8.1.1 Relationships and E-R diagrams

Name: _____ Class: _____ Date: _____ **Total: 30 marks**

KEY WORDS relationship • cardinality • link table • composite key

Objective

Translate rules about a school into relationships, place foreign keys, and draw an entity-relationship diagram. Use the tables and keys from sheet 8.1.

1 Key knowledge and worked examples

■ One relationship, read in both directions

A **relationship** connects entities. **Cardinality** describes how many instances can be related. Ask both questions: for ONE item on this side, how many items can be on the other side?

School rule	Relationship
Each student has one student card; each card belongs to one student	One-to-one
A tutor group has many students; each student belongs to one group	One-to-many
A student can join many clubs; each club can have many students	Many-to-many

The rules of the situation determine cardinality. A sample containing only one student in a group does not prove the relationship is one-to-one.

■ Worked example: from sentences to a diagram

Each student belongs to exactly one tutor group. A group may have no students yet or may have many students.

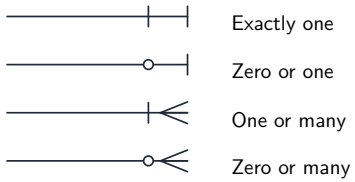
1. Start with one GROUP. It can have many STUDENT records, so the many end belongs next to STUDENT.
2. Start with one STUDENT. It belongs to one GROUP, so the one end belongs next to GROUP.
3. Put GroupID as a foreign key in STUDENT: one value records that student's group.

GROUP	
GroupID	GroupName
G1	Oak

STUDENT		
StudentID	Name	GroupID
S1	Mei	G1
S2	Lin	G1

An **entity-relationship (E-R) diagram** documents the entities and their relationships. Boxes represent entities. The line ends show cardinality. The diagram above shows maximum cardinality using one bar for one and a crow's foot for many. These are not arrowheads.

■ Add optional participation only when required



A circle means **zero allowed**; a bar means **one**; a crow's foot means **many**. Read the symbols beside the entity being counted. For the GROUP-STUDENT rules, one GROUP has zero or many STUDENT records; one STUDENT has exactly one GROUP. When only one/many is asked, mark those clearly at both ends. Add minimum participation consistently if you show it.

■ Worked example: why clubs need a link table

Putting a list such as `C1, C2` in STUDENT.ClubID would store several values in one cell. Adding ClubID1, ClubID2 and ClubID3 would limit the number of clubs. Instead, record each membership as one row in a **link table** .

Underlined fields in the link table form ONE composite primary key.

STUDENT	
StudentID	Name
S1	Mei
S2	Lin

MEMBERSHIP	
StudentID	ClubID
S1	C1
S1	C2
S2	C1

CLUB	
ClubID	Name
C1	Chess
C2	Music

Read one membership row `(S1, C2)` as: student S1 belongs to club C2. StudentID and ClubID are both foreign keys. Together they form the composite primary key because

each student may join each club only once.

The stored design has two one-to-many relationships: STUDENT to MEMBERSHIP and CLUB to MEMBERSHIP. Each MEMBERSHIP record refers to exactly one student and exactly one club. Repeated foreign-key values are normal; repeated copies of names and emails are unnecessary.

■ Worked example: when the pair is not enough

A student may attend the same club on different dates. `(StudentID, ClubID)` cannot identify every attendance: the pair repeats over time. If attendance is limited to once per day, use `(StudentID, ClubID, AttendDate)` or a separate AttendanceID. State the rule that makes your chosen key unique.

■ A method for independent design

Identify the entities, write each relationship in both directions, draw its line ends, then add keys. Place the foreign key on the many side of a one-to-many relationship. For one-to-one, a foreign key on either chosen side must also be unique; optional participation helps decide which side is suitable. Resolve many-to-many with a link table and check what one link-table record means.

2 Practice

2.1 The diagram shows GROUP and STUDENT. State the relationship type and write one sentence reading it in each direction. [3]

2.2 Each student belongs to one group. GROUP has primary key GroupID and STUDENT has primary key StudentID. State which table needs the foreign key and name it. Explain your choice. [3]

2.3 The MEMBERSHIP table below allows each student to join each club once. Name both foreign keys and the composite primary key. Explain what the first row records. [4]

MEMBERSHIP	
StudentID	ClubID
S1	C1
S1	C2
S2	C1

2.4 Each student has exactly one card, and each card belongs to exactly one student. Draw the E-R relationship between STUDENT and CARD. Include cardinality at both ends. [2]

2.5 ATTENDANCE has StudentID, ClubID and AttendDate. A student can attend the same club on different dates, at most once per day. Explain why `(StudentID, ClubID)` is insufficient and state a suitable composite key. [2]

ATTENDANCE		
StudentID	ClubID	AttendDate
S1	C1	09/10
S1	C1	16/10

3 Exam-style questions

3.1 A teacher may teach several courses and a course may have several teachers. Design tables for TEACHER, COURSE and their relationship. Give every primary and foreign key. Draw the E-R diagram for the stored design. Assume a teacher-course pair is recorded once. [6]

3.2 A library stores MEMBER, BOOK_COPY and LOAN. One member can have many loans over time; one physical copy can appear in many loans over time. Each loan belongs to one member and one copy. A member or copy can exist without a loan. Draw an E-R diagram, label cardinality at both ends of each line, and name the primary and foreign keys. A member may borrow the same copy again later. [6]

3.3 A school stores STUDENT and CLUB. Both contain a unique ID and a name. Students can join several clubs, and clubs can have several students. A designer adds three fields, ClubID1, ClubID2 and ClubID3, to STUDENT. Explain two problems with this design, then give a suitable replacement table and its keys. [4]

STUDENT
StudentID, Name
ClubID1, ClubID2, ClubID3

Solutions

2.1 One-to-many. One GROUP can have many STUDENT records. Each STUDENT belongs to one GROUP.

2.2 STUDENT holds the foreign key, named GroupID and referring to GROUP.GroupID. Each student needs one group reference; storing student IDs in GROUP would require several values or repeated fields.

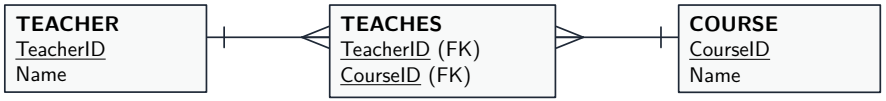
2.3 StudentID refers to STUDENT.StudentID and ClubID refers to CLUB.ClubID. The primary key is `(StudentID, ClubID)`. The first row records student S1's membership of club C1.

2.4

One at the STUDENT end and one at the CARD end. Exactly-one participation is shown.

2.5 The same student-club pair occurs on different dates. Use `(StudentID, ClubID, AttendDate)`.

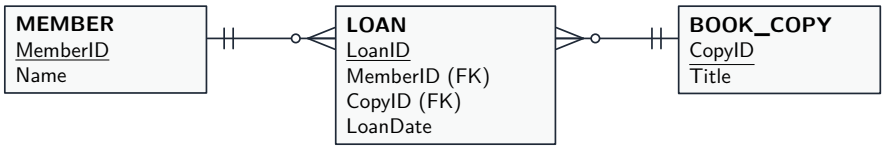
3.1



Underlined fields in the link table form ONE composite primary key.

Two entity primary keys; composite link-table key; both foreign keys; two correct one-to-many relationships.

3.2



FK = foreign key. Each loan has one member and one physical copy.

Three entities; two correct one-to-many relationships; entity primary keys; LoanID primary key; both foreign keys. LoanID allows the same member-copy pair to recur.

3.3 The numbered club fields form a repeating group and limit a student to three clubs. Use MEMBERSHIP with composite primary key `(StudentID, ClubID)`; both fields are foreign keys to their corresponding entity tables. Each row records one membership.