Relational Database Design Theory (I)

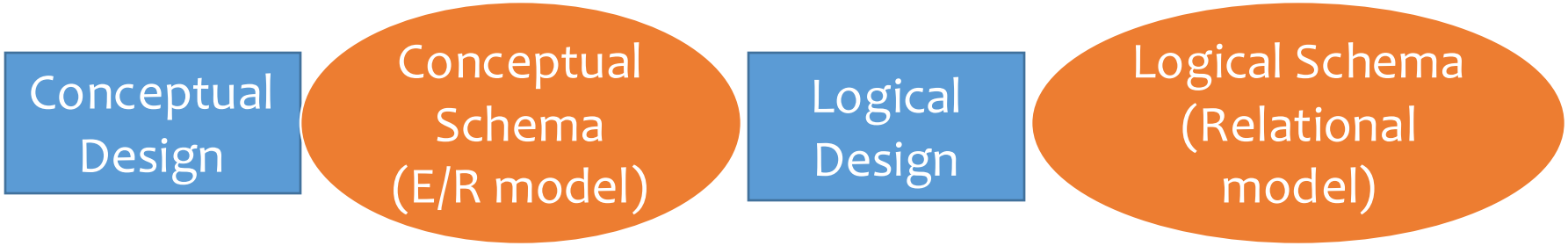
CS348

Instructor: Sujaya Maiyya

Announcements

- Assignment 2 is released
  - Due by Oct 31st

Database Design – where are we?



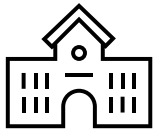
- Understand the real-world domain being modeled and constrained
- Entity-Relationship model
- Translate E/R diagram to relational data model
- (Refine a good database schema)
- Create DBMS schema (using DDL SQL)

Case Study

- Consider a simple university DB:

Instructors

Departments



Courses

Students

- External application constraints such as:
  - Each instructor has name, salary, and department
  - Each instructor is officially affiliated with one department
  - Each department has one building and one budget
  - Each student can have at most one advisor from each department

Case Study

Redundant data replication! (CS, DC, 20000) repeated k times if there are k instructors in CS!

- Possible Design: one large table InstructorDep with one row for each instructor

instructorID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

Fail to capture corner cases!

- If the building of CS is changed to E4?
- If the only instructor in Physics retires?
- If new department (w/o yet an instructor) is added?

Case Study

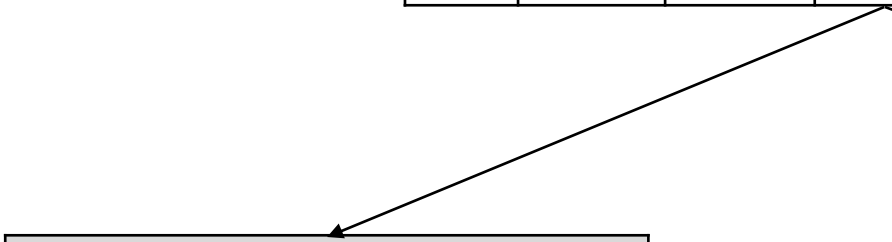
- Possible Design: consider the following schema for courses with one row for each course offering

CourseID	term	instructorName	capacity
CS348	S23	Sujaya	100
CS341	W25	Lap Chi	80
CS348	W25	Semih	100
CS348	S25	Xiao	100
CS350	W19	Salem	130

- Is there any redundancy? Unclear!
  - Depends on the external application constraints
  - If courses have one associated capacity (independent of term): Redundant
  - Otherwise, repetition may be necessary

Solution To Redundancy: Decompositions

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000



Inst			
iID	name	salary	depName
111	Alice	5000	CS
222	Bob	4000	Physics
333	Carl	5200	CS
444	Diana	5500	CS

Dep		
depName	bldng	budget
CS	DC	20000
Physics	PHY	30000

Decompositions: A good example

Break down a complex database schema into smaller, more manageable pieces

instructorID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

instructorID	name	salary	depName
111	Alice	5000	CS
222	Bob	4000	Physics
333	Carl	5200	CS
444	Diana	5500	CS

⋈

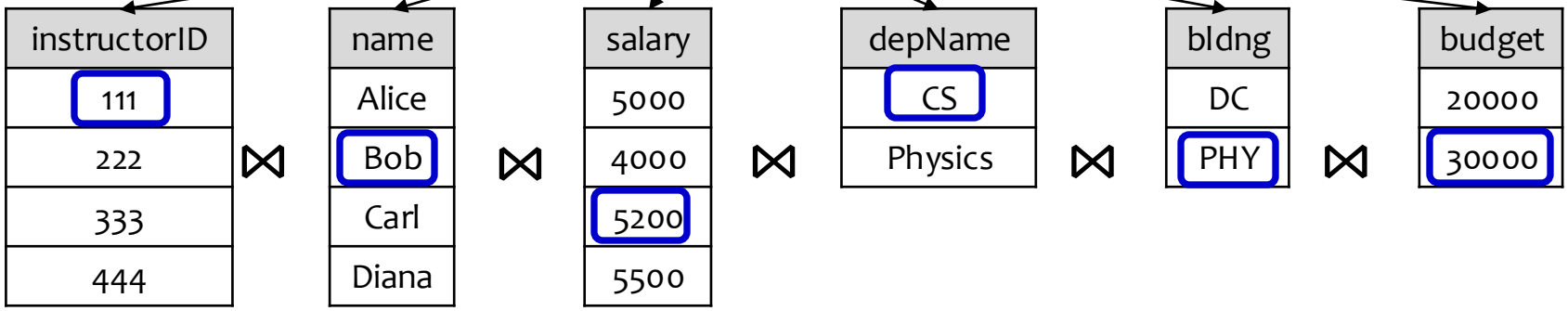
depName	bldng	budget
CS	DC	20000
Physics	PHY	30000

Loss less decomposition
$R = R1 \bowtie R2$

instructorID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

Decompositions: A bad example

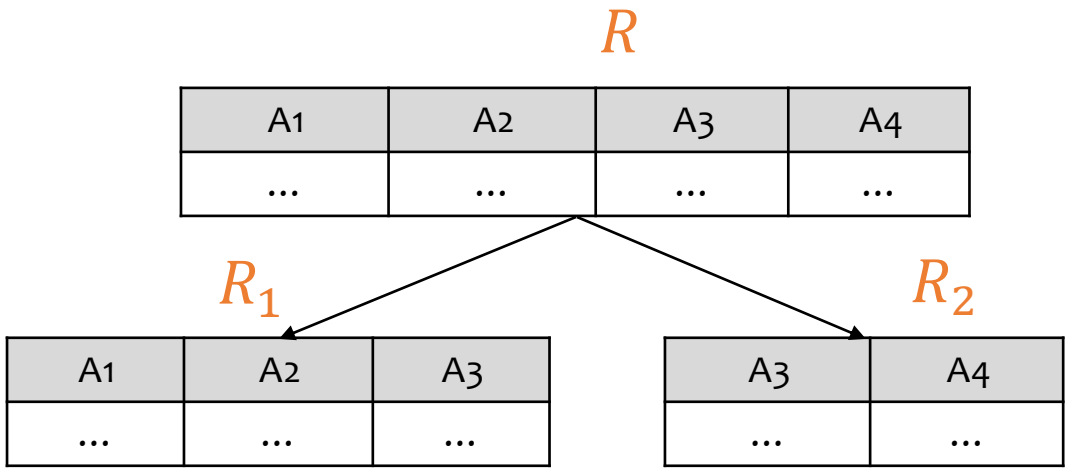
instructorID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000



instructorID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
111	Bob	5200	CS	PHY	30000
…	…	…	…	…	…

Lossy? But I got more rows! Can't tell what's fact and what's not, so we lose information!

Decompositions: must be lossless



- R is decomposed into R_1 and R_2
  - Attribute (R_1) ∪ Attribute(R_2) = Attribute (R)
  - R_1 and R_2 are the projections of R onto Attribute (R_1) and Attribute(R_2)
- Lossless decomposition gives $R = R_1 \bowtie R_2$

For any tuple $(a, b, c, d) \in R$, $(a, b, c) \in R_1$ and $(c, d) \in R_2$; then $(a, b, c) \bowtie (c, d) \in R$

Decompositions: second requirement

- R2 (Locality of Constraints): If the app had a constraint C, we would prefer to check C in a single relation

High level question we answer in this topic:

How to decompose a database to be lossless & (preferably) retain locality of constraints?

Normal Forms

- Given a set of constraints about the real-world facts that an application will store, how can we formally separate “good” and “bad” relational database schemas?

- Normal Forms (NF): Normalization helps in reducing data redundancy and improving data integrity, making it easier to manage and maintain databases.

Outline For Today

1. Application Constraints and Decompositions
2. Functional Dependencies
3. Boyce-Codd Normal Form (BCNF) & BCNF Decomposition Alg.
4. Dependency Preservation and 3rd Normal Form

A Motivation Example

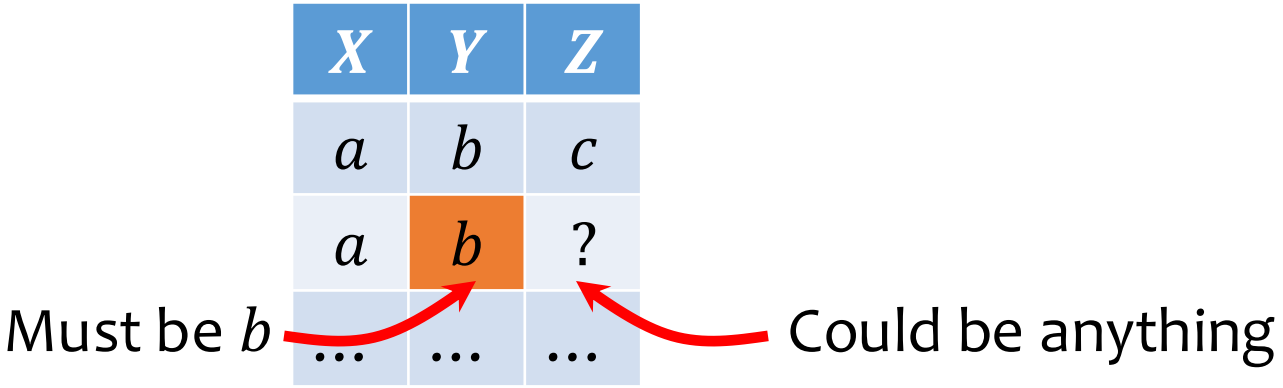
- Consider the following schema for InstructorDept

instructorID	name	salary	depName	bldng	budget

  - Each instructorID has 1 name and salary
    - instructorID determines name and salary
  - Each depName has 1 building and 1 associated budget
    - depName determines bldng and budget
  - Each instructorID, depName is unique in InstructorDep
    - instructorID and depName together determine all remaining attributes, including name, salary, bldgn and budget
  - How about instructorID and name together determining name? This is trivial!

Functional dependencies

- $X \rightarrow Y$ means that whenever two tuples in R agree on all the attributes in X, they must also agree on all attributes in Y



Formal definition

Let t[A] be a tuple t's projection on attributes A

Let X, Y be sets of attributes

- A FD $X \rightarrow Y$ holds in a relation R if any given pair of tuples t_1 and $t_2 \in R$ with $t_1[X] = t_2[X]$, we must have $t_1[Y] = t_2[Y]$.

- We say X determines Y
- A FD $X \rightarrow Y$ holds in a relation R means that $X \rightarrow Y$ holds on all instances of R

Redefining "keys" using FD's

A set of attributes *K* is a key for a relation R if

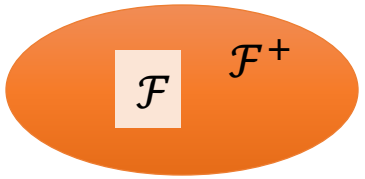
- *K* → all (other) attributes of R
  - That is, K is a "super-key"
- No proper subset of *K* satisfies the above condition
  - That is, K is minimal

Closure of FD sets: $\mathcal{F}^+$

- How do we know what additional FDs hold on a schema R?

- A set $\mathcal{F}$ of FDs logically implies $X \to Y$ if $X \to Y$ holds in all instances of R that satisfy $\mathcal{F}$

- The closure of a FD set $\mathcal{F}$ (denoted as $\mathcal{F}^+$):
  - The set of all FDs that are logically implied by $\mathcal{F}$
  - Informally, $\mathcal{F}^+$ includes all of the FDs in $\mathcal{F}$, i.e., $\mathcal{F} \subseteq F^+$, plus any dependencies they imply.



Armstrong's Axioms

- Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$ (trivially)

  instructorID, name → instructorID

- Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ (trivially)

  If instructorID → salary , then
  instructorID, name → salary, name

- Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

  If instructor ID → depName and depName → budget,
  then instructorID → budget

Implications of Armstrong's Axioms

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

  i. $X \rightarrow YZ$

  ii. $YZ \rightarrow Y$ (by reflexivity); $YZ \rightarrow Z$ (by reflexivity)

  iii. $X \rightarrow Y$ (by transitivity); $X \rightarrow Z$ (by transitivity)

- Union: If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

- Pseudo-transitivity: If $X \rightarrow Y$ and $YZ \rightarrow T$ then $XZ \rightarrow T$

- Using Armstrong's Axioms, you can prove or disprove a (derived) FD given a set of (base) FDs

Prove a FD in $\mathcal{F}^+$

instructorID, projID → funds

$\mathcal{F}$ includes:

instructorID → name
projID → projName, projDep
instructorID, projID → hours
projDep, hours → funds

- projID → projName, projDep
- projID → projDep (decomposition)
- instructorID, projID → instructorID, projDep (augmentation)
- instructorID, projID → hours
- instructorID, projID → instructorID, hours, projDep (union)
- instructorID, projID → hours, projDep (decomposition)
- hours, projDep → funds
- instructorID, projID → funds (transitivity)

Compute $\mathcal{F}^+$ from $\mathcal{F}$

- Start with closure $\mathcal{F}^+ = \mathcal{F}$
- For each FD f in $\mathcal{F}^+$
  - Apply reflexivity and augmentation rules on f
  - Add the resulting FD to $\mathcal{F}^+$
- For each pair of FDs f1 and f2 in $\mathcal{F}^+$
  - If f1 and f2 can be combined using the transitivity rule, add the resulting FD to $\mathcal{F}^+$
- Repeat until no new FD can be added to $\mathcal{F}^+$

Reasoning with $\mathcal{F}^+$

Given a relation R and a set $\mathcal{F}$ of FD's

- Does another FD $X \to Y$ follow from $\mathcal{F}$?
  - Compute $\mathcal{F}^+$ with respect to $\mathcal{F}$
  - If $(X \to Y) \in \mathcal{F}^+$, then $X \to Y$ follows from $\mathcal{F}$

- Is K a key of R?
  - If $(K \to R) \in \mathcal{F}^+$, K is a super key
  - Still need to verify that K is *minimal* (how?)
    - Hint: For any proper subset X of K, $(X \to R) \notin \mathcal{F}^+$

Attribute closure: Z^+

Given the relation schema R and a set $\mathcal{F}$ of FDs

- The closure of attributes X (denoted as X^+) is the set of all attributes $\{A_1, A_2, \ldots A_k\}$ functionally determined by X (that is, $X \rightarrow A_1 A_2 \ldots \mathrm{A_K}$)

- Algorithm for computing the closure Compute$X^+(X, \mathcal{F})$:
  - Start with closure $= X$
  - If $Z \rightarrow Y$ is in $\mathcal{F}$ and Z is already in the closure, then also add Y to the closure
  - Repeat until no new attributes can be added

In class exercise

- Given a relation R(ABCDEFG) under a set $\mathcal{F}$ of FDs, compute$X^+(\{B, F\}, \mathcal{F})$?

$\mathcal{F}$ includes:

$A, B \rightarrow F$

$A \rightarrow C$

$B \rightarrow E, D$

$D, F \rightarrow G$

FD	Z^+
initial	B, F
$B \rightarrow E, D$	B, F, E, D
$D, F \rightarrow G$	B, F, E, D, G

In class exercise

- Given a relation EmpProj (SIN, pnum, hours, ename, pname, ploc, allowance) under a set $\mathcal{F}$ of FDs, computeX^+({pnum, hours}, $\mathcal{F}$)?

$\mathcal{F}$ includes:

SIN, pnum → hours
SIN → ename
pnum → pname, ploc
ploc, hours → allowance

FD	Z^+
initial	*pnum, hours*
pnum → pname, ploc	*pnum, hours, pname, ploc*
ploc, hours → allowance	*pnum, hours,pname, ploc, allowance*

In class exercise

- Given a relation EmpProj (SIN, pnum, hours, ename, pname, ploc, allowance) under a set $\mathcal{F}$ of FDs, compute$X^+(\{\text{SIN, pnum}\}, \mathcal{F})$?

$\mathcal{F}$ includes:

SIN, pnum → hours
SIN → ename
pnum → pname, ploc
ploc, hours → allowance

FD	Z^+
initial	*SIN, pnum*
SIN → ename	*SIN, pnum , ename*
pnum → pname, ploc	*SIN, pnum , ename, pname, ploc*
SIN, pnum → hours	*SIN, pnum , ename, pname, ploc, hours*
ploc, hours → allowance	*SIN, pnum , ename, pname, ploc, hours, allowance*

- Compute$X^+(\{\text{SIN, pnum, hours}\}, \mathcal{F})$?

Reasoning with X^+

Given a relation R and a set $\mathcal{F}$ of FD's

- Does another FD $X \rightarrow Y$ follow from $\mathcal{F}$?
  - Compute X^+ with respect to $\mathcal{F}$
  - If $Y \subseteq X^+$, then $X \rightarrow Y$ follows from $\mathcal{F}$

- Is K a key of R?
  - Compute K^+ with respect to $\mathcal{F}$
  - If K^+ contains all the attributes of R, K is a super key
  - Still need to verify that K is *minimal* (how?)
    - Hint: check the attribute closure of its proper subset, i.e., Check that for no set X formed by removing attributes from K is $K^+$the set of all attributes

Alternative: Compute $\mathcal{F}^+$ from X^+

Subset-closure enumeration

- List every subset $X \subseteq R$
- Compute its attribute closure X^+ under $\mathcal{F}$
- Then all FDs $X \to Y$ with $Y \subseteq X^+$ lie in $\mathcal{F}^+$

What is next?

- Functional dependencies
  - Armstrong's axioms
  - Closure of FDs
  - Closure of attributes

- Next lecture: Decomposition
  - Boyce-Codd Normal Form (BCNF)
  - Third Normal Form (3NF)