Relational Database Design Theory (II)

Spring, 2024

Announcements

- Assignment (I) due: Apr 7, 2024
  - Please test your queries extensively before submission.
- Assignment (II) has been released.

Functional dependencies

Let $X = \{A_1, \ldots, A_n\}$ and $Y = \{B_1, \ldots, B_m\}$ be sets of attributes.

Definition [Functional dependency]

A functional dependency (FD) is of the form

$$X \rightarrow Y$$

that requires the attributes of X functionally determining the attributes Y.

In particular, a relation R satisfies $X \rightarrow Y$ if for every two tuples t_1 and t_2 of R

$$\wedge_{i=1}^{n} t_1[A_i] = t_2[A_i] \rightarrow \wedge_{j=1}^{m} t_1[B_j] = t_2[B_j].$$

- FD's are unique-value constraints.
- A FD $X \rightarrow Y$ holds on a relational schema R if every instance of R satisfies $X \rightarrow Y$.
- If $Y \subseteq X$, then $X \rightarrow Y$ is trivial.

Notation convention

- $A_1 \dots A_n$ represents $\{A_1, \dots, A_n\}$.
- Attributes: A, B, C, D, E
- Sets of attributes: X, Y, Z
- XY represents $X \cup Y$

Anomalies in a bad design

sid	cid	cname	room	grade
123	AI-3613	Database	1-108	A+
223	AI-3613	Database	1-108	B+
123	CS-101	CS Intro.	3-325	A
334	CS-101	CS Intro.	3-325	A-
345	ICE-1404P	Database	2-203	A

Table: R(sid, cid, cname, room, grade)

- Insertion anomaly: Cannot add data to db due to the absence of other data.
  - What happens if we want to add a new course CS2950?
- Deletion anomaly: Lose unintended information as a side effect when deleting tuples.
  - What happens if the student with sid 345 quit the course ICE-1404?
- Update anomaly: To update info of one tuple, we may have to update others as well.
  - What happens if the room of AI-3613 is changed?

Normalization theory

- Decide whether a particular relation schema R is in "good" from.
- In the case that R is not in "good" form, decompose R into a set of relation schemas $\{R_1, R_2, \ldots, R_n\}$ such that each R_i is in good form (normal form).
- The resulting decomposition should avoid anomalies.

A better design

Goal: Decompose R into R_1 and R_2 s.t.

$$R = R_1 \bowtie R_2$$

sid	cid	cname	room	grade
123	AI-3613	Database	1-108	A+
223	AI-3613	Database	1-108	B+
123	CS-101	CS Intro.	3-325	A
334	CS-101	CS Intro.	3-325	A-
345	ICE-1404P	Database	2-203	A

Table: R(sid, cid, cname, room, grade)

s_id	c_id	grade
123	AI-3613	A+
223	AI-3613	B+
123	CS-101	A
334	CS-101	A-
345	ICE-1404P	A

Table: R_1(sid, cid, grade)

c_id	cname	room
AI-3613	Database	1-108
CS-101	CS Intro.	3-325
ICE-1404P	Database	2-203

Table: R_2(cid, cname, room)

- $F = \{\text{cid} \to \{\text{cname}, \text{room}\}, \quad \{\text{sid}, \text{cid}\} \to \text{grade}\}$.
- cid is a superkey of R_2, i.e., $\text{cid} \to \{\text{cid}, \text{cname}, \text{room}\}$.

Decomposition criteria

- Lossless join

  Be able to reconstruct the original relation by joining smaller ones.

- Redundancy and anomalies avoidance

  Avoid unnecessary redundancy and anomalies.

- Dependency preservation

  Minimize the cost to check the integrity constraints defined in terms of FD's.

Lossless join decomposition

Let R be a relation schema consists of attributes $A_1, \ldots, A_n$.

A decomposition of relation schema R is to replace R by

$$R_1, \ldots, R_k$$

for some $k \geqslant 2$ such that

- Each R_i contains a subset of $\{A_1, \ldots, A_n\}$ for $i = 1, \ldots, k$, and
- Every attribute of R appears as an attribute of at least one of the new relations.

Definition

A decomposition $R_1, \ldots, R_n$ of R is lossless join if for every instance I of R, it holds that

$$I = I(R_1) \bowtie \ldots \bowtie I(R_n).$$

With lossless join decomposition, we are able to reconstruct the original relation via join.

Lossless join decomposition (cont'd)

Lemma 1

Suppose that R is decomposed into R_1 and R_2. If either $R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2$, then the decomposition is join lossless.

Proof. Let I be an relation instance of R.

1. $I \subseteq \Pi_{R_1}(I) \bowtie \Pi_{R_2}(I)$ holds for all instances.
2. $\Pi_{R_1}(I) \bowtie \Pi_{R_2}(I) \subseteq I$.

Assume w.l.o.g. that $R_1 \cap R_2 \to R_1$. Let t be a tuple in $\Pi_{R_1}(I) \bowtie \Pi_{R_2}(I)$, we show that $t \in T$.

There are tuples $t_1, t_2 \in I$ such that

$$\Pi_{R_1}(t_1) = \Pi_{R_1}(t) \text{ and } \Pi_{R_2}(t_2) = \Pi_{R_2}(t).$$

Since $\Pi_{R_1 \cap R_2}(t_1) = \Pi_{R_1 \cap R_2}(t_2)$ and I satisfies $R_1 \cap R_2 \to R_1$, we have $\Pi_{R_1}(t_2) = \Pi_{R_1}(t)$.

It follows that $t_2 = t$. Thus t is also in I. □

Boyce-Codd Normal Form

Definition [Boyce-Codd Normal Form]

A relation schema R is in Boyce-Codd Normal Form (BCNF) w.r.t. a set F of FD's if for every FD $X \to Y$ in the closure F^+ with $X \subseteq R$ and $Y \subseteq R$, one of the following holds:

- $X \to Y$ is trivial.
- X is a superkey of R, i.e., $X \to R$ is in F^+.

A database scheme is in BCNF if every relation scheme in it is in BCNF.

Example

- $R = (A, B, C)$, $F = \{A \to B, B \to C\}$. Then R is not in BCNF.
- $R_1 = (A, B)$, $R_2 = (B, C)$, $F = \{A \to B, B \to C\}$. Then both R_1 and R_2 are in BCNF.

Why using BCNF

A	B	C
1	2	3
...	...	...
1	4	?

Table: $R(A, B, C)$ with FD $\{A \rightarrow C\}$

- If a table is not in BCNF, then some attributes' value can derived using FDs.
  - In the table $R(A, B, C)$, the missing value must be 3 by the FD rule $A \rightarrow C$.
- BCNF: every attribute in every tuple contains data that cannot be inferred by FDs.
  - If a relation is in BCNF, then no redundancy can be observed by means of FDs.

BCNF decomposition algorithm

Input: A schema R and a set F of FD's
Output: A BCNF decomposition $\{R_1, \ldots, R_n\}$ of R
1. $\mathcal{D} \leftarrow \{R\}$;
2. while ex. some $R' \in \mathcal{D}$ that is not in BCNF do
3. choose a non-trivial $X \to Y$ in F^+ with $XY \subseteq R'$ and $X \not\to R'$;
4. $R_1 \leftarrow XY$; $R_2 \leftarrow X \cup (R' \setminus XY)$;
5. $\mathcal{D} \leftarrow (\mathcal{D} \setminus \{R'\}) \cup \{R_1, R_2\}$; // decompose R' to R_1 and R_2;
6. return $\mathcal{D}$;

Figure: BCNF decomposition algorithm

Example

Let $R = (A, B, C, D, E)$ and $F = \{A \to B, BC \to D\}$.

- $\mathcal{D}_1 = \{(A, B), (A, C, D, E)\}$ // using $A \to B$
- $\mathcal{D}_2 = \{(A, B), (A, C, D), (A, C, E)\}$ // using $AC \to D$

Remark. Every decomposition step is lossless.

Dependency preserving decomposition

Definition

Let F be a set of FD's on a schema R, and let $R_1, \ldots, R_n$ be a decomposition of R. The restriction of F to R_i is the set F_i of all FD's in F^+ that include only attributes of R_i.

Definition

Let F be a set of FD's on a schema R. A decomposition $R_1, \ldots, R_n$ of R is dependency preserving w.r.t. F if

$$F^+ = (\bigcup_{i=1}^{n} F_i)^+,$$

where F_i is the restriction of F to R_i.

A decomposition preserves dependencies if its original FD's do not span multiple tables.

BCNF and dependency preserving

Example

Let $R = (A, B, C)$ and $F = \{A \to B, B \to C, A \to C\}$

- A BCNF decomposition of R is $\{R_1 = (A, B),\ R_2 = (B, C)\}$.
- Another BCNF decomposition of R is $\{R'_1 = (A, C),\ R'_2 = (A, B)\}$.

Question. Which decomposition is dependency preserving?

Remark. BCNF decomposition does not warrant dependency preservation.

Third Normal Form (3NF)

Definition [Third Normal Form]

A relation schema R is in Third Normal Form (3NF) w.r.t. a set F of FD's if for every FD $X \to Y$ in F^+ at least one of the following holds:

- $X \to Y$ is trivial
- X is a superkey
- Every attribute in $Y \setminus X$ is contained in a candidate key of R.

Similarly, a database schema is in 3NF if every relation schema in it is in 3NF.

Remark. If R is in BCNF, then R is in 3NF.

3NF example

student_id	advisor_id	dept
125	15733	CS
125	14698	EE
224	14698	EE
246	15733	CS

Table: R(student_id, advisor_id, dept)

Two FD's defined over R

- student_id, dept $\rightarrow$ advisor_id
- advisor_id $\rightarrow$ dept

1. R has two candidate keys
   - {student_id, dept}
   - {student_id, advisor_id}
2. R is not in BCNF but in 3NF.
3. Redundancy and update anomaly in 3NF.

Remark. We can show that R has no dependency preserving BCNF decompositions.

Canonical cover (review)

- A set of FD's F defines a set of unique-value constraints.
- We want a minimal set F' of FD's to reduce constraint checking cost.
- F' should be equivalent to F to ensure correctness.

Definition

A canonical cover F_c for F is a set of FD's equivalent to F such that

- No FD in F_c contains an extraneous attribute.
- Each LHS of a FD in F_c is unique.

A canonical cover F_c of F is a minimal set of FD's equivalent to F.

3NF synthesis algorithm

Input: A schema R and a set F of FD's
Output: A 3NF decomposition $\{R_1, \ldots, R_n\}$ of R
1. computes F_c; $\mathcal{D} \leftarrow \{\}$;
2. for each $X \to Y \in F_c$ do
3. $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{R_i(X, Y)\}$;
4. if no relation schema in $\mathcal{D}$ contains a candidate key of R then
5. $\quad$ let Z be a candidate key of R;
6. $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{R'(Z)\}$;
7. remove redundant relations; // optional
8. return $\mathcal{D}$;

Figure: 3NF synthesis algorithm

3NF synthesis algorithm example

$R = (A, B, C, D, E)$, $F = \{AB \to C, C \to B, A \to D\}$.

R has two candidate keys: ABE, ACE.

1. F is already a canonical cover.
2. Add $R_1(A, B, C)$, $R_2(B, C)$ and $R_3(A, D)$ to $\mathcal{D}$.
3. Add $R_4(A, B, E)$ or $R_4(A, C, E)$ to $\mathcal{D}$.
4. Remove $R_2(B, C)$ from $\mathcal{D}$ since it is part of $R_1(A, B, C)$.

Correctness (I)

- Dependency preservation follows from $F_c^+ = F^+$ directly.
- Lossless join since at least one schema in $\mathcal{D}$ contains a candidate key of R.
- 3NF. Every R_i in $\mathcal{D}$ is in 3NF.

Lemma 2

Let F be a set of FD's holds on a schema R and $R_1, \ldots, R_n$ be a decomposition of R. Furthermore, assume the following:

- For every $X \rightarrow Y$ in F, there exists some R_i that contains all the attributes in XY.
- At least one schema in the decomposition contains a candidate key of R.

Then the decomposition $R_1, \ldots, R_n$ is join lossless.

Correctness (II)

Claim. Let R_i be a schema generated from a FD $X \to Y$ in F_c and $X' \to A$ be an arbitrary non-trivial FD in F_c^+ with $A \in Y$ and $X' \subseteq XY$. Then X' is a superkey of R_i.

Proof. We show that if X' is not a superkey of R_i, then A is extraneous in $X \to Y$.

By assumption, there exists an attribute $B \in X$ s.t. $B \notin (X')^+$. Otherwise, X' is a superkey.

It follows that $F_c \setminus \{X \to Y\}$ implies $X' \to A$. Then

$$(F_c \setminus \{X \to Y\}) \cup \{X \to Y \setminus \{A\}\} \text{ implies } X \to Y.$$

As a consequence, $A \in Y$ is extraneous for $X \to Y$ in F_c. Contradiction. □

More normal forms

- 1st Normal Form (1NF)
- 2ed Normal Form (2NF)
- 3rd Normal Form (3NF)
- Boyce-Codd Normal Form
- 4th & 5th Normal Forms

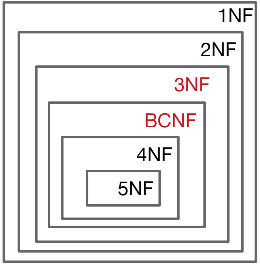

Figure: Normal Forms

Recap

- Lossless join decomposition
- Dependency preserving decomposition
- BCNF and BCNF decomposition algorithm
- 3NF and 3NF synthesis algorithm