COL362/632 Introduction to Database Management Systems

Database Design – Decomposition & Normalization

Kaustubh Beedkar

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Recap

Problems with "bad" schema

-

ActorFilm						
id	**firstname**	**lastname**	**dob**	**title**	**year**	**language**
1	Priyanka	Chopra	1992	Don	2006	Hindi
1	Priyanka	Chopra	1992	Don-II	2011	Hindi
2	Anthony	Hopkins	1937	MI-IV	2011	English
2	Anthony	Hopkins	1937	Valkyrie	2008	English
3	Bill	Nighty	1949	Valkyrie	2008	English

- Problems
  - Redundancy
  - Update anomalies
  - Delete anomalies

Recap

Functional Dependency (FD)

- Type of constraint: FD $X \rightarrow Y$ holds in relation R, if $\forall t_1, t_2 \in R$ whenever $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$
- Trivial & non-trivial FDs
- Inferring FDs – Amstrong's Axioms
- Closure of FDs
  - Given a set S of FDs, the closure S^+ is set of all FDs that can be derived from S
- Closure of attributes (attribute closure method)
  - Given a set S of FDs, can a new FD F be derived from S?

Keys of a Relation

- Superkey
- Candidate key
- Primary attributes
- Primary key

Decomposition

A relation R can be **decomposed** into instances

- R_1 and R_2
- where $[R_1] \cup [R_2] = [R]$
- $R_1 = \pi_{[R_1]}(R)$
- $R_2 = \pi_{[R_2]}(R)$

Schema Decomposition

Example

ActorFilm						
id	**firstname**	**lastname**	**dob**	**title**	**year**	**language**
1	Priyanka	Chopra	1992	Don	2006	Hindi
1	Priyanka	Chopra	1992	Don-II	2011	Hindi
2	Anthony	Hopkins	1937	MI-IV	2011	English
2	Anthony	Hopkins	1937	Valkyrie	2008	English
3	Bill	Nighty	1949	Valkyrie	2008	English

Actor			
id	**firstname**	**lastname**	**dob**
1	Priyanka	Chopra	1992
2	Anthony	Hopkins	1937
3	Bill	Nighty	1949

ActorIdFilm			
id	**title**	**year**	**language**
1	Don	2006	Hindi
1	Don-II	2011	Hindi
2	MI-IV	2011	English
2	Valkyrie	2008	English
3	Valkyrie	2008	English

Is decomposition solving the problems?

Schema Decomposition

Good decomposition should

1. Minimize redundancy
2. Avoid information loss (**lossless-join**)
3. Preserve functional dependencies (**dependency preserving**)
4. Have reasonable query performance

Schema Decomposition

Information Loss (example)

ActorIdFilm			
a_id	**title**	**year**	**language**
1	Don	2006	Hindi
1	Don-II	2011	Hindi
2	MI-IV	2011	English
2	Valkyrie	2008	English
3	Valkyrie	2008	English

ActorFilmYear		
a_id	**title**	**year**
1	Don	2006
1	Don-II	2011
2	MI-IV	2011
2	Valkyrie	2008
3	Valkyrie	2008

YearLanguage	
year	**language**
2006	Hindi
2011	Hindi
2011	English
2008	English

- Decompose R(id, title, year, language) into
  1. R_1(id, title, year)
  2. R_2(year, language)
- **Problem** Don-II in English or Hindi?Information loss!

Schema Decomposition

Recall A relation R can be **decomposed** into instances

- R_1 and R_2
- where $[R_1] \cup [R_2] = [R]$
- $R_1 = \pi_{[R_1]}(R)$
- $R_2 = \pi_{[R_2]}(R)$

Lossless-join Decomposition

A schema decomposition is lossless-join if $R_1 \bowtie R_2 \equiv R$

Schema Decomposition

When does a decomposition lead to lossless-join?

- Given: R and set of F FDs
- A decomposition of R into R_1 and R_2 is **lossless-join** iff at least one the following FDs
  1. $[R_1] \cap [R_2] \rightarrow [R_1]$
  2. $[R_1] \cap [R_2] \rightarrow [R_2]$

  is in the closure F^+

Example

- Consider relation $R(A, B, C, D)$ and FD $F = B \rightarrow CD$

  Lossless-join
  - Decomposition into $R_1(A, B, C)$ and $R_2(B, D)$
  - $\{A, B, C\} \cap \{B, D\} = \{B\}$
  - $B \rightarrow BD \in F^+$

  Lossy-join
  - Decomposition into $R_1(A, B, C)$ and $R_2(D)$
- **Q: What about decomposition into $R_1(A, B, C)$ and $R_2(AC)$?**

Schema Decomposition

Preserving Functional Dependencies

Recall A relation R can be **decomposed** into instances

- R_1 and R_2
- where $[R_1] \cup [R_2] = [R]$
- $R_1 = \pi_{[R_1]}(R)$ has FD F_1
- $R_2 = \pi_{[R_2]}(R)$ has FD F_2
- and F_1 and F_2 are computed from some FD F

Dependency Preserving Decomposition

A schema decomposition is dependency preserving if by enforcing F_1 over R_1 and F_2 over R_2, we can enforce F over R

Schema Decomposition

Example (dependency preserving decomposition)

Employee(e_id, first_name, last_name, dob, retirement_date)

- e_id $\rightarrow$ first_name, last_name, dob
- dob $\rightarrow$ retirement_date

can be decomposed into

- R_1(e_id, first_name, last_name, dob)
- R_2(dob, retirement_date)

Schema Decomposition

Example

R_1	
A	**B**
a_1	b
a_2	b

R_2	
A	**C**
a_1	c
a_2	c

R		
A	**B**	**C**
a_1	b	c
a_2	b	c

Consider $R(A, B, C)$

- $A \rightarrow B$
- $BC \rightarrow A$

can be decomposed into

- $R_1(A, B)$
- $R_2(A, C)$
- But, $R_1 \bowtie R_2$ violates $BC \rightarrow A$

Normal Forms

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF
- ...

Boyce-Codd normal form (BCNF)

A relation R is in BCNF **iff** for every FD $X \rightarrow Y$, at least one of conditions hold

- $X \rightarrow Y$ is trivial FD
- X is a superkey

Examples

PAN	name	age	account_no
AY101	abc	30	9019
AY101	abc	30	8019
BX201	xyz	23	7218
CZ301	pqr	25	5454

FD: PAN $\rightarrow$ name, age

- key = {PAN, account_no}
- Relation is not in BCNF!

PAN	name	age
AY101	abc	30
BX201	xyz	23
CZ301	pqr	25

FD: PAN $\rightarrow$ name, age

- key = {PAN}
- Relation is in BCNF!

BCNF Decomposition

If R is not in BCNF due to $X \rightarrow Y$, decompose R into

1. R_1 such that $[R_1] = X^+$
2. R_2 such that $[R_2] = [R] \setminus X^+ \cup X$

repeat recursively on R_1 and R_2 until no BCNF violations

BCNF decomposition example

PAN	name	age	account_no
AY101	abc	30	9019
AY101	abc	30	8019
BX201	xyz	23	7218
CZ301	pqr	25	5454

FD: PAN $\rightarrow$ name, age

Decompose into

1. R_1 (PAN, name, age)
2. R_2 (PAN, account_no)

PAN	name	age
AY101	abc	30
BX201	xyz	23
CZ301	pqr	25

PAN	account_no
AY101	9019
AY101	8019
BX201	7218
CZ301	5454

Normal Forms

BCNF decomposition

- removes redundancy (of certain types)
- is **lossless-join**
- is **not always** dependency preserving

Recall previous examples

- $R(A, B, C)$ with $A \to B$ decomposes into:
  1. $R_1(A, B)$
  2. $R_2(A, C)$

  Always satisfies the lossless-join criteria

- $R(A, B, C)$ with $A \to B$ and $BC \to A$ decomposes into
  1. $R_1(A, B)$ with $A \to B$
  2. $R_2(A, C)$ with no FDs

Normal Forms

Consider Films(actor_id, name, film_id, title, genre, length)

- a_id $\rightarrow$ name
- film_id $\rightarrow$ title, genre, length

Q: Candidate key?

Q: Is Films is BCNF?

Third Normal Form (3NF)

A relation R is in 3NF if whenever $X \rightarrow Y$, one of the following is true

1. $X \rightarrow Y$ is trivial FD
2. X is superkey
3. $\forall a \in Y \setminus X$, a is prime attribute

Note: If R is in BCNF $\implies$ it is in 3NF

Normal Forms

Consider R(A,B,C) with

- AB $\rightarrow$ C
- C $\rightarrow$ A

Q: is R in 3NF or BCNF?

Normalization

- Not always good.
  - Performance loss, if R_1 and R_2 are always used as $R_1 \bowtie R_2$
  - Data warehousing – queries typically involve large number of joins
- But, crucial for a “good” database design
  1. Application
  2. ER model
  3. ER to Relational Schema
  4. **Normalization: refine schema**
  5. Populate