CS 186 Discussion #6

Relational Algebra/Modeling
BCNF

Logistics

- Homework
  - Homework 3 due 10/19
  - Homework 4 + 5 deadlines updated on syllabus
- Midterm
  - Midterm 1 grades released
  - Midterm 2 is coming… 11/09
- Mid-semester evaluations

Relational Algebra

- Represent query execution plan with operators
  - More on query optimization later

Basic Operators

- SELECTION (σ)
- PROJECTION (π)
- CROSS-PRODUCT (x)
- SET-DIFFERENCE ($-$)
- UNION (U)

Basic Operators

- SELECTION (σ) — filter rows
- PROJECTION (π) — filter columns
- CROSS-PRODUCT (x)
- SET-DIFFERENCE (–)
- UNION (U)

Compound Operators

- INTERSECTION ($\cap$)
- JOINs
    - NATURAL JOIN ($\bowtie$)
    - THETA JOIN ($\bowtie_\theta$)
        - θ: R.sid = S.sid

Relational Algebra Worksheet

```
Songs(song_id, song_name, album_id, weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Albums(album_id, album_name, artist_id, year_released, genre)
```

- 1. Find the name of the artists who have albums with a genre of either ‘pop’ or ‘rock’.

```
SELECT artist_name
FROM Artists, Albums
WHERE Artists.artist_id = Albums.artist_id
AND (genre = ‘pop’ OR genre = ‘rock’);
```

Relational Algebra Worksheet

```
Songs(song_id, song_name, album_id, weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Albums(album_id, album_name, artist_id, year_released, genre)
```

- 1. Find the name of the artists who have albums with a genre of either ‘pop’ or ‘rock’.

SELECT artist_name
FROM Artists, Albums
WHERE Artists.artist_id = Albums.artist_id
AND (genre = ‘pop’ OR genre = ‘rock’);

$\pi_{\text{artist\_name}} ((\sigma_{\text{genre = ‘pop’} \vee \text{genre = ‘rock’}} \text{Albums}) \bowtie \text{Artists})$

Relational Algebra Worksheet

```
Songs(song_id, song_name, album_id, weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Albums(album_id, album_name, artist_id, year_released, genre)
```

- 2. Find the name of the artists who have albums of genres ‘pop’ AND ‘rock’.

SELECT artist_name
FROM Artists, Albums A1, Albums A2
WHERE Artists.artist_id = A1.artist_id
AND Artists.artist_id = A2.artist_id
AND A1.genre = ‘pop’ AND A2.genre = ‘rock’);

Relational Algebra Worksheet

```
Songs(song_id, song_name, album_id, weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Albums(album_id, album_name, artist_id, year_released, genre)
```

- 2. Find the name of the artists who have albums of genres ‘pop’ AND ‘rock’.

$$\pi_{artist\_name}((\sigma_{genre = \text{'pop'}} Albums) \bowtie Artists)$$
$$\cap$$
$$\pi_{artist\_name}((\sigma_{genre = \text{'rock'}} Albums) \bowtie Artists)$$

Relational Algebra Worksheet

```
Songs(song_id, song_name, album_id, weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Albums(album_id, album_name, artist_id, year_released, genre)
```

- 5. Find the names of all artists who do not have any albums.

```
SELECT artist_name
FROM Artists
WHERE artist_id NOT IN
(SELECT artist_id
FROM Albums)
```

Relational Algebra Worksheet

`Songs(song_id, song_name, album_id, weeks_in_top_40)`

`Artists(artist_id, artist_name, first_year_active)`

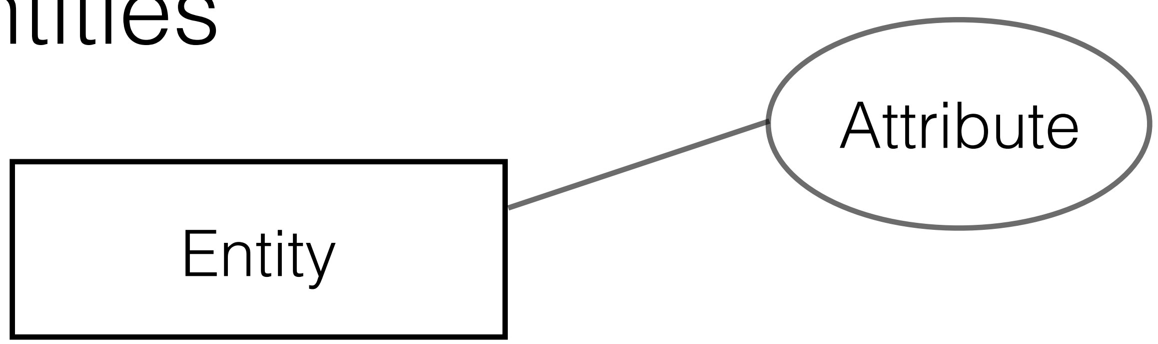
`Albums(album_id, album_name, artist_id, year_released, genre)`

- 5. Find the names of all artists who do not have any albums.

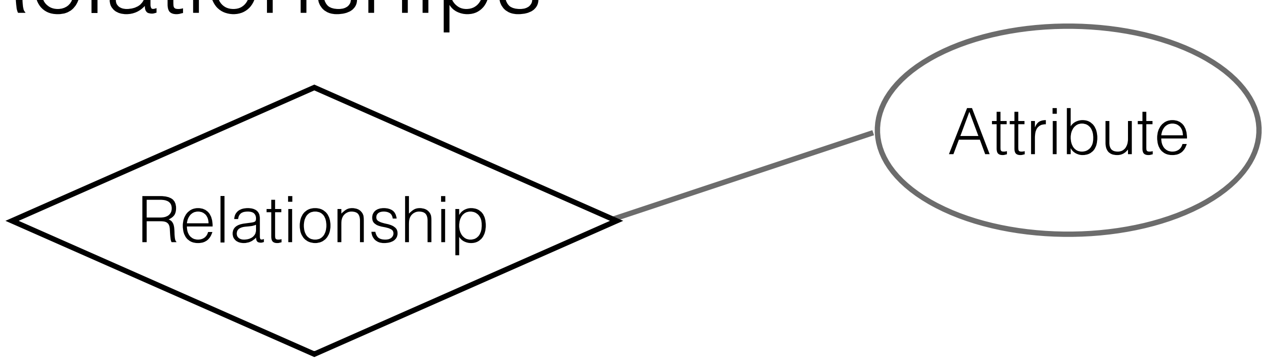
$$\pi_{\text{artist\_name}}\,(\text{Artists} \bowtie (\pi_{\text{artist\_id}}\,\text{Artists} - \pi_{\text{artist\_id}}\,\text{Albums}))$$

Relational Modeling

- Entity-Relationship model
- Entities



  - Weak Entities
- Relationships



Constraints

- Key Constraints
    - at *most* one
- Participation Constraints
    - at *least* one
- “Key constraint with total participation”
    - *exactly* one
- One-to-many? Many-to-one? One-to-one?

Constraints

- Key Constraints
  - at *most* one <- specifies many-to-one/one-to-many/one-to-one
- Participation Constraints
  - at *least* one
- “Key constraint with total participation”
  - *exactly* one

ER Diagrams Worksheet

Functional Dependencies

- X → Y
  - *X determines Y*
  - For every pair of tuples in R, if X is the same, Y must be the same.
- K → [all attributes of R]
  - K is a superkey! Why?
    - “No two distinct tuples can have the same values in all key fields”
  - What about primary keys?

Rules of Inference

- 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 $Z \rightarrow Y$

Closure

- Given a set of FDs F, finding the closure F^+ is <u>extremely</u> difficult; all implicit FDs are in F^+!
- Instead, lets compute <u>Attribute Closures</u>:
  - Given an attribute X,
    - "X^+ = All attributes A such that $X \to A$ is in F^+"
    - X^+ = All attributes that X determines (given F)

FD Worksheet

Flights(**F**light_no, **D**ate, f**R**om, **T**o, **P**lane_id), ForeignKey(**P**lane_id)

Planes(**P**lane_id, t**Y**pe)

Seat(**S**eat_no, **P**lane_id, **L**egroom), ForeignKey(**P**lane_id)

What are the functional dependencies?

FD Worksheet

`Flights(Flight_no, Date, fRom, To, Plane_id), ForeignKey(Plane_id)`

`Planes(Plane_id, tYpe)`

`Seat(Seat_no, Plane_id, Legroom), ForeignKey(Plane_id)`

What are the functional dependencies?

- FD → RTP
- P → Y
- SP → L

FD Worksheet

$F = \{AB \rightarrow C, A \rightarrow D, D \rightarrow E, AC \rightarrow B\}$

What are the attribute closures?

A:

AB:

B:

D:

FD Worksheet

$F = \{AB \rightarrow C, A \rightarrow D, D \rightarrow E, AC \rightarrow B\}$

What are the attribute closures?

A: ADE

AB: ABCDE

B: B

D: DE

Normal Form

- Avoids redundancies and anomalies
- Guidance on whether decomposition is needed

Boyce-Codd Normal Form

- R is in BCNF if for every FD $X \rightarrow A$ that holds over R, one of the following statements is true:
  - $A \in X$; that is, it is a trivial FD
  - X is a superkey
- No redundancy: contains only information that cannot be inferred with FDs

Normalization

- Decompose into multiple relations
  - Some problems:
    - Lossiness: Can we reconstruct the original relation?
    - Dependency Preserving: Have we lost any dependencies?
    - Some queries more expensive (need joins)

Decomposition into BCNF

- If $X \rightarrow Y$ violates BCNF…
  - Decompose R into $R - Y$ and XY
- Repeat until in BCNF (guaranteed to terminate)

- *Lossless, but not dependency preserving*
- *Final result depends on order of decomposition*

Lossless Join Decomposition

- Lossless join decomposition with respect to *F* if, for every instance *r* of *R*: $\pi_X(r) \bowtie \pi_Y(r) = r$

- Necessary and sufficient test:
  - R decomposed to X and Y is lossless iff:
    - $X \cap Y \rightarrow X$ OR $X \cap Y \rightarrow Y$
    - Common attributes contain key for either X or Y

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

ABEFG decomposes into BEFG, and AG

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

ABEFG decomposes into BEFG, and AG

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

ABEFG decomposes into BEFG, and AG

BEFG decomposes into BEG, and FG

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

ABEFG decomposes into BEFG, and AG

BEFG decomposes into BEG, and FG

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Decomposes into R – CD = ABEFG, and ABCD

ABEFG decomposes into BEFG, and AG

BEFG decomposes into BEG, and FG

Final Relations: ABCD, AG, BEG, FG

Worksheet: Decomposition

Decompose R = ABCDEFG into BCNF, given:

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

Final Relations: ABCD, AG, BEG, FG

Are dependencies preserved?

- Check: If $(F_X \cup F_Y)^+ = F^+$, dependency preserving

How to 3NF?

- Add a constraint to BCNF: A (from X → A) can be part of some candidate key!
    - This preserves dependencies.

1. Decompose into BCNF first, using a minimal cover for F
    - Resulting relations do not violate 3NF by default
2. Find dependencies X → A that are not preserved, and add XA to the decomposition
    - These don't violate 3NF either

Minimal Covers

- By definition, a minimal cover G of F:
  - Has the <u>same closure</u> as F
  - Has single attributes on the right side of each dependency
  - Cannot be modified without changing the closure
    - Consider AB → CD

Minimal Cover Algorithm

1. Put the FDs in standard form
   - Every dependency is of the form $X \rightarrow A$, where A is a *single attribute*
2. Minimize the left side of each FD
   - See if you can delete attributes from the left while preserving equivalence to F^+
3. Delete redundant FDs
   - (while preserving equivalence)

Worksheet: Minimal Cover

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

1. Convert to standard form

Worksheet: Minimal Cover

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

1. Convert to standard form
    - $AB \rightarrow C$, $AB \rightarrow D$
    - $C \rightarrow E$, $C \rightarrow F$
    - $G \rightarrow A$
    - $G \rightarrow F$
    - $CE \rightarrow F$

Worksheet: Minimal Cover

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

2. Minimize the left sides

- $AB \rightarrow C$, $AB \rightarrow D$
- $C \rightarrow E$, $C \rightarrow F$
- $G \rightarrow A$
- $G \rightarrow F$
- $CE \rightarrow F$

Worksheet: Minimal Cover

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

2. Minimize the left sides

- $AB \rightarrow C$, $AB \rightarrow D$
- $C \rightarrow E$, $C \rightarrow F$
- $G \rightarrow A$
- $G \rightarrow F$
- $\cancel{CE \rightarrow F}$ $C \rightarrow F$

Worksheet: Minimal Cover

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

3. Delete redundant FDs
   - $AB \rightarrow C$, $AB \rightarrow D$
   - $C \rightarrow E$, $C \rightarrow F$
   - $G \rightarrow A$
   - $G \rightarrow F$
   - $C \rightarrow F$

Worksheet: Minimal Cover

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

Final Answer:

- AB → C, AB → D
- C → E, C → F
- G → A
- G → F