CSE 180, Database Systems I
Shel Finkelstein, UC Santa Cruz
Lecture 10

Chapter 7: Relational DB Design Theory (Functional Dependencies, etc.)
Sections 7.1 - 7.5, 7.8
Part 2

Database System Concepts, 7th Ed.

Important Notices

CSE 180 Final Exam is on **Thursday March 19, 4:00 – 7:00pm**, in our usual classroom.

- **No** early/late Finals. **No** make-up Finals. **No** devices.
- Includes a Multiple Choice Section and a Longer Answers Section.
  - Bring Red Scantron sheets (ParSCORE form number f-1712) sold at Bookstore, and #2 pencils for Multiple Choice Section. (Ink and #3 pencils don't work.)
- Covers entire quarter, with slightly greater emphasis on second half of quarter.
- You may bring in one double-sided 8.5 by 11 sheet, with anything that you can read unassisted printed or written on both sides of the paper.
  - **No sharing** of sheets is permitted.
- Seating pattern for Final will be same as for Midterm.
- You **must** show your UCSC ID at end of Final.
- Practice Final from Fall 2019 (2 Sections) was posted on Piazza under Resources→Exams on Sunday, March 1.
  - Some material on Practice Final hasn't been covered yet in our class.
  - Solution will be posted on Monday, March 9… but take it yourself first, rather than just reading the solution.
- See Piazza notice @194 about Final, which has reminder about Course Grading.
  - After the class ends, your course score will be determined by your scores on Gradiance, Lab Assignments and Exams.
  - **You won't be able to do any additional work afterwards to improve your grade.**

Important Notices

- Lab4 assignment has been posted on Piazza under Resources→Lab4.
  See Piazza announcement about Lab4.
  Please read it soon!
  - Lab4 is due **Sunday, March 8**, by 11:59pm (2 week assignment).
    - Load Data for Lab4 has been posted; you'll need to use that Load Data to complete Lab4.
  - Section 6 of the Lab4 pdf (Testing) was corrected on Saturday, Feb 29 to say that reduceSomeTicketPricesFunction should return **total of all the ticketPrice reductions made**, agreeing with description in Section 5 of the Lab4 pdf.
  - Your solution should be submitted via Canvas as a zip file.
    - Late Lab Assignments will not be accepted.
    - Be sure that you submit the correct file!
  - Lab4 deals with material (JDBC, Stored Functions) from previous Lecture, Lecture 8.
    - Additional Piazza announcements describe some differences between PostgreSQL Stored Functions (which you'll use for Lab4) and the PSM standard.
  - Lab4 will be discussed at Lab Sections.

Important Notices

- Midterm been discussed in class and returned in class.
  - Deadline for questions about Midterm grade is past; it was this **Wednesday, March 4**.
- Gradiance #5 is due **Saturday, March 14** by 11:59pm.
- Lab3 grades have been unmuted.
  - Deadline for questions about Lab3 is **Thursday, March 12**.
- Attend Lectures, Lab Sections, Office Hours and LSS Tutoring.
  - See Piazza notices about LSS Tutoring with Jeshwanth Bheemanpally.
- Winter 2020 Student Experience of Teaching Surveys - SETs are now open.
  - SETs close on Sunday March 15 at 11:59pm.
  - Instructors **are not** able to identify individual responses.
  - Constructive responses help improve future courses.

Normal Forms

Given a relation schema, we want to understand whether it is a good design or a bad design.

- Intuitively, a good design is one that does not store data redundantly, and does not lead to anomalies.

If we know that rank determines salary_scale, which is a better design?

Employees(eid, name, addr, rank, salary_scale)

OR

Employees2(eid, name, addr, rank)
Salary_Table(rank, salary_scale)

Remember that sometimes database designers **may choose** to live with redundancy in order to improve query performance. But then they'll have to cope with anomalies, which can be difficult.

First Normal Form (1NF)

- A relation schema is in *first normal form (1NF)* if the type of every attribute is atomic.
- Very basic requirement of the relational data model.
  - Not based on FDs.
  - Every other Normal Form we'll discuss assumes 1NF, although definition doesn't mention that.

Example:

R(ssn: char(9), name: string, age: int)

- All our examples so far have been in 1NF.

Example of a non-first normal form relation:

R(ssn: char(9), name: Record[firstname: string, lastname: string], age: int, children: Set(string))

Second Normal Form (2NF)

- Not particularly important
  - We won't discuss this.
  - (Neither does the textbook.)

Keeping FDs Simple

- We proved in previous Lecture that:

  If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

  and: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

- So from now on, we'll assume that right side of every FD is a single attribute.
  - For example, instead of writing $AC \rightarrow BDE$, we will write:

    $AC \rightarrow B$, $AC \rightarrow D$, $AC \rightarrow E$

Boyce-Codd Normal Form (BCNF)

- Let r(R) be a relation schema, $\mathcal{F}$ be a set of FDs that holds for r, with A an attribute in R, and X as a subset of the attributes R.

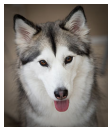
- r(R) is in *Boyce-Codd Normal Form (BCNF)* if
  - For every FD X → A in $\mathcal{F}$, at least one of following is true:
    - X → A is a trivial FD (i.e., A ∈ X) or,
    - X is a superkey.

- BNCF is desirable for avoiding redundancy.
  - Recall our Employees2/Salary_Table example.

Boyce-Codd Normal Form (BCNF)

- Let r(R) be a relation schema, $\mathcal{F}$ be a set of FDs that holds for r, with A an attribute in R, and X as a subset of the attributes R.

- r(R) is in *Boyce-Codd Normal Form (BCNF)* if
  - For every FD X → A in $\mathcal{F}$, at least one of following is true:
    - X → A is a trivial FD (i.e., A ∈ X) or,
    - X is a superkey.



- BNCF is desirable for avoiding redundancy.
  - Recall our Employees2/Salary_Table example.

Is this Relation in BCNF?

A	B	C
a1	b1	c1
a1	b2	c1

- The only functional dependency given is A → C.
- (to fill in)

Is this Relation in BCNF?

A	B	C
a1	b1	c1
a1	b2	c1

- The relation is not in BCNF because:

  $A \rightarrow C$ is not a trivial FD and A is not a superkey.

- Given that $A \rightarrow C$, we can infer that C value of second tuple must also be c1.
- But note that c1 is redundantly stored (twice).

Third Normal Form (3NF)

- Let r(R) be a relation schema, $\mathcal{F}$ be a set of FDs that holds for r, with A an attribute in R, and X as a subset of the attributes R.

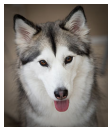
- r(R) is in *third normal form (3NF)* if
  - For every FD X → A in $\mathcal{F}$, at least one of following is true:
    - X → A is a trivial FD (i.e., A ∈ X), or
    - X is a superkey, or
    - A is part of some key of r(R).

- Note that red condition says that A is part of some key for r, not some superkey for r.

Third Normal Form (3NF)

- Let r(R) be a relation schema, $\mathcal{F}$ be a set of FDs that holds for r, with A an attribute in R, and X as a subset of the attributes R.

- r(R) is in *third normal form (3NF)* if
  - For every FD X → A in $\mathcal{F}$, at least one of following is true:
    - X → A is a trivial FD (i.e., A ∈ X), or
    - X is a superkey, or
    - A is part of some key of r(R).


- Note that red condition says that A is part of some key for r, not some superkey for r.

BCNF/3NF Example 1

Consider r(A, B, C, D)

with FD: A → D

Note: Trivial FDs and FDs based on Primary Keys are implicit, and are often not listed, because 3NF obviously covers all of those.

- Is it in BCNF?
- Is it in 3NF?

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

BCNF/3NF Example 2

Now consider r(A, B, C, D)

with FD's: A → D, and D → A.

- Note that BCD is also a key for r.

- Is it in BCNF?
- Is it in 3NF?

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

- There is still redundancy in R, even though it is in 3NF!

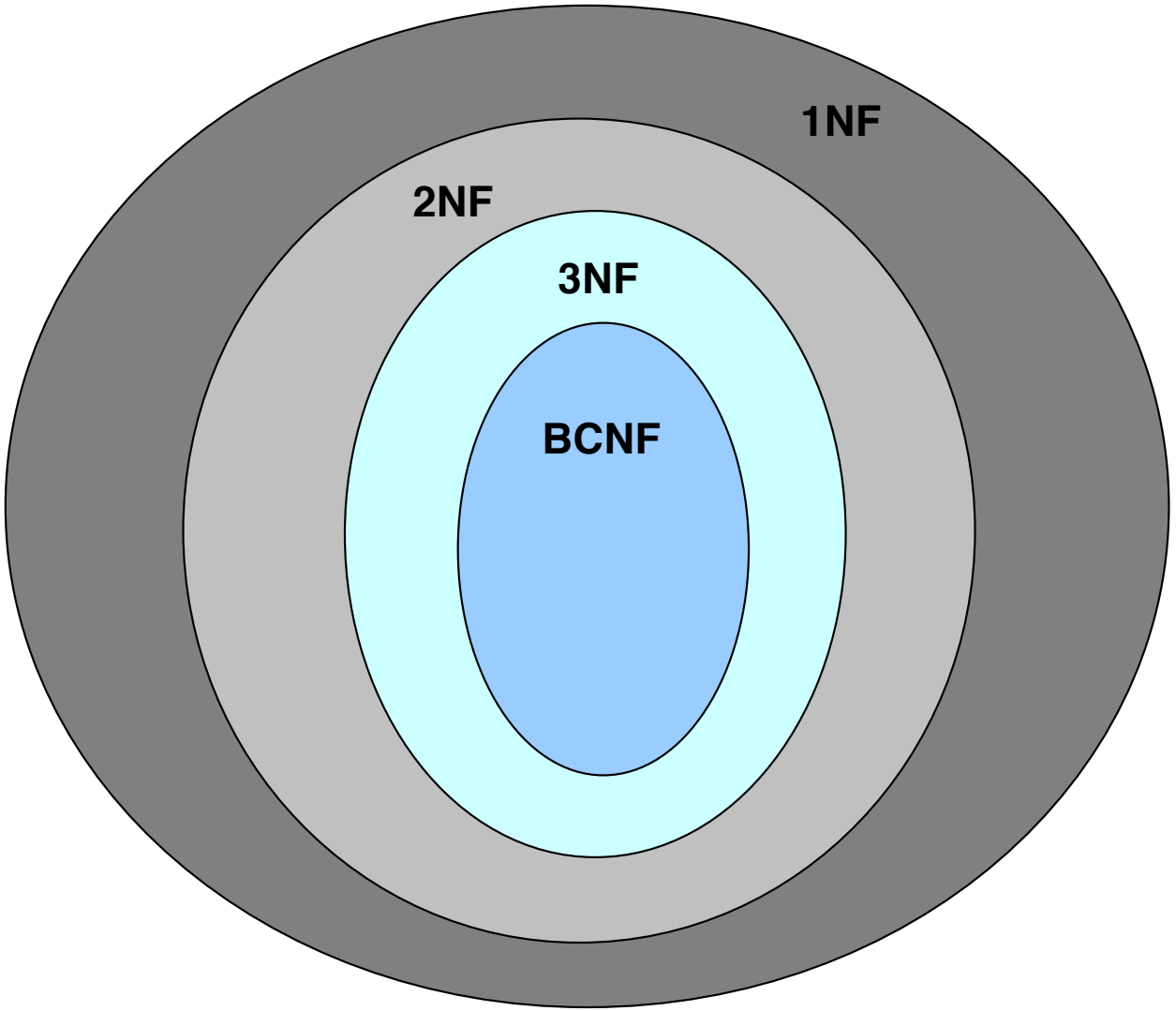
BCNF and 3NF

- By definition, any BCNF relation must also be a 3NF relation.
- Definition says:

  … if at least one of the following holds for each FD $X \rightarrow A$:

  - $X \rightarrow A$ is a trivial dependency (i.e., $A \in X$). BCNF, 3NF
  - X is a superkey. BCNF, 3NF
  - A is part of some key of r(R). 3NF
- However, a 3NF relation is not always in BCNF.
  - Example 2 is an example of a 3NF relation that is **not** in BCNF.

Relationships Among Normal Forms – The Big Picture



BCNF/3NF Example 3

Company_Info(<u>emp</u>, dept, manager)

dept → manager

Is it in BCNF?

Is it in 3NF?

BCNF/3NF Example 4

r(city, street, zip)

city, street → zip

zip → city

The above FDs are true of most post office policies. Note that a city may have multiple zips, but a zip must be in a single city.

Is it in BCNF?

Is it in 3NF?

- Despite 3NF, there can be Redundancy: The association of a zip with a city could appear in multiple records of r.
- **So although r is in 3NF, there can be Anomalies:**
  - zip → city. So if the city is changed in one (city, street, zip) record, but is not changed for another (city, street, zip) record that has the same zip, that's an anomaly.

BCNF/3NF Example 5

Customers(ssn, name, address)

ssn → name

ssn → address

Is it in BCNF?

Is it in 3NF?

Algorithm for Testing Whether a Relation is in BCNF using Attribute Closure

Given r(R) and $\mathcal{F}$, determine whether r is in BCNF.

- For each FD X→Y ∈ $\mathcal{F}$ such that Y ⊈ X (i.e., the FD is non-trivial), compute X^+.
  - If every such X is a superkey (i.e., $X^+ = R$), then r(R) is in BCNF.
  - If there is a set X of attributes such that $X^+ \neq R$, then r(R) is not in BCNF.

Example Reminder: BCNF Testing

- CompanyInfo(emp, dept, manager)
  - emp → dept, dept → manager
  - $dept^+ \neq$ attr(CompanyInfo).
  - Hence CompanyInfo **is not** in BCNF.

- Customers(ssn, name, address)
  - ssn → name
  - ssn → address
  - ssn^+ = attr(Customers)
  - Hence Customers **is** in BCNF.

- R(city, street, zip)
  - city, street → zip
  - zip → city
  - $zip^+ \neq$ attr(R)
  - Hence R **is not** in BCNF.

Binary Relations and BCNF

Is r(A,B) in BCNF?

- We haven't told you the Functional Dependencies on r(A,B)!
  - Consider all the possibilities,
- **Fact**: Any binary relation schema is in BCNF.
  - Why?

"Improving" a non-BCNF Relation

How can we "improve" a relation r that is not in BCNF?

- **Approach**: Decompose ("break up") relation r into smaller relations, so that each smaller relation is in BCNF.
- We did this when we decomposed Employees, separating out Salary_Table because of the FD: rank → salary_scale

  Employees2(<u>eid</u>, name, addr, rank)

  Salary_Table(<u>rank</u>, salary_scale)

Decomposition of a Relation

A *decomposition of a relation r(R)* is defined by sets of attributes X_1, …, X_k (which don't have to be disjoint) such that:

1. Each $X_i \subseteq R$
2. $X_1 \cup X_2 \cup \ldots \cup X_k = R$

For a decomposition, we will write π_{Xi} (r) as r_i

Examples:

- CompanyInfo(emp, dept, manager)
  - $CompanyInfo_1$(emp, dept), $CompanyInfo_2$(dept, manager)
- r(A,B,C,D,E,F,G)
  - r_1(A,C), r_2(A,B,C,D), r_3(C,D,E,F,G)

Goals for Redesigning Schema Using A Decomposition

1. The decomposition Eliminates Anomalies.
2. The decomposition doesn't lead to any "extra data" (for any instance of r) when the r_i's are re-joined back together.
   - Such Decompositions are called Lossless Decompositions.
   - Why must the Natural Join of all the r_i's always give at least all the data that was in r?
3. Dependency Preservation: (not required for Final)
   - The FD's on r_i are the FD's in $\mathcal{F}^+$ that mention only its attributes.
   - The decomposition is Dependency-Preserving if when the r_i's are re-joined back together, the FD's that were on the r_i's imply all of the original FD's in $\mathcal{F}$.

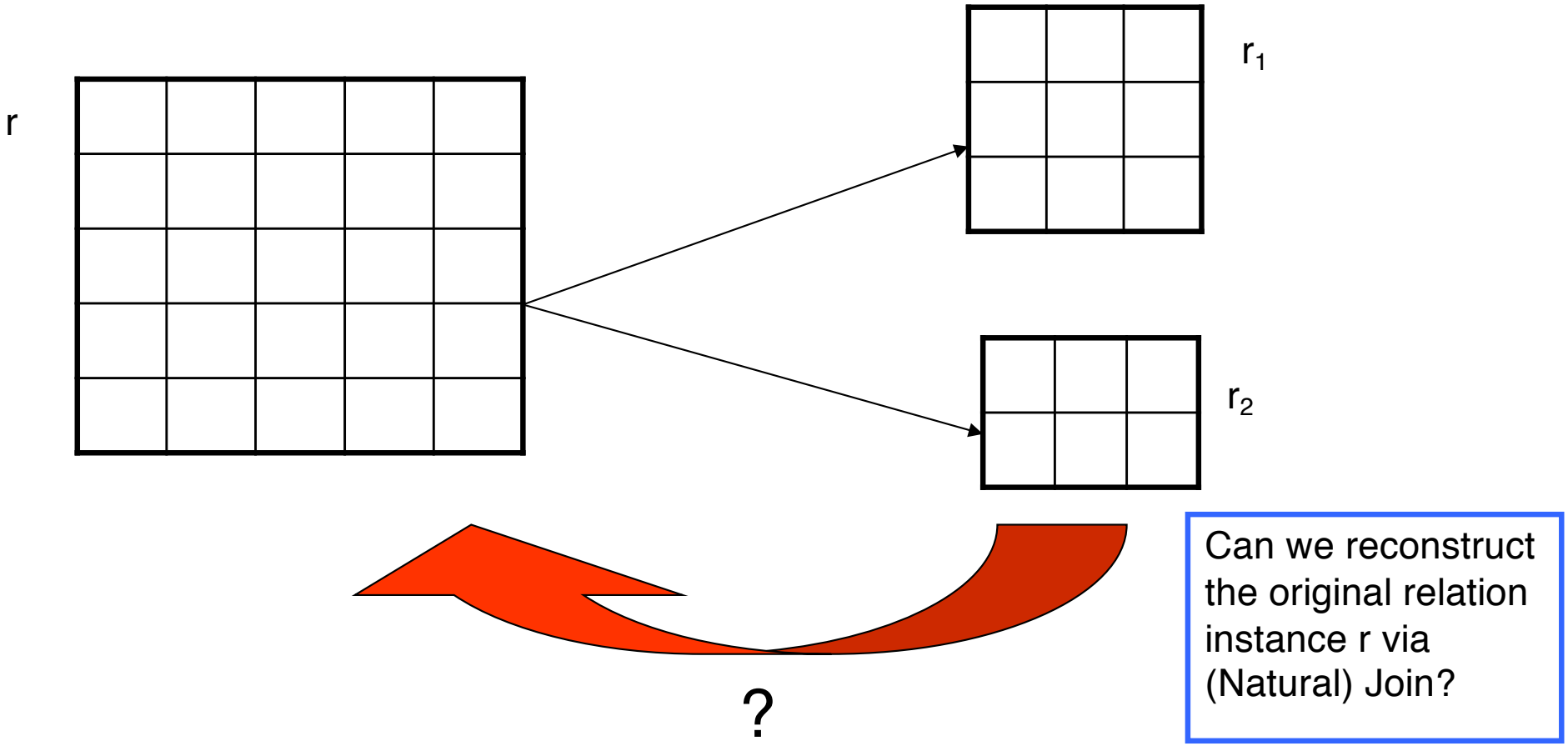
Are All BCNF Decompositions Good?

- Is it always possible to decompose r so that each smaller relation is in BCNF?
  - YES
  - One strategy: decompose r into a set of relation schemas $r_1, \ldots, r_k$ such that each r_i is a binary relation schema.

- Are all BCNF decompositions good?
  - NO

Decomposing a Relation

- Suppose that we have decomposed r into r_1 and r_2.
  - Given an Can we get back the original instance r by (Natural) Joining
    r_1 and r_2?instance of r, we decompose it into r_1 and r_2.



Lossless Decomposition

In general, can we obtain r by Natural Joining r_1 with r_2 … with r_k?

- That is, must it always true for any instance r, that:
  $r = r_1 \bowtie r_2 \bowtie ... \bowtie r_k$ (Natural Join) ?

More precisely:

- Let r(R) be a relation schema and $\mathcal{F}$ be a set of FDs over r.
- A decomposition of r into *k* schemas, with attribute sets X_1 , …, X_k, is a *Lossless Decomposition with respect to* $\mathcal{F}$ if:

For <u>every</u> instance of r that satisfies $\mathcal{F}$:

$$r = \pi_{X1}(r) \bowtie \ \ldots \bowtie \pi_{Xk}(r)$$
$$= r_1 \bowtie r_2 \bowtie ... \bowtie r_k$$

Lossless Decomposition Example 1

- Let r(A,B,C) be a relation schema with no functional dependencies
- Is the decomposition of r into schemas r1(A,B) and r2(B,C) a Lossless Decomposition?

Instance rx

A	B	C
a1	b1	c1
a1	b1	c2
a1	b2	c3

$\pi_{A, B}(rx)$

A	B
a1	b1
a1	b2

$\pi_{B,C}(rx)$

B	C
b1	c1
b1	c2
b2	c3

$\pi_{A, B}(rx) \bowtie \pi_{B,C}(rx)$

A	B	C
a1	b1	c1
a1	b1	c2
a1	b2	c3

Lossless Decomposition Example 2

Instance ry

A	B	C
a1	b1	c1
a2	b1	c2

$\pi_{A, B}(ry)$

A	B
a1	b1
a2	b1

$\pi_{B,C}(ry)$

B	C
b1	c1
b1	c2

$\pi_{A, B}(ry)$ 1 $\pi_{B,C}(ry)$

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

Lossy!

- By projecting on r1(A,B) and r2(B,C), some information may be lost (sometimes).
- We no longer know that (a1,b1,c2) did not exist in the original relation!
- Hence r_1(A,B) and r_2(B,C) is not a Lossless Decomposition of r.

FD's and Lossless Decompositions

- Let r(A,B,C) be a relation schema
- Is the decomposition of R into schemas r_1(A,B) and r_2(B,C) a Lossless Decomposition if we know B → C?
  - ry is not a legal instance, since it does not satisfy B → C.
  - rx, however, is a legal instance with respect to B → C.
- But that doesn't prove that r_1(A,B) and r_2(B,C) is a Lossless Decomposition in the presence of the FD B → C.
  - Is it Lossless?
  - Yes; see later slides in this lecture!!

Lossless Decomposition Example 3

CompanyInfo(emp, salary, dept, manager)

emp → salary, dept, manager

dept → manager

- CompanyInfo is not in BCNF because of dept → manager.
- Let's decompose into CompanyInfo_1(emp, salary) and CompanyInfo_2(dept, manager).

Instance of CompanyInfo:

(Bolt, 85K, Math, Tromb)

(Montgomery, 90K, Math, Tromb)

(Brandt, 88K, CS, Pohl)

Lossless Decomposition Example 3 (continued)

- Decompose instance of CompanyInfo(emp, salary, dept, manager)
  - $CompanyInfo_1$(emp, salary)

    (Bolt, 85K)

    (Montgomery, 90K)

    (Brandt, 88K)
  - $CompanyInfo_2$(dept, manager)

    (Math, Tromb)

    (CS, Pohl)

- $CompanyInfo_1 \bowtie CompanyInfo_2 = CompanyInfo_1 \times CompanyInfo_2$ has 6 tuples in it.

- But our instance of CompanyInfo had only 3 tuples in it!
  - Hence the decomposition of CompanyInfo(emp, salary, dept, manager) into $CompanyInfo_1$(emp, salary) and $CompanyInfo_2$(dept, manager) is not a Lossless Decomposition.

A Necessary and Sufficient Condition for Lossless Decomposition

- We would like our decompositions to be Lossless, and we'd like to be able to decide when a decomposition is Lossless.

Let r(R) be a relation and $\mathcal{F}$ be set of FDs that hold over r.

Fact: A decomposition of r(R) into two relation schemas $r_1(R_1)$ and $r_2(R_2)$ is Lossless if and only if $\mathcal{F}^+$ contains either:

1. $R_1 \cap R_2 \rightarrow R_1$, or
2. $R_1 \cap R_2 \rightarrow R_2$

That is, the intersection of the attributes R_1 and R_2 is a superkey of either r_1 or r_2

Testing Whether a Decomposition is a Lossless Decomposition

Fact: A decomposition of r(R) into two relation schemas $r_1(R_1)$ and $r_2(R_2)$ is Lossless if and only if $\mathcal{F}^+$ contains either:

1. $R_1 \cap R_2 \rightarrow R_1$, or
2. $R_1 \cap R_2 \rightarrow R_2$

That is, the intersection of the attributes R_1 and R_2 is a superkey of either r_1 or r_2

- This **Fact** works only for decompositions into **two** relations.
  - And note that it's not the definition of Lossless Decomposition!
- "The Chase" (not described in our textbook) is a procedural algorithm for checking whether any decomposition is a Lossless Decomposition.
  - We won't discuss "The Chase" in this class.

Two More Lossless Examples

- Decompose r(A,B,C) into r_1(A,B) and r_2(B,C), with $\mathcal{F}$ being the empty set.
  - Since B → AB and B → BC are not in $\mathcal{F}^+$, this decomposition is not a Lossless Decomposition.

- CompanyInfo(emp, salary, dept, manager)

  emp → salary, dept, manager

  dept → manager
  - CompanyInfo is not in BCNF.
  - Decompose into CompanyInfo_1(emp, salary) and CompanyInfo_2(dept, manager)
  - Since FDs {} → emp,salary and {} → dept, manager are not in $\mathcal{F}^+$, this decomposition is not a Lossless Decomposition.

A Final Lossless Example

Employees(eid, name, addr, rank, salary_scale)

with FD: rank → salary_scale

Decomposition:

Employees2(eid, name, addr, rank)

Salary_Table(rank, salary_scale)

Employees2 ∩ Salary_Table = {rank}

rank → attr(Salary_Table).

Therefore, the decomposition is Lossless.

Functional-Dependency Theory Roadmap

- We now consider the formal theory that tells us which functional dependencies are implied logically by a given set of functional dependencies.
- We'll define Normal Forms that may help us avoid Redundancy and Anomalies.
- Then we'll define Decompositions of relations into multiple relations (similar to our Employee/Rank example), and we'll explain what it means for a decomposition to be "Lossless".
- Silberschatz textbook includes algorithms to decompose relations losslessly into "Normal Forms".
  - We'll discuss one simple approach.
- But we will briefly mention what it means for a decomposition to be Dependency-Preserving.
- And we'll end by explaining that 2 out of 3 ain't bad.
  - We'd like 3 nice properties, but we might only be able to get 2.
  - Anomaly avoidance, Lossless Decomposition, Dependency-Preservation

A Simple Approach to Convert a non-BCNF Schema into a BCNF Schema

- Suppose that DB schema has a relation r(R) that is not in BCNF.
  - Then there must be at least one non-trivial FD X → Y such that X is not a superkey for r(R).
  - We'll can assume that none of the attributes in X are also in Y.
    - If some X attributes are in Y, just get rid of them.
- Replace r(R) in our schema with the two different relations:

  r(R) – Y with FDs that don't include attributes of Y

  X U Y with FD X → Y
- If resulting DB schema is not in BCNF, do this again!
- This is what we did when we began Design Theory lectures!

  Employees(eid, name, addr, rank, salary_scale)
  with FD: rank → salary_scale

  was replaced by

  Employees2(eid, name, addr, rank)

  Salary_Table(rank, salary_scale)

Decomposition and Normalization

Given a relation schema and functional dependencies, it is always possible to decompose schema into a set of **BCNF** relations that:

- 1) *Eliminates Anomalies,* and is
- 2) a *Lossless Decomposition.*
- However, the schema might not always be
  3) *Dependency-Preserving.*

Given a relation schema and functional dependencies, it is always possible to decompose schema into a set of **3NF** relations that:

- is 2) a *Lossless D*ecomposition*,* and is
- *3) Dependency-Preserving*.
- However, the schema might not always
  1) *Eliminate Anomalies.*