DBMS Week 5 TA Session

Redundancy

- Having multiple copies of same data in the database
- It leads to **anomalies**

Anomaly

- Inconsistencies that can arise due to data changes in a database with insertion, deletion, and update

Types of Anomalies

- Insertions Anomaly
- Deletion Anomaly
- Update Anomaly

Redundancy and Anomaly

- Redundancy $\implies$ Anomaly
- Dependency $\implies$ Redundancy
- Good Decomposition $\implies$ Minimization of Dependency
- Normalization $\implies$ Good Decomposition

Functional Dependencies

- It is a relationship between two sets of attributes.
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes
- Let R be a relational schema, $\alpha \subseteq R$ and $\beta \subseteq R$
- The functional dependencies or **FD** is $\alpha \rightarrow \beta$ holds on R if and only if for any legal relations $r(R)$, whenever any two tuples t1 and t2 of r agree on the attributes α, they also agree on the attributes β.

$$t1[\alpha] = t2[\alpha] \implies t1[\beta] = t2[\beta]$$

Example

StudentID	Semester	Lecture	TA
1234	6	Numerical methods	John
1221	4	Numerical methods	Smith
1234	6	Visual computing	Bob
1201	2	Numerical methods	Peter
1201	2	Physics II	Simon

- StudentID $\rightarrow$ Semester
- {StudentID, Lecture} $\rightarrow$ TA
- {StudentID, Lecture} $\rightarrow$ {TA, Semester}

Trivial Dependencies

- A functional dependency is trivial if it satisfies below condition
- $\alpha \to \beta$ **is trivial if** $\beta \subseteq \alpha$

Example

- ID, name $\to$ ID
- name $\to$ name

Note

- Relationship between a set functional dependencies F and its closure F^+
- $F \subseteq F^+$

Armstrong's Axioms

- **Reflexivity:** if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
- **Augmentation:** if $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$
- **Transitivity:** if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

- **Union:** if $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds
- **Decomposition:** if $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds
- **Pseudotransitivity:** if $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds

Closure of Attribute set

- The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.

Example

Consider a $R(A, B, C, G, H, I)$ and $FD = \{A \to B, G \to CH, BC \to GI\}$

- $(AC)^+ = AC$
- $(AC)^+ = ACB$
- $(AC)^+ = ACBGI$
- $(AC)^+ = ACBGIH$

Candidate Key

- A set of minimal attributes that uniquely identifies a tuple/row

Super Key

- Super set of any candidate key is Super Key

Prime Attribute

- Attributes that are present in the candidate key is prime attribute

Note

- Every Candidate key is a Super Key, $C.K \subseteq S.K$
- If closure of any attribute covers all the attributes in a relation, then it is a SuperKey and if it is minimal, then it is called Candidate key

Formula's to find maximum number of Super Keys

- N attributes, only 1 candidate key(by single attribute) = 2^{N-1}
- N attributes,1 candidate key formed k by attributes = 2^{N-k}
- N attributes, each attribute is a candidate key = $2^N - 1$

- N attributes, 2 different candidate key(formed by one or many attribute)
  - Maximum number of super keys = (S.K with A1) + (S.K with A2) - (S.K with both A1 and A2)
  - Max no of Super Keys = $2^{N-m} + 2^{N-n} - 2^{N-(m+n)}$

Example

studentInfo{enrollment_num, class, section, roll, name}.

{enrollment_num} and *{class, section, roll}* are two possible candidate keys.

What is the maximum number of possible superkeys of studentInfo?

- Max no of Super Keys = $2^{N-m} + 2^{N-n} - 2^{N-(m+n)}$

$$= 2^{5-1} + 2^{5-3} - 2^{5-(1+3)}$$
$$= 2^4 + 2^2 - 2^1$$
$$= 16 + 2 - 2$$
$$= 18$$

Extraneous Attribute

- An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies

Let's consider $\alpha \to \beta$

1. If $X \in \alpha$, $(\alpha - X)^+$ contains β, then it is extraneous attribute.

2. If $X \in \beta$, Compute α^+ using $F^{'} = (F - \{\alpha \to beta\}) \cup (\alpha \to (\beta - X))$,
   if α^+ contains X, then it is extraneous attribute.

Example for 1st condition

- $F = \{A \rightarrow B, AB \rightarrow C\}$

Let's check B for is an extraneous attribute or not

Now, $F^{'} = \{A \rightarrow B, A \rightarrow C\}$

$(AB - B)^{+}$ = A^{+} = ABC

We got B attribute after taking an closure.

$\therefore$ B is an extraneous attribute

Example for 2nd condition

- $F = \{A \to BC, AB \to CD\}$

Let's check C for is an extraneous attribute or not

Now, $F^{'} = \{A \to BC, AB \to D\}$

$(AB)^{+}$ = AB^{+} = $ABCD$

We got C attribute after taking an closure.

$\therefore$ C is an extraneous attribute

Canonical Cover

- A canonical cover is a set of functional dependencies that is equivalent to a given set of functional dependencies but is minimal in terms of the number of dependencies

Steps to find an canonical cover

1. Write all single attributes on right hand side
2. Remove redundant functional dependencies by Armstrong axioms
3. Remove extraneous attributes

Example

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

$A \rightarrow B$, $A \rightarrow C$

~~A-->B~~ (by redundancy)

$B \rightarrow C$

~~A-->BC~~ (by redundancy, extraneous attribute)

Canonical cover is $A \rightarrow B$, $B \rightarrow C$

Equivalence of Functional Dependency

1. F covers G - $G \subseteq F^+$
   - All relation should satisfy on G while taking the closure of F
2. G covers F - $F \subseteq G^+$
   - All relation should satisfy on F while taking the closure of G

- If it satisfies both F is equivalent to G

Example

$$F1 = \{P \rightarrow Q, Q \rightarrow R, R \rightarrow P\}$$
$$F2 = \{P \rightarrow R, Q \rightarrow R, R \rightarrow Q\}$$

Let's check F1 covers F2

Taking Closure on F1

- $P^+ = PQR$
- $Q^+ = QRP$
- $R^+ = RPQ$

F1 covers F2 because by taking the closure on F1, all the relation satisfies on F2

Example (continued)

$$F1 = \{P \to Q, Q \to R, R \to P\}$$
$$F2 = \{P \to R, Q \to R, R \to Q\}$$

Let's check F2 covers F1

Taking Closure on F2

- $P^+ = PQR$
- $Q^+ = QR$
- $R^+ = RQ$

F2 not covers F1 because by taking the closure on R, it's not able satisfy the dependency $R \to P$ on F1

Loseless Join Decomposition

- Original relation decomposed into smaller relations in such a way that it can be reconstructed through natural joins without any loss of information

- $R_1 \bowtie R_2 = R$
- $R_1 \cap R_2 \neq \phi$
- $R_1 \cup R_2 = R$
- $R_1 \cap R_2 \rightarrow R_1 \; or \; R_2$

- If all the above conditions satisfies, then it's a loseless decompostion

Example

A relation $R(A, B, C, D)$ with functional dependencies
$FD = \{A \to B, B \to C, C \to D\}$ is decomposed into three relations as follows:

$R1(A, B), R2(B, C), R3(C, D)$

- $R_1 \cup R_2 \cup R_3 = R$

- $R_1 \cap R_2 = B$
- $B^+ = BCD$ (R_2 relation)

- Now, $R_{12}(A, B, C)$
- $R_{12} \cap R_2 = C$
- $C^+ = CD$ (R_3 relation)

Dependency Preservation

- When a relation is decomposed, it is important to ensure that functional dependencies that held in original relation still maintained.
- It helps to prevent anomalies and ensures data integrity
- $F \subseteq F^+$, F^+ should preserve all dependency in F

Note

- Loseless Decompostion and Dependency preservation are independent to each other.

Example

A relation $R(A, B, C, D)$ with functional dependencies $FD = \{A \to B, B \to C, C \to D, D \to A\}$ is decomposed into three relations as follows:

$R1(A, B), R2(B, C), R3(C, D)$

$F_1 = \{A \to B, B \to A\}$ based on R_1
$F_2 = \{B \to C, C \to B\}$ based on R_2
$F_3 = \{C \to D, D \to C\}$ based on R_3

Example (Continued)

$F^+ = F_1 \cup F_2 \cup F_3$

$F^+ = \{A \to B, B \to A, B \to C, C \to B, C \to D, D \to C\}$

Now, $D \to A$ is missing F^+

Let's take D^+ closure on F^+

$D^+ = DCBA$

$\therefore D \to A$ dependency is preserved after decomposing it.