Database Systems

Chapter 3: Relational Model Normallization

session 1: Normalization and Functional Dependency

Outline

1. Normalization
2. Functional Dependency
3. Armstrong's axiom
4. Closure of a set attributes
5. Super Key and candidate key

Introduction to Normalization

- ❑ Transforming EER diagrams into relations often results in well-structured relations.
  - However, how do you recognize a poor table structure, and how do you produce a good or well-structured table?
  - There is no guarantee that all anomalies are removed after the transformation.

  ⇒The answer to these questions involves **normalization**

- ❑**Normalization** is a formal process for deciding which attributes should be grouped together in a relation so that all anomalies are removed.
- ❑ **Normalization** is the process of successively reducing relations with anomalies to produce smaller, well-structured relations

Introduction to Normalization

- Purpose of Normalization

  1. Minimize data redundancy, thereby avoiding anomalies and conserving storage space

  2. Simplify the enforcement of referential integrity constraints

  3. Make it easier to maintain data (insert, update, and delete)

  4. Provide a better design that is an improved representation of the real world and a stronger basis for future growth

Data Redundancy and Update Anomalies

- Well-Structured Relations
  - A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data redundancy
- Data redundancy is the duplication of same data in the table. The redundant data increases the size of database and creates issues called update anomalies which may leads to data inconsistency and inaccuracy.

Data Redundancy and Update Anomalies

- Relations that have redundant data may have problems called **update anomalies**.
- Three types of update anomalies
  - **Insertion Anomaly**–adding new rows forces user to create duplicate data
  - **Deletion Anomaly**–deleting rows may cause a loss of data that would be needed for other future rows
  - **Modification Anomaly**–changing data in a row forces changes to other rows because of duplication

Data Redundancy and Update Anomalies

- Example: Employee2 Relation

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

Question–Is this a relation? **Answer–Yes: Unique rows and no multivalued attributes**

Question–What's the primary key? **Answer–Composite: Emp_ID, Course_Title**

Data Redundancy and Update Anomalies

- **Anomalies** in this Relation
  - Insertion–can't enter a new employee without having the employee take a class
  - Deletion–if we remove employee 140, we lose information about the existence of a Tax Acc class
  - Modification–giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities

Data Redundancy and Update Anomalies

- Another example: Product2 relation

ProductNo	Name	UnitPrice	SuppNo	SName	TelNo	ContactPerson
P2344	17 inch Monitor	200	S8843	ABX Technics	56334532	Teresa Ng
P2346	19 inch Monitor	250	S8843	ABX Technics	56334532	Teresa Ng
P4590	Laser Printer	650	S9884	SoftSystem	55212233	Fatimah
P5443	Color Laser Printer	750	S9898	ID Computers	77617709	Larry Wong
P6677	Color Scanner	350	S9990	ITN Suppliers	56345505	Tang Lee Huat
P7700	3 in 1 Printer	400	S9990	ITN Suppliers	56345505	Tang Lee Huat

  - Is the Product2 relation redundant data?

Data Redundancy and Update Anomalies

⇒One solution to deal with this problem is to decompose the relation into smaller relations so that redundancy can be minimized.

⇒ 2 relation: Product and Supplier, they are link together using the supplier number

Supplier

SuppNo	SName	TelNo	ContactPerson
S8843	ABX Technics	56334532	Teresa Ng
S9884	SoftSystem	55212233	Fatimah
S9898	ID Computers	77617709	Larry Wong
S9990	ITN Suppliers	56345505	Tang Lee Huat

Product

ProductNo	Name	UnitPrice	SuppNo
P2344	17 inch Monitor	200	S8843
P2346	19 inch Monitor	250	S8843
P4590	Laser Printer	650	S9884
P5443	Color Laser Printer	750	S9898
P6677	Color Scanner	350	S9990
P7700	3 in 1 Printer	400	S9990

htt

Functional dependency

- A **functional dependency** is a constraint between two attributes or two sets of attributes in which one attribute or group of attributes determines the value of another.
- For any relation R, attribute B is functionally dependent on attribute A *if for each value of A determines only one value of B.*
- The functional dependency of B on A, written **A→ B**

Functional dependency

- Examples:
  - Employe ID determines employee name

    empID → empName
  - Project number determines project name and location

    pNumber → { pName, pLocation}
  - Employee ID and Project number determine the hours per week that the employee work on the project

    { empID, pLocation} →hours

Functional dependency

- In other words:

Let A , B be *sets* of attributes in a relation R
We write A → B or say A *functionally determines* B
if for any tuples t_1 and t_2:

$$t_1[A] = t_2[A] \text{ implies } t_1[A] = t_2[B]$$

"If any 2 tuples of R agree on attribute A value, then they must also agree on B value"

Functional dependency

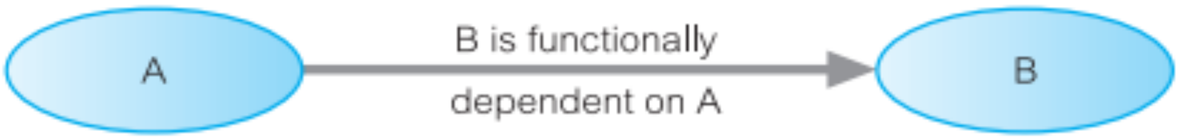
Example: On relation R, check whether these following FDs exist or not?

- ❑A->D
- ❑AB->D
- ❑E->A
- ❑A->E?

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c2	d2	e1
a2	b1	c3	d3	e1
a2	b1	c4	d3	e1
a3	b2	c3	d1	e1

Characteristics of Functional Dependency

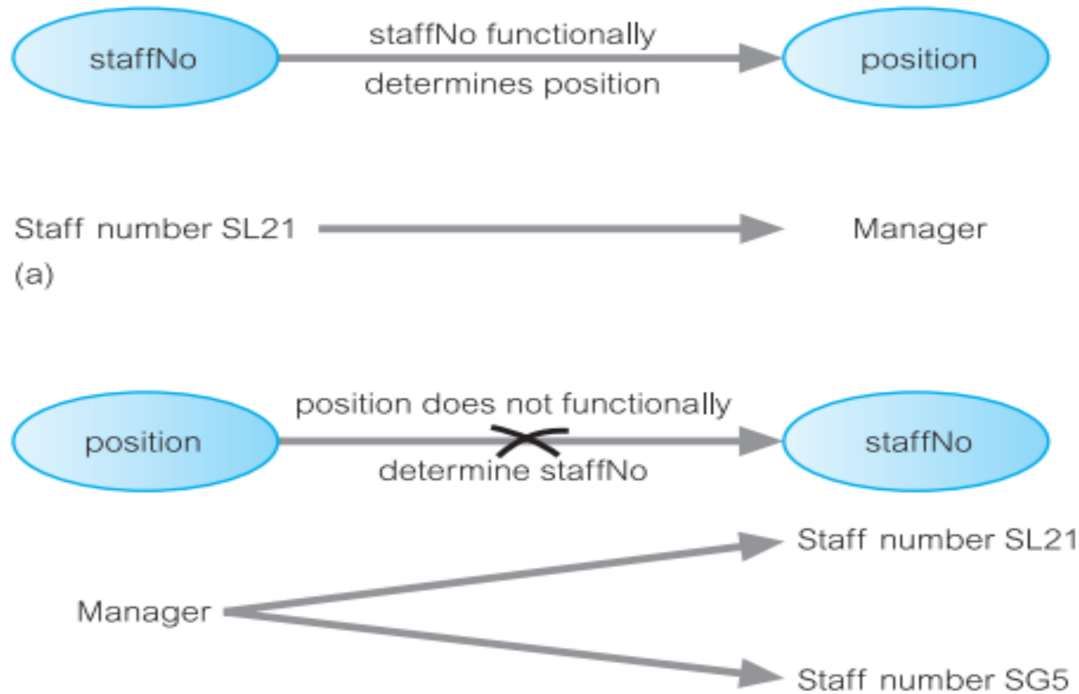
- Diagrammatic representation



- Determinant attribute A
  - Is an attribute or a group of attributes on the left-hand side of the arrow in a functional dependency
- B is the dependent attribute

Characteristics of Functional Dependency

- Example:



(a)staffNo functionally determines position

(staffNo → position)

(b) position does *not* functionally determine staffNo

(position → staffNo)

Characteristics of Functional Dependency

- Three type of Functional Dependency
  - Full Functional Dependency:
    - A → B is a fully dependency if B is **fully functionally dependent** on A, but not on any proper subset of A.
  - Partial Functional Dependency
    - Occurs the determinant is only part of the primary key
    - Ex: if (A, B) → (C, D), B → C and (A,B) is a primary key, then B → C is a partial dependency
  - Transitive Functional Dependency
    - Occurs when an attribute is functionally dependent on another *non-prime* attribute.
    - Ex: X → Y, Y → Z, and X is the primary key, then X → Z is a transitive dependency because X determines the value of Z via Y.

Characteristics of Functional Dependency

❑ Example 1: Staff relation

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

- staffNo → branchNo: is a full functional dependency
- staffNo, sName → branchNo: is not a full functional dependency, because branchNo is also functionally dependent on a subset of (staffNo, sName), namely staffNo

Characteristics of Functional Dependency

- Example 2: StaffBranch relation

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

These following functional dependencies:

- staffNo → sName, position, salary, branchNo, bAddress
- branchNo → bAddress

=> branchNo → bAddress is a transitive functional dependency

Characteristics of Functional Dependency

- Main characteristics of functional dependency used in normalization:
  - There is a one-to-one relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side of a functional dependency
  - They hold for *all* time
  - The determinant has the *minimal* number of attributes necessary to maintain the dependency with the attribute(s) on the right-hand side.

Armstrong's axiom

- Given that *X*, *Y*, and *Z* are sets of attributes in a relation *R*, three simple rules:
  1. **Reflexivity**: If *Y* is a subset of *X* *(or X ⊇ Y)*, then *X* → *Y*
     Reflexivity property is also known as trivial dependency
  2. **Augmentation**: If *X* → *Y*, then *XZ* → *YZ*
  3. **Transitivity:** If *X* → *Y* and *Y* → *Z*, then *X* → *Z*

Armstrong’s axiom

- From these rules, we can derive these rules:

4.Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

5.Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

6.Pseudo transitivity:

If $X \rightarrow Y$ and $YZ \rightarrow W$, then $XZ \rightarrow W$

4.Accumulation: If $X \rightarrow YZ$ and $Z \rightarrow V$, then $X \rightarrow YZV$

5.Extension: If $X \rightarrow Y$ and $W \rightarrow Z$, then $WX \rightarrow YZ$

Armstrong's axiom

- Example: Provide 3 FDs

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Inferred FD	Which Rule did we apply ?
4. {Name, Category} -> {Name}	Reflexivity (1)
5. {Name, Category} -> {Color}	Transitive (4 -> 1)
6. {Name, Category} -> {Category}	Reflexivity (1)
7. {Name, Category -> {Color, Category}	Union(5,6)
8. {Name, Category} -> {Price}	Transitive (7 -> 3)

Closure of a set of Attributes

- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X.
- Example:
  - Let us consider a relation R(A,B,C,D) and functional dependency set F = { A -> B, C -> D, B -> C}.
  - Then $(AB)^+$ is attribute closure of (AB) is all the attributes determined by AB together.
  - $(AB)^+$ = { A, B, C, D}, because A, B are added to set from reflexive property, C and D by transitive property.

Closure of a set of Attributes

- **Algorithm**: Determining X^+, the Closure of the set of attribute X under F

  $X^+ = X$;
  repeat
    $oldX^+ = X^+$;
    for each functional dependency $Y \rightarrow Z$ in F do
    If $X^+ \supset Y$ then $X^+ = X^+ \cup Z$;
  until ($X^+ = oldX^+$);

Closure of a set of Attributes

Example: R (A, B, C, D, E, G, H)

$F = \{B \rightarrow A; DA \rightarrow CE; D \rightarrow H; GH \rightarrow C; AC \rightarrow D\}$

- $X = \{AC\} \Rightarrow X^+ = \{AC\}^+$
  - $X^{(0)} = \{A,C\}$, $\{A,C\} \rightarrow \{D\}$
  - $X^{(1)} = \{A,C,D\}$, $\{A, D\} \rightarrow \{C,E\}$
  - $X^{(2)} = \{A,C,D,E\}$, $\{D\} \rightarrow \{H\}$
  - $X^{(3)} = \{A,C,D,E,H\}$
  - $X^+ = X^{(3)}$
- **So, {AC}+ = {A, C, D, E, H}**
- $X = \{B, D\}$ -> X^+?

Closure of a set of Attributes

- Example:

name → color
category → department
color, category → price

Closures:

name^+ = {name, color}
$\{\text{name, category}\}^+$ = {name, category, color, department, price}
color^+ = {color}

Closure of a set of Attributes

- Ex1: R (A, B, C, D, E, F)

  $F = \{AB \rightarrow C; AD \rightarrow E; B \rightarrow D; AF \rightarrow B\}$

  - Compute $\{AB\}^+$; $\{AF\}^+$

- Ex2: R(A,B,C,D,E,G)

  $F = \{ A \rightarrow C; A \rightarrow EG; B \rightarrow D; G \rightarrow E\}$

  - Compute $\{AB\}^+$; $\{CGD\}^+$

Closure of a set of Attributes

Why Do We Need Closure?

- With closure we can find all FD's easily
- In order *to verify whether or not a functional dependency $X \rightarrow Y$ could be derived from a set F of FDs*, the following simple test could be applied
  - ✓ Compute X^+
  - ✓ Check if $Y \in X^+$

Closure of a set of Attributes

Example:

- Does rule $AB \rightarrow EH$ follow from F = { $AB \rightarrow E$, $AG \rightarrow J$, $BE \rightarrow I$, $E \rightarrow G$, $GI \rightarrow H$}
- Compute $(AB)^+ = ABEIGHJ$
- $EH \subseteq (AB)^+$

So $AB \rightarrow EH$ is inferred from F.

Super key and candidate key

- Super Key: A set of attributes that uniquely determines all the attributes of a relation is known as super key.
- A set X of attributes in R is a super key of R if and only if **X^+ contains all attributes of R**. In other words, X is a super key if and only if it determines all other attributes.
- Example: Let a relation R(A,B,C,D) if A is a super key of the relation then A -> B, A -> C and A -> D (A determines all the attributes (A, B, C, D))

Super key and candidate key

- Candidate key: A candidate key is also a super key but it is minimal. That is, there is no proper subset of candidate key that is super key.
- If we remove any attribute from candidate key it cannot determine all the attributes of the relation.

Finding Super Key or Candidate Key

For each set of attributes X

1. Compute X+
2. If X^+ = set of all attributes then X is a super key
3. If X is minimal, then it is a key

Finding Super Key or Candidate Key

```
Product(name, price, category, color)
```

```
{name, category} → price
{category} → color
```

What is a key?

Example of Finding Super Key or Candidate Key

```
Product(name, price, category, color)
```

```
{name, category} → price
{category} → color
```

$\{name, category\}^+ = \{name, price, category, color\}$

= the set of all attributes

⟹ this is a superkey

⟹ this is a key, since neither **name** nor **category** alone is a superkey

Example of Finding Super Key or Candidate Key

- R(A,B,C,D) and F = { A -> B, C -> D}.
  - Let A^+ = {A, B}
  - B^+ = {B }

We can check other attribute closure to find out whether they determine all attributes or not.

  - $\{AC\}^+$ = (A,C,B, D)
  - $\{ABCD\}^+$ = (A B C D)
  - $\{ABC\}^+$ = (A B C D)

- So, AC, ABCD, ABC are super keys. And AC is a minimal which is essential to determine all attributes, so it is a candidate key.

An efficient algorithm to finding candidate Keys

- Suppose L contains those attributes that occur only on the left-hand side of FDs (or L is the set of attribute not occurred on the right- hand side of any FD)
- Suppose R contains those attributes that occur only on the right-hand side of FDs
- Suppose M contains those attributes that occur on both side of FDs

We only focus only on L and M to find all keys

Tip: *The attributes that are not in right-hand side of FDs are the essential components of candidate key*

Example

- Given R (A, B, C, D)

  $F = \{AB \rightarrow C, C \rightarrow B, C \rightarrow D\}$

- What are all the super keys of R?

  L = A

  M = B, C

  R = D

  - Compute L: A^+= A
  - Try adding B in M to A^+, compute AB^+= ABCD, so AB is a super key.
  - Try adding C in M to A+, compute AC^+ = ABCD, so AC is super key.

  =>AB, AC are also candidate keys.

Observe that prime attribute are A, B, C and non-prime is D attribute.

Example

- Given R(A,B,C,D,E)

  F= {A->C, C->BD, D->A}
- What are all the super keys, candidate keys of R?

Observe that E doesn't appear on the right hand side of any FDs. So every candidate keys must contain E. And B only appears on the right hand side, so no candidate keys contain B.

L = E

M = A, C, D

R = B

We have L^+= E^+= E so E is not a super key.

Try to testing AE, CE, DE

AE+= AECBD so **AE is a super key and also a candidate key** since neither A nor E is a super key.

Similar to the above, **CE, DE are also candidate keys.**

Exercises

- For each Ex1, Ex2:
  - What are all the keys of R?
  - What are all the super-keys for R that are not candidate keys?
- Ex1: R (A, B, C, D)

  $F = \{BC \rightarrow D, D \rightarrow A, A \rightarrow B\}$
- Ex2: R (A, B, C, D)

  $F = \{AD \rightarrow B, AB \rightarrow C, BC \rightarrow D, CD \rightarrow A\}$

Thank You !