OCR A Level Computer Science

Your notes

3.2 Databases

Contents

Your notes

Relational Databases

What is a Database?

- A **Database** is an organised collection of data
- It allows **easy storage, retrieval, and management** of information
- **Electronic databases** offer a number of key benefits:
  - Easier to add, delete, modify and update data
  - Data can be backed up and copied easier
  - Multiple users, from multiple locations, can access the same database at the same time

Database Terminology

Term	Definition
Field	A single piece of data in a record
Record	A group of related fields, representing one data entry
Table	A collection of records with a similar structure
Primary key	A unique identifier for each record in a table. Usually an ID number
Compound primary key (sometimes called composite)	A combination of (2 or more) fields that is unique for all records
Foreign key	A field in a table that refers to the primary key in another table. Used to link tables and create relationships
Secondary key	A field or fields that are indexed for faster searching

Database Management System (DBMS)	Software used to manage databases. Examples include MySQL, Oracle, Microsoft SQL Server, PostgreSQL

Your notes

Primary, Foreign & Secondary Keys

Visualising a database

Table:orders

order_id	product	total	customer_id
1	Paper	500	5
2	Pen	10	2
3	Marker	120	3
4	Ruler	1000	1
5	Erasers	20	4

Table:customers

customer_id	forename	surname	phone	country
1	John	Smith	07373 929122	UK
2	Iram	Iravani	07234 543422	Iraq
3	Wu	Zhang	04563 523427	China
4	Anne	James	09378 482894	USA
5	Khalid	Shirvani	02343 536522	France

- This database has 2 **tables** : orders and customers
- **Fields**:
  - In the orders table are *order_id, product, total and customer_id*
  - In the customers table are *customer_id, first_name, last_name, phone and country*
- A **record** would be 1 complete row in either table

- order_id is the **primary key** for the orders **table**
  customer_id is the **primary key** for the customers **table**
- In the orders **table** customer_id is a **foreign key** - it links an order back to the customer that made the order in the customer **table**


Your notes

Indexing (Secondary Keys)

- **Indexing** is a technique used to speed up data retrieval in a **database**
- It works in a similar way to the index in a book
- If a student had a maths book and wanted to find the section on factorising, they could start at the first page and look at each page in turn until they found the section they wanted
- But this would be slow, so it is better to look in the **index** to find where the factorising section is and just go directly to it
- Likewise in a database, certain columns can be **indexed** so that the DBMS does not have to look at every single **record** during a search and can just go to the relevant **records** directly
- This can greatly **speed up searches**
- Fields that are indexed are known as **secondary keys**

Flat File vs Relational Database

- A **flat file database** is one that stores all data in a single table
- It is simple and easy to understand but causes data redundancy, inefficient storage and is harder to maintain
- A **relational database** is one that organises data into multiple tables
- It uses keys to connect related data which reduces data **redundancy**, makes efficient use of storage and is easier to maintain

Consider this example **flat file** table of students


Your notes

ID	first_name	last_name	Personal tutor	FormRoom
1	sam	smith	Roger Hinds	6b
2	fred	lynch	Jess Little	8j
3	depak	noor	Roger Hinds	6b
4	archie	henns	Mary Kent	8k
5	helga	jordan	Mary Kent	8k
6	lizzy	bell	Mary Kent	8k
7	xavier	horten	Jack Berry	3m

- This table has redundant data - the tutor and form room information repeats
- This is inefficient
- If a tutor changed their name we would need to find all instances of that name and change them all
- Missing any would mean the **table** had inconsistent data

A **relational** database would solve this issue:

- A new **table** could be created to store the tutor information and the tutor information in the student table could be moved to the new **table**. Then a **foreign key** in the student **table** (TutorID) could link a student to their tutor

Your notes

ID	first_name	last_name	TutorID
1	sam	smith	1
2	fred	lynch	2
3	depak	noor	1
4	archie	henns	3
5	helga	jordan	3
6	lizzy	bell	3
7	xavier	horten	4

TutorID	Tutor Name	FormRoom
1	Roger Hinds	6b
2	Jess Little	8j
3	Mary Kent	8k
4	Jack Berry	3m

- Now the name of each tutor and their form room is stored only once
- This means if they change only one piece of data, the data is updated in the entire **database** and **Inconsistency** is avoided

Worked Example

Customers' details are stored in the flat file database table Customer. An extract of the table is shown below

CustomerID	Surname	Title	Phone	CarReg
JJ178	James	Mr	(0121) 343223	DY51 KKY
HG876	Habbick	Miss	(01782) 659234	PG62 CRG
EV343	Elise	Mrs	(07834) 123998	HN59 GFR
PG127	Pleston	Mr	(07432) 234543	JB67 DSF

Describe one problem that would arise with the flat file database structure if a customer wanted to insure more than one car at the same time

[2]

Answer:

One mark per pair:

Only one customer entry allowed (because of key field)...

so would not be able to add second entry **[1]**

Customer data already present/would be repeated...

resulting in redundant data/wasted space/inconsistencies should changes be made **[1]**

Your notes

Data Normalisation

Your notes

First Normal Form (1NF)

What is Normalisation?

- Normalisation is the process of **organising a database** to reduce data **duplication** and improve data **accuracy** and **consistency**
- Achieved by applying a set of guidelines (**forms**), each with specific **rules** and **requirements**
- Enhances database **efficiency** and **maintainability**
- Provides **consistency** within the database

First Normal Form (1NF)

For a **table** to be in first normal form it must:

- **Contain atomic values**
  - Each column in a table must contain single, indivisible values
- **Have no repeating groups**
  - Columns must not contain arrays or lists of values
- **Have unique column names**
  - Each column must have a unique name within the table
- **Have a unique identifier (primary key)**
  - Each row must have a unique identifier to distinguish it from other rows

This customers table below has no primary key and the name is stored in one field so is not atomic. This table is not in first normal form

Your notes

Table:customers

name	phone	country
John Smith	07373 929122	UK
Iram Iravani	07234 543422	Iraq
Wu Zhang	04563 523427	China
Anne James	09378 482894	USA
Khalid Shirvani	02343 536522	France

This customers table below has a primary key and the name is stored in two fields so it is atomic

This table is in **first normal form**

Table:customers

customer_id	forename	surname	phone	country
1	John	Smith	07373 929122	UK
2	Iram	Iravani	07234 543422	Iraq
3	Wu	Zhang	04563 523427	China
4	Anne	James	09378 482894	USA
5	Khalid	Shirvani	02343 536522	France

Second Normal Form (2NF)

Second Normal Form (2NF)

For a table to be in second normal form it must:

- **Fulfil all 1NF requirements**
- **Only apply to tables with a compound primary key**
- **Have full functional dependency**
  - **All non-prime attributes (attributes not part of the primary key) must be fully dependent on the primary key**
- **Have no partial dependencies**

- Non-prime attributes must not depend on only part of the primary key (in case of a composite primary key)
- Separate tables should be created for partially dependent attributes

In this table below, Course Title only depends on part of the compound primary key (the course code) and not the Date so this table is not in second normal form

Course (pk)	Date(pk)	Course Title	Room	Capacity	Available
SQL101	03/01/2020	SQL Basics	4A	12	4
DB202	03/01/2020	Database Design	7B	14	7
SQL101	04/05/2020	SQL Basics	7B	14	10
SQL101	15/05/2020	SQL Basics	12A	8	8
CS50	31/05/2020	C Programming	4A	12	11

Third Normal Form (3NF)

Third Normal Form (3NF)

For a table to be in third normal form it must:

- **Fulfil all 2NF requirements**
- **Have no transitive dependencies**
  - Non-prime attributes must not depend on other non-prime attributes
- **Have each non-prime attribute dependent solely on the primary key, not on other non-prime attributes**
- **Have separate tables for attributes with transitive dependencies, and the tables should be linked using a foreign key**

In this table below, the certificate depends on the title - this a transitive dependency and so this table is not in third normal form

FilmID	Title	Certificate	Description
12034	Saw IV	18	Eighteen and over
12035	Spiderman 2	12A	Age 12 and over
12036	Shrek	U	Universal

Your notes

Your notes

Examiner Tips and Tricks

For a table to be in **second normal form** it has to be in **first normal form**.

For a table to be in **third normal form** it has to be in **second normal form**.

So if asked for the rules of either second or third normal form make sure you say this.

Worked Example

An airport holds details of flights in a database using the table Flight. An extract of the table is shown below.

FlightID	FlightNumber	DestinationCode	DestinationName	DepartureDate	DepartureTime
1355	OC0089	JFK	John F. Kennedy	03/07/18	09:50
1453	CS1573	LHR	Heathrow	03/07/18	10:30
1921	OC7750	JFK	John F. Kennedy	04/07/18	08:30
1331	AM0045	YHZ	Halifax	04/07/18	14:25
1592	HB0326	RTM	Rotterdam	04/07/18	19:10
1659	CS0123	LHR	Heathrow	04/07/18	07:20

The airline wishes to ensure the database is normalised.

i) Describe why the database can be considered to be in First Normal Form

[2]

ii) Describe why the database can be considered to be in Second Normal Form

[2]

iii) Describe why the database can not be considered to be in Third Normal form

[2]

Answer:

No Repeating fields/data. Data is atomic. Has a primary Key **[2]**

Is in First Normal Form. Every field is dependent on the primary key **[2]**

Has a transitive relationship. A non-key field depends on another non-key field. DestinationName depends on DestinationCode **[2]**


Your notes

Entity Relationship Diagrams

Your notes

Entity Relationship Types

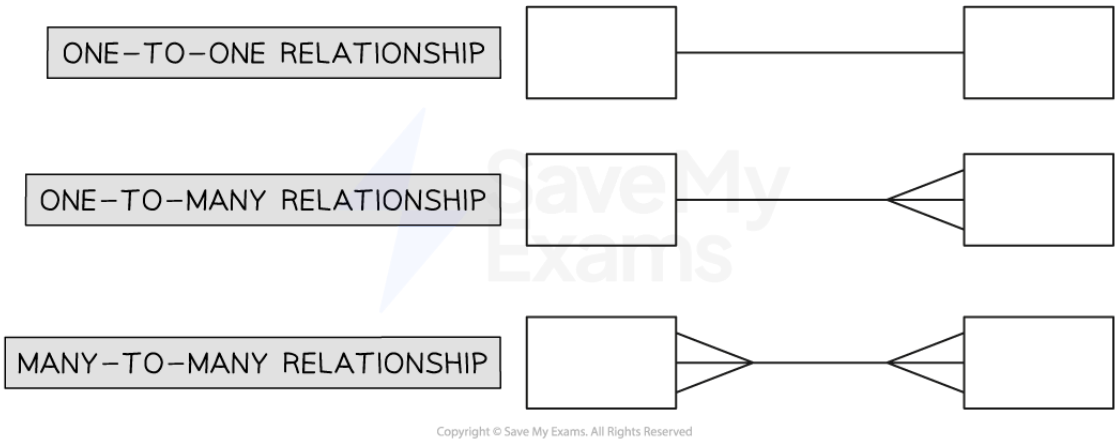
What is an Entity?

- An **entity** in something worthy of capturing and storing data about e.g. *students*, *orders*, *products*, *courses*, *customers*
- Entities become tables in a **relational database**
- Relational databases store different entities in **separate tables**. Linking tables depends on the relationships between entities.
- There are 3 types of (sometimes called degrees of) **relationships:**
  - **One to one**
  - **One to many**
  - **Many to many**
- Imagine a company has
  - A table of products
  - A table of customers
  - A table of the orders the customers have made
- What is the **relationship** between a customer and an order?
  - **One** customer can make multiple (**many**) orders
  - But each order relates to a specific (**one**) customer
  - So the relationship between customer and order is **one to many**
- Now consider the **relationship** between a product and an order
  - An order could have more than one (**many**) products on it
  - A product could be on more than one (**many**) order
  - So the **relationship** between order and product is **many to many**
- **One to one** relationships also exist but are not very common in databases

Drawing Entity Relationship Diagrams

Your notes

- **Entity Relationship Diagrams** (**ERDs**) are simple diagrams that represent the entities (tables) that will be in a database and the relationships between these entities
- The entities are drawn as boxes with the **entity** name in
- The relationships are drawn in as what is known as 'crow's feet notation'
- This is how to draw the relationships in the exam:



- The names of the entities would go inside the boxes

Examiner Tips and Tricks

These diagrams are simple but tell us some important things about the database:

- The names of all the tables
- Which tables will have a foreign key - when an entity has a 'many' relationship against it that means it will have a foreign key in it that links to the primary key of the connected entity

Worked Example

An insurance company's offices have a large number of black and white printers

The company's technicians keep accurate records of the printers in the building, and the quantity of toner cartridges in stock, in a flat file database. An extract of the database is shown:

Your notes

Printer Model	Location	Notes	Cartridge Code	Quantity in stock	Re-order URL
LasPrint LP753	office 3		LP-7XB	12	www.megacheapprint.com/toner / LP-7XB
LasPrint LP710	office 6	drum replaced	LP-7XB	12	www.megacheapprint.com / toner / LP-7XB
Zodiac ZN217	reception		Zod17	4	www.zodiaclaserprinting.com / shop / Z17
Zodiac ZN217	conference Room 2	had to add RAM	Zod17	4	www.megacheapprint.com / toner / LP-7XB
LasPrint LP753	office 8		LP-7XB	12	www.megacheapprint.com / toner / LP-7XB

A relational database is created with three tables:

- **PrinterModel: this stores all the data about each model of printer**
- **PrinterInstance: this stores the data about each individual printer in the building**
- **Cartridge: this stores information about the toner cartridges**

Draw an entity-relationship diagram to show the relationships between the three tables

[4]

Answer:

Entities and relationships drawn using standard notation **[1]**

Cartridge linked to PrinterModel, PrinterModel linked to PrinterInstance with no other links **[1]**

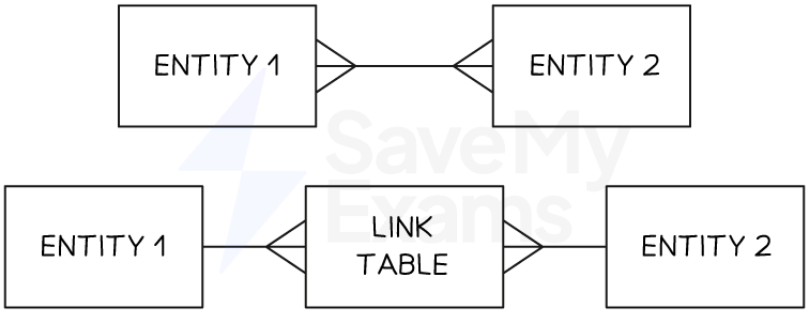
1:M relationship from Cartridge to Printer Model **[1]**

1:M relationship from PrinterModel to PrinterInstance **[1]**

Resolving Many to Many Relationships

- **Many to many** relationships cannot be implemented into an actual database and need to be '**resolved**'

- This involves creating a new table known as a '**link**' table that goes between the entities
- This new table **links** the entities that have the **many to many** relationships together



Your notes

Capturing, Selecting, Managing & Exchanging Data

Your notes

Capturing Data

How can Data be Captured?

- Forms
- OMR (Optical Mark Recognition)
- OCR (Optical Character Recognition)
- Sensors
- Barcodes
- Data Mining

Forms

- Collect user input
- Organise data in structured format
- Common in web applications

Your notes

PLACE OF SUBMISSION* SELECT COUNTRY

APPLICANT'S INFORMATION

SURNAME

GIVEN NAME*

HAVE YOU EVER CHANGED YOUR NAME? IF YES, CLICK THE BOX AND GIVE DETAILS ☐

SEX* SELECT GENDER

DATE OF BIRTH*

COUNTRY OF BIRTH* SELECT COUNTRY

PLACE OF BIRTH*

CURRENT NATIONALITY* SELECT COUNTRY

NATIONAL IDENTIFICATION NUMBER

VISIBLE MARK*

APPLICANT'S PASSPORT DETAILS

PASSPORT NUMBER*

DATE OF ISSUE*

DATE OF EXPIRY*

PLACE OF ISSUE*

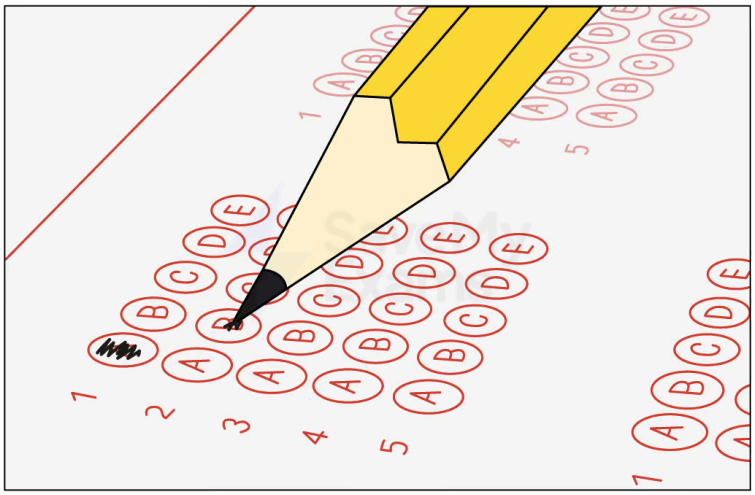
PREVIOUS PASSPORT NUMBER

A data capture form

OMR (Optical Mark Recognition)

Your notes

- Detects marked areas on paper by using a special machine to read the marks
- Used for exams, surveys, lottery tickets
- Automates data collection and entry



An OMR form being filled in

OCR (Optical Character Recognition)

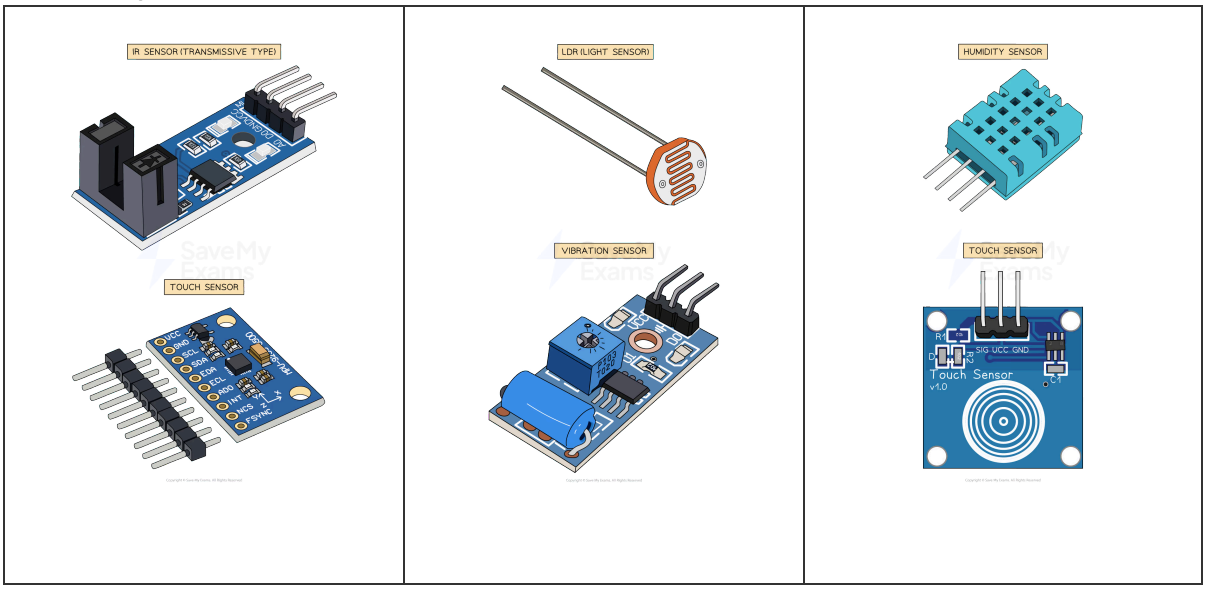
- Converts printed or handwritten text into digital format
- Useful for digitising documents
- Assists in searching and editing text

Sensors

- Devices that detect and respond to changes in environment
- Convert physical signals into digital data
- Facilitates automated data collection and real-time monitoring
- Used in various applications:
  - Temperature sensors
  - Pressure sensors

- Proximity sensors
- Light sensors
- Motion sensors
- Humidity sensors
- Gas sensors
- Force sensors
- Acoustic sensors
- Magnetic sensors

Your notes



Different Types of Sensor

Barcodes

- Machine-readable representation of data using parallel lines of varying widths and spacings
- Commonly used for tracking items, inventory management, and point-of-sale systems
- Two main types:
  - 1D (One-dimensional) barcodes: represent data using parallel lines (e.g. on products in shops)
  - 2D (Two-dimensional) barcodes: use geometric patterns like squares, dots, or hexagons to store data (e.g. QR code)
- Advantages of Barcodes

- Fast and accurate data entry
- Reduces human error

A Typical Barcode Scanner

Data Mining

- Process of discovering hidden patterns, correlations, and insights from large datasets
- Involves techniques and algorithms from fields like machine learning, statistics, and artificial intelligence
- Supports decision-making by transforming raw data into valuable information
- Applications of data mining
  - **Marketing**: Customer segmentation, market basket analysis, and targeted advertising
  - **Finance**: Fraud detection, credit scoring, and portfolio management
  - **Healthcare**: Disease prediction, patient clustering, and drug discovery

Your notes

- **Manufacturing**: Quality control, predictive maintenance, and supply chain optimisation
- **Telecommunications**: Network monitoring, customer analysis, and service improvement


Your notes

Selecting Data

Query By Example (QBE)

- User-friendly method for constructing database **queries** using a visual interface
- Allows users to search for data by providing an example of the desired output
- **Key Features**
  - Visual representation: QBE uses a grid or form-based interface, where users can enter criteria in columns representing database fields
  - Intuitive: Users don't need to learn complex query syntax, making it accessible for non-technical users
  - Flexible: Allows for simple to complex queries, including filtering, sorting, and joining multiple tables
- **How QBE Works**
- Users enter criteria in the appropriate columns or fields in the QBE grid or form
- The QBE system translates the user's input into an equivalent SQL query or other query language
- The query is executed against the database, and the results are displayed to the user
- **Common Query Operations**
- **Filtering**: Users can specify conditions or criteria in the QBE grid to retrieve specific records (e.g., all customers from a particular city)
- **Sorting**: Users can indicate the desired sorting order for the results (e.g., ascending or descending by last name)
- **Joining**: Users can combine data from multiple related tables by specifying relationships between the tables in the QBE grid
- **Aggregation**: Users can perform calculations or summaries on the data (e.g., counting the number of records, calculating averages, or summing values)

Benefits and Drawbacks of Query By Example (QBE)

Benefits	Drawbacks

Easy to learn and use, even for non-technical users	Less powerful and flexible than SQL for complex queries and data manipulation
Visual interface makes it simple to understand and modify queries	May not support advanced database features, such as stored procedures or **triggers**
Provides a more accessible way to perform database searches compared to writing SQL queries	Can be slower than SQL queries for large datasets or complex operations


Your notes

Managing & Exchanging Data

Managing Data

- After a database has been created, there must be easy ways of being able to **manage the data**, this includes:
  - **Add new data**
  - **Edit/modify existing data**
  - **Delete data**
- This can be achieved by:
  - A **database manipulation language** (DML) such as **SQL**
  - Built in **facilities of a Database Management System** (DBMS)

Exchanging Data

CSV (Comma Separated Values)

- **Simple** data exchange format
- Stores tabular data in **plain text**
- Uses **commas to separate** values
- **Widely supported** and easy to parse

```
student_id,first_name,subject,predicted_grade,teacher
001,"Ella","Computer Science",9,"RHA"
002,"Fynn","Computer Science",8,"RHA"
003,"Zarmeen","Computer Science",9,"RHA"
004,"Zahra","Computer Science",5,"RHA"
005,"Harrison","Computer Science",9,"RHA"
```

Your notes

JSON (JavaScript Object Notation)

- **Lightweight** data interchange format
- **Human-readable** and **easy to analyse**
- Uses **Key-value pairs**
- Common in **web applications**

```
'{"first_name":"Ella", "subject":"Computer Science", "predicted_grade":9}'
'{"first_name":"Fynn", "subject":"Computer Science", "predicted_grade":8}'
'{"first_name":"Zarmeen", "subject":"Computer Science", "predicted_grade":9}'
'{"first_name":"Zahra", "subject":"Computer Science", "predicted_grade":5}'
'{"first_name":"Harrison", "subject":"Computer Science", "predicted_grade":9}'
```

EDI (Electronic Data Interchange)

- **Standardised** electronic communication method
- Transfers **documents and data** between businesses
- **Reduces** paper usage and manual processes
- **Streamlines** transactions and increases **efficiency**

APIs (Application Programming Interfaces)

- A set of **protocols** that allow **different software applications** to **communicate** with each other
- Enables developers to **integrate different services** and functionalities into their applications
- Facilitates **data exchange** between applications and platforms

- Can be **RESTful** (Representational State Transfer), **SOAP** (Simple Object Access Protocol), or other types
- **Advantages of APIs**
  - Encourages **code reuse** and modular programming
  - **Simplifies** application development by providing pre-built functionalities
  - Facilitates **seamless integration** of different services
  - Promotes innovation by enabling developers to **build upon existing technologies**


Your notes

Memory Sticks

- **Portable storage devices**, also known as USB flash drives or thumb drives
- Use **flash memory** to store and transfer data between computers and other devices
- Connect to devices via **USB** (Universal Serial Bus) port

Benefits and Drawbacks of Memory Sticks

Benefits	Drawbacks
Easy to use: Plug-and-play functionality with no need for additional software	**Limited storage capacity** compared to external hard drives
Portable: Compact size allows for easy transport and storage	**Data loss risk** due to physical damage, loss, or theft
Durable: No moving parts, making them resistant to physical damage	**Slower transfer speeds** compared to other storage solutions
Compatible: Widely supported across different operating systems and devices	

Email

- **Electronic communication system** used for **exchanging messages and files** between users
- **Requires internet access** and an email account with an email service provider

Benefits and Drawbacks of Email

Benefits	Drawbacks

Fast: Allows for near-instant communication across the world	**Limited file size:** Most email service providers impose restrictions on attachment sizes
Convenient: Accessible on various devices, including computers, smartphones, and tablets	**Security risks**: Vulnerable to phishing attacks, spam, and data breaches
Organised: Stores and organises messages in folders, such as Inbox, Sent, and Drafts	**Privacy concerns**: Email messages and attachments can be intercepted, read, or modified by unauthorised parties
Versatile: Supports the attachment of various file types and sizes	**Reliability**: Delivery issues can occur due to server problems, incorrect email addresses, or spam filters


Your notes

SQL

Your notes

Selecting Data (Single Table)

What is SQL?

- **SQL** (Structured Query Language) is a programming language used to interact with a **DBMS**.
- The use of SQL allows a user to:
  - **Select data (single table)**
  - **Select data (multiple tables)**
  - **Insert data**
  - **Delete records**
  - **Delete tables**

Selecting Data (Single Table) Commands

Command	Description	Example
SELECT	Retrieves data from a database table	**SELECT * FROM** users; (retrieves all data from the 'users' table) **SELECT** name, age **FROM** users (retrieves names and ages from the 'users' table)
FROM	Specifies the tables to retrieve data from	**SELECT** name, age **FROM** users; (retrieves names and ages from the 'users' table)
WHERE	Filters the data based on a specified condition	**SELECT * FROM** users **WHERE** age > 30; (Retrieves users older than 30)
LIKE	Filters the data based on a specific pattern	**SELECT * FROM** users **WHERE** name **LIKE** 'J%';

		(retrieves users whose names start with 'J')
AND	Combines multiple conditions in a **WHERE** clause	**SELECT * FROM** users **WHERE** age > 18 **AND** city = 'New York'; (retrieves users older than 18 and from New York)
OR	Retrieves data when at least one of the conditions is true	**SELECT * FROM** users **WHERE** age < 18 **OR** city = 'New York'; (retrieves users younger than 18 or from New York)
WILDCARDS	'*****' and '**%**' symbols are used for searching and matching data '*****' used to select all columns in a table '**%**' used as a wildcard character in the LIKE operator	**SELECT * FROM** users; (retrieves all columns for the 'users' table) **SELECT * FROM** users **WHERE** name **LIKE** 'J%'; (retrieves users whose names start with 'J')


Your notes

Examples

- Select all the fields from the Customers table

Command:

```
SELECT * FROM Customers;
```

Output:

ID	Name	Age	City	Country
1	John Doe	30	New York	USA
2	Jane Doe	25	London	UK
3	Peter Lee	40	Paris	France

- Select the ID, name & age of customers who are older than 25


Your notes

Command:

```
SELECT ID, name, age
FROM Customers
WHERE Age > 25;
```

Output:

ID	Name	Age
1	John Doe	30
3	Peter Lee	40

- Select the name and country of customers who are from a country that begins with 'U'

Command:

```
SELECT Name, Country
FROM Customers
WHERE Country LIKE 'U%';
```

Output:

Name	Country
John Doe	USA
Jane Doe	UK

- Select all fields of customers who are from 'London' or 'Paris'

Command:

```
SELECT *
FROM Customers
WHERE City = 'London' OR City = 'Paris';
```

Your notes

Output:

ID	Name	Age	City	Country
2	Jane Doe	25	London	UK
3	Peter Lee	40	Paris	France

Worked Example

Customers' details are stored in the flat file database table Customer. An extract of the table is shown below

CustomerID	Surname	Title	Phone	CarReg
JJ178	James	Mr	(0121) 343223	DY51 KKY
HG876	Habbick	Miss	(01782) 659234	PG62 CRG
EV343	Elise	Mrs	(07834) 123998	HN59 GFR
PG127	Pleston	Mr	(07432) 234543	JB67 DSF

Write the SQL statement that would show only the CustomerID and Surname fields for customers with the Title "Miss" or "Mrs"

[4]

Answer:

SELECT CustomerID, Surname **[1]**
FROM Customer **[1]**
WHERE Title="Miss" **[1]**
OR Title = "Mrs" **[1]**

Selecting Data (Multiple Tables)

Command	Description	Example
Nested SELECT	A select within another select statement (nested). A mini select within the main one	**SELECT** * **FROM** users **WHERE** age > (**SELECT** AVG(age) **FROM** users); (retrieves users with an age greater than the average age)
JOIN (INNER JOIN)	Combines data from two or more tables based on a related column	**SELECT** users.name, orders.order_id **FROM** users **JOIN** orders **ON** users.user_id = orders.user_id; (retrieves user names and their corresponding order IDs)


Your notes

Examples

Table: Employees

ID	Name	Salary	Department	City
1	Fynn Roberts	45000	HR	London
2	Zarmeen Azra	52000	Sales	Manchester
3	Ella Stanley	39500	Marketing	Birmingham

- Select all fields for employees whose salary is bigger than the average salary of all employees

Command:

```
SELECT *
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

Output:

ID	Name	Salary	Department	City
2	Zarmeen Azra	52000	Sales	Manchester


Your notes

Inserting Data

Command	Description	Example
INSERT	Adds new data to a database table	**INSERT INTO** users (name, age) **VALUES** ('John Doe',25); (inserts a new user with the name 'John Doe' and age 25)

Example

Table: Employees

ID	Name	Salary	Department	City
1	Fynn Roberts	45000	HR	London
2	Zarmeen Azra	52000	Sales	Manchester
3	Ella Stanley	39500	Marketing	Birmingham

- Insert a new employee into the Employees table with the 'Name', 'Salary', 'Department' and 'City' fields

Command:

```
INSERT INTO Employees (Name, Salary, Department, City)
VALUES ('George Rope', 47250, 'Sales', 'Leeds');
```

Output:

ID	Name	Salary	Department	City

1	Fynn Roberts	45000	HR	London
2	Zarmeen Azra	52000	Sales	Manchester
3	Ella Stanley	39500	Marketing	Birmingham
4	George Rope	47250	Sales	Leeds


Your notes

Deleting Records

Command	Description	Example
DELETE	Removes data from a database table	**DELETE FROM** users **WHERE** age < 18; (deletes all users younger than 18 from the 'users' table) **DELETE FROM** users **WHERE** name="John"; (deletes a record where the name is John)

Example

Table: Employees

ID	Name	Salary	Department	City
1	Fynn Roberts	45000	HR	London
2	Zarmeen Azra	52000	Sales	Manchester
3	Ella Stanley	39500	Marketing	Birmingham
4	George Rope	47250	Sales	Leeds

- Delete all records from the Employees table whose department is 'Marketing'

Command:

```
DELETE FROM Employees
WHERE Department = 'Marketing' ;
```

Your notes

Output:

ID	Name	Salary	Department	City
1	Fynn Roberts	45000	HR	London
2	Zarmeen Azra	52000	Sales	Manchester
3	George Rope	47250	Sales	Leeds

Worked Example

A database stores information about songs on a music streaming service. One of the tables called Song has the fields Title, Artist, Genre and Length

A band called RandomBits removes their permission for their songs to be streamed. The company removes all the songs belonging to RandomBits from their service.

Write an SQL statement that will remove all songs by RandomBits from the table Song

[2]

Answer:
DELETE FROM Song **[1]**
WHERE Artist = "RandomBits" **[1]**

Deleting Tables

Command	Description	Example
DROP	Deletes a table in a database	**DROP TABLE** users; (deletes the 'users' table)

Example


Your notes

Table: Employees

ID	Name	Salary	Department	City
1	Fynn Roberts	45000	HR	London
2	Zarmeen Azra	52000	Sales	Manchester
3	Ella Stanley	39500	Marketing	Birmingham
4	George Rope	47250	Sales	Leeds

- Delete the Employees table

Command:

```
DROP TABLE Employees;
```

Output

```
"Table deleted successfully"
```

Your notes

Referential Integrity

What is Referential Integrity?

- Ensures **consistency** between related tables in a relational database
- Maintains valid **relationships** between **primary** and **foreign** keys
- There should not be foreign keys for which a matching primary key in the linked table does not exist
- **Foreign key constraints**
  - Value in a foreign key field must either:
    - Match a primary key value in the related table, or
    - Be **null** (if allowed)
  - Enforce **referential integrity**
  - Rules:
- **Cascade actions**
  - **CASCADE**: automatically makes changes to related records
  - **SET NULL**: sets foreign key value to null in related records
  - **SET DEFAULT**: sets foreign key value to its default in related records
  - **NO ACTION/RESTRICT**: prevents changes if related records exist
  - Update or delete actions will take effect everywhere in the database automatically
  - Types:

Benefits and Drawbacks of Referential Integrity

Benefits	Drawbacks
Ensures data consistency and accuracy	Can impact performance due to additional checks
Prevents orphaned records	May require additional planning and design

- Maintaining **referential integrity**
  - Use database management systems (DBMS) with **built-in support**
  - Implement triggers to **enforce** custom referential integrity rules
  - Regularly **validate** and clean up data to ensure **consistency**


Your notes

Worked Example

A hotel uses a computer system to keep track of room bookings. The hotel staff are able to query a database to discover which rooms are booked or which rooms are free

The hotel booking database enforces referential integrity.

Explain what is meant by the term 'referential integrity' and how this could potentially be broken

[2]

Answer:

Database/relationships are consistent and each foreign key links to an existing/valid primary key **[1]**
If a primary key is deleted, foreign keys that link to it are no longer valid so they should also be deleted - this is known as a cascaded delete **[1]**


Your notes

Transaction Processing

What is a Transaction?

- A **transaction** is a sequence of database operations treated as a single unit of work
- Ensures data **consistency** and **integrity** during simultaneous access
- Example: money transfer between bank accounts. The transaction here would be a withdrawal from one account and a deposit into another. Both operations must happen together or neither should happen
- Problems that arise from transaction processing
  - **Concurrency**: When multiple transactions are executed simultaneously, they might try to access or modify the same data, leading to inconsistencies
  - **Deadlock**: This occurs when two or more transactions are waiting for each other to release resources, causing them to wait indefinitely
  - **Data Integrity**: Transactions might leave the database in an inconsistent state if they fail in the middle of execution
  - **Isolation**: In a multi-user environment, one user's transaction might affect another user's transaction, leading to unpredictable outcomes
  - **Durability**: If a system fails after a transaction has been confirmed, it may result in loss of data
- Solutions to overcome problems that arise in transaction processing
  - **Locking**: Implementing a locking mechanism ensures that no two transactions can access the same data simultaneously. There are two types of locks: shared locks and exclusive locks
  - **Deadlock Prevention or Deadlock Detection**: Deadlocks can be prevented by ordering the way in which resources are requested or by having a timeout mechanism
  - **Logging and Recovery**: By keeping a log of all changes, the system can recover to a consistent state after a crash
  - **Commit and Rollback**: The commit operation saves all changes made in the transaction as permanent. The rollback operation reverts the changes made in the transaction
  - **Two-Phase Commit Protocol**: In **distributed systems**, this protocol ensures that a transaction is committed in all participating nodes, or none at all, to maintain consistency
  - **Concurrency control**

- Manages simultaneous access to data in a multi-user environment
- Techniques:
  - **Locking**: prevents multiple transactions from accessing the same data simultaneously
  - **Timestamping**: assigns a unique timestamp to each transaction

- **Transaction management**
  - Use database management systems (DBMS) with built-in transaction support
  - Implement custom transaction logic using SQL or other query languages
  - Test and monitor transactions to ensure ACID compliance and performance


Your notes

ACID

ACID (Atomicity, Consistency, Isolation, Durability)

- A set of rules that all Database Management Systems (**DBMS**) must use to ensure **data integrity**
- Guarantee **reliable processing** of transactions
- **Atomicity**
  - All operations in a transaction succeed or fail as a whole
  - If any operation fails, the transaction is rolled back
  - Ensures partial transactions do not occur
- **Consistency**
  - Ensures data remains in a consistent state after transactions
  - Transactions must follow database rules and constraints
  - Starts with a consistent state and ends with a consistent state
- **Isolation**
  - Transactions are isolated from each other
  - Intermediate states are not visible to other transactions
  - Prevents conflicts and data inconsistencies
- **Durability**
  - Committed transactions persist even in case of system failures
  - Ensures data is not lost once a transaction is complete

Your notes

Worked Example

RestaurantReview is a website that allows users to leave reviews and ratings for different restaurants

The website uses a database with the following structure

The database management system ensures referential integrity is maintained

Whenever a review is added to the system, the restaurant's average rating is updated

Describe what is meant by the term 'Atomic' in the context of ACID transactions. You should refer to the example of a review being added

[2]

Answer:

A transaction / review can only fully complete or not complete / cannot partially complete **[1]**

In this case, it should not be possible for the review to be added without the (average) rating being updated **[1]**

Record Locking

- **Record locking** is a technique used in database management systems (**DBMS**) to **prevent conflicting access** to data by multiple transactions or processes
- It **ensures data consistency** and **integrity** when multiple users or processes try to read, modify, or delete records simultaneously
- **Key Concepts**
  - Lock: A mechanism that prevents access to a database record by other transactions while a specific transaction is using the record
  - Lock granularity: Refers to the size of the locked data, ranging from a single row to an entire table
- **Types of Locks**

- Shared lock (Read lock): Allows multiple transactions to read a record simultaneously, but prevents modifications or deletions until the lock is released
- Exclusive lock (Write lock): Allows only one transaction to access and modify a record, blocking other transactions from reading or writing to the locked record until the lock is released

- **Benefits of Record Locking**
  - Maintains data consistency and integrity by preventing conflicting modifications
  - Allows concurrent read access to records without compromising data consistency
  - Improves database performance by enabling multiple users to access data simultaneously, while managing access to prevent conflicts


Your notes

Redundancy

- **Redundancy** can occur when the same piece of data is stored in more than one table in a database. This can either be by accident or by design
- It can lead to **inconsistencies** in the data and/or **wasted storage space**