DSA5104

Principles of Data Management and Retrieval

Lecture 5: Schema Refinement

Questions about Homework-1

- Q3. Return me the authors who have papers in VLDB conference before 2002 after 1995.

- 1995 < year < 2002

Questions about Homework-1

- Q4. Return me the authors who have cooperated both with "H. V. Jagadish" and "Divesh Srivastava".

- Write one paper with H and D together?

- Write one paper with H and another with D?

Questions about Homework-1

- Q9: Return me the number of papers written by H. V. Jagadish, Yunyao Li, and Cong Yu.

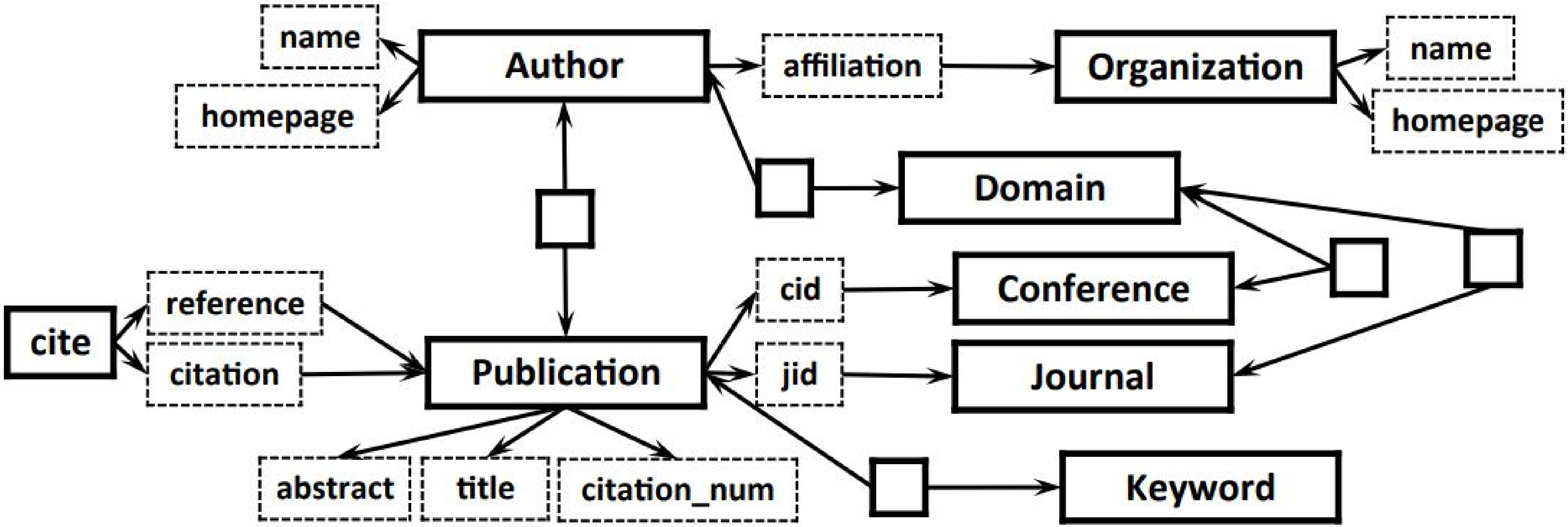
- Papers written by 3 of them.

Questions about Homework-1

- What to do with authors with the same name?
- E.g., Return me the authors…
- They are different authors. Return ids or other information as well.

Questions about Homework-1

- Cite Table



1. For tuple (a, b) in Cite, look for papers with pid = a , and pid = b in the publication table.
2. Examine their relationship using Google.

Recap

- Steps in Database Design
  - Requirements Analysis
  - Conceptual Design
    - ER Model (ER Diagram)
    - Entity sets, Relationship sets, Attributes
  - Logical Design
    - Translating ER Diagram to Relational Schema
  - Schema Refinement
    - FDs, F+, Attribute closure
    - BCNF
    - Decomposition
  - Physical Design - Indexes, disk layout
  - Security Design - Who accesses what, and how

Schema Refinement

Boyce-Codd Normal Form (BCNF)

- Relation R with FDs F is in BCNF if, for all X → A in F+
  - A ⊆ X (called a trivial FD), or
  - X is a superkey for R.

- In other words: “R is in BCNF if the only non-trivial FDs over R are key constraints.”

- Q: How to know if a set of attributes X is superkey of R?

Attribute Closure

- Typically, just check if $X \rightarrow Y$ is in F+. Efficient!
  - Compute attribute closure of X (denoted X+) wrt F.

    X+ = Set of all attributes A such that $X \rightarrow A$ is in F+
    - X+ := X
    - Repeat until no change (fixpoint):

      for $U \rightarrow V \subseteq F$,

      if $U \subseteq X+$, then add V to X+
  - Check if Y is in X+
  - If Y is in X+, then $X \rightarrow Y$ is in F+

- The above approach can also be used to check for keys of a relation R.
  - If X+ = R, then X is a superkey for R. ($X^+ = R$ means X^+ = {all attributes of R})
  - Q: How to check if X is a "candidate key" (minimal)?
  - A: For each attribute A in X, check if (X – A)+ = R. If (X – A)+ != R for every A in X, then X is a minimal superkey, i.e., a candidate key of R.

Why is BCNF Useful?

- If R is in BCNF, every field of every tuple stores **useful info** that cannot be inferred via FDs alone.
  - Say we know FD $X \rightarrow A$ holds for this example relation:
  - Can you guess the value of the missing attribute?
  - Yes, so relation is not in BCNF

X	Y	A
x	y1	a
x	y2	?

Example

- SNLRWH has FDs S → SNLRWH and R → W
- Q: Is this relation in BCNF?
  - No. The second FD causes a violation; R is not a superkey.
  - W values repeatedly associated with R values.

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Hourly_Emps

Decomposition of a Relation Scheme

- How to normalize a relation?
  - *Decompose* into multiple **normalized** relations

- Suppose R contains attributes $A_1 \ldots A_n$.

- A *decomposition* of R consists of replacing R by two or more relations such that:
  - Each new relation scheme contains a subset of the attributes of R, and
  - Every attribute of R appears as an attribute of at least one of the new relations.

Decomposing a Relation

- Easiest fix is to create a relation RW to store these associations (R → W), and to remove W from the main schema:

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Hourly_Emps2

R	W
8	10
5	7

Wages

- Q: Are both of these relations are now in BCNF?
- A: Yes. S → SNLRH is ok, as is R → W.

Quick Check

- In the picture above suppose $X \rightarrow A$. Then (true/false)
  - We need more information to know the value of the question mark
  - The question mark must be an a

X	Y	A
x	y1	a
x	y2	?

- After decomposition
  - No columns are replicated across tables
  - Resulting tables have the same cardinality

Problems with Decompositions

- There are three potential problems to consider:

1) May be ***impossible*** to reconstruct the original relation! (Lossiness)
    - Fortunately, not in the SNLRWH example.

2) Dependency checking may require joins.
    - Fortunately, not in the SNLRWH example.

3) Some queries become more expensive.
    - e.g., How much does Guldu earn?

Tradeoff: Must consider these 3 problems vs. redundancy.

Lossless Decomposition (example)

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

$\bowtie$

R	W
8	10
5	7

$=$

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Lossy Decomposition (example)

A	B	C
1	2	3
4	5	6
7	2	8

$A \rightarrow B;\ B \rightarrow C$

Lossy Decomposition (example)

A	B	C
1	2	3
4	5	6
7	2	8

$A \rightarrow B; B \rightarrow C$

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

Lossy Decomposition (example)

A	B	C
1	2	3
4	5	6
7	2	8

$A \rightarrow B;\ B \rightarrow C$

⇒

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

A	B
1	2
4	5
7	2

$\bowtie$

B	C
2	3
5	6
2	8

$=$

Lossy Decomposition (example)

A	B	C
1	2	3
4	5	6
7	2	8

$A \rightarrow B;\ B \rightarrow C$

$\Rightarrow$

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

A	B
1	2
4	5
7	2

$\bowtie$

B	C
2	3
5	6
2	8

$=$

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3

Lossless Join Decompositions

- Defn: Decomposition of R into X and Y is *lossless-join* w.r.t. a set of FDs F if, for every instance r that satisfies F:

$$\Pi_X(r) \bowtie \Pi_Y(r) = r$$

- It is always true that $r \subseteq \Pi_X(r) \bowtie \Pi_Y(r)$
  - When the relation is equality, the decomposition is lossless-join.

- Definition extended to decomposition into 3 or more relations in a straightforward way.

- ***It is essential that all decompositions used to deal with redundancy be lossless!*** *(Avoids Problem #1)*

More on Lossless Decomposition

- Theorem: The decomposition of R into X and Y is **lossless with respect to F** *if and only if* the closure of F contains:

  $$X \cap Y \to X, \quad \text{or}$$
  $$X \cap Y \to Y$$

- Example: decomposing ABC into AB and BC is lossy, because their intersection (i.e., "B") is not a key of either resulting relation (AB or BC).

- Useful corollary: If $X \to Z$ holds over R and $X \cap Z$ is empty, then decomposition of R into R-Z and XZ is loss-less (b/c X is a superkey of XZ).

- Just like in our BCNF example, X is Rating, Z is Wage. Clearly Rating intersect Wage is empty. So decomposing into SNLRH and RW is lossless.

Lossless Decomposition (example)

A	B	C
1	2	3
4	5	6
7	2	8

A	C
1	3
4	6
7	8

B	C
2	3
5	6
2	8

A → B; C → B

A	C
1	3
4	6
7	8

⋈

B	C
2	3
5	6
2	8

=

A	B	C
1	2	3
4	5	6
7	2	8

But, now we can't check A → B without doing a join!

Dependency Preserving Decomposition

- Dependency preserving decomposition (Intuitive):
  - A decomposition where the following is true:
    If R is decomposed into X, Y and Z,
    and we enforce FDs individually on each of X, Y and Z,
    then all FDs that held on R must also hold on result.
    (Avoids Problem #2 on our list.)

Dependency Preserving Decomposition

- Dependency preserving decomposition (Intuitive):
  - A decomposition where the following is true:
    If R is decomposed into X, Y and Z,
    and we enforce FDs individually on each of X, Y and Z,
    then all FDs that held on R must also hold on result.
    (Avoids Problem #2 on our list.)

- Definition – Projection of set of FDs F:
  If R is decomposed into X and Y, the projection of F on X (denoted F_X)
  is the set of FDs $U \rightarrow V$ in F^+
  such that all of the attributes U, V are in X.

  - F+ : closure of F, not just F

Dependency Preserving Decompositions (Cont.)

- Definition: Decomposition of R into X and Y is ***dependency preserving*** if $(F_X \cup F_Y)^+ = F^+$
  - i.e., if we consider only dependencies in the closure F^+ that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in F^+.
  - (just the formalism of our intuition above)

Dependency Preservation

- Critical to consider F^+ in this definition:
  - E.g., Given relation ABC and FDs $F = \{A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A\}$, decomposed into AB and BC.
  - *Is this dependency preserving? Is $C \rightarrow A$ preserved?*

Dependency Preservation

- Critical to consider F^+ in this definition:
  - E.g., Given relation ABC and FDs $F = \{A \to B,\ B \to C,\ C \to A\}$, decomposed into AB and BC.
  - *Is this dependency preserving? Is $C \to A$ preserved?*

- Note: F^+ contains $F \cup \{A \to C,\ B \to A,\ C \to B\}$,
  - E.g., with $A \to B$, $B \to C$, by transitivity, $F^+ \supseteq \{A \to C\}$
  - Next, examine $(F_{AB} \cup F_{BC})^+$

Dependency Preservation

- Critical to consider F^+ in this definition:
  - E.g., Given relation ABC and FDs $F = \{A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A\}$, decomposed into AB and BC.
  - *Is this dependency preserving? Is $C \rightarrow A$ preserved?*

- Note: F^+ contains $F \cup \{A \rightarrow C, B \rightarrow A, C \rightarrow B\}$,
  - E.g., with $A \rightarrow B$, $B \rightarrow C$, by transitivity, $F^+ \supseteq \{A \rightarrow C\}$
  - Next, examine $(F_{AB} \cup F_{BC})^+$

- With $F^+ = F \cup \{A \rightarrow C, B \rightarrow A, C \rightarrow B\}$
  - $F_{AB} \supseteq \{A \rightarrow B, B \rightarrow A\}$; $F_{BC} \supseteq \{B \rightarrow C, C \rightarrow B\}$
  - So, $(F_{AB} \cup F_{BC})^+ \supseteq \{B \rightarrow A, C \rightarrow B\}$
  - Hence, $(F_{AB} \cup F_{BC})^+ \supseteq \{C \rightarrow A\}$

Quick Check

- True/False:
  - In a lossless decomposition, the resulting tables join back together to give the original data.
  - In lossy decompositions, the result of the re-join could be missing tuples from the original.

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: A → B
     b. Compute A+

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$
     b. Compute A+
     c. Create $R_1 = A+$, $R_2 = A \cup (R - A+)$

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$
     b. Compute A+
     c. Create $R_1 = A+$, $R_2 = A \cup (R - A+)$
     d. Compute all FDs on R_1 and R_2

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$
     b. Compute A+
     c. Create $R_1 = A+$, $R_2 = A \cup (R - A+)$
     d. Compute all FDs on R_1 and R_2
     e. Repeat step 1 on R_1 and R_2 separately

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$
     b. Compute A+
     c. Create R_1 = A+, $R_2 = A \cup (R - A+)$
     d. Compute all FDs on R_1 and R_2
     e. Repeat step 1 on R_1 and R_2 separately
  2. Stop when all relations are in BCNF or are two attributes or fewer

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$
     b. Compute A+
     c. Create $R_1 = A+$, $R_2 = A \cup (R - A+)$
     d. Compute all FDs on R_1 and R_2
     e. Repeat step 1 on R_1 and R_2 separately
  2. Stop when all relations are in BCNF or are two attributes or fewer
     - All relations with two or fewer attributes are always in BCNF

Decomposition into BCNF

- Consider relation R with FDs F
  1. Check if R is in BCNF, if not:
     a. Pick a violation FD: $A \rightarrow B$
     b. Compute A+
     c. Create $R_1 = A+$, $R_2 = A \cup (R - A+)$
     d. Compute all FDs on R_1 and R_2
     e. Repeat step 1 on R_1 and R_2 separately
  2. Stop when all relations are in BCNF or are two attributes or fewer
     - All relations with two or fewer attributes are always in BCNF

- This is a lossless decomposition that is guaranteed to terminate.
  - Finite number of columns to partition

Decomposition into BCNF - Example

- Relation R = CSJDPQV, key C, JP → C, SD → P, J → S
  - {contractid, supplierid, projectid, deptid, partid, qty, value}

Decomposition into BCNF - Example

- Relation R = CSJDPQV, key C, JP → C, SD → P, J → S
  - {contractid, supplierid, projectid, deptid, partid, qty, value}
  - SD → P is a violation, so decompose into SD+ = SDP, and {SD} ∪ {R - SD+} = CSJDQV

Decomposition into BCNF - Example

- Relation R = CSJDPQV, key C, JP → C, SD → P, J → S
  - {contractid, supplierid, projectid, deptid, partid, qty, value}
  - SD → P is a violation, so decompose into SD+ = SDP, and {SD} ∪ {R - SD+} = CSJDQV
  - SDP is already in BCNF. For CSJDQV, J → S is a violation. So decompose into JS and CJDQV

Decomposition into BCNF - Example

- Relation R = CSJDPQV, key C, JP → C, SD → P, J → S
  - {contractid, supplierid, projectid, deptid, partid, qty, value}
  - SD → P is a violation, so decompose into SD+ = SDP, and {SD} ∪ {R − SD+} = CSJDQV
  - SDP is already in BCNF. For CSJDQV, J → S is a violation. So decompose into JS and CJDQV
  - So we end up with SDP, JS, and CJDQV all in BCNF

Decomposition into BCNF - Example

- Relation R = CSJDPQV, key C, JP → C, SD → P, J → S
  - {contractid, supplierid, projectid, deptid, partid, qty, value}
  - SD → P is a violation, so decompose into SD+ = SDP, and {SD} ∪ {R - SD+} = CSJDQV
  - SDP is already in BCNF. For CSJDQV, J → S is a violation. So decompose into JS and CJDQV
  - So we end up with SDP, JS, and CJDQV all in BCNF

  - *Is this a dependency preserving decomposition? Is JP → C preserved?*

  - Note: several functional dependencies may cause violation of BCNF
  - The order of which we "deal with" them could lead to very different sets of relations

BCNF and Dependency Preservation

- In general, there may not be a dependency preserving decomposition into BCNF.

BCNF and Dependency Preservation

- In general, **there may not be a dependency preserving decomposition into BCNF.**
- E.g., CSZ, CS → Z, Z → C
  - Can't decompose while preserving 1st FD; not in BCNF.
  - The second functional dependency violates BCNF, so we have to decompose it which means we will lose our dependency preservation

BCNF and Dependency Preservation

- In general, **there may not be a dependency preserving decomposition into BCNF.**
- E.g., CSZ, CS → Z, Z → C
  - Can't decompose while preserving 1st FD; not in BCNF.
  - The second functional dependency violates BCNF, so we have to decompose it which means we will lose our dependency preservation

- Similarly, decomposition of CSJDPQV into SDP, JS and CJDQV is not dependency preserving
  (w.r.t. the FDs JP → C, SD → P and J → S).
  – However, it is a lossless join decomposition.
  – In this case, adding JPC to the collection of relations gives us a dependency preserving decomposition.
    - But JPC tuples are stored only for checking the FD (*Redundancy!*)

Decomposition into 3NF

- Third Normal Form (3NF)

- Obviously, the algorithm for lossless decomposition into BCNF can be used to obtain a lossless join decomp into 3NF (typically, can stop earlier) but does not ensure dependency preservation.

- To ensure dependency preservation, one idea:
  - If $X \rightarrow Y$ is not preserved, add relation XY.

    Problem is that XY may violate 3NF!
- Refinement: Instead of the given set of FDs F, use a *minimal cover for F*.

Summary of Schema Refinement

- BCNF: each field contains data that cannot be inferred via FDs
  - Ensuring BCNF is a good heuristic

Summary of Schema Refinement

- BCNF: each field contains data that cannot be inferred via FDs
  - Ensuring BCNF is a good heuristic

- Have a relation non in BCNF? Try decomposing into BCNF relations.
  - Must consider whether all FDs are preserved!

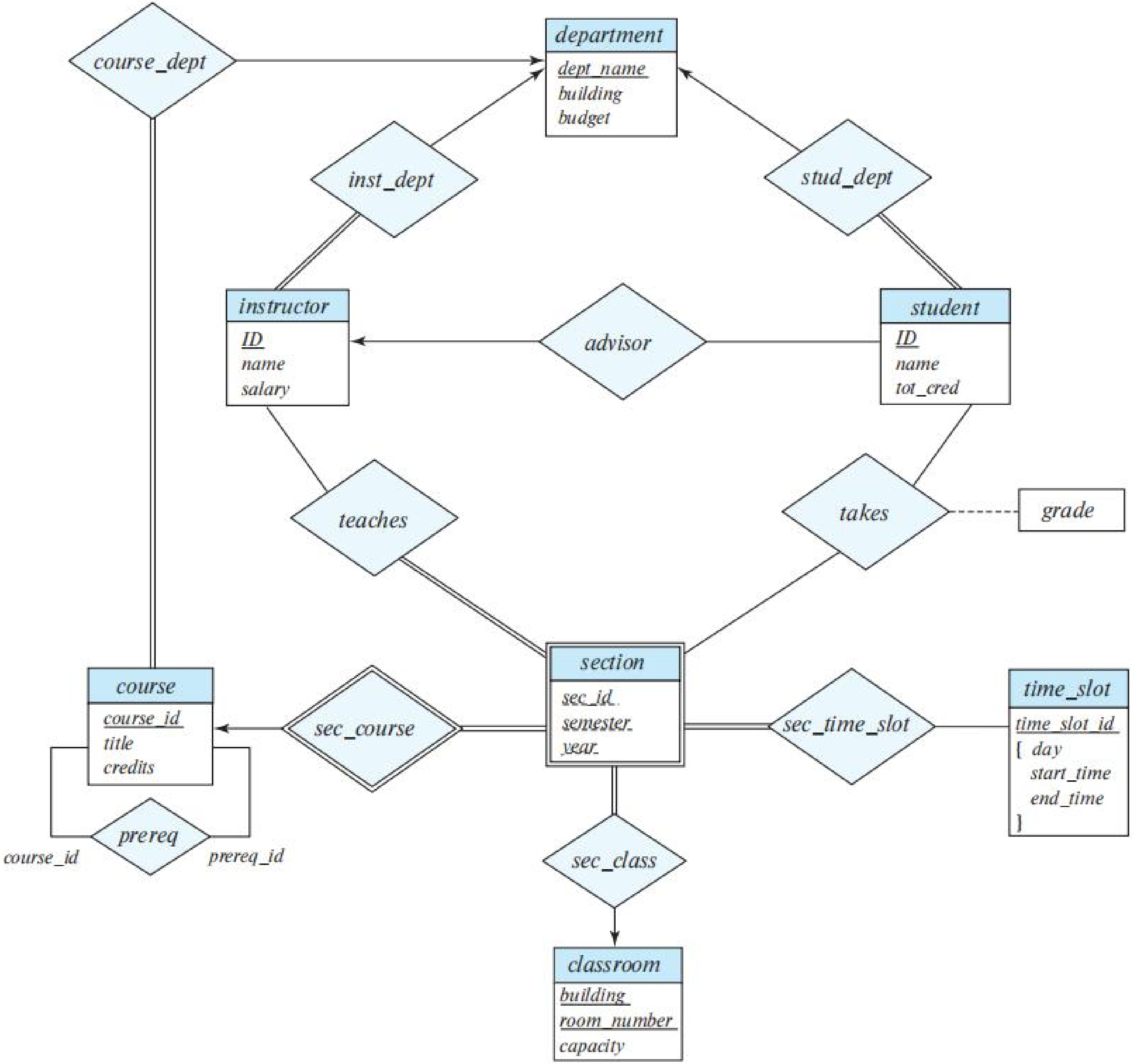
Summary of Schema Refinement (Cont.)

- What to do when a lossless, dependency preserving decomposition into BCNF is impossible?
  - There is a more permissive Third Normal Form (3NF)
  - But you will have redundancy. Beware. You will need to keep it from being a problem in your application code.

Steps in Database Design

- Requirements Analysis
- Conceptual Design
  - ER Model (ER Diagram)
  - Entity sets, Relationship sets, Attributes
- Logical Design
  - Translating ER Diagram to Relational Schema
- Schema Refinement
  - FDs, F+, Attribute closure
  - BCNF
  - Decomposition
- Physical Design - Indexes, disk layout
- Security Design - Who accesses what, and how

ER Diagram for University Database



Schema Diagram for University Database