8.1 Tables, keys and database concepts

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

KEY WORDS entity • candidate key • primary key • foreign key • index

Objective

Read a database table, choose keys, and explain how linked tables keep school-club data consistent. Then use the same ideas in a new situation.

1 Key knowledge and worked examples

■ Start with the data: who has joined a club?

A school needs students' contact details and club memberships. If every club keeps its own file, one student's email may be copied into several files. Changing only one copy leaves different answers to the same question.

CHESS file

StudentID	Email
S1	mei.new@school.cn

MUSIC file

StudentID	Email
S1	mei.old@school.cn

Same student, two copies of one contact detail.

Data redundancy is unnecessary duplication of data. **Data inconsistency** means copies of the same fact disagree. **Data dependence** means programs depend on a file's structure; changing that structure may require changing the programs. A DBMS separates the stored structure from the programs using it. Linked tables reduce unnecessary duplication; they do not automatically make every fact correct.

■ Read one table before learning its names

STUDENT

StudentID	Name	Email
S1	Mei	mei@school.cn
S2	Lin	lin@school.cn
S3	Lin	lin2@school.cn

Column: Email = one named property.
Row S1 = one student record.
Cell: mei@school.cn = one stored value.

- An **entity** is a type of thing we store data about: here, a student. The **STUDENT table** stores its instances.
- A **record**, or **tuple**, is one row: all the stored details of one student.
- A **field**, or **attribute**, is one named property, such as Email. `mei@school.cn` is a field value, not the field's name.
- A **relation** is a table in the relational model. A database can contain several related tables.

■ Worked example: choose a key, rather than guessing

The school guarantees that StudentID and Email are each unique and present for every student. Names may repeat.

Choice	Can identify every student?	Reason
Name	No	Two students can share a name
StudentID	Yes	Unique, present and stable
Email	Yes	Guaranteed unique here

A **candidate key** is a minimal field or set of fields that uniquely identifies a record. Minimal means no field can be removed without losing that guarantee. Choose one candidate as the **primary key**: use StudentID here. Email remains an alternative candidate key. Uniqueness must be a rule for all valid data, not a coincidence in three sample rows.

A **composite key** contains more than one field. For memberships, `(StudentID, ClubID)` identifies one student's membership of one club. Either ID alone may repeat; the pair must not repeat. A **secondary key** is used to search or sort records, for example Name, and need not be unique.

■ Worked example: follow a foreign key

STUDENT

StudentID	Name
S1	Mei
S2	Lin

S1 matches S1

MEMBERSHIP

StudentID (FK)	ClubID
S1	C1
S1	C2
S2	C1

Underlined: primary key. FK: foreign key. S1 can occur in several memberships.

StudentID is a primary key in STUDENT. In MEMBERSHIP it is a **foreign key** referring to STUDENT.StudentID. Follow `S1` to find Mei's details. Her name and email belong in STUDENT, so they need not be copied into every membership row. Foreign keys can repeat.

Referential integrity requires each non-null foreign key to match an existing referenced key. A new membership for `S9` is rejected if S9 is missing from STUDENT. A null foreign key is allowed only when that field permits nulls. Deleting a referenced student must obey the declared rule: reject the deletion or handle dependent records consistently, for example by cascading deletion.

■ **Worked example: a small index**

Name INDEX	
Name	Record
Lin	S2
Lin	S3
Mei	S1

record reference →

STUDENT	
StudentID	Name
S1	Mei
S2	Lin
S3	Lin

An **index** is an extra lookup structure. Searching Name can use the index to locate matching records without scanning every table row. It costs storage. Inserts, deletions and changes to indexed values also require index maintenance. An index and a secondary key are related ideas, but an index is the lookup structure, not the field itself.

2 Practice

2.1 Use the STUDENT data below. Give the entity, one field name, and the number of records. [3]

STUDENT		
StudentID	Name	Year
S1	Lin	12
S2	Mei	12
S3	Lin	13

2.2 A school guarantees unique StudentID and unique Email. Names may repeat. Identify both candidate keys and choose a suitable primary key. Explain why Name is unsuitable. [4]

STUDENT		
StudentID	Name	Email
S1	Lin	lin1@school.cn
S2	Lin	lin2@school.cn

2.3 MEMBERSHIP allows each student to join each club once. Explain why neither StudentID nor ClubID alone is a suitable primary key, then state a suitable key. [3]

MEMBERSHIP	
StudentID	ClubID
S1	C1
S1	C2
S2	C1

2.4 STUDENT contains S1 and S2. StudentID in MEMBERSHIP is a foreign key and cannot be null. Which proposed row is rejected: `(S2, C3)` or `(S8, C3)`? Explain why. Assume both club references are valid. [2]

STUDENT
StudentID
S1
S2

Proposed MEMBERSHIP	
StudentID	ClubID
S2	C3
S8	C3

2.5 Name is indexed so office staff can find students by name. State one benefit and one cost of this index. [2]

STUDENT
StudentID
Name

INDEX
Name

3 Exam-style questions

3.1 Two club files store different email addresses for the same student. Define data redundancy and data inconsistency using this example. Explain how linked tables reduce this problem. [4]

CHESS	
ID	Email
S4	a@school.cn

MUSIC	
ID	Email
S4	b@school.cn

3.2 A library has MEMBER(MemberID, Name) and LOAN(LoanID, MemberID, LoanDate). MemberID uniquely identifies a member and LoanID uniquely identifies a loan. Define **field**, **entity** and **foreign key**, giving an example of each from this database. [6]

MEMBER
MemberID
Name

LOAN
LoanID
MemberID
LoanDate

3.3 The library has a Name index on MEMBER. Explain why Name can be a secondary key but cannot be guaranteed to be a candidate key. State one effect of changing an indexed name. [3]

MEMBER	
MemberID	Name
M1	Chen
M2	Chen

3.4 Member M7 has an existing loan. A librarian tries to delete M7 from MEMBER while leaving the loan unchanged. Explain the integrity problem and describe two possible DBMS rules that would preserve referential integrity. [4]

MEMBER	
MemberID	Name
M7	Mei

LOAN	
LoanID	MemberID
L3	M7

Solutions

2.1 Student is the entity. StudentID, Name or Year is a field. There are three records.

2.2 StudentID and Email are candidate keys. Choose StudentID as primary key. Name can repeat, so it cannot uniquely identify each student.

2.3 StudentID repeats for different clubs. ClubID repeats for different students. Use the composite primary key `(StudentID, ClubID)`.

2.4 `(S8, C3)` is rejected: no STUDENT record has StudentID S8, so the foreign key has no matching referenced key.

2.5 The index locates matching records without scanning every row. It uses extra storage or must be maintained when indexed data changes.

3.1 Redundancy: the same student's email is stored in more than one file. Inconsistency: those copies disagree. Store the student's email once in STUDENT; club memberships refer to that record by StudentID, so an email update is made in one place.

3.2 Field: a named property/column, for example LoanDate in LOAN. Entity: a type of thing about which data is stored, for example a member. Foreign key: field(s) referring to a key in another table, for example LOAN.MemberID referring to MEMBER.MemberID.

3.3 Name can be used to search or sort records. Different members can share a name, so uniqueness is not guaranteed. Changing an indexed name requires updating the index, adding maintenance work.

3.4 The loan would refer to a member record that no longer exists, breaking referential integrity. RESTRICT/NO ACTION can reject the deletion while loans refer to M7. ON DELETE CASCADE can delete the referencing loans as part of the deletion. Cascading may be unsuitable when loan history must be retained; the rule is a design choice. Setting the reference to null is another valid choice only if nulls are permitted.