Relational Database Design Theory (I)

Introduction to Database Management

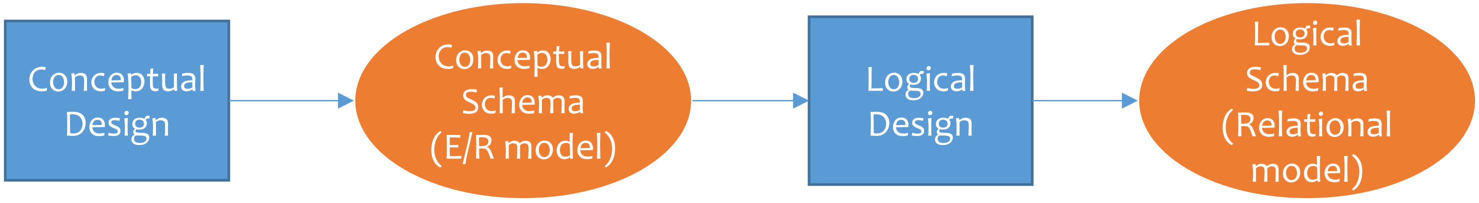
CS348 Fall 2022

Announcements (Tue. Oct 18)

- Assignment 1's grade was released last Thur
  - Partial solution is available on Learn
  - Appeal deadline is this Thur
- Milestone 1 is due this Thur, Oct 20, 11:59pm
  - Basic score is 45 points, capped by 49 points
  - Contribute $\frac{\min(s1,49)}{45} * 30$ to the final project score
- Assignment 2 is released
  - Cover lectures till lecture 10
  - Due by Thur, Oct 27, 11:59pm

Design process – where are we?

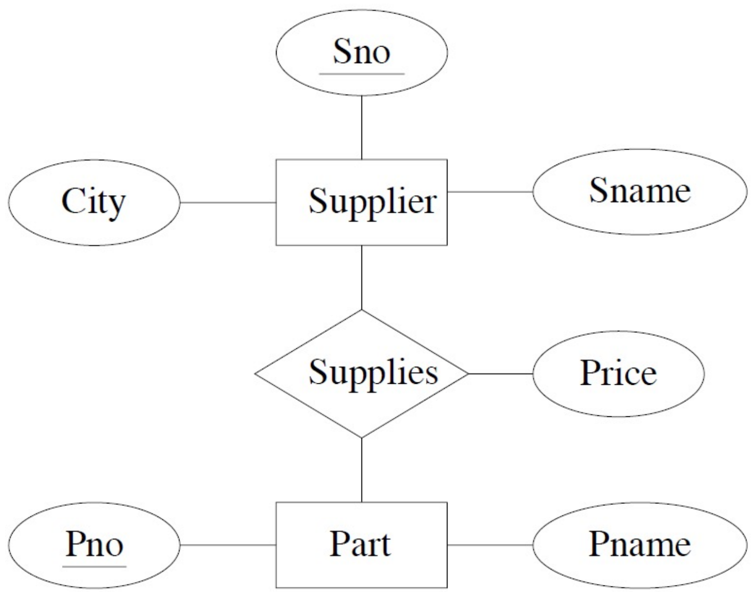
- Schema refinement



- What are relational design principles?

A Parts/Suppliers database example

- Each type of part has a name and an identifying number and may be supplied by zero or more suppliers.
- Each supplier has an identifying number, a name, and a contact location for ordering parts.
- Each supplier may offer the part at a different price.



Parts/Suppliers example (cont.)

- An instance

Suppliers

Sno	Sname	City
S1	Magna	Ajax
S2	Budd	Hull

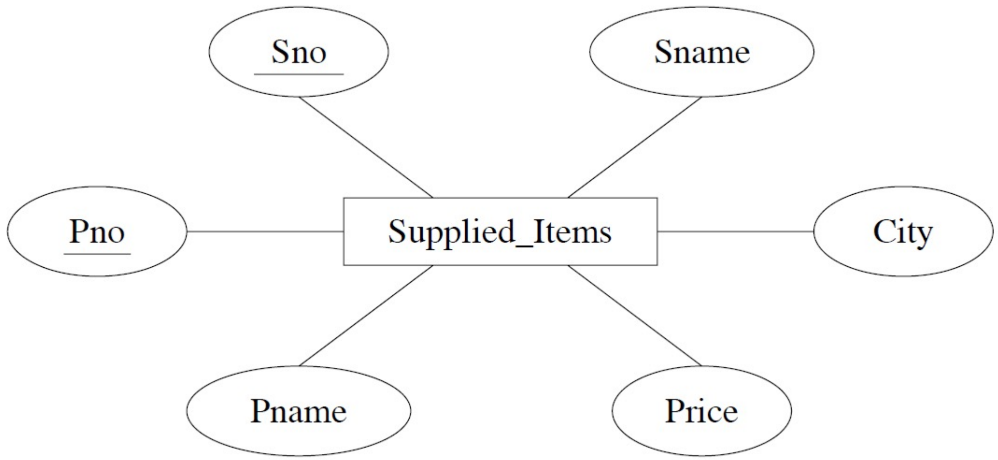
Parts

Pno	Pname
P1	Bolt
P2	Nut
P3	Screw

Supplies

Sno	Pno	Price
S1	P1	0.50
S1	P2	0.25
S1	P3	0.30
S2	P3	0.40

Alternate Parts/Suppliers database



Supplied_Items

Sno	Sname	City	Pno	Pname	Price
S1	Magna	Ajax	P1	Bolt	0.50
S1	Magna	Ajax	P2	Nut	0.25
S1	Magna	Ajax	P3	Screw	0.30
S2	Budd	Hull	P3	Screw	0.40

Change anomalies

- The single-table schema suffers from:
  - Update anomalies (e.g. change supplier name)
  - Insert anomalies (e.g. add a new item)
  - delete anomalies (e.g. S1 no longer supplies Nut)
  - Likely increase in space requirements

Supplied_Items

Sno	Sname	City	Pno	Pname	Price
~~S1~~	~~Magna~~ Xi	~~Ajax~~	~~P1~~	~~Bolt~~	~~0.50~~
~~S1~~	~~Magna~~ Xi	~~Ajax~~	~~P2~~	~~Nut~~	~~0.25~~
~~S1~~	~~Magna~~ Xi	~~Ajax~~	~~P3~~	~~Screw~~	~~0.30~~
S2	Budd	Hull	P3	Screw	0.40

Change anomalies

- The single-table schema suffers from:
  - Update anomalies (e.g. change supplier name)
  - Insert anomalies (e.g. add a new item)
  - delete anomalies (e.g. S1 no longer supplies Nut)
  - Likely increase in space requirements

- The multi-table schema does not have these problems.

Suppliers

Sno	Sname	City
~~S1~~	~~Magna~~ X!	~~Ajax~~
S2	Budd	Hull

Parts

Pno	Pname
P1	Bolt
P2	Nut
P3	Screw

Supplies

Sno	Pno	Price
~~S1~~	~~P1~~	~~0.50~~
~~S1~~	~~P2~~	~~0.25~~
~~S1~~	~~P3~~	~~0.30~~
S2	P3	0.40

Another alternate

- Is more tables always better?

Snos

Sno
S1
S2

Snames

Sname
Magna
Budd

Cities

City
Ajax
Hull

Pnums

Pnum
I1
I2
I3

Pnames

Pname
Bolt
Nut
Screw

Prices

Price
0.50
0.25
0.30
0.40

- Information about relationships is lost

Designing good databases

- Goals
  - A methodology for evaluating schemas (detecting anomalies)
  - A methodology for transforming bad schemas into good ones
- How do we know an anomaly exists?
- What should we do if an anomaly exists?

Schema decomposition: avoid anomalies while retaining all info in the instances.

Integrity constraints (e.g. dependencies between attributes) → lead to anomalies

Supplied_Items

Sno	Sname	City	Pno	Pname	Price
S1	Magna	Ajax	P1	Bolt	0.50
S1	Magna	Ajax	P2	Nut	0.25
S1	Magna	Ajax	P3	Screw	0.30
S2	Budd	Hull	P3	Screw	0.40

Design Theory

- Detect anomalies: Functional dependencies   This lecture

- Repair anomalies: Schema decomposition

Functional dependencies

- Consider the following relation schema

EmpProj

SIN	PNum	Hours	EName	PName	PLoc	Allowance

SIN → EName

1. SIN determines employee name
PNum→ PName, PLoc
2. Project number determines project name and location
3. Allowances are always the same for the same number of hours at the same location
PLoc, Hours → Allowance

- A functional dependency (FD) has the form $X \rightarrow Y$, where X and Y are sets of attributes in a relation R

Functional dependencies

- A functional dependency (FD) has the form $X \to Y$, where X and Y are sets of attributes in a relation R

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

X	Y	Z
a	b	c
a	b	?
…	…	…

Must be b

Could be anything

  - If X is a superkey of R , then X → R (all the attributes)

Functional dependencies

- Consider the following relation schema

EmpProj

<u>SIN</u>	<u>PNum</u>	Hours	EName	PName	PLoc	Allowance

1. SIN determines employee name

   SIN → EName

2. Project number determines project name and location

   PNum→ PName, PLoc

3. Allowances are always the same for the same number of hours at the same location

   PLoc, Hours → Allowance

- How about SIN and EName determines Ename?
  - Trivial FD

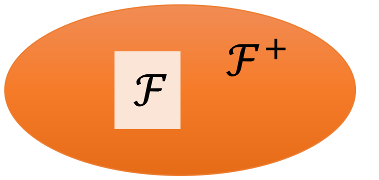
  SIN,EName → EName

Closure of FD sets

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

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

- The closure of a FD set $\mathcal{F}$ (denoted $\mathcal{F}^+$):
  - 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.



Rules of FD's

- Armstrong's axioms
  - Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$
  - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

SIN,EName → EName

SIN, Z → Ename, Z

- Rules derived from axioms
  - Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  - Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

PNum→ PName, PLoc

PNum→ Pname
PNum→ PLoc

☞Using these rules, you can prove or disprove an FD given a set of FDs

Example for proving a FD

Prove SIN, PNum → Allowance

1. *SIN, PNum* → *Hours* (∈ $\mathcal{F}$)
2. *PNum* → *PName,PLoc* (∈ $\mathcal{F}$)
3. *PLoc,Hours* → *Allowance* (∈ $\mathcal{F}$)

$\mathcal{F}$ includes:

SIN, PNum → *Hours*
SIN → *EName*
PNum → *PName,PLoc*
PLoc, Hours → *Allowance*

Example for proving a FD

$\mathcal{F}$ includes:

SIN, PNum → Hours
SIN → EName
PNum → PName,PLoc
PLoc, Hours → Allowance

Prove SIN, PNum → Allowance

1. *SIN, PNum → Hours* (∈ $\mathcal{F}$)
2. *PNum → PName,PLoc* (∈ $\mathcal{F}$)
3. *PLoc,Hours → Allowance* (∈ $\mathcal{F}$)
4. *SIN, PNum → PNum (reflexivity)*
5. *SIN, PNum → PName,PLoc (transitivity, 4 and 2)*
6. *SIN, PNum → PLoc (decomposition, 5)*
7. *SIN, PNum → PLoc,Hours (union, 6 and 1)*
8. *SIN, PNum → Allowance (transitivity, 7 and 3)*

Example for proving a FD

Prove SIN, PNum → Allowance

1. SIN, PNum → Hours (∈ $\mathcal{F}$)
2. PNum → PName,PLoc (∈ $\mathcal{F}$)
3. PLoc,Hours → Allowance (∈ $\mathcal{F}$)
4. SIN, PNum → PNum (reflexivity)
5. SIN, PNum → PName,PLoc (transitivity, 4 and 2)
6. SIN, PNum → PLoc (decomposition, 5)
7. SIN, PNum → PLoc,Hours (union, 6 and 1)
8. SIN, PNum → Allowance (transitivity, 7 and 3)

$\mathcal{F}$ includes:

SIN, PNum → Hours
SIN → EName
PNum → PName,PLoc
PLoc, Hours → Allowance

SIN, PNum

PLoc, Hours, Allowance, ..

Attribute closure of {SIN, PNum}

Attribute closure

- The closure of attributes Z in a relation R (denoted Z^+) with respect to a set of FDs, $\mathcal{F}$, is the set of all attributes $\{A_1, A_2, \ldots\}$ functionally determined by Z (that is, Z $\rightarrow A_1 A_2 \ldots$)

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

Example for computing attribute closure

$\text{Compute}Z^+(\{PNum, Hours\}, \mathcal{F})$:

$\mathcal{F}$ 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***

$$PNum, Hours \rightarrow PLoc, Allowance$$

Using attribute closure

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

- 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?) [Exercise]
    - Hint: check the attribute closure of its proper subset.

Design Theory

- Detect anomalies: Functional dependencies
  - Closure of FDs (rules, e.g. Armstrong's axioms)
  - Attribute closure
- Repair anomalies: Schema decomposition
  - (next lecture)