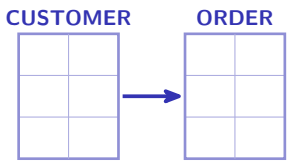
Databases

AS Computer Science ▪ Topic 8

A-Level Computer Science



What we will cover

1. Tables and keys 8.1
2. E-R diagrams 8.1.1
3. Normalisation 8.1.2
4. The DBMS 8.2
5. SQL: DDL and DML 8.3
6. SQL: joins and aggregates 8.3.1
7. Recap

1 Tables and keys

Sheet 8.1 – what you need to know

You must be able to:

- define entity, field, record and each kind of key, with an example
- explain referential integrity, including on a delete
- say why a database beats separate files

What the exam asks

"Give the definition of the following database terms, using an example from the database for each." **[6]**

"Tick one box to identify whether the database is in Third Normal Form (3NF) or not." **[2]**

"Explain what is meant by referential integrity, and how it applies to this database." **[2]**

Practise on exercise sheet 8.1, then try the real exam questions on past-paper sheet 8.1.

Why not just use files?

Separate files for each program duplicate data and go out of step.

duplication

the same customer stored in sales, accounts and support

inconsistency

one file is updated; the others are not

no sharing

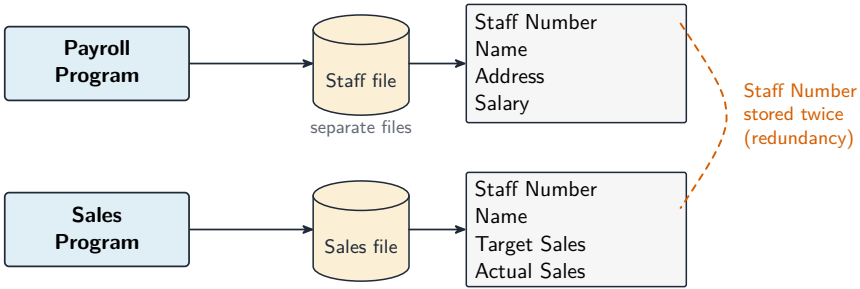
each program owns its files – no single source of truth

hard queries

“all orders for this customer” means writing a new program

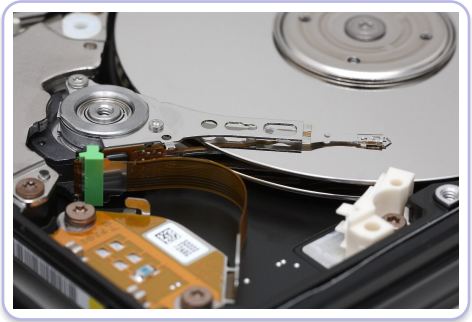
Before databases: one file per program

Programs stored data in **flat files 平面文件** – usually **one file per program**.



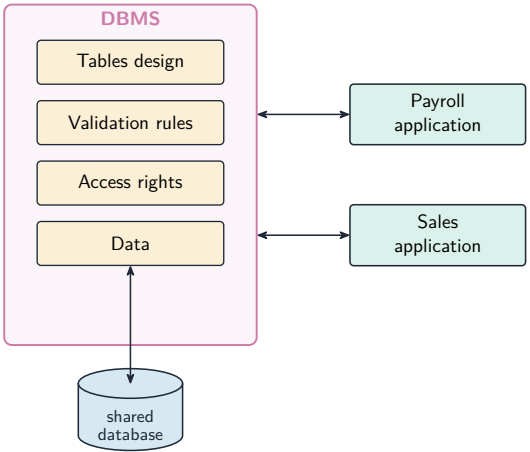
Its **limitations** at scale: redundancy (the same fact in several files), inconsistency (the copies disagree), dependence on each program's format, and no shared **security**.

The files sit on the storage devices



Flat files on a disk: each program owns its own, which is exactly where the trouble starts.

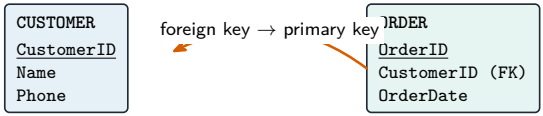
The database approach



A **relational database** 关系数据库 stores data in tables, managed by one **DBMS** 数据库管理系统 that all programs share. Each fact is stored once.

The relational model

- a **table** 表 of **records** 记录 (rows, also **tuples** 元组)
- and **fields** 字段 (columns, also **attributes** 属性)
- **primary key** 主键 – identifies a record uniquely
- **foreign key** 外键 – matches a primary key in another table



Referential integrity 参照完整性: no foreign key may point at a record that is not there.

The other kinds of key

composite key 复合键	a primary key made of two or more fields together, because no single field is unique
candidate key 候选键	any field, or set of fields, that *could* have been chosen as the primary key
secondary key 次键	a non-primary field kept **indexed** 索引, so searches and joins on it run fast

> **Indexing** 索引 costs space and slows writes, so you index the fields you actually search on – not every field.

Terms with an example, and referential integrity

`HOUSE(HouseID, Street, Bedrooms, OwnerID)`
`OWNER(OwnerID, Name, Phone)`

entity	a thing data is stored about: a house
field	one item of data about it: `Bedrooms`
record (tuple)	all the data about one house: one row
foreign key	`OwnerID` in HOUSE: the primary key of another table

"Explain referential integrity and how it applies." [2]

Every foreign key value must match an existing primary key: a house cannot have an `OwnerID` that is not in OWNER.

An owner with houses is deleted

Restrict: the delete is refused.
Cascade: the owner's houses are deleted too.
Either way, no house is left an orphan.

2 E-R diagrams

Sheet 8.1.1 – what you need to know

You must be able to:

- draw 1:1, 1:M and M:N relationships
- read the relationship off where the foreign key sits
- break M:N with a link table and a composite key

What the exam asks

"Complete the entity-relationship (E-R) diagram. ACTOR, FILM_FACT, FILM_ACTOR" **[2]**

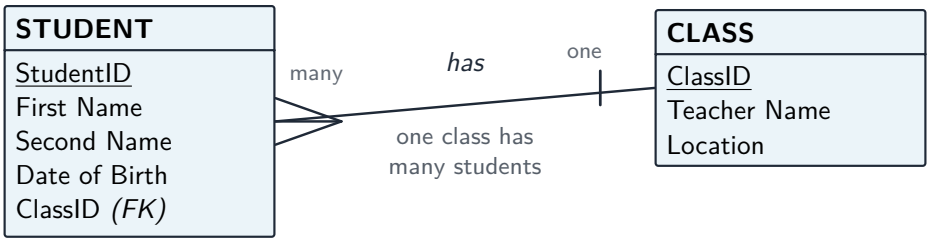
"Explain why the table FILM_ACTOR has a composite primary key." **[2]**

"Describe three benefits of this approach." **[3]**

Practise on exercise sheet 8.1.1, then try the real exam questions on past-paper sheet 8.1.1.

Entity-relationship diagrams

Cardinality 基数 at each end: **one-to-one** (1:1), **one-to-many** 一对多 (**1:M**) or **many-to-many** 多对多 (**M:N**).

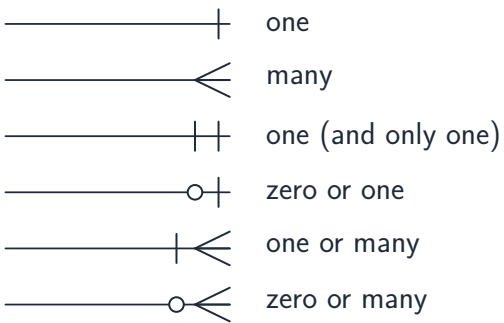


one class, many students

M:N needs a **link table** 连接表 holding two **foreign keys**.

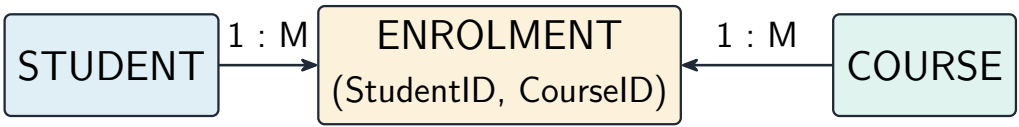
Crow's-foot notation

Read the symbol at each END of the line: it gives that side's **cardinality** 基数.



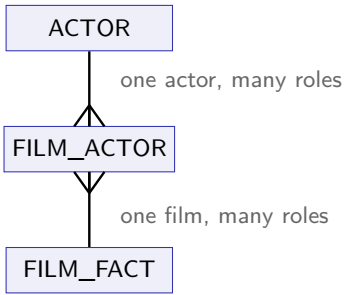
A link table breaks up M:N

A link table resolves a many-to-many relationship into two one-to-many relationships



a link table turns many-to-many into two one-to-many relationships

Reading an E-R diagram off the tables



- the table holding the foreign key is the "many" end
- a link table `FILM_ACTOR(ActorID, FilmID)` turns M:N into two 1:M

"Explain why FILM_ACTOR has a composite primary key." [2]

Neither ActorID nor FilmID is unique alone (an actor is in many films); the pair is unique. If an actor could appear twice, add the date to the key.

Fields `Club1`, `Club2`, `Club3` in one table: a repeating group. Use a link table instead.

3 Normalisation

Sheet 8.1.2 – what you need to know

You must be able to:

- put a table into 1NF, 2NF and 3NF
- give the highest normal form of a design, with a reason
- write a 3NF design with every key marked

What the exam asks

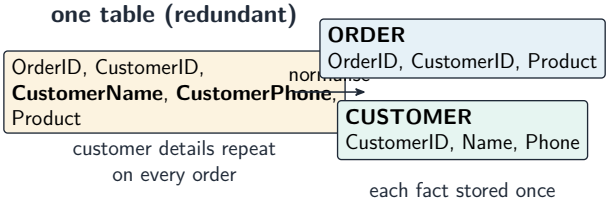
"Explain how to modify the table to put it into First Normal Form (1NF)." **[4]**

"Give the highest level of Normal Form the database is in and justify your choice." **[3]**

"Write a normalised database design. All tables must be in 3NF." **[4]**

Practise on exercise sheet 8.1.2, then try the real exam questions on past-paper sheet 8.1.2.

Normalisation



Normalisation 规范化 cuts redundancy through **normal forms** 范式:

- **1NF**: atomic fields, a key
- **2NF**: depend on the whole key
- **3NF**: no **transitive dependency** 传递依赖

Find the entities and attributes, pick a primary key for each, split repeating fields (1NF), split fields depending on part of the key (2NF), then on a non-key field (3NF).

1NF and 2NF, worked

UNF: `STUDENT(StudentID, Name, Subjects)`**, with "Maths, Physics" in one cell**

1NF: one value per cell, no repeating groups. Move subjects to their own table:

`STUDENT(StudentID, Name)`
`STUDENT_SUBJECT(StudentID, Subject)`

1NF: `ENTRY(StudentID, ExamID, Mark, ExamDate)`

`ExamDate` depends on `ExamID` only: a partial dependency. 2NF moves it out:

`ENTRY(StudentID, ExamID, Mark)`
`EXAM(ExamID, ExamDate)`

Write each table as `TABLE(Key, Field, …)` with the primary key underlined. Justify the highest NF by naming the dependency that breaks the next one.

Worked example: to third normal form

Not 3NF: `TeacherName` depends on `ClassID`, not on the key –

StudentID	Name	ClassID	TeacherName
S01	Li Wei	C1	Mr Chen
S02	Anna	C1	Mr Chen
S03	Ben	C2	Ms Roy

split off the transitive fact

StudentID	Name	ClassID
S01	Li Wei	C1
S02	Anna	C1
S03	Ben	C2

FK

ClassID	TeacherName
C1	Mr Chen
C2	Ms Roy

Now “Mr Chen” is stored once – renaming a teacher touches one row.

4 The DBMS

Sheet 8.2 – what you need to know

You must be able to:

- describe the DBMS tools: data dictionary, query processor, developer interface
- explain how a DBMS keeps data secure and consistent
- explain record locking when many users work at once

What the exam asks

"Describe two ways in which the DBMS can be used to ensure the security of the customer data." **[4]**

"Identify three other items stored in a data dictionary." **[3]**

"Explain how a DBMS supports data integrity." **[3]**

Practise on exercise sheet 8.2, then try the real exam questions on past-paper sheet 8.2.

The DBMS

A **DBMS** 数据库管理系统 manages the database centrally – **data management** 数据管理 and **data modelling** 数据建模 define the **logical schema** 逻辑模式, and it also handles **backup** 备份 and recovery, per-user security, **transactions** 事务 that wholly succeed or wholly fail, a **query processor** 查询处理器 to run queries, and a **developer interface** 开发者接口 of tools and APIs:

data dictionary 数据字典	**concurrent access** 并发访问	**transactions** 事务
views 视图	backup & security	**query processor** 查询处理器

What a DBMS provides, and the problems it removes

the tools	a data-dictionary editor, a query builder, a forms builder, a report generator, user management, and an SQL editor
data redundancy 数据冗余	file-based storage repeats the same fact in several files; a database stores it once
data inconsistency 数据不一致	one copy of the data is updated and another is not
data integrity 数据完整性	constraints and keys enforced by the DBMS itself, not by each program
data security 数据安全	one place to set who may read and who may write
data independence	programs are written against the logical view, so the storage can change without rewriting them

Redundancy causes inconsistency, and inconsistency destroys integrity. The three arguments are one chain, and the exam expects it stated as one.

Security, the data dictionary and integrity

"Describe two ways the DBMS can ensure the security of customer data." [4]

Access rights: each user can read or change only certain tables.
Passwords: only authorised users log in.
Views: users see only the fields they need.
Backups and encryption of the stored data.

Many users at once

Record locking: while one user updates a record, others cannot change it, so no update is lost.

"Identify three items stored in a data dictionary." [3]

Table and field names; data types and field sizes; primary and foreign keys; validation rules; relationships; indexes; access rights.

"Explain how a DBMS supports data integrity." [3]

Validation rules on input; referential integrity between tables; data stored once, so no inconsistent copies.

5 SQL: DDL and DML

Sheet 8.3 – what you need to know

You must be able to:

- write `CREATE TABLE` with primary, composite and foreign keys
- write `INSERT`, `UPDATE`, `DELETE` and `ALTER`
- total with `SUM … AS` and `GROUP BY`

What the exam asks

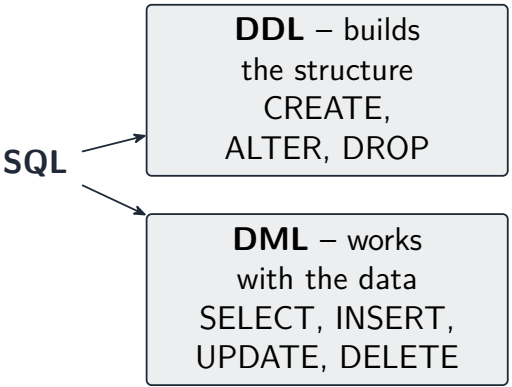
"Write an SQL script to output the customer ID, name and total cost of the orders not paid." **[4]**

"Write an SQL script to define the table REPAIR_PART. Include constraints…" **[5]**

"Write an SQL script to return the total amount of all the invoices sent in 2023 that have been paid." **[3]**

Practise on exercise sheet 8.3, then try the real exam questions on past-paper sheet 8.3.

SQL – DDL and DML



DDL

```
CREATE TABLE CUSTOMER
(CustomerID INTEGER PRIMARY KEY,
Name VARCHAR(50) NOT NULL);
```

- **DDL** 数据定义语言: structure
- **DML** 数据操纵语言: the data

SQL's two halves, and the exam's SQL rules

DDL 数据定义语言	Data Definition Language – creates or changes the structure: `CREATE TABLE`, `ALTER TABLE`, keys and constraints
DML 数据操纵语言	Data Manipulation Language – works on the data: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
the types	`INTEGER`, `REAL`, `VARCHAR(n)`, `CHAR(n)`, `DATE`, `TIME`, `BOOLEAN`, `DECIMAL(p,s)`
exact names	use the table and field names from the question, spelled as given
quoting	single quotes round strings (`'Smith'`); do not quote numbers
the operators	`LIKE 'A%'` matches anything starting with A (`%` any string, `_` one character); `IN (1,2,3)`; `BETWEEN 10 AND 20`

Combine conditions with `AND` / `OR` / `NOT`, and end every statement with a semicolon. Marks are lost on those details, not on the logic.

DDL: a table with a composite key and a foreign key

Define REPAIR_PART, with its constraints

```
CREATE TABLE REPAIR_PART(
  RepairID INTEGER NOT NULL,
  PartID VARCHAR(6) NOT NULL,
  Quantity INTEGER,
  PRIMARY KEY (RepairID, PartID),
  FOREIGN KEY (RepairID) REFERENCES REPAIR(RepairID),
  FOREIGN KEY (PartID) REFERENCES PART(PartID)
);
```

- a composite key is one `PRIMARY KEY` clause listing both fields
- each foreign key names the table and field it references
- `ALTER TABLE PART ADD Supplier VARCHAR(30);` adds a field later

DML: change the data, and total it

Add, change, remove

```
INSERT INTO PART (PartID, Name, Price)
VALUES ('P0042', 'Fan', 12.50);

UPDATE PART SET Price = 13.00
WHERE PartID = 'P0042';

DELETE FROM PART
WHERE PartID = 'P0042';
```

Total of the invoices paid in 2023

```
SELECT SUM(Amount) AS Total2023
FROM INVOICE
WHERE Paid = TRUE
AND InvoiceDate BETWEEN #01/01/2023#
  AND #31/12/2023#;
```

Each customer's unpaid total

```
SELECT CUSTOMER.CustomerID, Name,
SUM(Cost)
FROM CUSTOMER INNER JOIN ORDERS
ON CUSTOMER.CustomerID =
ORDERS.CustomerID
WHERE Paid = FALSE
GROUP BY CUSTOMER.CustomerID, Name;
```

6 SQL: joins and aggregates

Sheet 8.3.1 – what you need to know

You must be able to:

- join tables with `INNER JOIN … ON`
- count, total and average with `GROUP BY`
- find and correct errors in an SQL script

What the exam asks

"Complete the SQL script to return the number of devices for the staff member Ali Khan." **[4]**

"Identify the feature that describes the relationship between data and its structure." **[1]**

"There are four errors in the script. Identify and correct each error." **[4]**

Practise on exercise sheet 8.3.1, then try the real exam questions on past-paper sheet 8.3.1.

SQL – querying data

SELECT

```
SELECT Name, Phone
FROM CUSTOMER
WHERE City = 'London'
ORDER BY Name ASC;
```

A **join** 连接 combines tables on a foreign key; **aggregate functions** 聚合函数 (`COUNT`, `SUM`) pair with `GROUP BY`.

```
SELECT name, grade
FROM students
WHERE grade > 80;
```

→

result

name	grade
Ann	88
Cara	91

Always put a `WHERE` on `UPDATE`/`DELETE`, or it hits every row.

Joins and counts: clause order, and the missing ON

How many devices does Ali Khan have?

```
SELECT COUNT(DEVICE.DeviceID)
FROM STAFF INNER JOIN DEVICE
ON STAFF.StaffID = DEVICE.StaffID
WHERE STAFF.FirstName = 'Ali'
AND STAFF.LastName = 'Khan';
```

Clause order: `SELECT`, `FROM`, `JOIN … ON`, `WHERE`, `GROUP BY`, `ORDER BY`.

Forgetting the join condition

Without `ON`, every STAFF row pairs with every DEVICE row: a Cartesian product, and the count is far too big.

Average mark per class

```
SELECT ClassID, AVG(Mark)
FROM RESULT
GROUP BY ClassID;
```

Find the four errors

The script

```
SELECT SUM(DeviceID)
FROM STAFF, DEVICE
WHERE StaffID = StaffID
OR LastName = Khan;
```

Other traps: dates as `#DD/MM/YYYY#`; `CREATE DATABASE Name;` before any table; `ON DELETE CASCADE` on a foreign key removes the child rows with the parent.

error	correction
`SUM` adds the IDs up	`COUNT(DeviceID)`
`StaffID = StaffID`	`STAFF.StaffID = DEVICE.StaffID`
`OR`	`AND`: both must hold
`Khan` unquoted	`'Khan'`

7 Recap

Exam tips

- Define the terms exactly: entity, attribute, **primary key, foreign key**, and the relationship types (1:1, 1:many, many:many).
- Give a reason at each normal form: **1NF** (no repeating groups), **2NF** (no partial dependency), **3NF** (no non-key dependency).
- Explain what a **DBMS** provides (data independence, security, integrity, concurrent access).
- Distinguish **DDL** (define the structure) from **DML** (query and change the data).

Topic 8 – key takeaways

1 Why databases

remove redundancy & inconsistency

2 Keys

primary identifies; foreign links tables

3 Normalisation

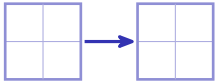
1NF $\rightarrow$ 2NF $\rightarrow$ 3NF; each fact once

4 SQL

DDL builds structure; DML queries data

Check yourself

1. What links one table to another?
2. How do you store a many-to-many relationship?
3. Which SQL keyword filters rows?
4. What does 3NF remove?



Answers **1.** a foreign key **2.** a link table **3.** `WHERE` **4.** transitive dependencies