Normalization

Presented by Stéphane Bressan

Learning Objectives

- Understand the rationale (anomalies) and definition of the main normal forms based on functional dependencies (2NF, 3NF and BCNF)
- Be able to decompose (or synthesize) a schema into a dependency preserving BCNF or 3NF.

lossless &

Anomalies

- Redundant storage
- Update anomalies
- Insertion anomalies
- Deletion anomalies

Anomalies: Example

Assume that the position determines the salary:

position → salary

company

eNumber	firstName	lastName	address	department	Position	salary
1XU3	Dewi	Srijaya	12a Jln Lempeng	Toys	Clerk	2000
4W3E	Izabel	Leong	10 Outram Park	Sports	Trainee	1200
3XXE	John	Smith	107 Clementi Rd	Toys	Clerk	2000
5SD2	Axel	Bayer	55 Cuscaden Rd	Sports	Trainee	1200
6RG5	Winnie	Lee	10 West Coast Rd	Sports	Manager	2500
755Y	Sylvia	Tok	22 East Coast Lane	Toys	Manager	2600
2SD3	Eric	Wei	100 Jurong drive	Toys	Assistant manager	2200
?	?	?	?	?	Security guard	1500

key

Redundant storage

Update anomaly

Potential deletion anomaly

Insertion anomaly

Decomposition

- The decomposition of a relation scheme R is a set if relation scheme R_i such that:

$$\bigcup_i R_i = R$$

Do not lose any of the columns

Union of all columns in all tables (Fragments)

= Col.s in original table

Normalization: Example

employee

eNumber	firstName	lastName	address	department	Position
1XU3	Dewi	Srijaya	12a Jln Lempeng	Toys	Clerk
4W3E	Izabel	Leong	10 Outram Park	Sports	Trainee
3XXE	John	Smith	107 Clementi Rd	Toys	Clerk
5SD2	Axel	Bayer	55 Cuscaden Rd	Sports	Trainee
6RG5	Winnie	Lee	10 West Coast Rd	Sports	Manager
755Y	Sylvia	Tok	22 East Coast Lane	Toys	Manager
2SD3	Eric	Wei	100 Jurong drive	Toys	Assistant manager

key

salary

Position	salary
Clerk	2000
Trainee	1200
Manager	2500
Assistant manager	2200
Security guard	1500

key

- Redundant storage?
  - NO
- Update anomaly?
  - NO
- Deletion anomaly?
  - NO
- Insertion anomaly?
  - NO

Lossless Decomposition: Example

- The decomposition is **lossless** if we can recover the initial table:

SELECT first_name, last_name, address, department, T2.position, salary
FROM T2, T3
WHERE T2.position = T3.position

Stitch back all data

Keep all columns & preserve all data

Lossless Decomposition: Counter Example

Flight Number	Departure time	Arrival time	Origin	Destination
SG12	12h00	13h00+	SIN	CDG
TG414	15h50	16h30	SIN	JKT
TG415	12h00	14h20	BKK	SIN

- **Flight Number** is the key

Original Table

Lossless Decomposition: Counter Example

Flight Number	Departure time	Arrival time	Origin	Destination
SG12	12h00	13h00+	SIN	CDG
TG414	15h50	16h30	SIN	JKT
TG415	12h00	14h20	BKK	SIN

Decomposition

Flight Number	Departure time	Origin
SG12	12h00	SIN
TG414	15h50	SIN
TG415	12h00	BKK

Losing functional dependency

?

Departure time	Arrival time	Destination
12h00	13h00+	CDG
15h50	16h30	JKT
12h00	14h20	SIN

Dunno which departure time for what flight

Lossless, Dependency Preserving

- The decomposition is lossy
- And we lost functional dependencies:

{**Flight Number**} → {**Arrival time**, **Destination**}

Non dependency preserving

Decomposition: Example

- Consider the relation scheme
  {C,S,J,D,P,Q,V}
- with FDs
  {C} → {S,J,D,P,Q,V}
  {J,P} → {C}
  {S,D} → {P}

Contracts(contractid, supplierid, projectid, deptid, partid, qty, value)

Decomposition: Example

P there cos {S,D} → {P}

- Consider the decomposition into
  {C,S,J,D,Q,V}, {S,D,P}

On both sides, tool to bring everything tgt

- The decomposition is lossless
  - We can recover the initial relation thanks to S,D
- However the functional dependency {J,P} → {C} is lost across the two relations…
- This decomposition is not dependency preserving

Dependency Preserving Decomposition

- The decomposition of a relation scheme
  - R with FDs F
  - Is a set of relation schemes R_i with FDs F_i
- The decomposition is dependency preserving if and only if

$$(\cup_i F_i)+ = F+$$

Too Much Decomposition

- It might be tempting to decompose to the extreme
- Evaluation of queries may be inefficient since it will involve combining several relations

join causes inefficiency

Too Much Decomposition: Example

Flight Number	Departure time	Origin
SG12	12h00	SIN
TG414	15h50	SIN
TG415	12h00	BKK

Flight Number	Arrival time	Destination
SG12	13h00+	CDG
TG414	16h30	JKT
TG415	14h20	SIN

Can actually decompose into 4 tables

Looking for a "Good" Design

- The designer needs guidelines:
  - Normalization theory
    - Minimal redundancy and no anomalies
    - Lossless decompositions
    - Dependency preserving decompositions
- But ultimately the designer needs to look at the workload (the queries and their efficiency requirement)

Second Normal Form (2NF)

- R is a relation schema, with the set F of FDs
- R is in 2NF if and only if
  - For all $X: X \subset R$
  - and, for all $A \in R$
  - such that there exists a FD: $X \rightarrow \{A\}$ in F+
- Then
  - $A \in X$ (*the FD is trivial*), or
  - X is not a proper subset of a candidate key for R, or
  - A is part of some candidate key for R
    (A is called a prime attribute)

Not a proper subset

Candidate key : minimal super key

Second Normal Form (2NF)

- Consider the relation scheme {A,B,C,D} with the FDs:
  - $\{A,B\} \rightarrow \{C,D\}$ and
  - $\{A\} \rightarrow \{D\}$
- {A,B} is a primary key (it is not a proper subset)
- {A} is a proper subset of a primary key
- {D} is not part of some candidate key
- This scheme is not in 2NF

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

Violation

{D} is not a prime attribute

Second Normal Form (2NF)

Decomposition ✓
lossless
A on both sides ✓
No dependency loss ✓

- E.g., Relation scheme {A,D} with FDs
  - $\{A\} \to \{D\}$ A is a candidate key
  - {A} is the primary key (i.e., not proper subset)
  - The scheme is in 2NF
- E.g., Relation scheme {A,B,C} with FDs
  - $\{A,B\} \to \{C\}$ {A,B} candidate key
  - {A,B} is the primary key
  - The scheme is in 2NF

Third Normal Form (3NF)

- R is a relation schema, with the set F of FDs
- R is in 3NF if and only if
  - For all $X: X \subset R$
  - And, for all $A \in R$
  - such that there exists a FD: $X \to \{A\}$ in F+
- Then
  - $A \in X$ (trivial FD), or
  - X is a superkey for R, or
  - A is part of some candidate key for R

3NF is always in 2NF
3NF stricter cos of 2nd cond.

Third Normal Form (3NF)

- Rationale:
  - If X is not a superkey for R then this dependency is partial and may cause redundancy
  - If A is not part of some candidate key for R then there is a transitive dependency:
    candidate key $\to X \to A$
  - Insertion/deletion/update anomalies

Boyce-Codd Normal Form (BCNF)

- R is a relation schema, with the set F of FDs
- R is in BCNF if and only if
  - For all $X: X \subset R$
  - And, for all $A \in R$
  - such that there exists a FD: $X \to \{A\}$ in F+
- Then
  - $A \in X$, or
  - X is a superkey for R

No last condition

Stricter, removing a possibility
BCNF always in 3NF

BCNF ⊂ 3NF ⊂ 2NF ⊂ 1NF *

- 2NF:
  - $A \in X$, or
  - X is not a proper subset of a candidate key for R, or
  - A is part of some candidate key for R
- 3NF:
  - $A \in X$, or
  - X is a superkey for R, or
  - A is part of some candidate key for R
- BCNF:
  - $A \in X$, or
  - X is a superkey for R → No last condition

All LHS must give ALL columns in F+ !!

Decomposition into BCNF

Let S be the initial set of schemes with FDs F
Until all relation schemes in S are in BCNF
  for each R in S
    if FD $X \to Y$ in F+ violates BCNF for R
      ($X \to R$ not hold, not superkey
      $X \cap Y = \emptyset$, not trivial) → Violate conditions
    then
      use $X \to X+$
      let S be $(S - \{R\}) \cup \{(R-X+) \cup X,\ X+\}$
  endfor
enduntil

Decomposition into BCNF

{c,d} → {e,f} not Trivial
{c,d} not superkey

$X \rightarrow Y$

X={c,d}

X+={c,d,e,f}

a	b	c	d	e	f

Find one culprit & break it into 2

a	b	c	d

c	d	e	f

Decomposition into BCNF

- The different possible orders* in which we may consider the dependencies violating BCNF in the algorithm application may lead to different decompositions

orders in which we consider the constraints violating the BCNF condition

There may be problems which may lose functional dependencies

May get sth non-dependency preserving

Remark: Projecting FDs

- If S is a fragment after decomposition of a relation R with FDs F
- The set of projected FDs on S is the set G of FDs
  - If $X \rightarrow Y$ is in G
    - Then X and Y are subsets of S
    - $X \rightarrow Y$ is in F+
  - If $X \rightarrow Y$ is in F+ and X and Y are subsets of S
    - Then $X \rightarrow Y$ is in G+

Decomposition into BCNF

- Let us consider the relation scheme R={A,B,C,D,E} and the FDs:
  - {A} → {B} → Non trivial, non-superkey
  - {A} → {E}
  - {C} → {D}

{A,C} → {A,B,C,D,E}

- Candidate key: $\boxed{\text{\{A, C\}}}$

BCNF
Minimal Cover
↳ {A} → {B,E}
{C} → {D}

* 3NF algo: Calc minimal cover.
↳ This case, already a minimal cover.

Decomp:
R1: (A,B,E)
R2: (C,D)
R3: (A,C)

Decomposition: R1 = (A,B)
R2 = (A,E)
R3 = (C,D)
R4 = (A,C)

If 3NF synthesis, usually can find BCNF (NOT ALWAYS)

Decomposition into BCNF

- R is not in BCNF
- Because (for instance):
  - {A} → {B} holds
  - It is not trivial
  - {A} is not a superkey

Decomposition into BCNF

- Pick {A} → {B} for decomposition
- Expand into {A} → {B,E}
- {A,B,C,D,E} becomes
  - {A,C,D} and {A,B,E}
- With FDs: (we need projected FDs)
  - {A} → {B}, on {A,B,E} ✓
  - {A} → {E}, on {A,B,E} ✓ } {A} is superkey BCNF
  - {C} → {D}, on {A,C,D} ✓ → Violation

Decomposition into BCNF

- Pick {C} → {D} for decomposition
- Expand into {C} → {D}
- {A,C,D} and {A,B,E} becomes
  - {A,C}, {C,D} and {A,B,E}
- With FDs: (*we need projected FDs)*

  {A} → {B}, *on {A,B,E}*

  {A} → {E}, *on {A,B,E}*

  {C} → {D}, *on {C,D}*

Decomposition into BCNF

- Finally the BCNF decomposition of R={A,B,C,D,E} with the FDs:

  {A} → {B},

  {A} → {E},

  {C} → {D}

- is: {A,C}, {C,D} and {A,B,E}

↳ Dependency preserving

Decomposition into BCNF

- BCNF decomposition may not be dependency preserving
- Example: {A,B,C} with FDs

  {A,B} → {C}

  {C} → {A}

  Decompose to CA, BC.

- The first FD is not preserved.

Divide & Conquer Algo

Decomposition into 3NF

Let S be the initial set of relation scheme R with FDs F

for each X⊂R such that X → {A_i} is in the minimal cover of F

  if no scheme contains X $\cup_i$ {A_i}

  then create relation with scheme X $\cup_i$ {A_i}

endfor

if no scheme contains a key for R

then create a relation with scheme with key for R

Dependency preserving

- The 3NF synthesis algorithm always finds a loosless dependency preserving decomposition

Credits

The content of this lecture is based on chapter 8 of the book "Introduction to database Systems"
By
S. Bressan and B. Catania, McGraw Hill publisher

Clipart and media are licensed from Microsoft Office Online Clipart and Media