CS186 - Week 13

Functional Dependencies
and Schema Refinement

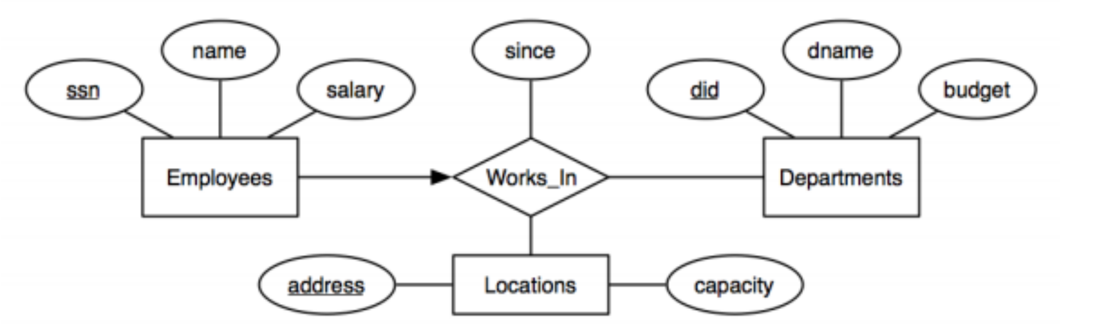
Today

- ER Modeling
    - Quick review
    - How to translate a model to SQL
- Functional Dependencies and Schema Refinement
    - Boyce-Codd Normal Form
    - Decomposition
        - Lossless? Dependency preserving?

Review: ER Diagrams

- Entity: “Thing”
  - Attribute: Properties of entities
  - Primary key underlined
- Relationships: How things interact
  - More meaningful by capturing constraints
- Ternary relations
- Weak Entities
  - Idea of ownership
  - Partial key

	Partial Participation	Total Participation
Non-Key	0 or more	1 or more
Key	0 or 1	Exactly 1



Translating ER to SQL

- Create one table per entity
- Create one table per relation (“grouping” entities together)
- DDL in SQL…

CREATE TABLE *table_name*

({ *column_name data_type*

[DEFAULT *default_expr*]

[*column_constraint* [, ...]] | *table_constraint* }

[, ...]) ;

Common Constraints: NOT NULL, DEFAULT, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK

– gsi_id NOT NULL, PRIMARY KEY(gsi_id), FOREIGN KEY(gsi_id) REFERENCES gsis

Common Data Types: CHAR(n), INTEGER, REAL, DATE

– address CHAR(200), start_date DATE, did INTEGER

Questions?

Do “ER to SQL” problem on worksheet.

ER to SQL Solution

```
CREATE TABLE Employees(
 ssn CHAR(11),
 name CHAR(100),
 salary REAL,
 PRIMARY KEY(ssn));

CREATE TABLE Departments(
 did INTEGER,
 dname CHAR(100),
 budget REAL,
 PRIMARY KEY(did));

CREATE TABLE Locations(
 address CHAR(200),
 capacity INTEGER,
 PRIMARY KEY(address));
```

```
CREATE TABLE Works_In(
 ssn CHAR(11),
 did INTEGER,
 address CHAR(200),
 since DATE,
 PRIMARY KEY(ssn),
 FOREIGN KEY(ssn) REFERENCES Employees,
 FOREIGN KEY(did) REFERENCES Departments,
 FOREIGN KEY(address) REFERENCES
Locations);
```

Functional Dependencies

- X → Y reads "X determines Y"
  - X and Y are set of attributes
  - Given any two tuples, if X values are the same, Y values <u>must also</u> be the same.
- Key → all attributes of R
- Can be used to detect redundancies and refine the schema

Armstrong's Axioms

- Reflexivity: if $X \supseteq Y$, then $X \rightarrow Y$
  - Example: $A \rightarrow A$, $AB \rightarrow A$
- Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
- Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Union: if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Decomposition: if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- **Common mistake**: if $XA \rightarrow YA$, **cannot infer** $X \rightarrow Y$.
  - Trivial example, $A \rightarrow YA$

Closures

- FD closure: F+
  - Set of all FDs implied by F
  - Includes trivial dependencies
  - Example: $F = \{A \rightarrow B, B \rightarrow C\}$
    - $F+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, A \rightarrow A, A \rightarrow AB, \ldots\}$
  - Expensive to compute
- Attribute closure: X+
  - Given just X, what can we determine?
  - Example: $F = \{A \rightarrow B, B \rightarrow C\}$
    - $A+ = ABC$

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { $B \to CD$, $D \to E$, $B \to A$, $E \to C$, $AD \to B$ }
  - B+

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { $B \rightarrow CD$, $D \rightarrow E$, $B \rightarrow A$, $E \rightarrow C$, $AD \rightarrow B$ }
  - B+ = B

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { $B \rightarrow CD$, $D \rightarrow E$, $B \rightarrow A$, $E \rightarrow C$, $AD \rightarrow B$ }
  - B+ = B

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { $B \rightarrow CD$, $D \rightarrow E$, $B \rightarrow A$, $E \rightarrow C$, $AD \rightarrow B$ }
  - B+ = B

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { B → CD, D → E, B → A, E → C, AD → B }
  - B+ = B = BCD

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { B → CD, D → E, B → A, E → C, AD → B }
  - B+ = B = BCD

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { $B \rightarrow CD$, $D \rightarrow E$, $B \rightarrow A$, $E \rightarrow C$, $AD \rightarrow B$ }
  - B+ = B = BCD = BCDE

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { B $\rightarrow$ CD, D $\rightarrow$ E, B $\rightarrow$ A, E $\rightarrow$ C, AD $\rightarrow$ B }
  - B+ = B = BCD = BCDE

More on Attribute Closures

- In general, how can we determine?
- Algorithm!
  - `Initialize X+ := X`
  - `Repeat until no change:`
    - `If U → V is in F such that U is in X+, add V to X+`
- Example:
  - R = ABCDE,
  - F = { B → CD, D → E, B → A, E → C, AD → B }
  - B+ = B = BCD = BCDE = ABCDE

Questions?

Do worksheet FD problem 1.

FD Problem 1 Solution

1. Consider the **Works_In(S**sn, **L**ot, **D**id, s**I**nce**)** relation from our previous example. If **S** (ssn) is a key for this relationship, what is the functional dependency we can infer from that? Abbreviate the attribute with the bolded/capitalized letter.

$S \rightarrow SLDI$

2. If employees in the same department are given the same parking lot, what additional functional dependency can we infer?

$D \rightarrow L$

Boyce-Codd Normal Form

- Idea: Schema design is hard, can mess up.
  - Can we ensure a reasonable design?
- BCNF ≈ “reasonable schema”
- Definition: Relation R with FDs F is in BCNF if,

for all $X \rightarrow A$ in F+

  - $X \rightarrow A$ is reflexive (called a trivial FD), or
  - X is a superkey for R
- If R in BCNF, then every field of every tuple cannot be inferred from FDs alone

Example

- Twitter: Users send Tweets
- For Users, keep track of Username, Email, BirthYear
- For Tweets, keep track of TweetId, Text, Date
- What are the FDs? How about superkeys?

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
  - ABEG, ABC

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
  - ABEG, ABC

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
  - ABEG, ABC

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
  - ABEG, ABC

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
  - ABEG, ABC

BCNF Decomposition

- Make a bad table good
  - What qualifies as bad?
- If $X \rightarrow A$ is a FD that violates BCNF, then decompose R into **R - A** and **XA**
  - Repeat as necessary
- Example 2: R = ABCEG;

  F = {$AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow G$, $E \rightarrow G$}
  - ABEG, ABC
  - ABE, EG, ABC
  - Done!

Decomposition Properties

Lossless-join: Can we reconstruct R?

- Decomposing R into X and Y is lossless-join iff
  - X or Y contains a key of the other relation.
  - (alternate, equivalent definition in lecture)
- Does BCNF always produce lossless-join?
  - Yes, mostly! We decompose $X \rightarrow A$ into R-A and XA.
  - Won’t hold if A contains some part of X.
- Lossiness is not allowed!

Decomposition Properties

Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC

  $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$

Decomposition Properties

Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC

  $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$

Decomposition Properties

Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC

  $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$

Decomposition Properties

Dependency-preserving: Can we verify all FDs without joins?

- Example: ABE, EG, ABC

  $F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow G, E \rightarrow G\}$
- Oh no! This dependency was not preserved.
- How to fix?
  - Add relation BCG
  - May break BCNF!

Questions?

Worksheet problems 2 and 3

FD Problem 2 Solution

Flight schema

```
Flights(Flight_no, Date, fRom, To, Plane_id), ForeignKey(Plane_id)
Planes(Plane_id, tYpe)
Seat(Seat_no, Plane_id, Legroom), ForeignKey(Plane_id)
```

1. Find the set of functional dependencies.

From the key constraints we get the following functional dependencies:
FD → FDRTP
P → PY
SP → SPL

2. Expand the FDs found above using Armstrong's axioms (you can omit the trivial and non interesting dependencies).

Using decomposition and transitivity (FD → FDRTP , P → PY) we can obtain:
FD → Y
Using decomposition, augmentation and transitivity (FD → FDRTP , SP → SPL) we can obtain:
FDS → L

FD Problem 3 Solution

1. Now consider the attribute set R = ABCDE and the FD set F = {AB → C, A → D, D → E, AC → B}. Compute the attribute closure for each of A, AB, B, and D.

- *A: {A, D, E}*
- *AB: {A, B, C, D, E}*
- *B: {B}*
- *D: {D, E}*

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

AB→CD => decompose ABCDEFG into <u>ABCD</u>, ABEFG
G→A => decompose ABEFG into <u>AG</u>, BEFG
G→F => decompose BEFG into <u>FG</u>, <u>BEG</u>
Final relations: **ABCD, AG, FG, BEG**.

Thank you!