CS 133: Databases

Fall 2019
Lec 22 – 11/26
Database Design
Prof. Beth Trushkowsky

Warm-up Exercise

(See exercise sheet. You can start before class.)

Given: X→Y and YW → Z

XW → YW (augmentation)
XW → Z (transitivity)

Rules of Inference

- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If Y ⊆ X, then X → Y
  - *Augmentation*: If X → Y, then XZ → YZ for any Z
  - *Transitivity*: If X → Y and Y → Z, then X → Z
- Some additional rules (that follow from AA):
  - *Union*: If X → Y and X → Z, then X → YZ
  - *Decomposition*: If X → YZ, then X → Y and X → Z
  - *Pseudo-transitivity*: If X → Y and YW → Z, then XW → Z

Goals for Today

- Learn how to decompose a relation to adhere to Boyce-Codd Normal Form (BCNF)
- Understand lossy vs. loss-less decompositions
- Reason about issues that can result even if a decomposition is loss-less

The Issue with Non-Key FDs

- Why does the FD *rating → hourly_wages* yield redundancy issues?

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

- *Rating* is a **non-key field**, so there could be **duplicate pairs** of particular {***rating, hourly_wages***} in this relation
- By separating {***rating, hourly_wages***} into its own relation, we resolve redundancy!
  - Can regain the original data via **natural join**

"Normal" Forms for a Schema

- *Idea*: decompose relation into two or more relations to remove redundancy. Decomposition **guided by FDs!**
- **Boyce-Codd Normal Form (BCNF)**
  - Adhere to simple conditions and anomalies caused by data redundancy cannot occur
- BCNF definition:
  A Relation R with FDs F is *in* BCNF if, **for all X → A in F⁺**
  - A ∈ X (a trivial FD), or
  - **X is a superkey for R**
- I.e.,: R is in BCNF if the **only non-trivial FDs** over R are ***key constraints***

Each tuple in *R* is an entity or relationship **identified by a key** and **described by other** attributes

Problems with Decompositions

- There are three potential problems to consider:
  1) May be **impossible to reconstruct the original** relation! (**Lossiness**)
  2) Checking functional dependencies may require joins
  3) Some queries become more expensive due to joins
     - e.g., *How much does Smiley earn*?

Lossiness (#1) cannot be allowed
#2 and #3 are design tradeoffs
Must consider these issues vs. redundancy

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

Lossy vs. Lossless Decomposition

- Example schema:
  Oversees(**P**rojectId, **E**mployeeId, **D**epartmentId)
- FDs:
  - E → P (an employee oversees only one project)
  - D → P (a dept works on only one project)
  - E → D (an employee only works with one dept for these projects)
- Example instance of Oversees:

Project	Employee	Department
Comet	Alice	Physics
Comet	Bob	Astronomy
Genomics	Carl	Biology
Genomics	Denise	Biology

Redundancy?

Lossy vs. Lossless Decomp (cntd)

- Redundancy with the FD **D → P**

Project	Employee	Department
Comet	Alice	Physics
Comet	Bob	Astronomy
Genomics	Carl	Biology
Genomics	Denise	Biology

- Proposed decomposition:

Project	Employee
Comet	Alice
Comet	Bob
Genomics	Carl
Genomics	Denise

Department	Project
Physics	Comet
Astronomy	Comet
Biology	Genomics

Lossy vs. Lossless Decomp (cntd)

- Redundancy with the FD **D → P**

Project	Employee
Comet	Alice
Comet	Bob
Genomics	Carl
Genomics	Denise

$\bowtie$

Department	Project
Physics	Comet
Astronomy	Comet
Biology	Genomics

Project	Employee	Department
Comet	Alice	Physics
Comet	Alice	Astronomy
Comet	Bob	Physics
Comet	Bob	Astronomy
....	....	...

Tuples not in original!

Lossy vs. Lossless Decomp (cntd)

- Decomposition attempt #2, for FD **D → P**:

Department	Employee
Physics	Alice
Astronomy	Bob
Biology	Carl
Biology	Denise

$\bowtie$

Department	Project
Physics	Comet
Astronomy	Comet
Biology	Genomics

Project	Employee	Department
Comet	Alice	Physics
Comet	Bob	Astronomy
Genomics	Carl	Biology
Genomics	Denise	Biology

Loss-less Decomposition

- 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:

$$\pi_X(r) \bowtie \pi_Y(r) = r$$

- Decomposition of R into X and Y is lossless with respect to F *if and only if* F^+ contains:

  $X \cap Y \to X$, **or**
  $X \cap Y \to Y$

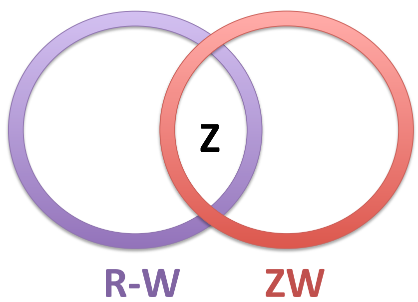
  *In other words, the common attributes **form a key** for X **or** Y*

Corollary: If Z → W holds over R and Z ∩ W is empty, then decomposition of R into **ZW** and **R-W** is loss-less.

- In “Oversees” example, decomposing into {E,P} and {D,P} is *lossy* because the intersection (i.e., Project) is not a key of either resulting relation

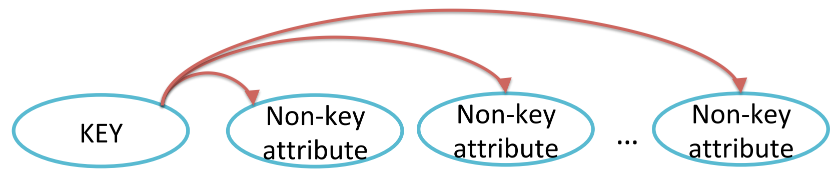
Loss-less Decomposition into BCNF

- Relation R has FDs F. If Z → W in F violates BCNF:
  - decompose R into **R - W** and **ZW** (guaranteed to be loss-less)



Reasoning about BCNF

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



Also recall that relations are *sets* of tuples

Reasoning about BCNF

- If relation R is in BCNF, then each field of a tuple provides a fact ***that cannot be inferred using FDs alone***
- Suppose we are told that the FD X → A holds for this relation:

X	Y	A
x	y1	a
x	y2	?

Possible to guess the value of the missing attribute!

We can infer missing value using the FD… this relation is *not* in BCNF.

Example 1: Is Hourly_Emps in BCNF?

Hourly_Emps

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

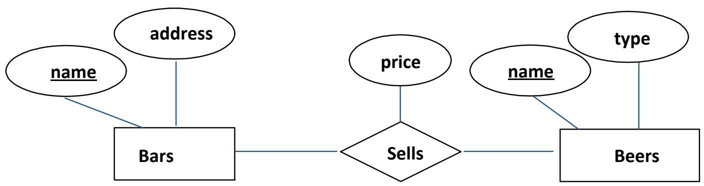
- SNLRWH has FDs

  S → SNLRWH
  R → W

In BCNF??

Example 2: Is Bar_Sells in BCNF?

- Combing Bars and Sells
  Bar_Sells (bar_name, beer_name, address, price)



- **FDs (for just Bar_Sells):**
  bar_name → address
  bar_name, beer_name → price

In BCNF??
(Exercise 2)

Examples: BCNF Decomposition

- Hourly_Emps

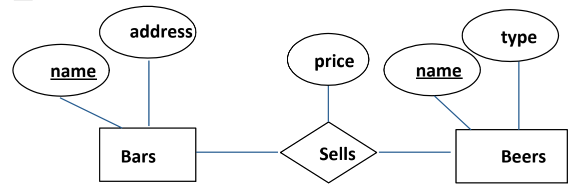
S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Hourly_Emps2

R	W
8	10
5	7

Wages

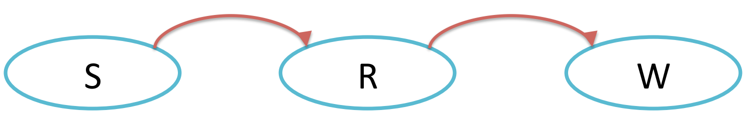
- Bar_sells



Bars(bar_name, address)   Sells(bar_name, beer_name, price)

Transitive Dependencies

- Violating FD involves *attribute(s) depending on non-key attribute(s)*

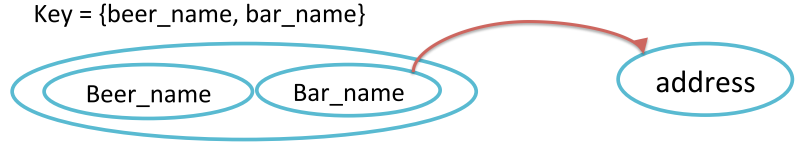


Hourly_Emps violating FD: R → W

Partial Dependencies

- Violating FD involves *attribute(s) depending on attribute(s) that are* ***proper subset of a key***

Key = {beer_name, bar_name}

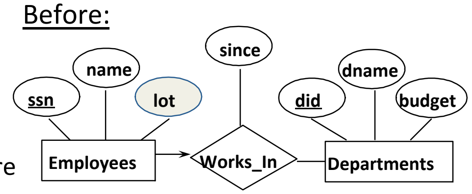


Bar_sells violating FD: bar_name → address

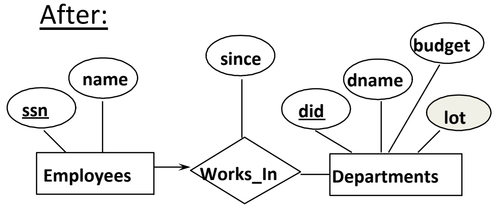
Refining an ER Diagram

- 1st diagram becomes:
  Workers(S,N,L,D,Si)
  Departments(D,M,B)
  – Lots associated w/ workers
- Suppose all workers in a dept are assigned the same lot: D → L
- Redundancy; fixed by:
  Workers2(S,N,D,Si)
  Dept_Lots(D,L)
  Departments(D,M,B)
- Can fine-tune this:
  Workers2(S,N,D,Si)
  Departments(D,M,B,L)

Before:



After:



Repeated Decomposition

- ***Repeated*** decomposition
  – May be needed to get set of relations that are in **BCNF**
  – Can confirm BCNF for original relation R using only FDs F, but *each decomposed relation* R_i must be checked for violating each [relevant] FD in $\mathbf{F^+}$
- Using *attribute closure* to check decomposed R_i
  – To confirm R_i is in BCNF: for each subset of attributes α in R_i, check that α^+ (under F):
    - Contains no attributes of $R_i - \alpha$, ***or***
    - Contains all attributes of R_i

Exercise 3: BCNF Decomposition

- Candidate key={id, advisorId}
- FD violation? Both!
- Decomposed into three relations:
  – R1 = {id, name, dorm}
  – R2 = {advisorId, advisorName}
  – R3 = {id, advisorId}

An Aside: *Multiple* Candidate Keys

- For relation ***Bars(bar_name, address)***, suppose we knew:
  – bar_name → address
  – address → bar_name

Either attribute could serve as primary key!

- When creating a relation in SQL, use *one* candidate key as the primary key
  – Enforce others using `UNIQUE` key word
  – Commonly used when use ***surrogate key*** as a primary key

Dependency Preservation

- Decomposed example from “Oversees”:
  - E → P (an employee oversees only one project)
  - D → P (a dept works on only one project)
  - E → D (an employee only works with one dept for these projects)

Project	Department
Comet	Physics
Comet	Astronomy
Genomics	Biology

Department	Employee
Physics	Alice
Astronomy	Bob
Biology	Carl
Biology	Denise

- How can we check **E → P** ??
  (an employee oversees only one project)

Dependency Preserving Decomposition

- **Dependency preserving decomposition** (Intuition):
  - If R is decomposed into X, Y and Z, *and* we enforce the FDs that hold individually on X, Y, and Z

    → then all FDs that were given to hold on R must also hold

- The **projection of F on attribute set X** (denoted F_X):
  - The set of FDs **U → V** in **F^+** (*closure of F , not just F!*) such that all of the attributes on both sides of the FD **are in X**
  - *That is: U and V are subsets of X*

Dependency Preserving Decompositions (Contd.)

- Decomposition of R into X and Y is ***dependency preserving*** if $(F_X \cup F_Y)^+ = F^+$
  - i.e., we can check FDs on X and Y independently

- “Oversees” example, continued:
  - X = {Employee, Department}, F_X = { E→ D }
  - Y = {Project, Department}, F_Y = { D → P }
  - Does $(F_X \cup F_Y)^+$ include E→ P?

  YES! (transitive property)

Exercise 4: Movie showings

a. {movie, city} and {movie, theater}

b. After decomposing on Theater → city, can’t preserve the FD movie, city→theater

Movie showings: decomposition issue

- Showings (movie, theater, city)
- FDs:
  - movie, city → theater
  - theater → city

  implied:
  movie, theater → city
- Decompose on theater → city:

Theater	City
ArcLight	Pasadena
iPic	Pasadena

⋈

Theater	Movie
ArcLight	The Martian
iPic	The Martian

Theater	City	Movie
ArcLight	Pasadena	The Martian
iPic	Pasadena	The Martian

Above decomposition could allow this to happen!

Violates FD
movie, city → theater

Third Normal Form (3NF)

- Definition: **for all X → A in F⁺**
  - A ∈ X (called a trivial FD), or
  - X is a superkey for R, **OR**
  - **A is a *part of some candidate key* for R**
- Allows FDs like non-key → partial key

A is called a *prime* attribute

- 3NF but not BCNF?
  - have overlapping composite candidate keys

Always possible to get a loss-less, dependency-preserving decomposition into 3NF!

(may contain redundancy)

Alternate Formulation of 3NF & BCNF

BCNF change

*Every ~~non-key~~ attribute must describe **a fact** about "**the key**,*
***the whole key**,*
*and **nothing but the key**,*
so help me Codd"

(1st Normal Form)
(2nd Normal Form)
(3rd Normal Form)

- Normal forms increasingly restrictive
  - 1st NF ⊃ 2nd NF ⊃ 3rd NF ⊃ Boyce-Codd NF