

8.1.1 Normalising to 3NF, E-R diagrams and the full relational vocabulary

Name: _____ Class: _____ Date: _____ Total: 31 marks

KEY WORDS relational database 关系数据库 • entity 实体 • primary key 主键
• foreign key 外键 • field 字段

Objective

Build the skills to answer the **database design** questions that sheet 8.1 only defines. That sheet states the limitations of a file-based approach and defines a few terms; this one is about the two things the unit is really assessed on: **normalising** 规范化 a real set of fields through 1NF, 2NF and 3NF while **naming the dependency** that breaks each form, and documenting a design as an **entity-relationship** (E-R) diagram. It also covers the terms 8.1 leaves out – **candidate key** 候选键, **secondary key** 次要键, **tuple** 元组, **attribute** 属性 and **indexing** 索引.

You must be able to:

- use the full terminology of the relational model, including candidate key, secondary key, tuple, attribute, referential integrity and indexing
- use an **E-R diagram** to document a database design, and resolve a **many-to-many** relationship with a **link table** 链接表
- show understanding of the normalisation process: 1NF, 2NF and 3NF
- explain why a given set of tables is, or is not, in 3NF, naming the dependency
- produce a normalised design from a description or from a given table

1 Worked examples

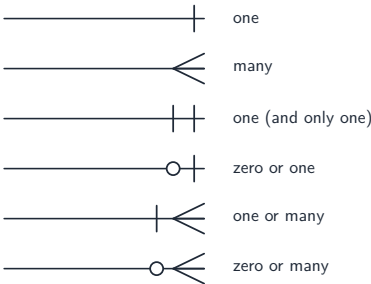
Study these first. Each one shows a **method** that a question later on this sheet needs.

- The vocabulary, in the exam’s own words

Term	What it is
entity	a thing the database stores data about; it becomes a table
table / relation	the set of records for one entity
record / tuple	one row: all the data about one instance of the entity
field / attribute	one column: one property of the entity
primary key	the attribute, or set of attributes, that uniquely identifies each record
candidate key	any attribute that <i>could</i> serve as the primary key; one candidate is chosen, the rest are alternates
secondary key	an attribute used to search or sort on, which need not be unique
foreign key	an attribute in one table that is the primary key of another, and is what links them
referential integrity	the rule that a foreign key must match an existing primary key, so no record can point at one that does not exist
indexing	building a lookup on a field so records can be found by it without reading every row: searches are faster, but the index takes storage and must be updated on every write

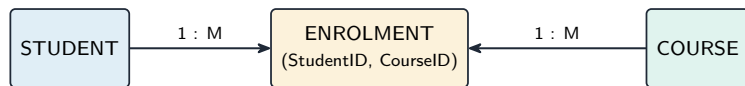
- **Candidate and primary are often confused.** In STUDENT(StudentID, Email, Name) both StudentID and Email are candidate keys, because either identifies a student; StudentID is chosen as the primary key, and Email stays a candidate.

■ E-R diagrams, and the many-to-many problem



An E-R diagram shows the entities as boxes and the relationships as lines, with the **cardinality** marked at each end: one-to-one, one-to-many or many-to-many.

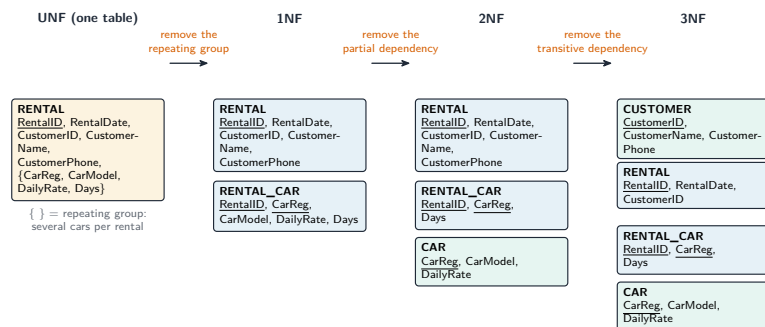
A relational database **cannot store a many-to-many relationship directly** – there is nowhere to put the foreign key, because either side would need many of them.



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

The fix is a **link table** (also called a junction or associative table): a third table holding the primary key of each side as a foreign key. One many-to-many becomes **two one-to-many** relationships. Its own primary key is usually the two foreign keys together, as a composite key.

■ Normalising: three questions, in order



1. **Is every cell a single value, with no repeating group?** If not, it is not in 1NF. Move the repeating group into its own table with a composite key.
 2. **If the key is composite, does every non-key field depend on the WHOLE key?** A field depending on only part of it is a **partial dependency** 部分依赖, and the table is not in 2NF.
 3. **Does every non-key field depend on the key alone?** A field depending on another non-key field is a **transitive dependency** 传递依赖, and the table is not in 3NF.
- An “explain why this is not in 3NF” answer must **name the dependency and the fields involved**. “It has repeated data” describes the symptom and earns nothing.

Worked example. *A theatre records each booking as*

```
BOOKING(BookingID, BookingDate, CustomerID, CustomerName,
        CustomerEmail, SeatNo, Price)
```

where one booking may cover several seats, and the price depends on which seat it is.

- **Not in 1NF:** SeatNo and Price are a **repeating group** – one booking has several seats. Move them out: BOOKING_SEAT(BookingID, SeatNo, Price), composite key (BookingID, SeatNo).
- **Not in 2NF:** in that new table, Price depends on SeatNo **alone**, which is only part of the composite key – a **partial dependency**. Move it: SEAT(SeatNo, Price), leaving BOOKING_SEAT(BookingID, SeatNo).
- **Not in 3NF:** in BOOKING, CustomerName and CustomerEmail depend on CustomerID, which is not the key – a **transitive dependency**. Move them: CUSTOMER(CustomerID, CustomerName, CustomerEmail).

The 3NF design is four tables, and every primary key is underlined:

```
CUSTOMER(CustomerID, CustomerName, CustomerEmail)
BOOKING(BookingID, BookingDate, CustomerID)
BOOKING_SEAT(BookingID, SeatNo)
SEAT(SeatNo, Price)
```

- Take the forms **in order**. Testing for 2NF before the repeating group is gone gives the wrong answer, because the composite key does not exist yet.

2 Practice

Now use the methods above.

2.1 Define candidate key and secondary key. [2]

2.2 State what is meant by referential integrity. [2]

2.3 State one benefit and one drawback of indexing a field. [2]

2.4 Give the database term for a **row** of a table and for a **column** of a table. [2]

2.5 State the **two** conditions a table must meet to be in **2NF**. [2]

3 Exam-style questions

3.1 A club records each event as

```
EVENT(EventID, EventName, EventDate, MemberID1, MemberID2, MemberID3)
```

where the three member fields hold the members who attended.

(a) State why this table is not in 1NF. [2]

(b) Give a 1NF design for this data. [2]

3.2 A table is defined as

```
ORDER_LINE(OrderID, ProductID, Quantity, ProductName, UnitPrice)
```

with the composite primary key (OrderID, ProductID).

(a) Name the dependency that stops this table being in 2NF, and state which fields are involved. [2]

(b) Give a 2NF design. [2]

3.3 A table is defined as

```
EMPLOYEE(EmployeeID, Name, DepartmentID, DepartmentName)
```

with EmployeeID as the primary key.

Explain why this table is not in 3NF, and give a 3NF design. [4]

3.4 In a school, a teacher may teach many courses and a course may be taught by many teachers.

(a) State why this relationship cannot be stored directly in two tables. [2]

(b) Describe the design that solves it, naming the tables and their keys. [2]

3.5 A vet's surgery records each visit: the date, the pet's name and species, the owner's name and telephone number, and the treatments given during that visit. One visit may involve several treatments, and each treatment has a fixed cost.

Produce a **3NF** design for this data. Underline each primary key and state each foreign key. [5]

Solutions

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

2.1

- a **candidate key** is any attribute, or set of attributes, that **could** be used as the primary key because it uniquely identifies each record
- a **secondary key** is an attribute used to **search or sort** the table on, and it need not be unique

2.2

- every **foreign key** value must match an existing **primary key** value in the table it refers to
- so a record cannot refer to one that does not exist, and a record still referred to cannot simply be deleted

2.3

- benefit: records can be found by that field without reading every row, so searching and sorting on it are much faster
- drawback: the index takes extra storage, and it must be updated every time a record is inserted, changed or deleted, which slows those operations

2.4

- a row is a **record**, also called a **tuple**
- a column is a **field**, also called an **attribute**

2.5

- it must already be in **1NF**
- and every non-key field must depend on the **whole** primary key, not on part of it – there must be no partial dependency

3.1

(a) **MemberID1**, **MemberID2** and **MemberID3** are a **repeating group**: the same piece of information is held in three numbered fields

- 1NF requires every field to hold a single atomic value and forbids repeating groups; this design also limits an event to three members and wastes space when fewer attend

(b) **EVENT(EventID, EventName, EventDate)**

- **ATTENDANCE(EventID, MemberID)** with the composite key (**EventID, MemberID**), where **EventID** is a foreign key

3.2

(a) a **partial dependency**

- **ProductName** and **UnitPrice** depend on **ProductID alone**, which is only part of the composite key (**OrderID, ProductID**)

(b) **ORDER_LINE(OrderID, ProductID, Quantity)**

- **PRODUCT(ProductID, ProductName, UnitPrice)**, with **ProductID** a foreign key in **ORDER_LINE**

3.3

- the table is not in 3NF because of a **transitive dependency**
- **DepartmentName** depends on **DepartmentID**, which is a **non-key** field, rather than on the primary key **EmployeeID**
- **EMPLOYEE(EmployeeID, Name, DepartmentID)**
- **DEPARTMENT(DepartmentID, DepartmentName)**, with **DepartmentID** a foreign key in **EMPLOYEE**
- the consequence is worth knowing: a department's name is stored once, so it cannot be spelt two ways, and renaming it is one change

3.4

(a) it is a **many-to-many** relationship

- there is nowhere to put the foreign key: a teacher would need many course keys and a course would need many teacher keys, and a field holds one value

(b) add a **link table**, for example **TEACHES(TeacherID, CourseID)**, whose primary key is the two together

- **TeacherID** and **CourseID** are both foreign keys, so the one many-to-many becomes **two one-to-many** relationships: **TEACHER** to **TEACHES** and **COURSE** to **TEACHES**

3.5

- **OWNER** separated out, because the owner's details depend on the owner and not on the visit – a transitive dependency
- **PET** separated out, with **OwnerID** as a foreign key
- **VISIT** holding the date and the pet, with **PetID** as a foreign key
- the treatments are a **repeating group**, so they need a link table **VISIT_TREATMENT** with the composite key
- **TREATMENT** holding the fixed cost, because the cost depends on the treatment alone – a partial dependency in the link table

```
OWNER(OwnerID, OwnerName, Telephone)
PET(PetID, PetName, Species, OwnerID)
VISIT(VisitID, VisitDate, PetID)
VISIT_TREATMENT(VisitID, TreatmentID)
TREATMENT(TreatmentID, TreatmentName, Cost)
```

- foreign keys: **OwnerID** in **PET**; **PetID** in **VISIT**; **VisitID** and **TreatmentID** in **VISIT_TREATMENT**
- names may differ; the marks are for the five tables, their keys and the relationships between them