Database Systems

Tutorial Week 9

Objectives

I. Review of normalisation concepts
II. Normalisation exercises

Anomalies

- 3 types
- What are they?
    - Update
    - Deletion
    - Insertion

Anomalies

Consider the following instance of the relation Allocation:

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
  - A data inconsistency that occurs when we have duplicate values, and we try and update the records, but not all values get updated
  - E.g. say the number of seats in Sidney Myer G09 increases from 20 to 30
    - In this entity, we have to update all rows where room = Sidney Myer G09

Anomalies

Consider the following instance of the relation Allocation:

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

- Deletion anomaly
  - An unintentional loss of certain attribute values when deleting another attribute value
  - E.g. if we remove COMP10001 from this table, we also delete the details of room EDS 6

Anomalies

Consider the following instance of the relation Allocation:

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

- Insertion anomaly
  - The inability to insert new attribute values due to the absence of other values
  - E.g. say a new room “NewAlice109” has been built but hasn’t been timetabled for any course or staff member
    - We can’t add “NewAlice109” to the table

Anomalies

- If a relation is in *denormalised* form, it can lead to these anomalies
- Normalisation helps us remove these anomalies and get rid of redundant data
- To achieve normal forms, we need to understand:
  - Functional dependencies of the attributes of a given relation
  - How to resolve partial dependencies and transitive dependencies

Functional Dependency (FD)

- An FD occurs when a *set* of attributes X $\{X_1, X_2, \ldots, X_n\}$ *determines* another *set* of attributes Y $\{Y_1, Y_2, \ldots, Y_n\}$ for every record of a relation
- In other words:
  - Each value of the set X is associated with only *one* value of the set Y
  - If two records have the same $X_1, X_2, \ldots, X_n$, then they must have the same $Y_1, Y_2, \ldots, Y_n$
  - So if we know X, then we also know Y
- Written: "X → Y"
- Read: "X determines Y"
- E.g. Students(<u>SID</u>, name)
  - SID → name

Determinants

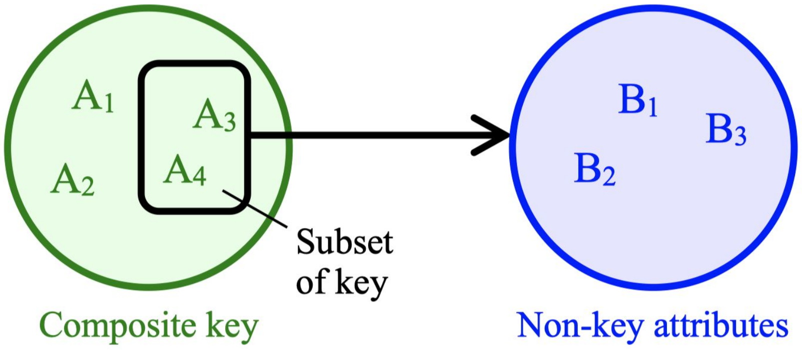
- These are attributes that determine the value of other attributes
- E.g. consider the relation: Person(<u>ssn</u>, name, birthdate, address, age)
  - FDs:
    - birthdate → age
    - ssn → name, birthdate, age, address
  - What are the determinants?
    - *birthdate*
    - *ssn*

Key and Non-Key Attributes

- A key is a set of attributes $\{A_1, A_2, \ldots, A_n\}$ such that
  - $\{A_1, A_2, \ldots, A_n\}$ functionally determines all other attributes
  - No subset of $\{A_1, A_2, \ldots, A_n\}$ functionally determines all other attributes
  - The key must be *minimal*
  - If an attribute is part of the key, it is a *key attribute*
- E.g. consider the relation: Person(<u>ssn</u>, name, birthdate, address, age)
  - What is the minimal key of the Person relation?
    - *ssn*
  - Why is {*ssn*, *name*} not a key?
    - It’s a “super key”, but it’s not the *minimal* set

Partial Functional Dependency

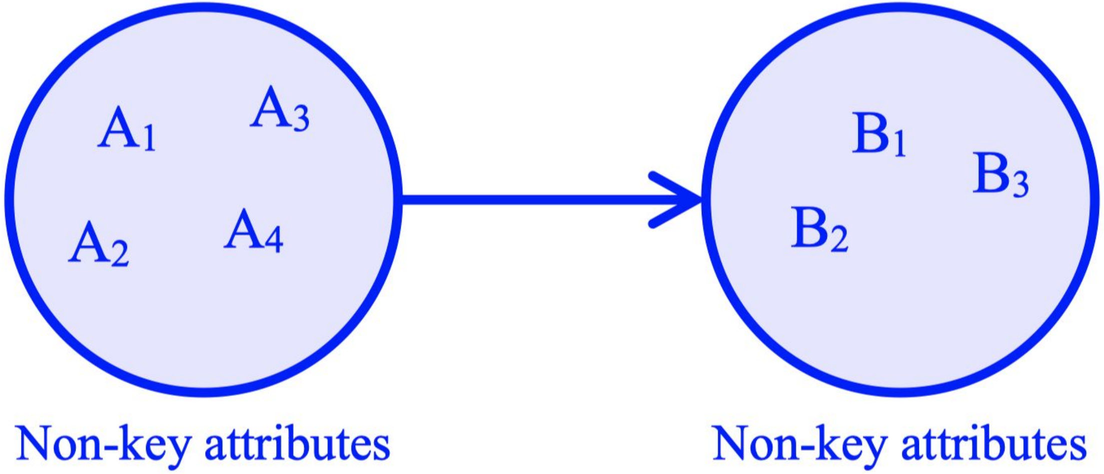
- Occurs when a *subset* of a composite key functionally determines one or more non-key attributes



- Consider a relation R(<u>A</u>, B, C, <u>D</u>) with a composite PK of (A, D), satisfying the functional dependencies $A \rightarrow B$, $D \rightarrow C$
- Functional dependencies like $A \rightarrow B$ and $D \rightarrow C$ are *partial functional dependencies*

Transitive Functional Dependency

- Occurs when a non-key attribute (or a subset of PK and non-key attributes) *determines* another non-key attribute



Armstrong’s Axioms

- Given a relation and a set of FDs, we can infer new FDs using rules known as Armstrong’s Axioms
- 3 important axioms
- What are they?
    1. Reflexivity
    2. Augmentation
    3. Transitivity

Armstrong's Axioms

- Reflexivity
  - Suppose A = {A_1, A_2, …, A_n} is a subset of attributes of R
  - Suppose B = {B_1, B_2, …, B_n} is another subset of attributes of R such that B is a subset of A
  - Reflexivity implies that:
    - $B \subseteq A \Longrightarrow A \rightarrow B$
  - E.g. consider the relation: Person(<u>ssn</u>, name, birthdate, address, age)
    - Suppose A = {*ssn, name*} and B = {*name*}
    - By reflexivity, *ssn, name* → *name*

Armstrong's Axioms

- Augmentation
  - Suppose A = $\{A_1, A_2, \ldots, A_n\}$ is a subset of attributes of R
  - Suppose B = $\{B_1, B_2, \ldots, B_n\}$ is a subset of attributes of R
  - Suppose C = $\{C_1, C_2, \ldots, C_n\}$ is a subset of attributes of R
  - Augmentation implies that:
    - $A \rightarrow B \Longrightarrow AC \rightarrow BC$
  - E.g. consider the relation: Person(<u>ssn</u>, name, birthdate, address, age)
    - Suppose A = {*ssn, name*}, B = {*name*} and C = {*age*}
    - By reflexivity, *ssn, name* → *name* (as before)
    - Now, by augmentation, *ssn, name, age* → *name, age*

Armstrong’s Axioms

- Transitivity
  - $A \rightarrow B$ and $B \rightarrow C \Rightarrow A \rightarrow C$
  - E.g. consider the relation: Person(ssn, name, birthdate, address, age)
    - $ssn \rightarrow birthdate$, $birthdate \rightarrow age \Rightarrow ssn \rightarrow age$

Normalisation and Normal Forms

- Normalisation is a technique used to improve relations to remove undesired redundancy
- The process is performed in stages:
  - To achieve First Normal Form (1NF), all **repeating groups** are identified to be decomposed into new relations
  - To achieve Second Normal Form (2NF), all **partial dependencies** are resolved and removed
  - To achieve Third Normal Form (3NF), all **transitive dependencies** are removed

Exercises — Question 1 (6 minutes)

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.

Exercises — Question 2 (6 minutes)

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 $AB \rightarrow C$ 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.

Exercises — Question 2

- Is the relation in 1NF?
  - Yep
  - There are no repeating groups
- Is the relation in 2NF?
  - Nope
  - There is a partial functional dependency B → D
    - B is part of a composite candidate key (AB and BC)
    - D is a non-key attribute
  - We need to decompose R to remove the partial dependency, but how?
    - Decompose R into R1 and R2
    - R1 contains all attributes except the RHS of the partial dependency
    - R2 contains attributes in the partial dependency, B and D
    - R1(A, B, C) and R2(B, D)
  - Now it's in 2NF
- Is the relation in 3NF?
  - Yep!
  - R1 and R2 don't have any partial or transitive functional dependencies

Exercises — Question 3 (10 minutes)

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 propertyNo and iDate 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.

Exercises — Question 3

- Is the relation in 1NF?
  - Yep
  - There are no repeating groups

Exercises — Question 3

- Is the relation in 2NF?
  - Nope
  - There is a partial functional dependency *propertyNo* → *pAddress*
    - *propertyNo* is part of a composite candidate key (*propertyNo* and *iDate*)
    - *pAddress* is a non-key attribute
  - We need to decompose StaffPropertyInspection to remove the partial dependency, but how?
    - Need to decompose property details into its own relation
    - Decompose StaffPropertyInspection into StaffPropertyInspection and Property
    - StaffPropertyInspection contains all attributes except the RHS of the partial dependency
      - StaffPropertyInspection(propertyNo, iDate, iTime, comments, staffNo, sName)
    - Property contains attributes in the partial dependency, *propertyNo* and *pAddress*
      - Property(propertyNo, pAddress)

Exercises — Question 3

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

Property(propertyNo, pAddress)

- Which attributes should we make PK, FK or PFK?
    - Original relation is StaffPropertyInspection(<u>propertyNo</u>, pAddress, <u>iDate</u>, iTime, comments, staffNo, sName) where *propertyNo* and *iDate* is the composite PK, so keep this the same for now
    - Make the determinant *propertyNo* the PK of the Property relation
    - Make *propertyNo* in StaffPropertyInspection a **PFK**

Exercises — Question 3

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

Property(propertyNo, pAddress)

Now it's in 2NF :)

Recall the FDs:

- propertyNo, iDate → iTime, comments, staffNo, sName
- propertyNo → pAddress
- staffNo → sName

Exercises — Question 3

- Are both relations in 3NF?
  - Property is in 3NF since there is no transitive dependency
  - StaffPropertyInspection is *not* in 3NF since there is a transitive dependency *staffNo → sName*
    - A non-key attribute (*staffNo*) is determining another non-key attribute (*sName*)
  - We need to decompose StaffPropertyInspection to remove the transitive dependency, but how?
    - Need to decompose staff details into its own relation
    - Decompose StaffPropertyInspection into StaffPropertyInspection and Staff
    - StaffPropertyInspection contains all attributes except the RHS of the transitive dependency
      - StaffPropertyInspection(propertyNo, iDate, iTime, comments, staffNo)
    - Staff contains attributes in the transitive dependency, *staffNo → sName*
      - Staff(staffNo, sName)

Exercises — Question 3

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

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

Staff(staffNo, sName)

- Which attributes should we make PK, FK or PFK?
  - After achieving 2NF, *propertyNo* is a PFK and *iDate* is a PK, so keep this the same
  - Make the determinant *staffNo* the PK of the Staff relation
  - Make *staffNo* in StaffPropertyInspection a **FK**

Exercises — Question 3

StaffPropertyInspection(propertyNo FK, iDate, iTime, comments, staffNo FK)

Property(propertyNo, pAddress)

Staff(staffNo, sName)

Now it’s in 3NF :)

Option: W8 or W9 Take Home Questions