RELATIONAL DATABASE DESIGN

Ongole Campus

Rajiv Gandhi University of Knowledge Technologies

Features of good relational designs

- Minimum data redundancy – no duplicated data.
- Consistent data – no inconsistent data.
- Provides efficient access of data.
- Supports data integrity in maintenance.
- No wastage of space.

Problems encountered in bad schema design

- Redundancy may lead to anomalies.
- Anomaly is possibility of certain data getting to inconsistent.
- There are three types of anomalies:
  - Insertion anomalies.
  - Deletion anomalies.
  - Update anomalies.

Solution

- If the table is too large to reduce the redundancy the table can be split in to multiple tables.
- Splitting a single table into multiple tables is called as decomposition.
- Decomposition should not lossy decomposition.

Solution

- After decomposition, if we are able to form the original table by natural join then it is called as lossless join decomposition

- Normalization will be the solution to have a relation schema in good form.

Atomic domains and first normal forms

- Domain of an attribute is ***atomic*** if the values of the attribute are considered indivisible.
- Example for atomic domain:

  id number: int

Atomic domains and first normal forms

First normal form:

A relation schema 'R' is in first normal form (1NF) if has

- The domains of all attributes should be atomic.
- Values of each attribute is single value from its domain.

Atomic domains and first normal forms

First normal form: Example

Id no	name	Contact_no
12345	Sravanthi	9898987651, 9876543222
12346	Sravan	9786756453, 8978675644

The above table is not in first normal form as it has multiple values in “contact_no” attributes.

Atomic domains and first normal forms

First normal form: Example (solution)

Id no	name	Contact_no
12345	Sravanthi	9898987651, 9876543222
12346	Sravan	9786756453, 8978675644

Id no	name	Contact_no_1	Contact_no_2
12345	Sravanthi	9898987651	9876543222
12346	Sravan	9786756453	8978675644

Even though the above table is in 1st normal form , still it is not acceptable because the person may have more than or less than two contact numbers. In this scenario we cannot add new columns.

Atomic domains and first normal forms

First normal form: Example (alternative solution)

Id no	name	Contact_no
12345	Sravanthi	9898987651, 9876543222, 99887789761
12346	Sravan	9786756453,

Id no	name	Contact_no
12345	Sravanthi	9898987651
12346	Sravan	9786756453
12345	Sravanthi	9876543222
12345	Sravanthi	99887789761

Of course, the above solution may solve previously addressed problem but in some cases it may lead to redundancy.

Atomic domains and first normal forms

First normal form: Example (alternative solution)

Id no	name	Contact_no
12345	Sravanthi	9898987651, 9876543222
12346	Sravan	9786756453, 8978675644

Id no	name
12345	Sravanthi
12346	Sravan

Id no	Contact_no
12345	9898987651
12346	9786756453
12345	9876543222
12345	99887789761

Functional dependencies

- Constraints on a legal relation.
- Association among attributes is known as "Functional Dependency" and it is represented as ($\rightarrow$).
- The values for certain set of attributes determines uniquely the values for other set of attributes in a relation.
- It generalized notion of key.

Functional dependencies

Let us consider a relation 'R' and A, B are two sets of non-empty attributes in a relation 'R'.

Then,

The Functional Dependency FD is represented as $A \rightarrow B$.

Where,

'A' functionally determines B.

Therefore, 'A' is called as "Determinant".

Functional dependencies

- A primary key is a special case of FD.
- A Functional Dependency "$A \rightarrow B$" says that

If two tuples agree on the values in the attribute 'A' then they also agree on the values in attributes 'B'.

i.e

$t1.A = t2.A$ then $t1.B = t2.B$

Here, A & B are two sets of attributes and t1 & t2 are two tuples.

Functional dependencies

Example:

emp_id	emp_name	contact	d_name
e1	Ajay	9888767654	ECE
e2	Vijay	7324566695	CSE
e3	shukla	9827445455	CSE
e4	guru	8783457644	ECE
e5	syamala	7454534512	CSE
e6	syamala	9838748563	CSE
e7	guru	7246123542	CSE
e8	yasaswi	8278356231	EEE

Emp_id → emp_name

Emp_id → contact

Emp_id → d_name

contact → emp_name

Functional dependencies

Use of functional dependencies:

- To test the whether then relations are legal under the given FDs.
  - *"r satisfies F"*, if the relation 'r' is legal under the given set of functional dependencies 'F'.
- Specifies the constraints on the set of legal relations.
  - *"F holds on R"*, if all legal relations on 'R' satisfies the set of functional dependencies 'F'.

Functional dependencies

Types of functional dependencies:

- Trivial FDs and Non-trivial FDs.
- Transitive FDs.
- Multi-valued FDs

Functional dependencies

Trivial FDs –

Let,

A and B are two sets of attributes in a relation

$B \subseteq A$

Then,

$A \rightarrow B$ is trivial functional dependency

Functional dependencies

Example for trivial FD:

emp_id	emp_name	contact	d_name
e1	Ajay	9888767654	ECE
e2	Vijay	7324566695	CSE
e3	shukla	9827445455	CSE
e4	guru	8783457644	ECE
e5	syamala	7454534512	CSE
e6	syamala	9838748563	CSE
e7	guru	7246123542	CSE

Emp_id, emp_name → emp_name

Functional dependencies

Non-Trivial FDs –

Let,

A and B are two sets of attributes in a relation

B **not subset of** A

Then,

$A \rightarrow B$ is non-trivial functional dependency

Functional dependencies

Example for non-trivial FD:

emp_id	emp_name	contact	d_name
e1	Ajay	9888767654	ECE
e2	Vijay	7324566695	CSE
e3	shukla	9827445455	CSE
e4	guru	8783457644	ECE
e5	syamala	7454534512	CSE
e6	syamala	9838748563	CSE
e7	guru	7246123542	CSE

Emp_id → emp_name

Functional dependencies

Transitive FDs –

Let,

A, B and C are three sets of attributes in a relation

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

Then,

$A \rightarrow C$

Functional dependencies

Example for transitive FD:

emp_id	emp_name	contact	d_name
e1	Ajay	9888767654	ECE
e2	Vijay	7324566695	CSE
e3	shukla	9827445455	CSE
e4	guru	8783457644	ECE
e5	syamala	7454534512	CSE
e6	syamala	9838748563	CSE
e7	guru	7246123542	CSE

Emp_id → contact

contact → d_name

Emp_id → d_name

Functional dependencies

Multi valued FDs – If multiple independent attribute sets are depends on single attribute set then it can be called as multi valued FD.

Let,

A, B and C are three sets of attributes in a relation

B and C are independent attribute sets

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

Then,

A has multi value FD.

Functional dependencies

Example for multi valued FD:

emp_id	emp_name	contact	d_name
e1	Ajay	9888767654	ECE
e2	Vijay	7324566695	CSE
e3	shukla	9827445455	CSE
e4	guru	8783457644	ECE
e5	syamala	7454534512	CSE
e6	syamala	9838748563	CSE
e7	guru	7246123542	CSE

Emp_id → emp_name

Emp_id → contact

Armstrong's axioms for FD

- Armstrong's Axioms are the inference rules or properties of the functional dependencies.
- These axioms are derived by william w. Armstrong in 1974.
- These rules reduces the set of functional dependencies.
- These axioms are –
  - Sound – They generate only the FDs that actually holds.
  - Complete – Generate all the FDs that hold.

Armstrong's axioms for FD

Basic rules:

- Reflexivity rule – It is trivial FD.
  - If $B \subseteq A$ then, $A \rightarrow B$
- Transitive rule –
  - If $A \rightarrow B$, $B \rightarrow C$ then, $A \rightarrow C$
- Augmentation rule –
  - If $A \rightarrow B$, then $AC \rightarrow BC$

Armstrong's axioms for FD

Additional rules:

- Union –
  - If $A \rightarrow B$ and $A \rightarrow C$ then, $A \rightarrow BC$
- Decomposition rule –
  - If $A \rightarrow BC$, then $A \rightarrow B$ and $A \rightarrow C$
- Pseudo transitive rule –
  - If $A \rightarrow B$, $BC \rightarrow D$ then, $AC \rightarrow D$

Closure of a set of FD's

- The set of all FDs logically implied by F is the closure of F.
- Closure of F is denoted as F^+.
- The closure of set of attributes covers all attributes based on given functional dependency.

Closure of a set of FD's

Algorithm to compute closure of set of FDs F:

- $F^+ = F$
- Repeat
  - For each FD 'f' in F^+
    - Apply reflexivity and augmentation rules on 'f'.
    - Add resulting FD to F^+.
  - For each pair of FD 'f1 and 'f2' in F^+
    - If f1 and f2 can combined on transitivity
    - Then, add resulting FD to F+.
- Until F^+ does not change further.

Closure of a set of FD's

Algorithm to compute X^+, closure of 'X' under set of FDs F:

- $X^+ = X$
- Repeat
  - For each FD '$A \rightarrow B$' in F.
    - If $A \subseteq X^+$ then, $X^+ = X^+ \cup B$.
- Until X^+ does not change further.

Closure of a set of FD's

Algorithm to compute X^+, closure of 'X' under set of FDs F:

- $X^+ = X$
- Repeat
  - For each FD '$A \rightarrow B$' in F.
    - If $A \subseteq X^+$ then, $X^+ = X^+ \cup B$.
- Until X^+ does not change further.

Closure of a set of FD's

Example: Given relation R with schema R(A,B,C,D,E,G,H,K) with set of FD's F:

$A \rightarrow B,\ B \rightarrow DE,\ E \rightarrow GH,\ K \rightarrow H,\ B \rightarrow K$

Find A^+ ?

$A^+ = (A)$

$= (A, B)$ $[A \rightarrow B]$

$= (A, B, D, E, K)$ $[B \rightarrow DE,\ B \rightarrow K]$

$= (A, B, D, E, K, G, H)$ $[E \rightarrow GH,\ K \rightarrow H]$

Closure of a set of FD’s

Uses of attribute closure:

- Testing for super key.
- Testing functional dependencies.
- Compute the closure of ‘F’.

Closure of a set of FD's

Testing for super key –

- To check whether given 'x' is super key of the relation R then follow the following steps
  - Computer closure of x i.e x^+.
  - Check all the attributes of R are in x^+.
    - If true then the 'x' is super key
    - Else 'x' is not super key.

Closure of a set of FD's

Testing functional dependencies –

- To check the FD $A \rightarrow B$ holds in R then follow the steps
  - If $B \subseteq A+$ then, $A \rightarrow B$ holds in R.

Closure of a set of FD's

Example: Given a relation R(A,B,C,D,E) and set of FD's 'F' are $A \rightarrow B$, $A \rightarrow C$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$. Check whether the FD "$CD \rightarrow AC$" is implied by the given FDs set or not.

$(CD)^+ = (CD)$

$= (CDE)$ $[CD \rightarrow E]$

$= (CDEA)$ $[E \rightarrow A]$

$= (CDEAB)$ $[A \rightarrow B]$

There fore, $CD \rightarrow AC$ hold on given relation R.

Minimal cover

- Minimal cover is minimal set of functional dependencies of given FDs.
- Also defined as irreducible set of functional dependencies of given set of FDs.
- It is also known as canonical cover.

Minimal cover

Example: Given a relation R (A,B,C) and set of FDs F as $\{A \rightarrow B, A \rightarrow C, B \rightarrow C\}$.

Here,

F_c i.e minimal of F can be reduced to $\{A \rightarrow B, B \rightarrow C\}$

Minimal cover

- Minimal set of F i.e F_c should be equivalent to the given set of FDs F.
- If the closures of two given FDs sets F1, F2 are equal then only we can say that the F1 and F2 are equivalent.

Minimal cover

If F_c is canonical form of an given FDs set F then,

- F and F_c should be equivalent.
- No FD in F_c contains an extraneous attribute either on LHS or in RHS.
- Each LHS of the FD in F_c should be unique.

Minimal cover

Algorithm to compute canonical cover for F:

- Repeat
  - Use union rule to replace FDs, which are in the form of $A \rightarrow B$, $A \rightarrow C$ with $A \rightarrow B, C$.
  - Find the extraneous attribute on either sides of FD $\alpha \rightarrow \beta$ in F_C. If extraneous attribute found delete the it from $\alpha \rightarrow \beta$.
- Until F_c does not change.

Note: Union rule may need to apply after removing the extraneous attributes deletion.

Minimal cover

Finding extraneous attribute in a FD $\alpha \rightarrow \beta$ in FD set F:

- To check $A \in \alpha$ is extraneous
  - Compute $(\{\alpha\} - A)^+$ using the dependencies in F.
  - If $(\{\alpha\} - A)^+$ contains β then A is extraneous in α else not.
- To check $A \in \beta$ is extraneous
  - Compute α^+ using the dependencies in F1 where

    $F1 = [F - (\alpha \rightarrow \beta)] \cup [\alpha \rightarrow (\beta - A)]$.
  - If α^+ contains A then A is extraneous β else not.

Minimal cover

Example: Given R(A, B, C) and set of FDs F are {$A \rightarrow BC$, $B \rightarrow C$, $A \rightarrow B$, $AB \rightarrow C$ } then find the canonical cover of F.

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

Perform union on F {$\mathbf{A \rightarrow BC}$, $B \rightarrow C$, $\mathbf{A \rightarrow B}$, $AB \rightarrow C$ }

F1 = {$A \rightarrow BC$, $B \rightarrow C$, $AB \rightarrow C$ }

Check A is extraneous in LHS of $AB \rightarrow C$

B^+ is (B, C) and C is in B^+ . Hence, A is extraneous. There fore delete A from $AB \rightarrow C$ on LHS.

The updated F1 is {$A \rightarrow BC$, $B \rightarrow C$}

Check C is extraneous in RHS of $A \rightarrow BC$

A^+ is (A, B, C) where FD set is {$A \rightarrow B$, $B \rightarrow C$} and C is in A^+ . Hence, C is extraneous. There fore after deletion of C from $A \rightarrow BC$ on RHS.

The updated F1 is {$A \rightarrow B$, $B \rightarrow C$}

The canonical cover is {$A \rightarrow B$, $B \rightarrow C$}

Finding candidate keys

- Minimal subset of super keys are called as **candidate keys**.
- Steps to find the candidate key.
  1. Find essential attributes.
  2. Determine the candidate key(s).
     - If all essential attributes together forms candidate key the it is the only one possibility of having the candidate key.
     - Else, essential attributes along with some of the non-essential candidate attributes forms multiple candidate key.

Finding candidate keys

Essential attributes –

- Attributes that are not present on RHS of the all the given functional dependencies.
- Example: Let R(A, B, C, D, E, F) be a relation scheme with the F {A → B, C → D, D → E} then, the attributes A, C and F are essential attributes.
- Attributes which are not the essential attributes are called as non-essential attributes.

Finding candidate keys

Example: Let R(A, B, C, D, E, F) be a relation scheme with the F {A → B, C → D, D → E} then find the candidate key.

Step 1: A, C, F are the essential attributes.

Step 2: $(ACF)^+ = \{A, B, C, D, E, F\}$ contains all the attributes of the given relational schema then the attributes **ACF** together forms candidate keys.

Decomposition

- The process of dividing a single relation into two or more sub-relations is called “Decomposition”.
- Properties of decomposition:
  - Loss less join decomposition.
  - Dependency preserving decomposition.

Decomposition

Loss less join decomposition –

- No information should lost from original relation during decomposition.
- When sub-relations are joined back the same relation should be obtained.

Decomposition

Loss less join decomposition –

Steps to check whether the given decomposition is lossy or lose less:

Let Relation R is split into R_1 and R_2 then,

- $Attr(R_1) \cup Attr(R_2) = Attr(R)$
- $Attr(R_1) \cap Attr(R_2) \neq NULL$
- $Attr(R_1) \cap Attr(R_2) \rightarrow Attr(R_1)$, $Attr(R_1) \cap Attr(R_2) \rightarrow Attr(R_2)$ i.e the common attribute should be candidate key for at least one of the sub-relation.

Decomposition

Loss less join decomposition –

A relation R(ABCD) is decomposed into $R_1(AB)$, $R_2(BC)$ and $R_3(BD)$ with FD set F is {A→ BC, B→C, C→D, D→B} then check for the loss less join decomposition.

Step-1: $AB \cup BC \cup BD = ABCD$

Step-2: $AB \cap BC \cap BD = B$

Step-3: B is key for $R_2(BC)$ because $B \rightarrow C$.

Decomposition

Dependency preservation –

- After decomposition the no functional dependency should be lost.
- Let 'R' is the original relation with FD set 'F' and then 'R' splits into R_1 and R_2 with F_1 and F_2 FD sets respectively.

  Then, the dependency preservation says that

  - $F_1 \cup F_2$ should be 'F'
  - Otherwise, the decomposition not preserving dependency.

Decomposition

Dependency preservation –

A relation R(ABCD) with FD set F is $\{A \rightarrow BC, B \rightarrow C, B \rightarrow D\}$ is decomposed into $R_1(ABC)$ and $R_2(BD)$ with FD sets $F_1 = \{A \rightarrow BC, B \rightarrow C\}$ and $F_2 = \{B \rightarrow D\}$ then check for the dependency preservation decomposition.

$$\{A \rightarrow BC, B \rightarrow C\} \cup \{B \rightarrow D\} = \{A \rightarrow BC, B \rightarrow C, B \rightarrow D\}$$

Therefore, the given decomposition preserves the dependency.

Definition of 2NF

A relation is said to be in 2NF if –

- It follows 1NF.
- Free from partial dependencies.

- If there is a PD in a relation then remove the partial dependent attributes from the original relation and place them in seperate relation along with the copy of determinant.

Definition of 2NF

Partial dependency –

All Non-key attributes in a table should be totally depend on part of primary key is called as "partial dependency(PD)". i.e

Let R (P,N,K) where,

P – Set of prime key attributes.

N – Set of non-prime key attributes.

K – Set of Candidate keys.

Then, **P** → **N** is known as partial dependency.

Definition of 2NF

Partial dependency – (Example)

Given R (A, B, C, D, E, F, G, H, I, J) with FDs set F = {AB → C, A → DE, B → F, F → GH, D → IJ} then check which of the functional dependencies are partial dependencies which are not.

Key: $(AB)^+ = \{A, B, C, D, E, F, G, H, I, J\}$

K – AB

P – A, B

N - C, D, E, F, G, H, I, J

A → DE, B → F are partial dependencies.

Definition of 2NF

Question: Given R(A, B, C, D, E, F, G, H, I, J) with FDs set F = { $AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$, $F \rightarrow GH$, $D \rightarrow IJ$} check whether the relation is in 2NF or not if not then make it to obey 2NF.

Key: $(AB)^+ = \{A, B, C, D, E, F, G, H, I, J\}$

$A \rightarrow DE$, $B \rightarrow F$ are partial dependencies.

Therefore, the given relation is not in the form of 2NF.

$R_1(ADEIJ)$, $R_2(BFGH)$, $R_3(ABC)$

$F1 = \{A \rightarrow DE, D \rightarrow IJ\}$, $F2 = \{B \rightarrow F, F \rightarrow GH\}$, $F3 = \{AB \rightarrow C\}$

Definition of 3NF

A relation is said to be in 3NF if –

- It follows 2NF.
- Free from transitive dependencies (every non – prime attribute of R is non-transitively depends on every key R).

- If there is a Transitive Dependency in a original relation then remove those transitive dependency attributes from 2NF table and place them in separate relation along with copy of determinant.

Definition of 3NF

Question: Given R(A, B, C, D, E, F, G, H, I, J) with FDs set F = { $AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$, $F \rightarrow GH$, $D \rightarrow IJ$} apply the normalization upto 3NF.

2NF –

$R_1(ADEIJ)$ - {$A \rightarrow DE$, $D \rightarrow IJ$}

$R_2(BFGH)$ - {$B \rightarrow F$, $F \rightarrow GH$}

$R_3(ABC)$ - {$AB \rightarrow C$}

Definition of 3NF

Question: Given R(A, B, C, D, E, F, G, H, I, J) with FDs set F = { $AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$, $F \rightarrow GH$, $D \rightarrow IJ$} apply the normalization upto 3NF.

R_1(ADEIJ) - {$A \rightarrow DE$, $D \rightarrow IJ$} Transitive dependency

R_2(BFGH) - {$B \rightarrow F$, $F \rightarrow GH$} Transitive dependency

R_3(ABC) - {$AB \rightarrow C$}

Therefore R_1 and R_2 are not following 3NF.

Definition of 3NF

Question: Given R(A, B, C, D, E, F, G, H, I, J) with FDs set F = { $AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$, $F \rightarrow GH$, $D \rightarrow IJ$} apply the normalization upto 3NF.

$R_1(ADEIJ)$ - {$A \rightarrow DE$, $D \rightarrow IJ$}

$R_{11}(ADE)$ - {$A \rightarrow DE$}

$R_{12}(DIJ)$ – {$D \rightarrow IJ$}

$R_2(BFGH)$ - {$B \rightarrow F$, $F \rightarrow GH$}

$R_{21}(BF)$ - {$B \rightarrow F$}

$R_{22}(FGH)$ – {$F \rightarrow GH$}

$R_3(ABC)$ - {$AB \rightarrow C$}

$R_{11}(ADE)$ - {$A \rightarrow DE$}

$R_{12}(DIJ)$ – {$D \rightarrow IJ$}

$R_{21}(BF)$ - {$B \rightarrow F$}

$R_{22}(FGH)$ – {$F \rightarrow GH$}

$R_3(ABC)$ - {$AB \rightarrow C$}

Definition of BCNF

BCNF – Boyce Codd Normal Form.

A relation is said to be in BCNF if –

- It already in 3NF.
- All determinants are keys.

References

- https://nptel.ac.in/courses/106/105/106105175/
- https://nptel.ac.in/courses/106/106/106106095/
- A. Silberschatatz, H. F. Korth and S. Sudarshan, Database System Concepts, 5/e, McGraw Hill,2006

Thank You