DA 512H: Database Management Systems

Schema Refinement (1)

Debanga Raj Neog
Mehta Family School of Data Science and Artificial Intelligence (MFSDS&AI)
IIT Guwahati

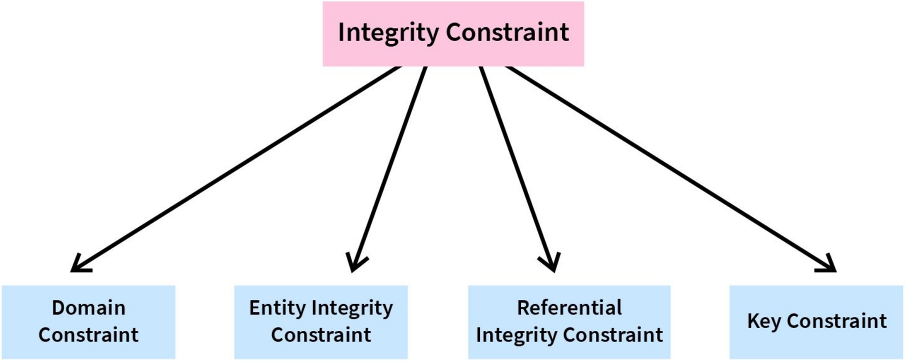
Lecture Plan

- Redundancy Problems
- Handling redundancy problems
    - NULL value and Decomposition
- Decomposition
    - Functional Dependency (FD)
    - Closure of FD
    - Attribute closure

Background

- Conceptual database design gives us a set of **relation schemas** and **integrity constraints** (ICs) that can be regarded as a good starting point for the final database design.

- ER design is a good starting point, but we still need techniques to detect schemas with potential problems and to refine such schemas to eliminate the problems.



Redundancy Problems

- Storing the same information redundantly in a database is the root cause of several problems

Redundancy Problems

- Storing the same information redundantly in a database is the root cause of several problems
- Four types of anomalies
    1. Redundant storage – Same information stored repeatedly

Redundancy Problems

- Storing the same information redundantly in a database is the root cause of several problems
- Four types of anomalies
  1. Redundant storage – Same information stored repeatedly
  2. Update anomaly – Can result in inconsistent data if all redundant data are not simultaneously update

Redundancy Problems

- Storing the same information redundantly in a database is the root cause of several problems
- Four types of anomalies
    1. Redundant storage – Same information stored repeatedly
    2. Update anomaly – Can result in inconsistent data if all redundant data are not simultaneously update
    3. Insertion anomaly – May not be possible to insert a record unless some other **unrelated** information is stored as well

Redundancy Problems

- Storing the same information redundantly in a database is the root cause of several problems
- Four types of anomalies
  1. Redundant storage – Same information stored repeatedly
  2. Update anomaly – Can result in inconsistent data if all redundant data are not simultaneously update
  3. Insertion anomaly – May not be possible to insert a record unless some other **unrelated** information is stored as well
  4. Deletion anomaly – May not be possible to delete a record without deleting some other **unrelated** record

1. Redundant storage

- What’s wrong?

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1102
Sindhu	MA518	1102

1. Redundant storage

- What's wrong?

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1102
Sindhu	MA518	1102

If every course is in only one room, contains ***redundant*** information!

2. Update Anomaly

- What's wrong?

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1205
Sindhu	MA518	1102

2. Update Anomaly

- What's wrong?

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1205
Sindhu	MA518	1102

If we update the room number for one tuple, we get inconsistent data = an ***update anomaly***

3. Insertion Anomaly

- What's wrong?

	CS348	**1103**

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1102
Sindhu	MA518	1102

3. Insertion Anomaly

- What’s wrong?

	CS348	**1103**

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1102
Sindhu	MA518	1102

We can’t reserve a room without students (an **unrelated** information) = an ***insert anomaly***

4. Delete anomaly

- All students have dropped the course!
- What's wrong?

Student	Course	Room
..	..	..

4. Delete anomaly

- All students have dropped the course!
- What’s wrong?

Student	Course	Room
..	..	..

If everyone drops the class, we lose what room the class is in(an **unrelated** information)! = a ***delete anomaly***

Addressing anomalies – NULL value

- Allow NULL value
  - Cannot resolve redundant storage or updation anomaly
  - Can partially resolve insertion and deletion anomalies
  - But may be limiting as in some cases as we cannot replace primary key with NULL

<u>RollNo</u>	Student	Course	Room
1201	Mahesh	MA518	1102
1202	Paes	MA518	1102
1203	Sindhu	MA518	1102

Addressing anomalies – NULL value

- Allow NULL value
  - Cannot resolve redundant storage or updation anomaly
  - Can partially resolve insertion and deletion anomalies
  - But may be limiting as in some cases as we cannot replace primary key with NULL

		CS348	**1103**

(Reserving a room without knowing the students)

1204	**Harry**		

(Registering a student without assigning a course)

RollNo	Student	Course	Room
1201	Mahesh	MA518	1102
1202	Paes	MA518	1102
1203	Sindhu	MA518	1102

Addressing anomalies - Decomposition

- Decomposition
  - Replace a relation with a collection of “smaller” relations

Student	Course	Room
Mahesh	MA518	1102
Paes	MA518	1102
Sindhu	MA518	1102

- Redundancy?
- Update anomaly?
- Delete anomaly?
- Insert anomaly?

Student	Course
Mahesh	MA518
Paes	MA518
Sindhu	MA518

Course	Room
MA518	1103

Problems with decomposition

- What problems (if any) will the decomposition cause?
  - May need to perform joins to answer queries. While performing can lose information.
    - To avoid losing information while performing joins, the decomposition should have lossless-join property
  - May not be able to enforce all the constraints on the smaller relations
    - The decomposition should have dependency-preserving property. That is enforce constraints on the original relation by enforcing some constraints on the smaller relations
  - If queries require join operation on the decomposed relations frequently, there can be performance issues
- Some of these properties we will discuss during the discussion of Normal Forms.

Problems with decomposition

- What can help in deciding whether to perform decomposition?
- Answer: **Normal Forms**
  - If a relation is in one of the normal forms, then it is known that certain problems cannot arise
- The theory behind normal forms is **functional dependencies**

Functional Dependency (FD)

- Functional Dependency is a relation between attributes of a relation schema
- Let R be a relation schema and X,Y be non-empty sets of attributes in R
- A **functional dependency** X $\rightarrow$ Y holds over relation R if, for every allowable instance *r* of R:
  - $t1 \in r,\ t2 \in r, \pi_X(t1) = \pi_X(t2)$ implies $\pi_Y(t1) = \pi_Y(t2)$
  - i.e., X $\rightarrow$ Y means whenever two tuples agree on X then they agree on Y (X and Y are *sets* of attributes.)
- K is a candidate key for R means that K $\rightarrow$ R
  - However, K $\rightarrow$ R does not require K to be *minimal*!

Example: Functional Dependencies

- The figure satisfies the FD AB $\longrightarrow$ C
- Now, if we add a tuple <a1, b1, c2, d1> than the FD will be violated

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

- Role of FDs in detecting redundancy:
  - Consider a relation R with 3 attributes, ABC.
    - No FDs hold: There is no redundancy here.
    - Given A $\longrightarrow$ B: Several tuples could have the same A value, and if so, they'll all have the same B value!

Finding FDs?

- We find out from application domain that a relation satisfies some FDs. Does it mean we have found all the FDs?
- There could be more FDs implied by the ones we have

Provided FDs:

1. Name $\longrightarrow$ Color
2. Category $\longrightarrow$ Department
3. Color, Category $\longrightarrow$ Price

Does it always hold? **Name, Category $\longrightarrow$ Price**

Answer: Find the **closure** of the provided FDs.
That is the set of all FDs that can be inferred from the provided FDs

Closure of FDs

- Set of all FDs implied by a given set F of FDs is called the closure of F
  - denoted as $\mathbf{F^+}$
- Answer: Three simple rules called **Armstrong's Axioms.**
  - Reflexivity: If $X \supseteq Y$, then $X \rightarrow Y$
  - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$
  - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Additional Rules
  - Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Ex: Inferred FD

Answer: Three simple rules called **Armstrong's Rules.**

- Reflexivity: If X ⊇Y, then X→ Y
- Augmentation: If X→ Y, then XZ→ YZ
- Transitivity: If X→Y and Y→ Z, then X→ Z

Additional Rules

- Union: If X→ Y and X→ Z, then X→YZ
- Decomposition: If X→YZ, then X→ Y and X→ Z

Provided FDs:

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

Which / how many other FDs hold?

Inferred FDs:

Inferred FD	Rule used
4. Name, Category ⟶ Name	?
5. Name, Category ⟶ Color	?
6. Name, Category ⟶ Category	?
7. Name, Category ⟶ Color, Category	?
8. Name, Category ⟶ Price	?

Ex: Inferred FD

Answer: Three simple rules called **Armstrong's Rules.**

- Reflexivity: If X ⊇Y, then X→ Y
- Augmentation: If X→ Y, then XZ→ YZ
- Transitivity: If X→Y and Y→ Z, then X→ Z

Additional Rules

- Union: If X→ Y and X→ Z, then X→YZ
- Decomposition: If X→YZ, then X→ Y and X→ Z

Inferred FDs:

Inferred FD	Rule used
4. Name, Category ⟶ Name	Reflexivity
5. Name, Category ⟶ Color	Transitive (4 -> 1)
6. Name, Category ⟶ Category	Reflexivity
7. Name, Category ⟶ Color, Category	Union (5 + 6)
8. Name, Category ⟶ Price	Transitive (7 -> 3)

Provided FDs:

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

Can we find an algorithmic way to do this?

Coming back to our question

Provided FDs (F):

1. {Name} $\longrightarrow$ {Color}
2. {Category} $\longrightarrow$ {Dept.}
3. {Color, Category} $\longrightarrow$ {Price}

Inferred FD	Rule used
4. Name, Category $\longrightarrow$ Name	Reflexivity
5. Name, Category $\longrightarrow$ Color	Transitive (4 -> 1)
6. Name, Category $\longrightarrow$ Category	Reflexivity
7. Name, Category $\longrightarrow$ Color, Category	Union (5 + 6)
8. Name, Category $\longrightarrow$ Price	Transitive (7 -> 3)

- Does the FD always hold? **Name, Category $\longrightarrow$ Price**
- Yes, because the FD is in the closure of F^+
- I just wanted to check $X \longrightarrow Y$, do I need to compute closure of the set of FDs F?
- No, compute the Attribute closure (X^+) wrt F, which is the set of attributes B
- $X \longrightarrow B$ can be inferred using the Armstrong axioms

Closure of a set of Attributes

- Given a set of attributes $X=\{A_1, \ldots, A_n\}$ and a set of FDs F:
- Then the closure, $\{A_1, \ldots, A_n\}^+$ is the set of attributes B, s.t.

$$\{A_1, \ldots, A_n\} \rightarrow B$$

Example: F =

name → color
category → department
color, category → price

Closures:

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

Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$ and set of FDs F.

Repeat until X doesn't change; **do**:

if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F

and $\{B_1, \ldots, B_n\} \subseteq X$

then add C to X.

Return X as X^+

Ex: Closure Algorithm

- Find all FD's implied by

$$\begin{aligned} &A,B \rightarrow C \\ &A,D \rightarrow B \\ &B \rightarrow D \end{aligned}$$

- Requirements

1. Non-trivial FD (i.e., no need to return A, B $\rightarrow$A)

Ex: Closure Algorithm

- Step 1: Compute X+, for every set of attributes X:

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

Given F =

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

Ex: Closure Algorithm

- Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \Phi$:

Given F =

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

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

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

Closure Algorithm

Name, Category $\longrightarrow$ Price ???

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

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

F =

name $\longrightarrow$ color

category $\longrightarrow$ dept

color, category $\longrightarrow$ price

Closure Algorithm

Name, Category $\longrightarrow$ Price ???

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

F =
name $\longrightarrow$ color
category $\longrightarrow$ dept
color, category $\longrightarrow$ price

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

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

Closure Algorithm

Name, Category $\longrightarrow$ Price ???

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

F =

name $\longrightarrow$ color

category $\longrightarrow$ dept

color, category $\longrightarrow$ price

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

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

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

Closure Algorithm

Name, Category $\longrightarrow$ Price ???   Yes

Start with X = $\{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

F =
name $\longrightarrow$ color
category $\longrightarrow$ dept
color, category $\longrightarrow$ price

{name, category}$^+$ =
{name, category}

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

{name, category}$^+$ =
{name, category, color, dept}

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

Ex-1

R(A,B,C,D,E,F)

$A,B \rightarrow C$
$A,D \rightarrow E$
$B \rightarrow D$
$A,F \rightarrow B$

- Check whether $AB \rightarrow E$, $AB \rightarrow F$ and $AF \rightarrow E$?

Ex-1

R(A,B,C,D,E,F)

$A,B \rightarrow C$
$A,D \rightarrow E$
$B \rightarrow D$
$A,F \rightarrow B$

Compute $\{A,B\}^+ = \{A, B, \qquad\qquad\qquad \}$

R(A,B,C,D,E,F)

$A,B \rightarrow C$
$A,D \rightarrow E$
$B \rightarrow D$
$A,F \rightarrow B$

Compute $\{A,B\}^+ = \{A, B, C, D \qquad\qquad \}$

R(A,B,C,D,E,F)

$A,B \rightarrow C$
$A,D \rightarrow E$
$B \rightarrow D$
$A,F \rightarrow B$

Compute $\{A,B\}^+ = \{A, B, C, D, E\}$

Assignment-1

R(A,B,C,D,E,F)

$A,B \rightarrow C$
$A,D \rightarrow E$
$B \rightarrow D$
$A,F \rightarrow B$

Compute $\{A, F\}^{+} = \{A, F, \quad B, D, C, E\}$

Summary

- Redundancy Problems
  - Storage redundancy, updation, insertion and deletion anomaly
- Handling Redundancy
  - NULL values and Decomposition
- Closure of FDs
  - Armstrong Rules – Reflexivity, Augmentation and Transitivity
  - Additional Rules – Union and Decomposition
- Attribute Closure algorithm