THE UNIVERSITY OF MELBOURNE

INFO20003 Database Systems

Xiuge Chen

Tutorial 9

2021.05.03

Agenda

1. **Review of normalization concepts - 15min**
2. **Normalisation exercises - 35min**

Review of normalization concepts

1. Anomalies

What are them?

Something wrong with the existing database
Like redundancy, error occurrence when manipulating data

Why we need to identify and fix them?

Make database more efficient and less error-prone
How: **Normalization**!

Types of Anomalies:

1. **update** anomaly
2. **deletion** anomaly
3. **insertion** anomaly

Review of normalization concepts

1. Anomalies Types:

Update anomaly: data inconsistency that results from data redundancy and partial update when one or more instances of duplicated data are updated but not all.

Deletion anomaly: unintentional loss of certain attribute values due to the deletion of other data for other attributes.

Insertion anomaly: the inability to add certain attributes to a database due to absence of other attributes.

Review of normalization concepts

1. Anomalies Example:

CourseNumber	Tutor	Room	Seats
INFO20003	Farah	Alice Hoy 109	30
COMP10001	Farah	EDS 6	25
INFO30005	Patrick	Sidney Myer G09	20
COMP20005	Alan	Sidney Myer G09	20

Update anomaly: suppose the room Sidney Myer G09 has been improved, and there are now 30 seats. In this single entity, we will have to update all other rows where room = Sidney Myer G09.
Deletion anomaly: If we remove COMP10001 from the above table, the details of room EDS 6 are also deleted.
Insertion anomaly: a new room “NewAlice109” has been built but has not yet been timetabled for any course or members of staff.

Review of normalization concepts

2. Functional dependency:

- Occurs when a subset of R’s attributes {A1, A2, ..., An} **determine** attributes {B1, B2, ..., Bn}
- If two records have the same A1, A2, ..., An then they have the same B1, B2, ..., Bn.
- A relation R satisfies a functional dependency (FD) if and only if the FD is true for every instance of R.
- Written as:

$$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n$$

Review of normalization concepts

3. Determinants:

- Attributes that **determine** the value of other attributes are called **determinants**
- Example:

Person (<u>ssn</u>, name, birthdate, address, age)

birthdate $\rightarrow$ age
ssn $\rightarrow$ name, birthdate, age, address

birthdate and ***ssn*** are determinants, as *birthdate* determines *age* and *ssn* determines the rest of the attributes.

Review of normalization concepts

4. Key and non-key attributes:

- A *key* is a set of attributes {A1, A2, ..., A*n*} for a relation R
- such that {A1, A2, ..., A*n*} **functionally determines** all other attributes of R and no subset of {A1, A2, ..., A*n*} functionally determines all other attributes of R. The key must be minimal.

Example:

Person (ssn, name, birthdate, address, age)

ssn is the **minimal key** of the Person relation
but {***ssn***, ***name***} is not (it is a “super key”).

Review of normalization concepts

5. Partial functional dependency

- Arises when one or more non-key attributes are functionally determined by a *subset* of the primary key.

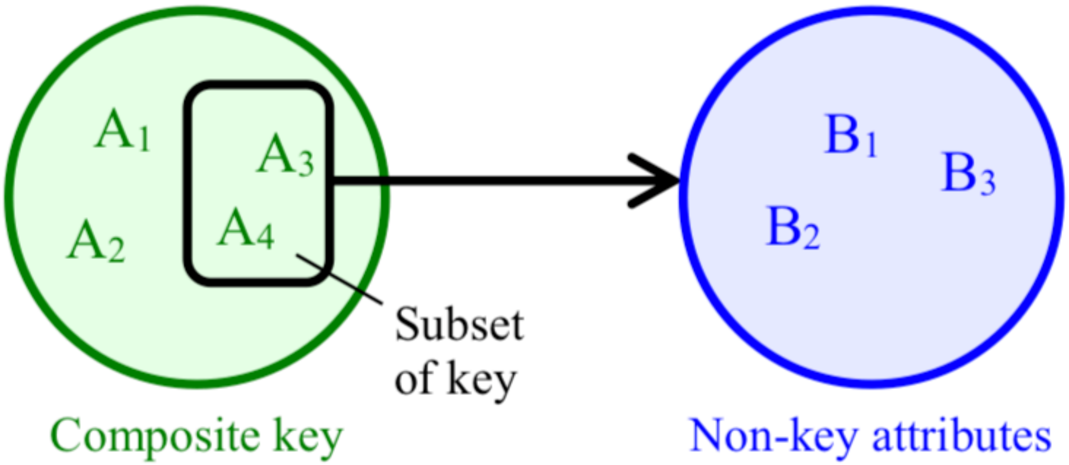

Figure 1: A partial functional dependency (subset of composite key determining some non-key attributes)

Review of normalization concepts

5. Partial functional dependency Examples:

- R (A, B, C, D)
- composite primary key: (A, D)
- functional dependencies: A → B, D → C.
- AD determines BC (AD → BC: AD can uniquely identify BC).
- Functional dependencies like A → B and D → C are called ***partial functional dependencies*.**
- Order (Order#, Item#, Desc, Qty)
- Order# and Item# are the keys.
- item description, Desc, can be determined by Item# alone (***partial functional dependencies***)

Review of normalization concepts

6. Transitive functional dependency

- A non-key attribute is determined by **another non-key attribute** (or by a subset of PK and non-key attributes), such a dependency is called a ***transitive functional dependency.***

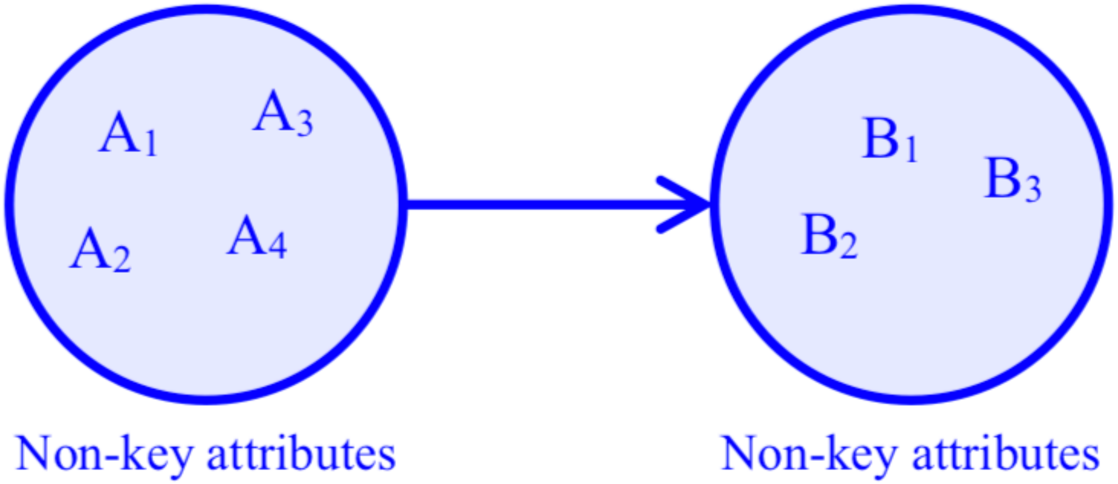

Figure 2: A transitive functional dependency (non-key attributes determining other non-key attributes)

Review of normalization concepts

7. Armstrong’s Axioms

- **What is it used for?**
  - Given a relation and a set of **functional dependencies** (FDs), we can discover **new functional dependencies** using some rules generally known as **Armstrong’s Axioms.**

- **Types**:
  - Reflexivity (also known as “trivial FDs”)
  - Augmentation
  - Transitivity

Review of normalization concepts

7. Armstrong's Axioms

- **Reflexivity**
  - R ({A_1, A_2, ..., A_n} and {B_1, B_2, ..., B_n}), B is a subset of A

$$B \subseteq A \Rightarrow A \rightarrow B$$

  - Example: Person (ssn, name, birthdate, address, age)

ssn, name → name

- **Augmentation**
  - Consider another subset of attributes {C_1, C_2, ..., C_n}

$$A \rightarrow B \Rightarrow AC \rightarrow BC$$

  - Example: Person (ssn, name, birthdate, address, age)

ssn, name, age → name, age

Review of normalization concepts

7. Armstrong's Axioms

- **Transitivity**

$$A \rightarrow B \text{ and } B \rightarrow C \Longrightarrow A \rightarrow C$$

- Example: Person (ssn, name, birthdate, address, age)

$$\textbf{ssn} \rightarrow \textbf{birthdate, birthdate} \rightarrow \textbf{age} \Longrightarrow \textbf{ssn} \rightarrow \textbf{age}$$

Review of normalization concepts

8. Normalization and normal forms

- Normalization: a technique used to iteratively improve relations to **remove undesired redundancy** by decomposing relations and eliminating anomalies.

- Process is iterative
- Process can be performed in stages generally referred to as Normal Forms.

Review of normalization concepts

8. Normalization and normal forms

- **First Normal Form (1NF)**, the relation is analysed and all **repeating groups** are identified to be decomposed into new relations.

- **Second Normal Form (2NF)**, all the **partial dependencies** are resolved/removed.

- **Third Normal Form (3NF):** all the **transitive dependencies** are removed.

Review of normalization concepts

Any questions?

Normalisation exercises

1. Consider the relation Diagnosis with the schema Diagnosis (DoctorID, DocName, PatientID, DiagnosisClass) and the following functional dependencies:

DoctorID → DocName

DoctorID, PatientID → DiagnosisClass

Consider the following instance of Diagnosis:

DoctorID	DocName	PatientID	DiagnosisClass
D001	Alicia	P888	Flu
D002	John	P999	Lactose intolerance
D003	Jennifer	P000	Flu
D002	John	P111	Fever

Identify **different anomalies** that can arise from this schema using the above instance.

Normalisation exercises

Q1: What is the key for Diagnosis

DoctorID	**DocName**	**PatientID**	**DiagnosisClass**
D001	Alicia	P888	Flu
D002	John	P999	Lactose intolerance
D003	Jennifer	P000	Flu
D002	John	P111	Fever

(DoctorID, PatientID)

since together they are sufficient to uniquely identify each record

Normalisation exercises

Q2: Does Insertion anomaly exist? What is it?

DoctorID	DocName	PatientID	DiagnosisClass
D001	Alicia	P888	Flu
D002	John	P999	Lactose intolerance
D003	Jennifer	P000	Flu
D002	John	P111	Fever

Yes.
Example:
Inserting data for a new doctor like DoctorID and DocName, we must insert data of at least one patient associated with the doctor.

Inability to insert records of particular fields is insertion anomaly.

Normalisation exercises

Q3: Does Deletion anomaly exist? What is it?

DoctorID	DocName	PatientID	DiagnosisClass
D001	Alicia	P888	Flu
D002	John	P999	Lactose intolerance
D003	Jennifer	P000	Flu
D002	John	P111	Fever

Yes.
Example:
Deleting patient's data can result in the loss of doctor's data as well resulting in deletion anomaly.

If delete P888 data -> lose record for the doctor named Alicia

Normalisation exercises

Q4: Does Update anomaly exist? What is it?

DoctorID	DocName	PatientID	DiagnosisClass
D001	Alicia	P888	Flu
D002	John	P999	Lactose intolerance
D003	Jennifer	P000	Flu
D002	John	P111	Fever

Yes.
Example:
One doctor may be associated with more than one patient.
An update anomaly may result if a doctor's name is changed for only one patient.

If fail to change the doctor's name from "John" to "John Miller" for both two records -> update anomaly.

Normalisation exercises

2. Consider a relation R (A, B, C, D) with the following FDs:

$$AB \rightarrow C,\ AC \rightarrow B,\ BC \rightarrow A,\ B \rightarrow D$$

The possible candidate keys of R are AB, AC, and BC, since each of those combinations is sufficient to uniquely identify each record.

Let's consider AB for instance. From AB → C we see that AB uniquely identifies C, and since B alone uniquely identifies D, AB together have covered CD, i.e. the entire set of attributes.

List all the **functional dependencies** that violate 3NF. If any, decompose R accordingly. After decomposition, check if the resulting relations are in 3NF, if not decompose further.

Normalisation exercises

2. Consider a relation R (A, B, C, D) with the following FDs:

$$AB \rightarrow C,\ AC \rightarrow B,\ BC \rightarrow A,\ B \rightarrow D$$

Notice: To be in 3NF, a relation should be in 2NF and all the transitive functional dependencies should be removed

Q1: Is This relation in 2NF?

No.

partial functional dependency B → D

Q2: How to decompose it?

Another relation

R1 (A, B, C) and R2 (B, D)

Normalisation exercises

3. Consider the following relation StaffPropertyInspection:

StaffPropertyInspection (propertyNo, pAddress, iDate, iTime, comments, staffNo, sName)

The FDs stated below hold for this relation:

propertyNo, iDate → iTime, comments, staffNo, sName

propertyNo → pAddress
staffNo → sName

From these FDs, it is safe to assume that <u>propertyNo</u> and <u>iDate</u> can serve as a **primary key**.

Your task is to normalise this relation to **3NF**. Remember in order to achieve 3NF, you first need to achieve 1NF and 2NF.

Normalisation exercises

3. Consider the following relation StaffPropertyInspection:

StaffPropertyInspection (<u>propertyNo</u>, pAddress, <u>iDate</u>, iTime, comments, staffNo, sName)

<u>propertyNo</u>, <u>iDate</u> → iTime, comments, staffNo, sName

<u>propertyNo</u> → pAddress
staffNo → sName

Q1: Is this relation in 1NF (repeating groups)?

Yes

Q2: Is this relation in 2NF (partial dependencies)?

No
<u>propertyNo</u> → pAddress
Decompose it!

Normalisation exercises

3. Consider the following relation StaffPropertyInspection:

StaffPropertyInspection (propertyNo, pAddress, iDate, iTime, comments, staffNo, sName)

propertyNo, iDate → iTime, comments, staffNo, sName

propertyNo → pAddress
staffNo → sName

A2: Decompose into 2 relations:

Property (propertyNo, pAddress)

PropertyInspection (propertyNo (FK), iDate, iTime, comments, staffNo, sName)

Q3: Is this relation in 3NF (transitive dependencies)?

No
staffNo → sName

Normalisation exercises

3. Consider the following relation StaffPropertyInspection:

StaffPropertyInspection (<u>propertyNo</u>, pAddress, <u>iDate</u>, iTime, comments, staffNo, sName)

<u>propertyNo</u>, <u>iDate</u> → iTime, comments, staffNo, sName

<u>propertyNo</u> → pAddress
staffNo → sName

A3: Decompose into 3 relations:

Property (<u>propertyNo</u>, pAddress)

Staff (<u>staffNo</u>, sName)

PropertyInspection (<u>propertyNo</u> (FK), <u>iDate</u>, iTime, comments, staffNo (FK))

Normalisation exercises

Any questions?