Databases

IGCSE Computer Science

What is a database?

A **database** 数据库 is an organised store of data, kept so that it is easy to search, sort and update. At IGCSE you work with a **single-table database** 单表数据库 — all the data is held in one table.

A library card catalogue is a database on paper — records you can search and sort by index

Large modern databases are stored on servers in data centres

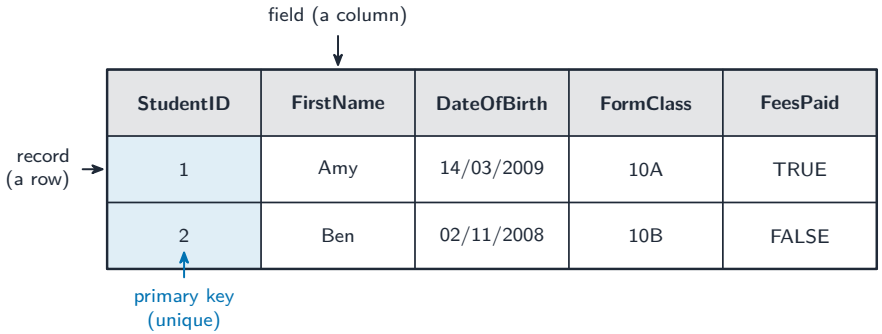
Records and fields

A database table is made of records and fields.

- A **record** 记录 is one row in the table — all the data about one thing (for example one student).
- A **field** 字段 is one column in the table — one piece of data that every record has (for example "First name").

StudentID	FirstName	DateOfBirth	FormClass	FeesPaid
1	Amy	14/03/2009	10A	TRUE
2	Ben	02/11/2008	10B	FALSE

Here each **row** is a record, and each **column** is a field.



Each row is a record and each column is a field; the primary key (StudentID) is unique for every record

Data types for fields

Each field stores one **data type** 数据类型. You choose the type that best fits the data.

text	a name
number	an age
date/time	a birthday
boolean	yes / no

Field types: text, number, date/time and boolean

Data type	Used for	Example
text/alphanumeric 文本	letters, digits and symbols	“10A”, “Amy”
character 字符	a single character	’M’
Boolean 布尔值	one of two values	TRUE / FALSE
integer 整数	a whole number	42
real 实数	a number with a decimal point	3.5
date/time 日期时间	a date or a time	14/03/2009

The primary key

A **primary key** 主键 is a field that holds a **unique** 唯一的 value for every record. No two records can have the same primary key, so it lets you pick out exactly one record.

In the table above, `StudentID` is a good primary key because every student has a different number. A field like `FormClass` would be a bad primary key, because many students share the same class.

Validation in a database

When data is put into a database, **validation** 验证 checks make sure it is sensible — for example a range check on an age field, or a presence check so a field is not left empty. (You saw these checks in topic 7.)

age field: range check 0–120

enter 16 → accepted

enter 200 → rejected

A range check accepts sensible values and rejects the rest

Structured Query Language (SQL)

Structured Query Language 结构化查询语言 (SQL) is a language used to **query** 查询 a database — to pick out the records you want. You must understand and complete SQL scripts.

SELECT, FROM and WHERE

- `SELECT` says **which fields** to show.
- `FROM` says **which table** to use.
- `WHERE` gives a **condition** 条件, so only matching records are shown.

```
SELECT name
FROM students
WHERE age > 15
```

gets the names of students older than 15

SELECT picks fields, FROM names the table, WHERE sets the condition

```
SELECT FirstName, FormClass
FROM Student
WHERE FeesPaid = TRUE;
```

This shows the first name and class of every student who has paid the fees.

Use `*` to select **all** fields:

```
SELECT *
FROM Student
WHERE FormClass = '10A';
```

ORDER BY

`ORDER BY` sorts the results. Use `ASC` for **ascending** 升序 (smallest first, A→Z) or `DESC` for **descending** 降序 (largest first, Z→A).

```
SELECT FirstName, DateOfBirth
FROM Student
ORDER BY DateOfBirth ASC;
```

AND and OR

Join conditions with `AND` (both must be true) or `OR` (at least one must be true).

```
SELECT FirstName
FROM Student
WHERE FormClass = '10A' AND FeesPaid = FALSE;
```

SUM and COUNT

- `SUM` adds up the values in a number field.
- `COUNT` counts how many records match.

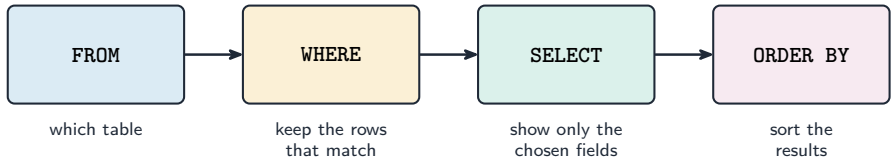
```
SELECT COUNT(StudentID)
FROM Student
WHERE FeesPaid = FALSE;
```

This counts how many students have not paid. `SUM` works the same way but adds a number field instead of counting rows.

Working out the output

To find the output of an SQL script, read it in this order:

1. `FROM` — which table;
2. `WHERE` — keep only the records that match the condition;
3. `SELECT` — show only the chosen fields;
4. `ORDER BY` — put the results in order.



Read an SQL query in this order: FROM (which table), WHERE (which rows), SELECT (which fields), ORDER BY (sort)

Following these steps, you can write down exactly which rows and columns the query returns.

Worked example. A `Book` table has the fields `Title`, `Author`, `Price` and `InStock`. Write a query showing the title and price of every book by Orwell that is in stock, cheapest first.

```
SELECT Title, Price
FROM Book
WHERE Author = 'Orwell' AND InStock = TRUE
ORDER BY Price ASC;
```

Build it in the reading order: `FROM` names the table; `WHERE` keeps only the matching records, and because there are two conditions they need `AND`; `SELECT` shows only the two fields asked for; `ORDER BY … ASC` sorts them. Text values go in **quotes**, and only the fields the question asks for belong in `SELECT` - adding `Author` just because you filtered on it is the commonest way to lose a mark here.

Exam tips

- A **record** is a row (all the data about one thing); a **field** is a column (one item that every record has).
- A **primary key** must be *unique* for every record, so it picks out exactly one record (StudentID, not FormClass).
- Learn the SQL parts: `SELECT` (which fields), `FROM` (which table), `WHERE` (the condition), `ORDER BY` (sort, ASC or DESC).
- Read a query in the order **FROM → WHERE → SELECT → ORDER BY** to work out its output.
- `COUNT` counts the matching records; `SUM` adds up a number field.