Exercise walk-through

8.1.1

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

A-Level Computer Science

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 The method

Method — 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 *could* 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

Method — The vocabulary, in the exam's own words

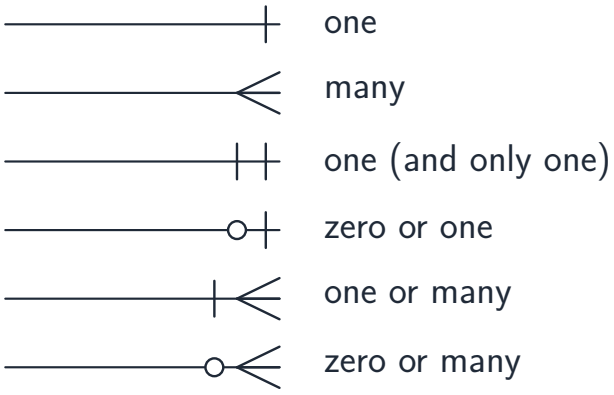
- **Candidate and primary are often confused.** In `STUDENT(StudentID, Email, Name)` both `StudentID` and `Email` are candidate keys, because either identifies a student; `StudentID` is chosen as the primary key, and `Email` stays a candidate.

Method — 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.
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.

Method — 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...



Method — 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...

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

Method — 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.

Method — Normalising: three questions, in order

Worked example. *A theatre records each booking as*

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

- **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)`.

Method — Normalising: three questions, in order

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

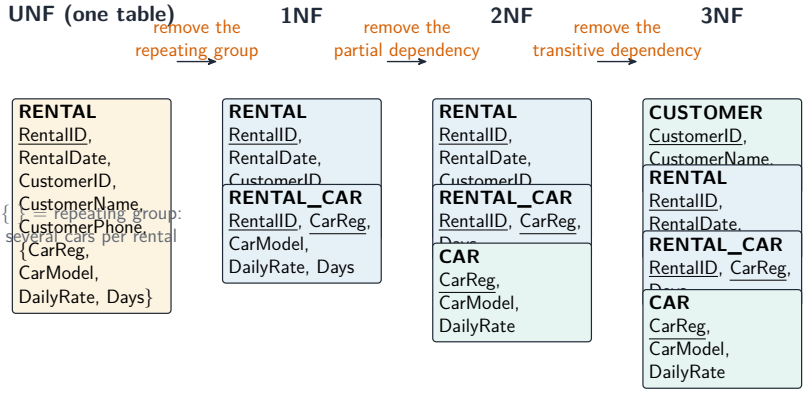
```
CUSTOMER(CustomerID, CustomerName, CustomerEmail)
BOOKING(BookingID, BookingDate, CustomerID)
BOOKING_SEAT(BookingID, SeatNo)
SEAT(SeatNo, Price)
```

Method — Normalising: three questions, in order

- 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.

Method — Normalising: three questions, in order

Worked example.



2 Practice

Practice 2.1 ▪ 2 marks

Define **candidate key** and **secondary key**.

Solution 2.1

Method: The vocabulary, in the exam's own words — p.4

Worked steps

- a **candidate key** is any attribute, or set of attributes, that **could** be used as the primary key because it uniquely identifies each record B1
- a **secondary key** is an attribute used to **search or sort** the table on, and it need not be unique B1

Practice 2.2 ▪ 2 marks

State what is meant by **referential integrity**.

Solution 2.2

Method: The vocabulary, in the exam's own words — p.4

Worked steps

- every **foreign key** value must match an existing **primary key** value in the table it refers to B1
- so a record cannot refer to one that does not exist, and a record still referred to cannot simply be deleted B1

Practice 2.3 ▪ 2 marks

State **one** benefit and **one** drawback of **indexing** a field.

Solution 2.3

Method: The vocabulary, in the exam's own words — p.4

Worked steps

- benefit: records can be found by that field without reading every row, so searching and sorting on it are much faster B1
- drawback: the index takes extra storage, and it must be updated every time a record is inserted, changed or deleted, which slows those operations B1

Practice 2.4 ▪ 2 marks

Give the database term for a **row** of a table and for a **column** of a table.

Solution 2.4

Method: The vocabulary, in the exam's own words — p.4

Worked steps

- a row is a **record**, also called a **tuple** B1
- a column is a **field**, also called an **attribute** B1

Practice 2.5 ▪ 2 marks

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

Solution 2.5

Worked steps

- it must already be in **1NF** B1
- and every non-key field must depend on the **whole** primary key, not on part of it – there must be no partial dependency B1

3 Exam-style

Exam-style 3.1 ▪ 4 marks

A club records each event as

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

where the three member fields hold the members who attended.

Exam-style 3.1 (a) ▪ 2 marks

State why this table is not in 1NF.

Solution 3.1 (a)

Method: Normalising: three questions, in order — p.9

Worked steps

`MemberID1`, `MemberID2` and `MemberID3` are a **repeating group**: the same piece of information is held in three numbered fields B1

- 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 B1

Exam-style 3.1 (b) ▪ 2 marks

Give a 1NF design for this data.

Solution 3.1 (b)

Worked steps

`EVENT(EventID, EventName, EventDate)` B1

- `ATTENDANCE(EventID, MemberID)` with the composite key `(EventID, MemberID)`, where `EventID` is a foreign key B1

Exam-style 3.2 ▪ 4 marks

A table is defined as

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

with the composite primary key (`OrderID, ProductID`).

Exam-style 3.2 (a) ▪ 2 marks

Name the dependency that stops this table being in 2NF, and state which fields are involved.

Solution 3.2 (a)

Method: Normalising: three questions, in order — p.9

Worked steps

a **partial dependency** B1

- `ProductName` and `UnitPrice` depend on `ProductID` **alone**, which is only part of the composite key `(OrderID, ProductID)` B1

Exam-style 3.2 (b) ▪ 2 marks

Give a 2NF design.

Solution 3.2 (b)

Worked steps

`ORDER_LINE(OrderID, ProductID, Quantity)` B1

- `PRODUCT(ProductID, ProductName, UnitPrice)`, with `ProductID` a foreign key in `ORDER_LINE` B1

Exam-style 3.3 ▪ 4 marks

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.

Solution 3.3

Method: Normalising: three questions, in order — p.9

Worked steps

- the table is not in 3NF because of a **transitive dependency** B1
- `DepartmentName` depends on `DepartmentID`, which is a **non-key** field, rather than on the primary key `EmployeeID` B1
- `EMPLOYEE(EmployeeID, Name, DepartmentID)` B1
- `DEPARTMENT(DepartmentID, DepartmentName)`, with `DepartmentID` a foreign key in `EMPLOYEE` B1
- 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

Exam-style 3.4 ▪ 4 marks

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

Exam-style 3.4 (a) ▪ 2 marks

State why this relationship cannot be stored directly in two tables.

Solution 3.4 (a)

Method: E-R diagrams, and the many-to-many problem — p.6

Worked steps

it is a **many-to-many** relationship B1

- 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 B1

Exam-style 3.4 (b) ▪ 2 marks

Describe the design that solves it, naming the tables and their keys.

Solution 3.4 (b)

Worked steps

add a **link table**, for example `TEACHES(TeacherID, CourseID)`, whose primary key is the two together B1

- `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` B1

Exam-style 3.5 ▪ 5 marks

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.

Solution 3.5

Method: Normalising: three questions, in order — p.9

Worked steps

- `OWNER` separated out, because the owner's details depend on the owner and not on the visit – a transitive dependency **B1**
- `PET` separated out, with `OwnerID` as a foreign key **B1**
- `VISIT` holding the date and the pet, with `PetID` as a foreign key **B1**
- the treatments are a **repeating group**, so they need a link table `VISIT_TREATMENT` with the composite key **B1**
- `TREATMENT` holding the fixed cost, because the cost depends on the treatment alone – a partial dependency in the link table **B1**

Solution 3.5

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

Solution 3.5

- 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