A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

8.1 Database Concepts

Name: _____ Class: _____ Date: _____ **Total: 22 marks**

KEY WORDS relational database 关系数据库 • entity 实体 • primary key 主键 • foreign key 外键 • integrity 完整性

Objective

Build the skills to answer exam questions on **database concepts** — why relational databases beat flat files, the relational terms, keys, relationships and **normalisation** 规范化.

You must be able to:

- state the limitations of a **file-based** approach and how a **relational database** 关系数据库 fixes them
- use the terms **table**, **record**, **field**, **primary key** 主键 and **foreign key** 外键
- read an **entity-relationship (E-R) diagram** and explain whether tables are in **3NF**

1 Worked examples

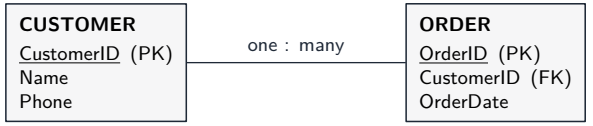
Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

■ File-based limits → relational database

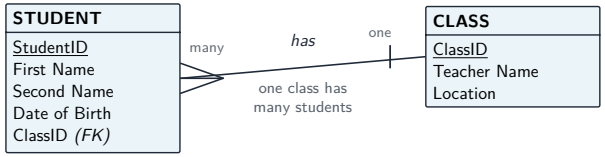
Storing data in separate **flat files** causes **data redundancy** (the same fact stored in several files), **data inconsistency** (copies updated separately disagree), and **data dependence** (programs tied to the file format). A relational database stores data once in linked **tables** managed by one DBMS, fixing all three.

■ Relational terms and keys

Term	Meaning
table (relation)	a grid for one entity, e.g. CUSTOMER
record (row)	one instance of the entity
field (column)	one piece of data about each record
primary key	field(s) that **uniquely identify** each record
foreign key	a field that refers to the primary key of another table



The **foreign key** CustomerID in ORDER refers to the **primary key** CustomerID in CUSTOMER, linking each order to its customer.



An entity-relationship diagram models the data

■ Normalisation (up to 3NF)

Normalisation removes redundancy. A table is in **Third Normal Form (3NF)** when every non-key field depends on **the key, the whole key, and nothing but the key** — i.e. atomic values (1NF), no partial dependency on part of a composite key (2NF), and no field depending on another non-key field (3NF). A **many-to-many** relationship is split using a **link (junction) table** holding two foreign keys.

■ Relationships and referential integrity

Two tables are linked by a **relationship** — **one-to-one**, **one-to-many** or **many-to-many**. The foreign key is placed on the **"many" side**. An **entity-relationship (E-R) diagram** shows these links.

Referential integrity 参照完整性 means every **foreign key** value must match an existing **primary key** in the other table (or be empty). It stops you recording, for example, an order for a customer who does not exist — the DBMS rejects a foreign key that has no matching record.

2 Practice

Now apply the methods above.

2.1 State **two** limitations of a file-based approach to storing data. [2]

2.2 Define **primary key** and **foreign key**. [2]

2.3 State what an **entity** is in a relational database. [1]

2.4 State what **First Normal Form (1NF)** requires. [1]

3 Exam-style questions

3.1 (a) State **two** problems caused by storing data in separate flat files. [2]

(b) State how a relational database removes one of them. [1]

3.2 Define each term: (a) record (b) field (c) primary key. [3]

3.3 Explain what it means for a table to be in **Third Normal Form (3NF)**. [3]

3.4 A school needs to record which students belong to which clubs; a student can join many clubs and a club has many students. Describe how to represent this **many-to-many** relationship using tables. [3]

3.5 (a) State what is meant by **referential integrity**. [2]

(b) A database has a `CUSTOMER` table and an `ORDER` table; one customer can place many orders. State the **relationship type** and which table holds the **foreign key**. [2]

Solutions

Each answer shows the steps, then the **result**.

2.1

- any two of: data redundancy (same data in several files); data inconsistency (copies disagree); data dependence (programs tied to the file format); weak security/sharing/querying

2.2

- primary key = field(s) that **uniquely identify** each record
- foreign key = a field that refers to the **primary key of another table**

2.3

- a **thing** the database stores data about (e.g. a customer, a book) — one table per entity

2.4

- all values are **atomic** (single values) — no repeating groups in a field

3.1

(a) any two: redundancy; inconsistency; data dependence; poor sharing/security

(b) e.g. each fact is stored **once** in a table, so there is no redundancy/inconsistency

3.2

- record = one **row** (one instance of the entity)
- field = one **column** (one piece of data)
- primary key = field(s) that **uniquely identify** a record

3.3

- every non-key field depends on **the whole primary key** and on **nothing but the key**
- values are atomic (1NF) and there is no partial dependency (2NF)
- no non-key field depends on another non-key field (no transitive dependency)

3.4

- create a **link (junction) table** (e.g. `STUDENT_CLUB`)
- it holds a foreign key to `STUDENT` and a foreign key to `CLUB`
- each row records one student-in-one-club, so many-to-many becomes two one-to-many relationships

3.5

(a) every foreign key value must match an existing primary key in the related table (or be null)

- so the database cannot hold a reference to a record that does not exist

(b) a **one-to-many** relationship

- the foreign key (`CustomerID`) is held in the `ORDER` table — the "many" side