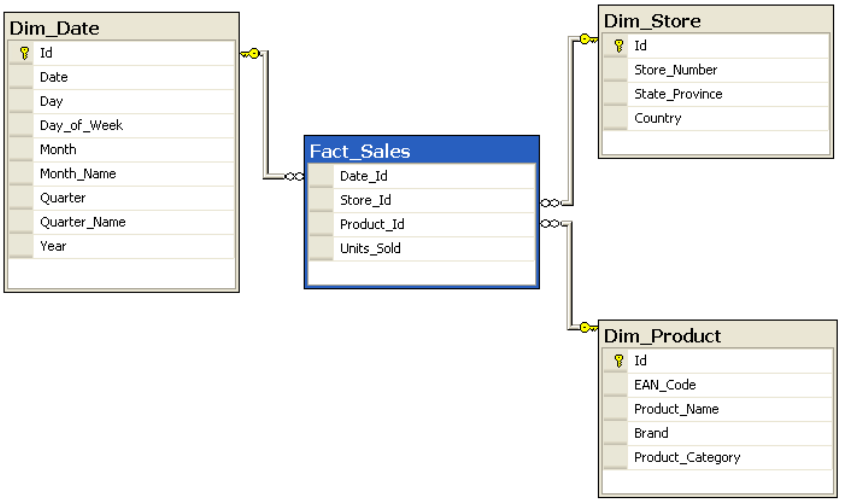


Image source:
https://en.wikipedia.org/wiki/Star_schema#/media/File:Star-schema-example.png

In a relational database, an entity is a table that stores all the data about a certain thing. For example, you may want to store information about all your customers in a database. You would then create a customer table (customer entity), which could look something like the one shown below:

C_NAME	C_PHONE	C_ADDRESS	C_POSTCODE	A_NAME	A_PHONE	TP	AMT	REN
Alfred Smith	082 345 2341	207 Willow St, Port Elizabeth	6390	Leah Hahn	084 259 2073	T1	R100.00	05-Apr-2021
Kathy Dunne	083 567 9012	556 Bad St, Cape Town	7100	Alex Alby	085 785 3938	S2	R250.00	16-Jun-2021
Paul Farris	076 782 1232	2148 High St,Benoni	1522	Leah Hahn	084 259 2073	T2	R850.00	22-sep-2021

C_NAME = customer name
C_PHONE = customer phone
C_ADDRESS = customer address
C_POSTCODE = customer postcode
A_NAME = agent name
A_PHONE = agent phone
TP = insurance type
AMT = insurance policy amount in thousands of R
REN = Insurance renewal date

The CUSTOMER table contains 3 records. Each record is composed of 9 fields: C_NAME, C_PHONE, C_ADDRESS, C_POSTCODE, A_NAME, A_PHONE, TP, AMT, and REN. Each record describes a specific customer; each customer is an instance of that entity.

If you were designing a database for a store, you might also have a product entity that stores all the information about the products you sell. Your database would then contain a product table and a customer table. The database would also save information about the relationship between the two entities. For example, it would store information about which products a particular customer brought on a particular date. How this is implemented in a relational database, however, is beyond the scope of this course.

NoSQL databases

A problem with relational databases is that their performance degrades as the volume of data increases. Many web applications have to store massive amounts of data. Imagine the amount of data that companies like Amazon and Google have to store, for example. Addressing this problem led to the development of NoSQL databases. You are using a NoSQL database every time you search for a product on

Amazon, watch a video on Youtube, or send a message to a friend on Facebook. NoSQL databases generally have the following characteristics:

- They are not based on the relational model
- They support distributed database architectures i.e. servers in different areas
- They provide high scalability, high availability, and fault tolerance
- They support very large amounts of sparse data
- They are geared toward performance rather than transaction consistency

There are several types of NoSQL databases. Some of these are briefly described below:

1. **Key-value store databases**: This is the simplest form of NoSQL database. Every item in the database is stored as a key (used to identify the value) and its value.

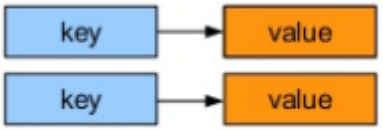


Image source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

2. **Column-oriented databases**: Like key-value store databases, a key is used to identify values but instead of the key identifying only one value, it can be used to identify multiple values.

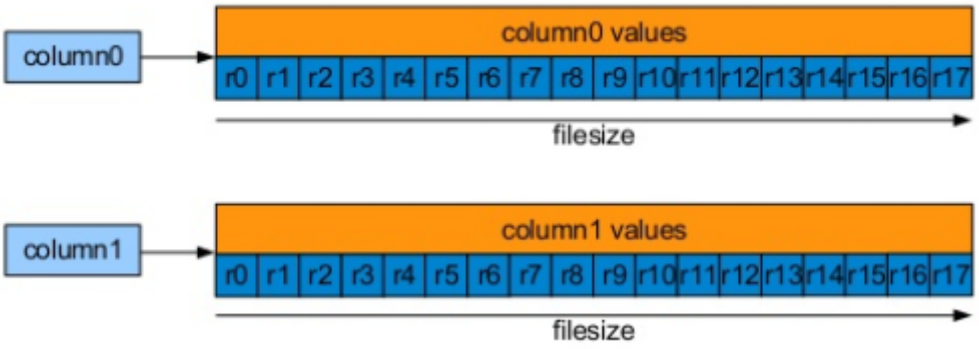


Image source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

3. **Document store databases**: With this type of database, a key is used to identify a particular document. Documents are stored in recognised formats like XML, JSON, PDF etc.

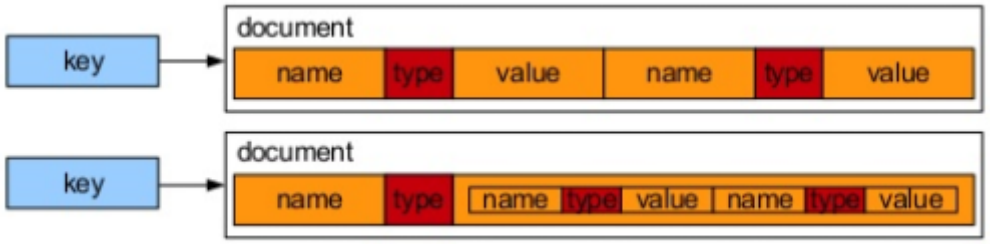
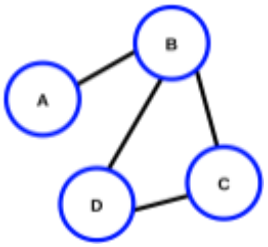


Image source:
https://www.slideshare.net/arangodb/introduction-to-column-oriented-databases

4. **Graph databases**: This database uses the graph data structure to store data. Graphs contain nodes (A, B, C, and D in the image below) and edges (the lines connecting the nodes, so AB, BD, BC, and DC in the image below). In graph databases, nodes are objects and edges are the relationships between these objects. Social networking applications, like Facebook, often store data using graph databases because this model is good for tracking the relationships between objects.



5. **Object-oriented databases**: These databases combine OOP and database principles. These databases are tied to specific programming languages.

MONGODB

You are going to be working with MongoDB. MongoDB is a document store, NoSQL database. A MongoDB is made up of collections and documents.

- **Collection**: A collection is a group of documents. It is similar to an entity or table when working with relational databases.

- **Documents**: In relational databases, records are stored in tables. An example of a record in the CUSTOMER table we considered earlier would be Alfred Smith. MongoDB uses documents instead of records (or rows in a table) to store data (i.e. with a MongoDB database, Alfred Smith's data would be stored in a document instead of in a row in the CUSTOMER table).

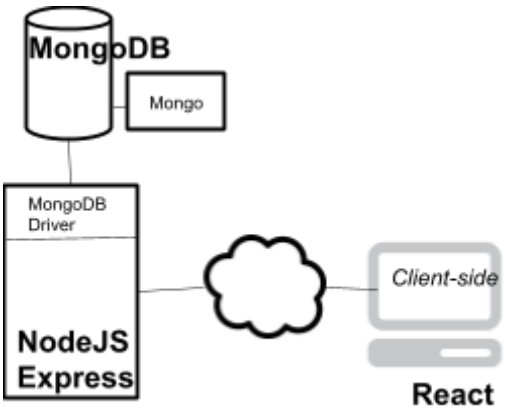
MongoDB uses BSON documents. BSON stands for Binary JSON. BSON uses JSON files and stores type information. JSON files are just text information; this means that it has to be parsed if you want to program with it. Since BSON stores type information it is quicker and more efficient to use than JSON.

Extra resource

Watch **this video** to let MongoDB's CTO and Co-Founder, Eliot Horowitz, give you a comprehensive introduction to MongoDB in under 5 minutes.

MONGODB IN A FULL STACK WEB APPLICATION

Before we start using MongoDB in the next task, it is important to first see how MongoDB fits into full-stack web development and what other tools are needed to get it to work properly.



Consider the image above. As you already know, to use your app, clients will interact with a web server that will be running Node.js. Node.js will route all requests and perform whatever server-side logic is needed by our app. If our user wants to access, add, or change information that needs to persist, they will need access to the Database. MongoDB is used for our database. An important component of MongoDB is Mongo. *Mongo is MongoDB's administrative shell*. It is a C++ program that allows you to execute instructions on the database from a command line interface. Mongo allows you to use the MongoDB query language.

For Node.js to be able to communicate with MongoDB, it also makes use of *MongoDB drivers*. The official MongoDB driver can be installed using NPM (more on this later).

Compulsory Task 1

Follow these steps:

- Create a file called **answers.txt**. Answer the following questions in answers.txt:
    - What are the key benefits of using MongoDB? (See **here**).
    - A key difference between relational databases and MongoDB is that MongoDB is schemaless. Explain what this means and the pros and cons of a schemaless database.
    - Companies like Amazon use NoSQL databases. Why do you think they choose to use NoSQL databases instead of relational databases?
    - Give an example of when it would be better to use a relational database rather than a NoSQL database. Justify your answer.

Compulsory Task 2

Follow these steps:

- Create a JSON document that stores all the personal information that a company like Hyperion might need from you as a student.

- What is the difference between a JSON and BSON document?

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

SPOT CHECK 1 ANSWERS

1. Data are raw facts that have not yet been processed to reveal their meaning. Information, on the other hand, is the result of processing the raw data to reveal its meaning. Data processing can be as simple as organising data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modelling.
2. (any 3)
    a. Improved data sharing
    b. Better data integration
    c. Minimised data inconsistency
    d. Improved data access
    e. Improved decision-making
    f. Increased end-user productivity