Discussion 9

DB Design

Announcements

Vitamin 9 (DB Design) due **Monday, March 25 at 11:59pm**

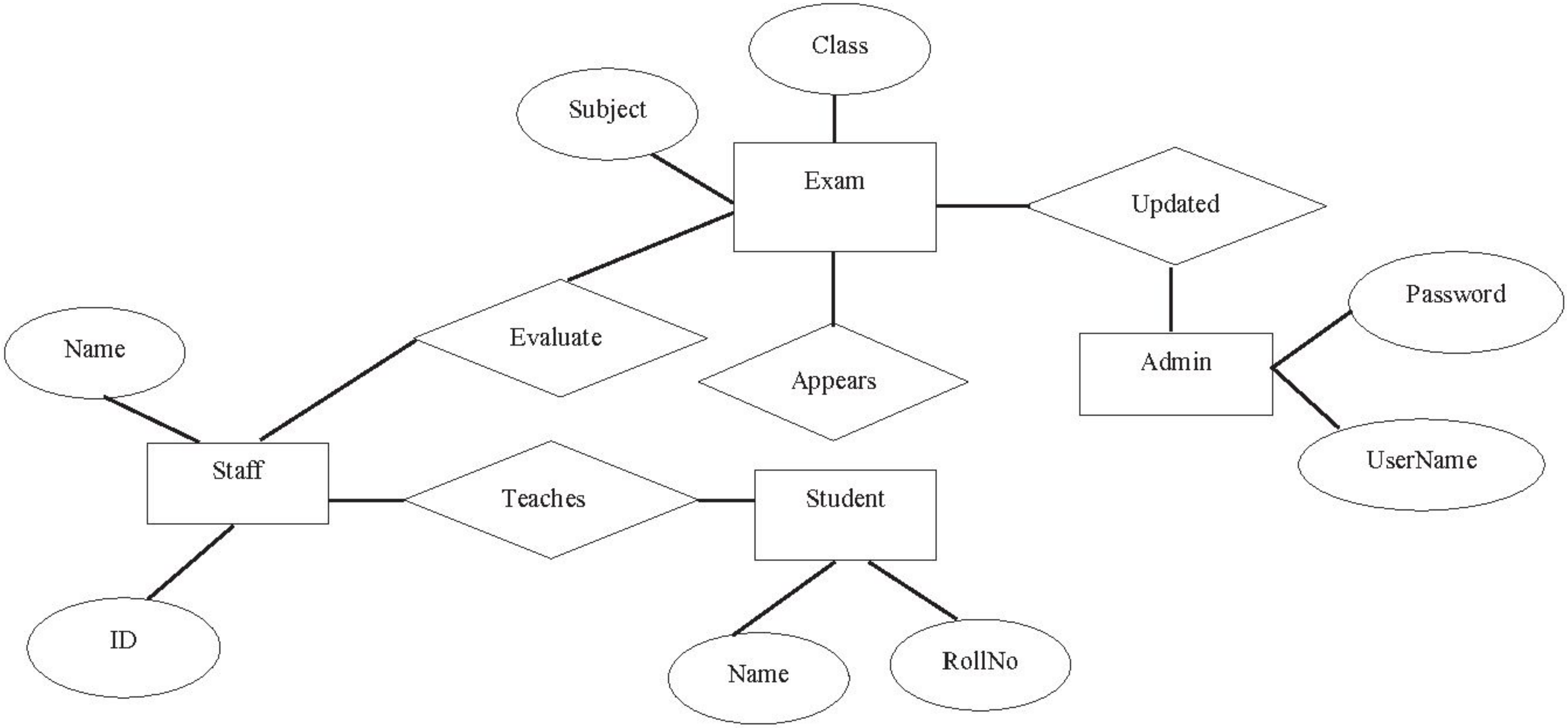
Midterm 2 - Thursday 4/4 from **7PM to 9PM**

Fill out this form for MT 2 conflicts

ER Diagrams

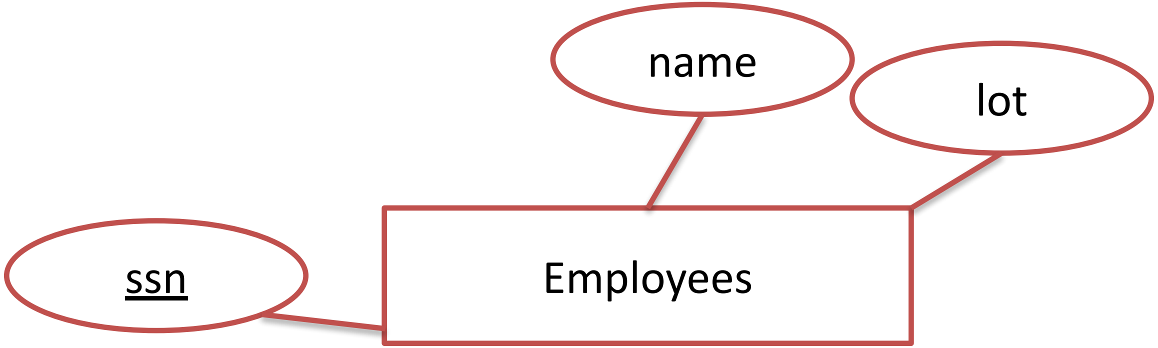
ER Intro

- Today we’ll focus on how to design DB schemas rather than how dbs are actually built
- Production DBs have a lot of tables with complicated relationships
- ER diagrams help design these schemas and document them


Class
Subject
Exam
Updated
Password
Name
Evaluate
Appears
Admin
UserName
Staff
Teaches
Student
ID
Name
RollNo

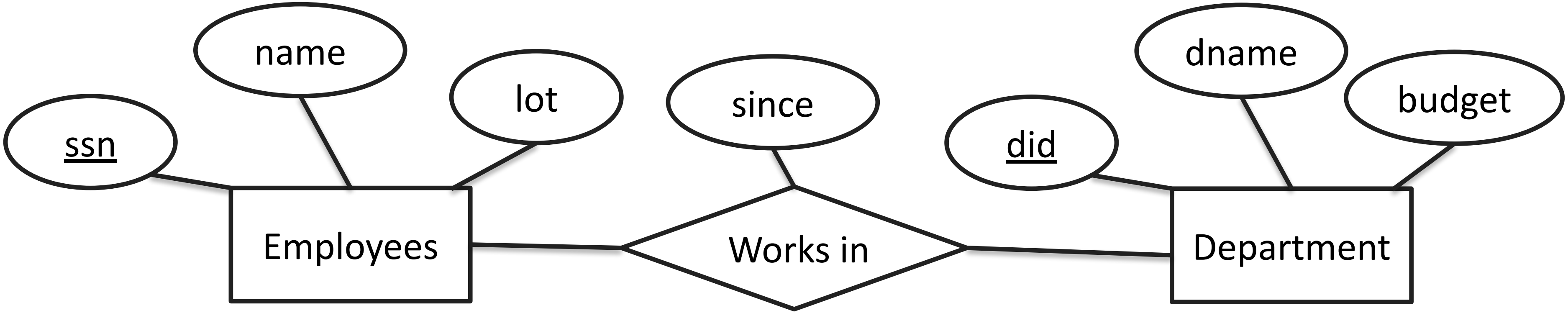
ER Diagrams: Entities

- **Entities** are real-world objects described with **attributes**
- An **entity set** is a collection of the same type of entities
  - All entities in an entity set have the same attributes
  - All entities set have a **key** (underlined)
  - Box around entity set, ellipse around attributes



ER Diagrams: Relationships

- A **relationship** is an association between 2+ entities
  - May be further described with attributes
- A **relationship set** is a collection of the same type of relationships (represented with a diamond)

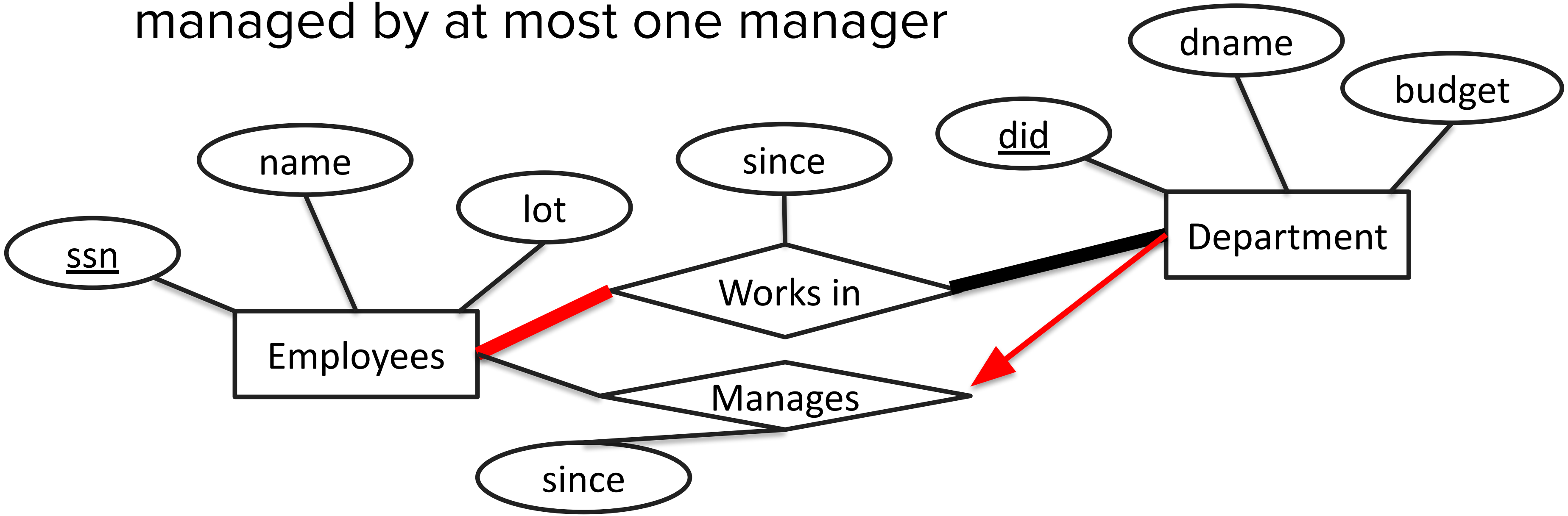


ER Diagrams: Constraints

- Key constraint 
  - at most one
- Participation constraint 
  - at least one
- Key constraint with total participation 
  - exactly one
- Non-key partial participation
  - 0 or more (no restrictions)

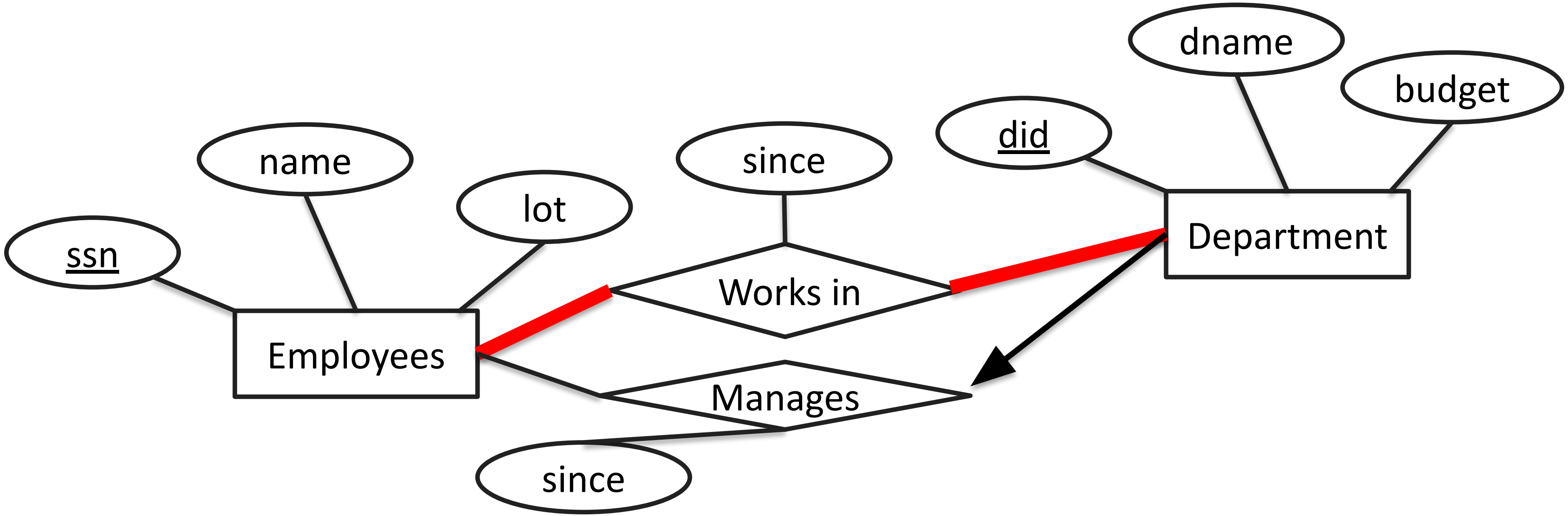
ER Diagrams: Constraints

- **participation constraint:** must be in relationship at least once - every employee must work in a department
- **key constraint**: at most once - every department may be managed by at most one manager



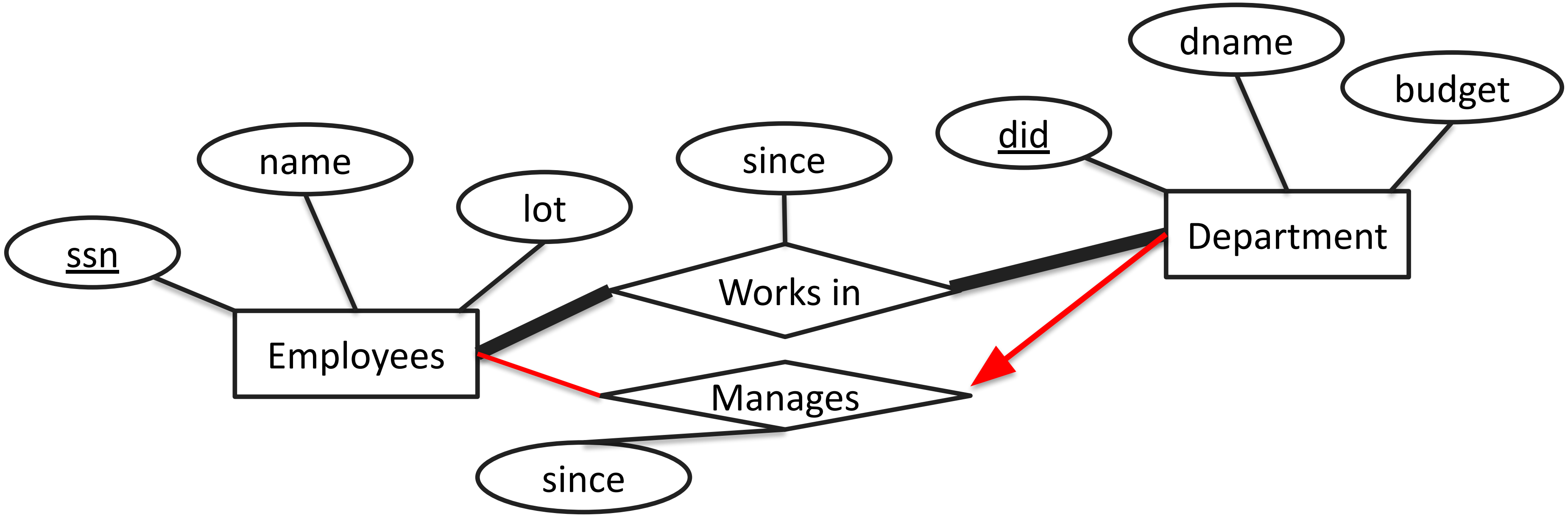
ER Diagrams: Constraints

- The Works In relationship is a **many-to-many** relationship
  - An employee may work in many departments
  - A department may have many employees



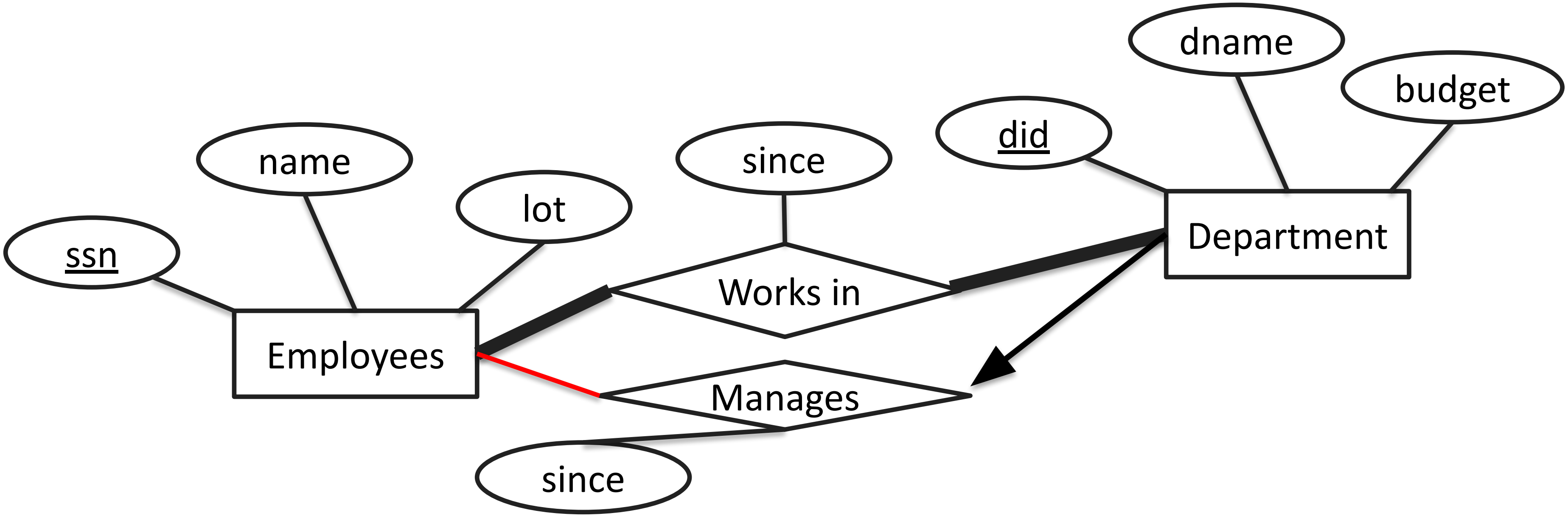
ER Diagrams: Constraints

- The Manages relationship is **1-to-many** (or **many-to-1**)
  - 1-to-many: An employee may manage many depts
  - many-to-1: A dept may be managed by at most 1 employee



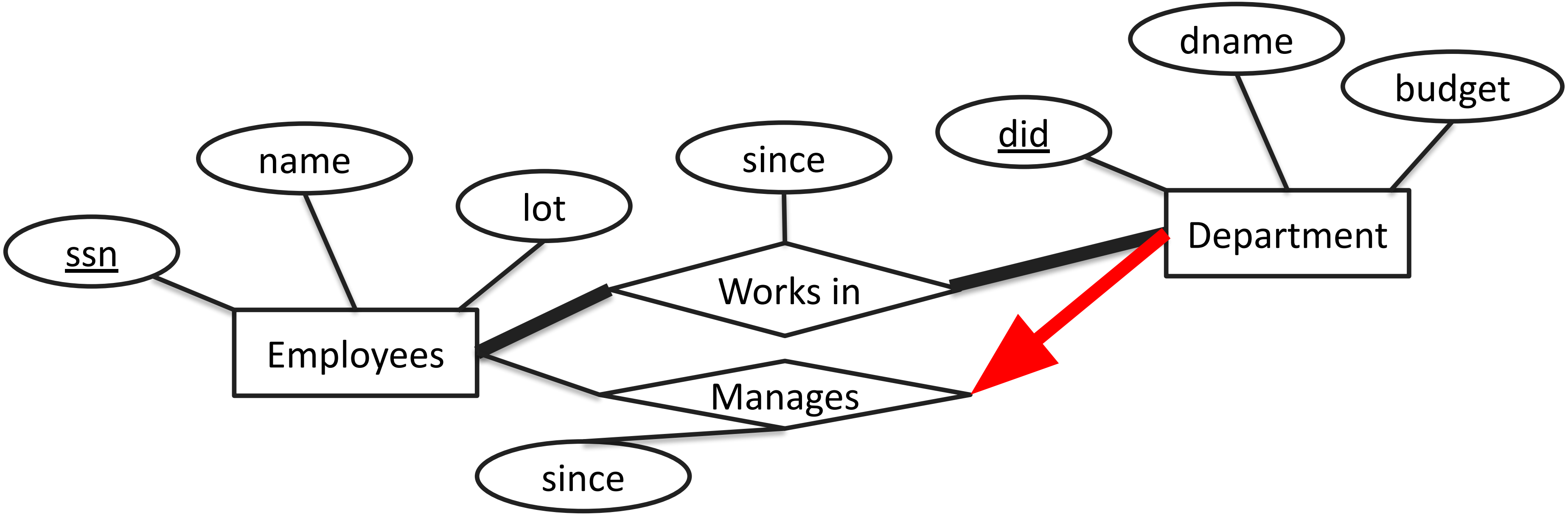
ER Diagrams: Constraints

- The Manages relationship has **optional participation** for employees
  - An employee does not have to manage a department



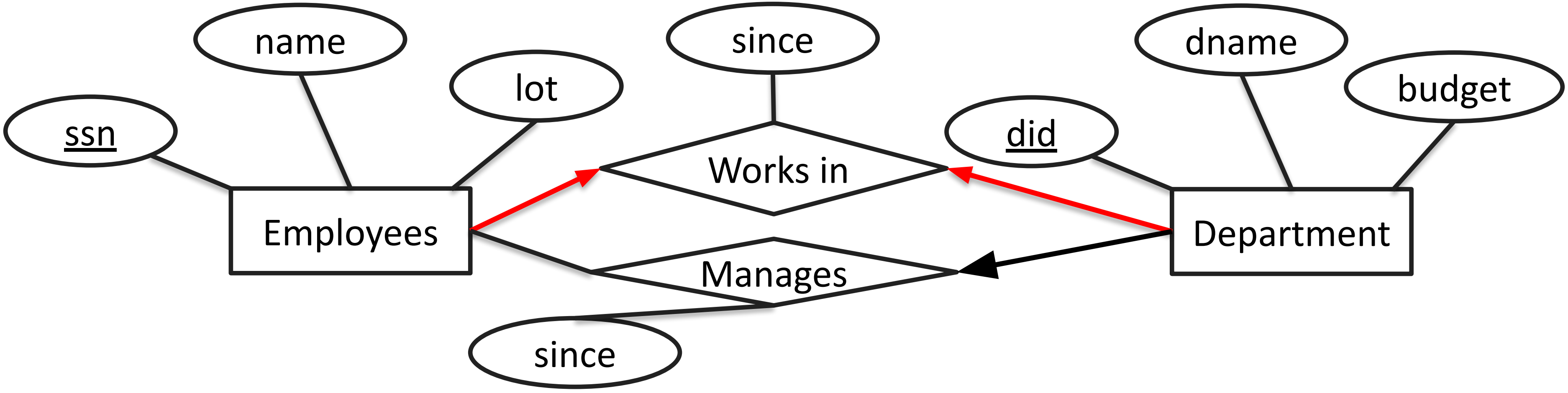
ER Diagrams: Constraints

- What if we change this arrow to be bolded instead?
- The Manages relationship now has a **key constraint** with **total participation** for Department
  - A department must be managed by exactly one manager



ER Diagrams: Constraints

- What if we change these lines to arrows instead?
- The Works in relationship is now a **1-to-1** relationship
  - A department can have at most one employee
  - An employee can work in at most one department

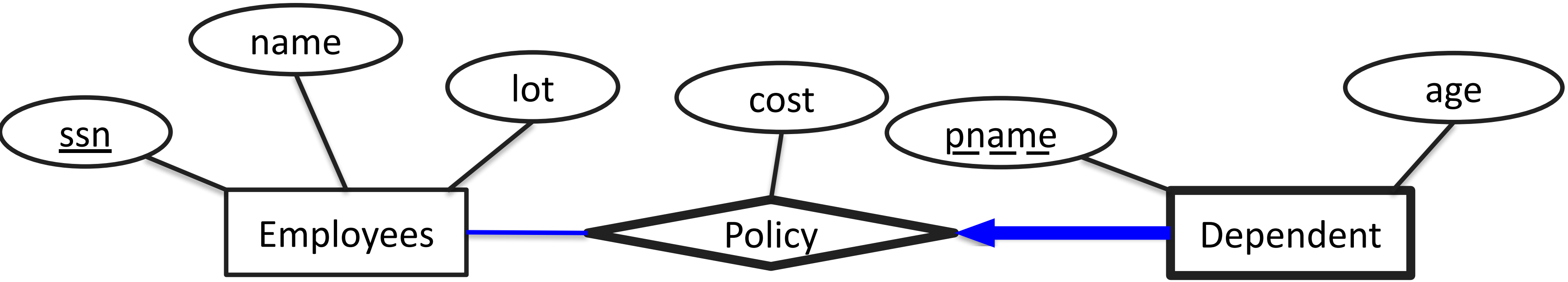


ER Diagrams: Weak Entities

- A **weak entity** is an entity that can be identified uniquely *only* with the key of *another* entity
  - Weak entities have a **partial key** (dashed underline), which identifies the entity when combined with owner entity's key
  - Must be a one-to-many relationship (1 owner entity, many weak entities), with total participation

ER Diagrams: Weak Entities

- A **weak entity** is an entity that can be identified uniquely *only* with the key of *another* entity (**owner entity**)



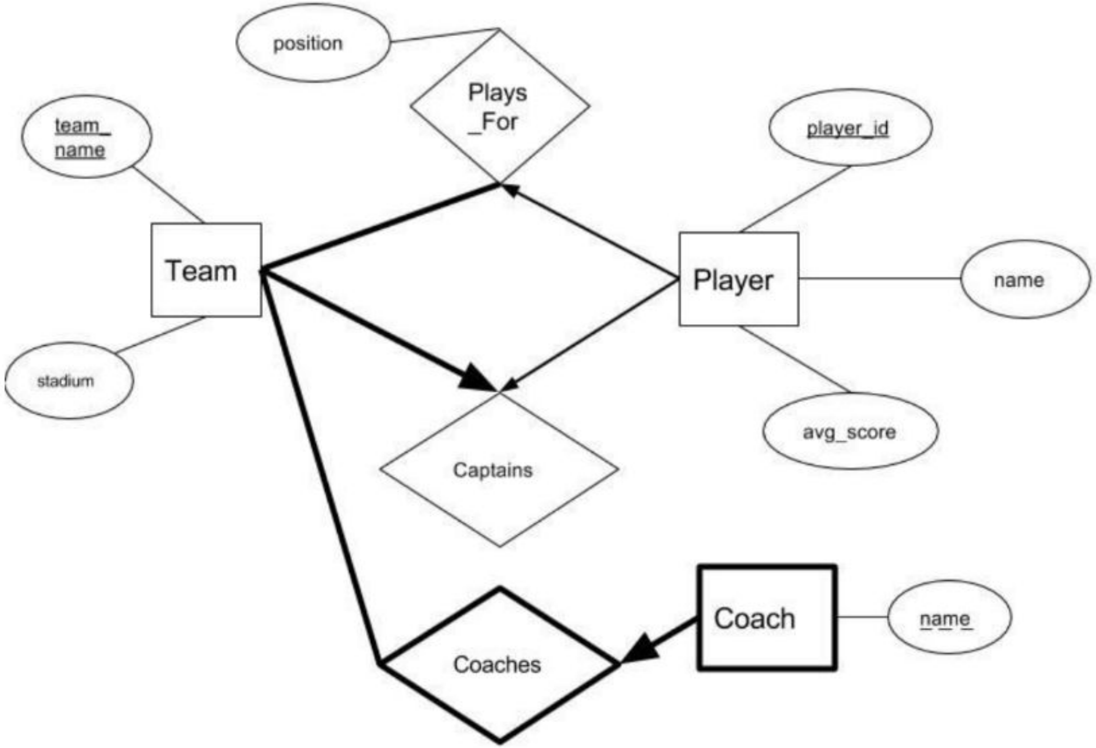
Worksheet: ER Diagrams

Worksheet: ER Diagram

We want to store sports teams and their players in our database. Draw an ER diagram corresponding to data given below:

- Every Team in our database will have a unique team_name and a stadium where they play their games.
- Each Coach has a name.
- Each Player will have a unique player_id, a name and an average score.
- Our database will contain who Plays_For which team and also the "position" that the player plays in. We also need to store who Captains a team, and who Coaches a team.
- Every Team needs players, and needs exactly one captain.
- Each Player can be on at most one team, but may currently be a free agent and not on any team
- Each team needs coaches and may have many.
- A Coach is uniquely identified by which team they coach.

Worksheet: ER Diagram



Functional Dependencies

FDs Intro

- In some data sets, if you already know a set of columns, you can use that information to infer the other columns

- Example: imagine that you have birthday and age columns in a table. Birthday uniquely determines age

- These relationships are called functional dependencies. We want to use them to eliminate redundancy

Functional Dependencies

Students have merit-based pay; their Rating determines their Wage. **Wage depends on Rating.**

SID (S)	Name (N)	Rating (R)	Wage (W)	Hours (H)
0001	Terri	8	15	40
0002	Garrett	5	10	30
0003	Lorene	5	10	30
0004	Heather	8	15	32
0005	Marjorie	2	7	30
0006	Victor	8	15	40
...	...	...	...	...

Functional Dependencies

Problem 1: Redundancy

SID (S)	Name (N)	Rating (R)	Wage (W)	Hours (H)
0001	Terri	8	15	40
0002	Garrett	5	10	30
0003	Lorene	5	10	30
0004	Heather	8	15	32
0005	Marjorie	2	7	30
0006	Victor	8	15	40
…	…	…	…	…

Functional Dependencies

Problem 2: Insert/delete/update anomalies

SID (S)	Name (N)	Rating (R)	Wage (W)	Hours (H)
0001	Terri	8	15	40
0002	Garrett	5	10	30
0003	Lorene	5	10	30
0004	Heather	8	15	32
0005	Marjorie	2	7	30
0006	Victor	8	15	40
...	...	...	...	...

Functional Dependencies: Anomalies

- **Update anomaly**: if we change a wage for one person, we have to change it for everyone
- **Insert anomaly**: if we want to insert a person with rating 10, we have to figure out the wage associated with it
- **Delete anomaly**: if we delete all employees with rating 8, we no longer know the wage value corresponding to rating 8 (what if we add a rating 8 person later?)

Functional Dependencies: Schema Decomposition

Solution: Move rating and wage information to a separate table

SID (S)	Name (N)	Rating (R)	Hours (H)
0001	Terri	8	40
0002	Garrett	5	30
0003	Lorene	5	30
0004	Heather	8	32
0005	Marjorie	2	30
0006	Victor	8	40
...	...	...	...

R → W

Rating (R)	Wage (W)
8	15
2	7
5	10

Functional Dependencies

- **functional dependency**: **X → Y** (X *determines* Y)
  - X, Y are *sets* of attributes
  - if attributes in X match, then attributes in Y must match
- **superkey**: X is a superkey of R if X → [all attributes of R]
- **candidate key**: a set of keys that determines all columns in a relation and no columns that can be removed and still be a superkey

Functional Dependencies: Inference Rules

- Armstrong's Axioms
  - Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$
  - Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$
    - $XZ \rightarrow YZ$ does NOT imply $X \rightarrow Y$
  - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  - $XZ \rightarrow Y$ does NOT imply $X \rightarrow Y$ and $Z \rightarrow Y$

Functional Dependencies: Closure

- The **closure** of a set of FDs F is F^+
  - set of all FDs implied by F
  - hard to find, exponential in # of attributes, so we use attribute closure instead

- The **attribute closure** of an attribute X given a set of FDs is X^+
  - set of all attributes A such that $X \rightarrow A$ is in F^+ (all attributes that can be determined by just X)

Functional Dependencies: Closure

- The **attribute closure** of an attribute X given a set of FDs is X^+
  - set of all attributes A such that $X \rightarrow A$ is in F^+ (all attributes that can be determined by just X)
  - Algorithm:
    - Closure = X;
    - Repeat until there is no change
      - If there is an FD $U \rightarrow V$ in F s.t. $U \subseteq$ closure,
        - set closure = closure $\cup$ V

Worksheet: FDs

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \to Y$, $XY \to YZ$, and $Y \to X$ where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	**y**	**z**
1	2	0
1	2	1
1	3	0
2	3	0

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \to Y$, $XY \to YZ$, and $Y \to X$ where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	y	z
1	2	0
1	2	1
1	3	0
2	3	0

None.

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \rightarrow Y$, $XY \rightarrow YZ$, and $Y \rightarrow X$ where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	**y**	**z**
1	2	1
1	3	1
2	3	0

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \rightarrow Y$, $XY \rightarrow YZ$, and $Y \rightarrow X$ where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	y	z
1	2	1
1	3	1
2	3	0

$XY \rightarrow YZ$.

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \rightarrow Y$, $XY \rightarrow YZ$, and $Y \rightarrow X$ where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	**y**	**z**
1	3	1
2	3	0

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \rightarrow Y$, $XY \rightarrow YZ$, and $Y \rightarrow X$ where $X = \{x\}$, $Y = \{y\}$, and $Z = \{z\}$. For each of the following relations, indicate which functional dependencies it might satisfy.

x	**y**	**z**
1	3	1
2	3	0

$X \rightarrow Y$, $XY \rightarrow YZ$.

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \rightarrow Y$, $XY \rightarrow YZ$, and $Y \rightarrow X$ where X = {x}, Y = {y}, and Z = {z}. For each of the following relations, indicate which functional dependencies it might satisfy.

x	y	z
1	3	1

Worksheet: FD #1

Consider a relation R(x, y, z) and the functional dependencies $X \rightarrow Y$, $XY \rightarrow YZ$, and $Y \rightarrow X$ where $X = \{x\}$, $Y = \{y\}$, and $Z = \{z\}$. For each of the following relations, indicate which functional dependencies it might satisfy.

x	y	z
1	3	1

$X \rightarrow Y$, $XY \rightarrow YZ$, $Y \rightarrow X$.

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

A^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

A^+

A → B (AB)

AB → AC (ABC)

BC → BD (ABCD)

ABCD

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

B^+, C^+, D^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

B^+, C^+, D^+

B, C, D

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

AB^+, AC^+, AD^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

AB^+, AC^+, AD^+

ABCD

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

BC^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

BC^+

BC → BD (BCD)

BCD

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

BD^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

BD^+

BD

Worksheet: FD #2

Consider the set F = {A ➜ B, AB ➜ AC, BC ➜ BD, DA ➜ C} of functional dependencies. Compute the following attribute closures.

CD^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

CD^+

CD

Worksheet: FD #2

Consider the set F = {A ➔ B, AB ➔ AC, BC ➔ BD, DA ➔ C} of functional dependencies. Compute the following attribute closures.

BCD^+

Worksheet: FD #2

Consider the set F = {A → B, AB → AC, BC → BD, DA → C} of functional dependencies. Compute the following attribute closures.

BCD^+

BCD

Worksheet: FD #3

Consider again the set F = {A ➔ B, AB ➔ AC, BC ➔ BD, DA ➔ C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

A

Worksheet: FD #3

Consider again the set F = {A → B, AB → AC, BC → BD, DA → C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

A

A^+ = ABCD

candidate key because A^+ is a minimal set of keys to cover all symbols in F

Worksheet: FD #3

Consider again the set F = {A → B, AB → AC, BC → BD, DA → C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

B, C, D

Worksheet: FD #3

Consider again the set F = {A → B, AB → AC, BC → BD, DA → C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

B, C, D

$B^+ = B$, $C^+ = C$, $D^+ = D$

neither because none of them cover all symbols in F

Worksheet: FD #3

Consider again the set F = {A ➜ B, AB ➜ AC, BC ➜ BD, DA ➜ C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

AB, AC, AD

Worksheet: FD #3

Consider again the set F = {A → B, AB → AC, BC → BD, DA → C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

AB, AC, AD

$AB^+ = AC^+ = AD^+ = ABCD$

superkeys because A is a candidate key, so any more symbols added to A is not the minimal set of symbols to cover all symbols in F and thus, a superkey

Worksheet: FD #3

Consider again the set F = {A ➔ B, AB ➔ AC, BC ➔ BD, DA ➔ C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

BC

Worksheet: FD #3

Consider again the set F = {A → B, AB → AC, BC → BD, DA → C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

BC

$BC^+ = BCD$

neither

Worksheet: FD #3

Consider again the set F = {A ➔ B, AB ➔ AC, BC ➔ BD, DA ➔ C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

BD, CD, BCD

Worksheet: FD #3

Consider again the set F = {A → B, AB → AC, BC → BD, DA → C}. Indicate whether the following sets of attributes are candidate keys, superkeys (but not candidate keys), or neither.

BD, CD, BCD

$BD^+ = BD$, $CD^+ = CD$, $BCD^+ = BCD$

neither

Boyce-Codd Normal Form (BCNF)

Normalization: Boyce-Codd Normal Form (BCNF)

- R is in BCNF if:
  - for every FD $X \rightarrow A$ that holds over R,
    - either $A \subseteq X$ OR X is a superkey
- No redundancy in BCNF
  - every field of every tuple contains some information that *cannot* be inferred from the FDs

Normalization: Boyce-Codd Normal Form (BCNF)

- We can decompose a relation R that is not in BCNF into multiple relations that are in BCNF

- For each FD $X \rightarrow Y$ in F^+:
  - If $X \rightarrow Y$ violates BCNF:
    - Decompose R into $(R - X^+) \cup X$ and X^+

- Final result depends on order of decomposition

BCNF Decomposition Example

Decompose R = ABCDE into BCNF, given the functional dependency set:

F = {A → E, C → E, B → CD, B → A}.

BCNF Decomposition Example

Decompose R = ABCDE into BCNF, given the functional dependency set:

F = {A → E, C → E, B → CD, B → A}.

Relations: {ABCDE} → {AE, ABCD}

- A is not a superkey. A^+ = AE
- A & E are in a relation together, so we split A+ and (R - A+) U A = ABCD out from the offending relation.

BCNF Decomposition Example

Decompose R = ABCDE into BCNF, given the functional dependency set:

F = {A ➔ E, C ➔ E, B ➔ CD, B ➔ A}.

Relations: {AE, ABCD}

- C & E are not both in a relation, so nothing happens.

BCNF Decomposition Example

Decompose R = ABCDE into BCNF, given the functional dependency set:

F = {A → E, C → E, B → CD, B → A}.

Relations: {AE, ABCD}

- B is a superkey for {ABCD}. B^+= ABCDE (from B → CD, C → E, B → A)
- B, C, & D are in a relation together, because B is a superkey, we don't split ABCD.

BCNF Decomposition Example

Decompose R = ABCDE into BCNF, given the functional dependency set:

F = {A → E, C → E, B → CD, B → A}.

Relations: {AE, ABCD}

- B is a superkey for {ABCD}.
- B & A are in a relation together, because B is a superkey, we don't split ABCD.

Decomposition Problem: Lossiness

- **lossiness**: we may not be able to reconstruct the original relation
  - doesn’t actually lose data, it generates bad data
- Decompose R into X and Y. Decomposition is lossless iff f^+ contains:
  - X INTERSECT Y ➔ X or
  - X INTERSECT Y ➔ Y
- BCNF is always lossless (good)

Decomposition Lossiness Example

Given R = ABC with F = B→C, if we decompose it into two relations AB and AC, and then join AB and AC on A, we aren't going to have the same rows as the original relation.

R

A	B	C
a1	b1	c1
a1	b2	c2

→

X

A	B
a1	b1
a1	b2

Y

A	C
a1	c1
a1	c2

→

X ⋈ Y

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

Dependency Preserving Decompositions

- **dependency preserving:** if we can enforce F+ individually on each table and this in turn enforces the FDs on the entire database

- Formalism: dependency preserving iff $(F_x \cup F_y)^+ = F^+$ where F_x are the FDs we can enforce just in relation X
  - For example: imagine we decomposed R = ABC into X=AB, Y=BC. If F: {A → B, A → C} this is not dependency preserving because we can't enforce the dependency A → C on either relation

- BCNF is not necessarily dependency preserving

Worksheet: Normal Forms

Worksheet: Normal Forms #1

Decompose R = ABCDEFG into BCNF, given the functional dependency set:

$F = \{AB \to CD, C \to EF, G \to A, G \to F, CE \to F\}$.

Worksheet: Normal Forms #1

Decompose R = ABCDEFG into BCNF, given the functional dependency set:

F = {AB → CD, C → EF, G → A, G → F, CE → F}.

Relations: {ABCDEFG} → {ABCDEF, ABG}

- AB^+ = ABCDEF, not a superkey for {ABCDEFG}
- (R - AB^+) U AB = ABG

Worksheet: Normal Forms #1

Decompose R = ABCDEFG into BCNF, given the functional dependency set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

Relations: {ABCDEF, ABG} → {ABCD, CEF, ABG}

- C^+= CEF, not a superkey for {ABCDEF}
- (ABCDEF - C^+) U C = ABCD

Worksheet: Normal Forms #1

Decompose R = ABCDEFG into BCNF, given the functional dependency set:

F = {AB → CD, C → EF, G → A, G → F, CE → F}.

Relations: {ABCD, CEF, ABG} ➜ {ABCD, CEF, AG, BG}

- G^+ = GAF, not a superkey for {ABG}
- (ABG - G^+) U G = BG

Worksheet: Normal Forms #1

Decompose R = ABCDEFG into BCNF, given the functional dependency set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

Relations: {ABCD, CEF, AG, BG}

- There aren't any relations with G&F both in them

Worksheet: Normal Forms #1

Decompose R = ABCDEFG into BCNF, given the functional dependency set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

Relations: {ABCD, CEF, AG, BG}

- CE → F does not violate BCNF since CE is a superkey for CEF

Worksheet (3.2): Normal Forms

Decompose R = ABCDEFG into BCNF, given the functional dependency set:
F = {AB → CD, C → EF, G → A, G → F, CE → F}.

Is the above decomposition lossless?

Yes, because BCNF is **always** lossless.

Worksheet (3.3): Normal Forms

Does the above decomposition preserve dependencies? Why/why not?

No, G → F is not represented in the closure of the union of each subrelation’s dependencies.

Attendance Link

https://cs186berkeley.net/attendance