CIS 560 – Database System Concepts

Lecture 11

# Decomposition Goals & Transactions in SQL

September 20, 2013

Credits for slides: Suciu, Chang, Ullman.



## Announcements

- HW4 will be posted tonight, due September 27th
- Exam 1 – October 7th (sample exam posted on KSOL)
- Project information posted on KSOL
- Project proposals
  - Information about team members and English description due September 24th
  - E/R diagram and relational schema due October 4th
- Proposal presentations October 9th and 11th

## Review

- Closure of a set of attributes?
- Key/superkey?
- What is a "bad" functional dependency?
- What does it mean for a relation to be in BCNF?

## BCNF Decomposition Algorithm

BCNF_Decompose(R)

find X s.t.: X ≠$X^+$ ≠ [all attributes]

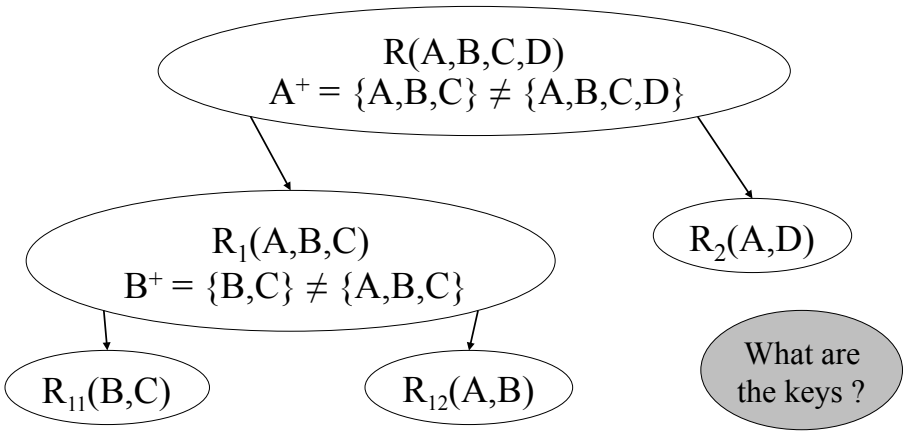
**if** (not found) **then** "R is in BCNF"

**let** Y = $X^+$ - X
**let** Z = [all attributes] - $X^+$
decompose R into R1(X ∪ Y) and R2(X ∪ Z)
continue to decompose recursively R1 and R2

R(A,B,C,D)

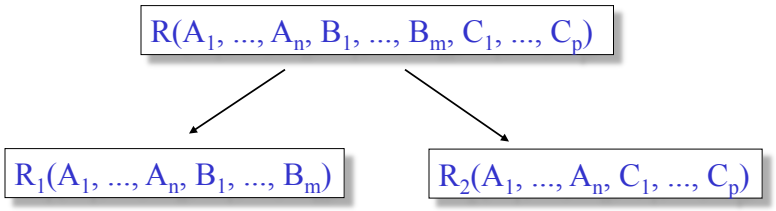
A → B
B → C

## Example




What happens if in R we first pick $B^+$? Or $\{A,B\}^+$?

## Decompositions in General




$R_1$ = projection of R on $A_1, ..., A_n, B_1, ..., B_m$
$R_2$ = projection of R on $A_1, ..., A_n, C_1, ..., C_p$

# Correct Decomposition

Sometimes it is correct:

| Name | Price | Category |
|---|---|---|
| Gizmo | 19.99 | Gadget |
| OneClick | 24.99 | Camera |
| Gizmo | 19.99 | Camera |

| Name | Price |
|---|---|
| Gizmo | 19.99 |
| OneClick | 24.99 |
| ~~Gizmo~~ | ~~19.99~~ |

| Name | Category |
|---|---|
| Gizmo | Gadget |
| OneClick | Camera |
| Gizmo | Camera |

Lossless decomposition

# Incorrect Decomposition

Sometimes it is not:

| Name | Price | Category |
|---|---|---|
| Gizmo | 19.99 | Gadget |
| OneClick | 24.99 | Camera |
| Gizmo | 19.99 | Camera |

What's incorrect?

| Name | Category |
|---|---|
| Gizmo | Gadget |
| OneClick | Camera |
| Gizmo | Camera |

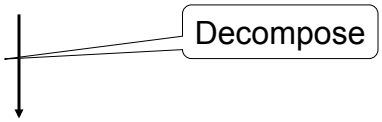
| Price | Category |
|---|---|
| 19.99 | Gadget |
| 24.99 | Camera |
| 19.99 | Camera |

Lossy decomposition

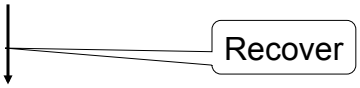
# Lossless Decompositions

A decomposition is *lossless* if we can recover:

R(A,B,C)



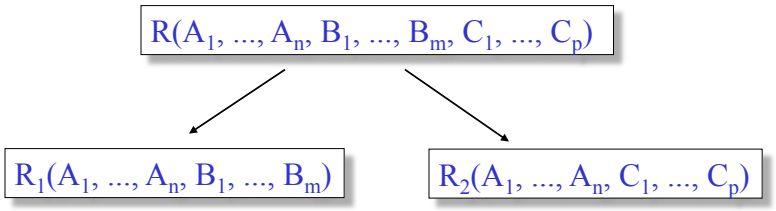

{ R1(A,B) , R2(A,C) }




R'(A,B,C) = R(A,B,C)

R' is in general larger than R. Must ensure R' = R

# Decompositions in General




If $A_1, ..., A_n \rightarrow B_1, ..., B_m$
Then the decomposition is lossless

Note: don't need $A_1, ..., A_n \rightarrow C_1, ..., C_p$

BCNF decomposition is always lossless. WHY ?

# Decomposition Based on BCNF is Necessarily Lossless

R(A, B, C), $A \rightarrow B$

BCNF: R1(A,B), R2(A,C)

Some tuple (a,b,c) in R
decomposes into (a,b) in R1
and (a,c) in R2

Recover tuple (a,b,c) in R?

# Decomposition Based on BCNF is Necessarily Lossless

R(A, B, C), $A \rightarrow B$

BCNF: R1(A,B), R2(A,C)

Some tuple (a,b,c) in R — (a,b',c') also in R
decomposes into (a,b) in R1 — (a,b') also in R1
and (a,c) in R2 — (a,c') also in R2

Recover tuples in R: (a,b,c), (a,b,c'), (a,b',c), (a,b',c') also in R?

Can (a,b',c) be a bogus tuple?

# General Decomposition Goals

- Elimination of anomalies
- Recoverability of information
  - Can we get the original relation back?
- Preservation of dependencies
  - Want to enforce FDs without performing joins

BCNF

3NF

Sometimes cannot decompose into BCNF without losing ability to check some FDs

# A Problem with BCNF

| Unit | Company | Product |
|---|---|---|

FD's: Unit → Company; Company, Product → Unit

So, there is a BCNF violation, and we decompose.

# A Problem with BCNF

| Unit | Company | Product |
|---|---|---|

FD's: Unit → Company; Company, Product → Unit

So, there is a BCNF violation, and we decompose.

| Unit | Company |
|---|---|

Unit → Company

| Unit | Product |
|---|---|

No FDs

In BCNF we lose the FD: Company, Product → Unit

# So What's the Problem?

| Unit | Company |
|---|---|
| Galaga99 | UI |
| Bingo | UI |

| Unit | Product |
|---|---|
| Galaga99 | databases |
| Bingo | databases |

No problem so far. All *local* FD's are satisfied.

Let's put all the data back into a single table again:

| Unit | Company | Product |
|---|---|---|
| Galaga99 | UI | databases |
| Bingo | UI | databases |

**Violates the dependency: company, product -> unit!**

## Preserving FDs

- What if, when a relation is decomposed, the X of an X→Y ends up only in one of the new relations and the Y ends up only in another?
- Such a decomposition is not "dependency-preserving."

## Solution: 3rd Normal Form (3NF)

A relation R is in 3rd normal form if :

Whenever there is a nontrivial dependency
$A_1, A_2, ..., A_n \rightarrow B$ for R ,
then $\{A_1, A_2, ..., A_n\}$ is a super-key for R,
or B is part of a key.

## 3NF vs. BCNF

- R is in BCNF if whenever X→A holds, then X is a superkey.
- Slightly stronger than 3NF.
- Example: R(A,B,C) with {A,B}→C, C→A
  - 3NF but not BCNF

Trade-offs

BCNF = no anomalies, but may lose some FDs
3NF = keeps all FDs, but may have some anomalies

- Everyday relational DBs
  - aim for BCNF, settle for 3NF

## Caveat

- Normalization is not the be-all and end-all of DB design.
- Example: suppose attributes A and B are always used together, but normalization theory says they should be in different tables.
  - decomposition might produce unacceptable performance loss (extra disk reads)

## Where We Are

What we have already learned so far

- Relational model of data
- Data manipulation language: SQL
- Views and constraints
- Database design (E/R diagrams & normalization)

But what if I want to update my data?

- Need to worry about transactions in SQL

21

## Transactions

- The problem: An application must perform *several* writes and reads to the database, as a unit.
  - Example: Two people attempt to book the last seat on a flight.
- Solution: multiple actions of the application are bundled into one unit called *Transaction*.
  - Transactions guarantee certain properties to hold that prevent problems.

22

## Turing Awards to Database Researchers

- Charles Bachman 1973 for CODASYL
- Edgar Codd 1981 for relational databases
- Jim Gray 1998 for transactions

23

## The World Without Transactions

- Just write applications that talk to databases
- Rely on operating systems for scheduling, and for concurrency control
- What can go wrong?
  - Three famous anomalies
  - Other anomalies are possible (but not so famous)

24