CS 4440 A

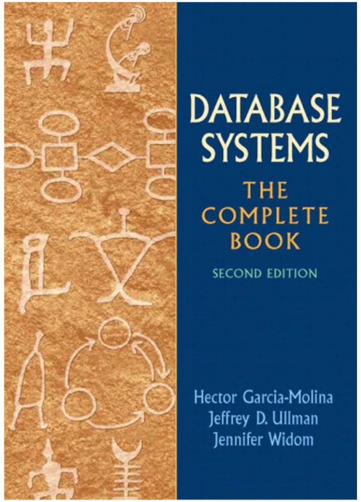
Emerging Database Technologies

Lecture 3
01/17/24

Reading Materials

Database Systems: The Complete Book (2nd edition)

- Chapter 3: Design Theory for Relational Databases (3.1 – 3.5)



Supplementary materials

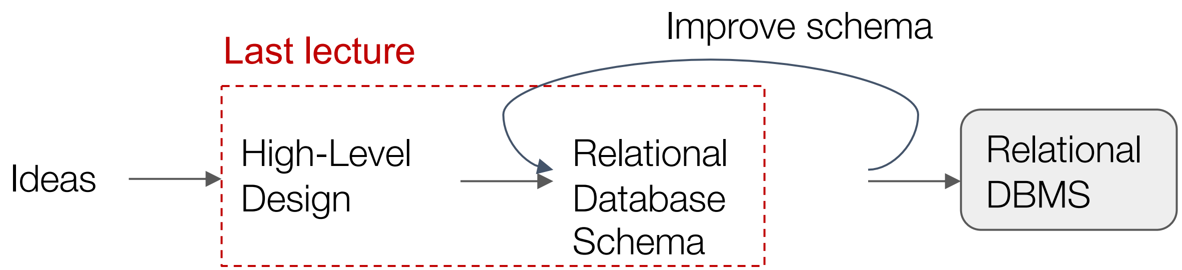
Fundamental of Database Systems (7th Edition)

- Chapter 14 - Basics of Functional Dependencies and Normalization for Relational Databases

Acknowledgement: The following slides have been adapted from EE477 (Database and Big Data Systems) taught by Steven Whang.

Design theory for relational databases

- There are many ways to design a relational database schema
  - E.g., we just learned how to use an E/R diagram
- It is also common to improve the initial schema (esp. eliminating redundancy)
  - Often, the problem is combining too much into one relation
- Fortunately, there is a well-developed design theory for good schema design
  - Functional dependencies, normalization, multivalued dependencies
  - One of the reasons Databases are powerful and so widely used



Agenda

Functional dependency

“Anomalies” in relation schemas

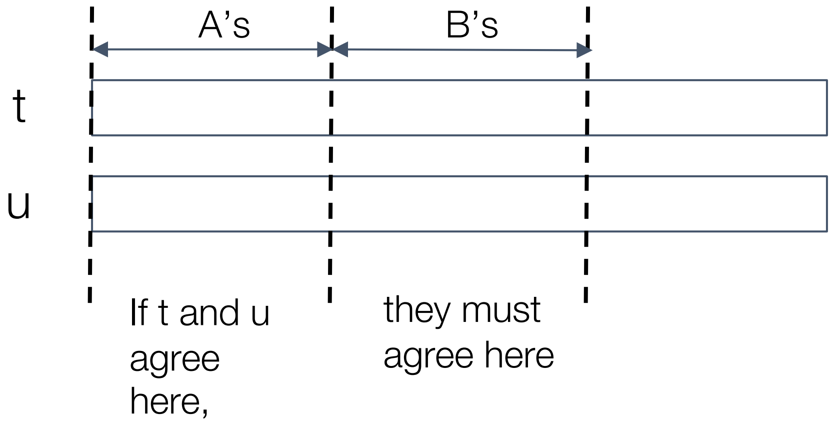
- Redundancy
- Update anomaly
- Deletion anomaly

“Normalization” to remove anomalies

- BCNF
- 3NF

Functional dependency (FD)

- A common constraint on a relation that generalizes the idea of a key
- Definition: if two tuples of R agree on all the attributes $A_1, A_2, \ldots, A_n$, they must also agree on (or functionally determine) $B_1, B_2, \ldots, B_m$
- Denoted as $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$
- Called “functional” because FD takes A values and produces unique B values



Functional dependency (FD)

- Consider the following relation, which tries to do “too much” and has redundancies
- Q: What are the FDs?

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Functional dependency (FD)

- Consider the following relation, which tries to do “too much” and has redundancies
- What are the FDs?

title, year → length, genre, studioName ✓

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Functional dependency (FD)

- Consider the following relation, which tries to do "too much" and has redundancies
- What are the FDs?

title, year → starName ✗

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Functional dependency (FD)

- FD are for all possible instances of a relation
- FDs can be used to decompose relations and eliminate redundancy
- It is common for the right side of an FD to be a single attribute
- In fact, $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ is equivalent to the set of FD's

$$A_1A_2 \ldots A_n \rightarrow B_1$$
$$A_1A_2 \ldots A_n \rightarrow B_2$$
$$\ldots$$
$$A_1A_2 \ldots A_n \rightarrow B_m$$

Key

- A set of attributes that functionally determine all other attributes
- And no proper subset does the same (i.e., a key is minimal)
- There can be multiple keys (there is no special role of the primary key here)

{title, year, starName} is a key
{title, year} is not a key because title year → starName is not an FD
{year, starName} is not a key because year starName → title is not an FD
{title, starName} is not a key because title starName → year is not an FD

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Superkey

- A set of attributes that contains a key
- Not necessarily minimal

{title, year, starName} is a key or superkey
{title, year, length, starName} is a superkey

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Reasoning about FDs

- Suppose we are told of a set of FDs that a relation satisfies
- Often we can deduce that relation must satisfy certain other FDs
  - Example: if a relation R(A, B, C) satisfies the FD's $A \rightarrow B$ and $B \rightarrow C$, R also satisfies $A \rightarrow C$
  - Proof: given two tuples (a, b_1, c_1), (a, b_2, c_2), we know that $b_1 = b_2$ and, therefore, $c_1 = c_2$ as well
- This ability to discover additional FDs is helpful for good relation schema design

Splitting/combining rule

- Splitting/combining can be applied to the right sides of FD's

$$A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$$

Splitting rule

Combining rule

$$A_1A_2 \ldots A_n \rightarrow B_1, \quad A_1A_2 \ldots A_n \rightarrow B_2, \quad \ldots, \quad A_1A_2 \ldots A_n \rightarrow B_m$$

Splitting/combining rule

- For example,

title year → length genre studioName

title year → length
title year → genre
title year → studioName

Splitting rule

- Splitting rule does not apply to the left sides of FD’s

title year → length

title → length
year → length

Trivial functional dependencies

- A constraint is trivial if it holds for every possible instance of the relation
- A trivial FD $A_1A_2 \ldots A_n \rightarrow B_1\ B_2\ \ldots\ B_m$ is where $\{B_1, B_2, \ldots B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$
  - E.g., title year → title
  - E.g., title → title
- Trivial dependency rule: $A_1A_2 \ldots A_n \rightarrow B_1\ B_2\ \ldots\ B_m$ is equivalent to $A_1A_2 \ldots A_n \rightarrow C_1\ C_2\ \ldots\ C_k$ where the C's are the B's that are not also A's
  - E.g., title year → title length is equivalent to title year → length

Closure of attributes

- The closure of $\{A_1, A_2, \ldots, A_n\}$ under the FD's in S is the set of attributes X where $A_1A_2 \ldots A_n \rightarrow X$ follows from the FD's of S
- Denoted as $\{A_1, A_2, \ldots, A_n\}+$

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

$\{A, B\}+$

A, B

Closure of attributes

- The closure of $\{A_1, A_2, \ldots, A_n\}$ under the FD's in S is the set of attributes X where $A_1A_2 \ldots A_n \rightarrow X$ follows from the FD's of S
- Denoted as $\{A_1, A_2, \ldots, A_n\}+$

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

{A, B}+

A, B, C

Closure of attributes

- The closure of $\{A_1, A_2, \ldots, A_n\}$ under the FD's in S is the set of attributes X where $A_1A_2 \ldots A_n \rightarrow X$ follows from the FD's of S
- Denoted as $\{A_1, A_2, \ldots, A_n\}+$

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

{A, B}+

A, B, C, D

Closure of attributes

- The closure of $\{A_1, A_2, \ldots, A_n\}$ under the FD's in S is the set of attributes X where $A_1A_2 \ldots A_n \rightarrow X$ follows from the FD's of S
- Denoted as $\{A_1, A_2, \ldots, A_n\}+$

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

$\{A, B\}+$

A, B, C, D, E

Closure of attributes

- The closure of $\{A_1, A_2, \ldots, A_n\}$ under the FD's in S is the set of attributes X where $A_1A_2 \ldots A_n \rightarrow X$ follows from the FD's of S
- Denoted as $\{A_1, A_2, \ldots, A_n\}+$

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

{A, B}+

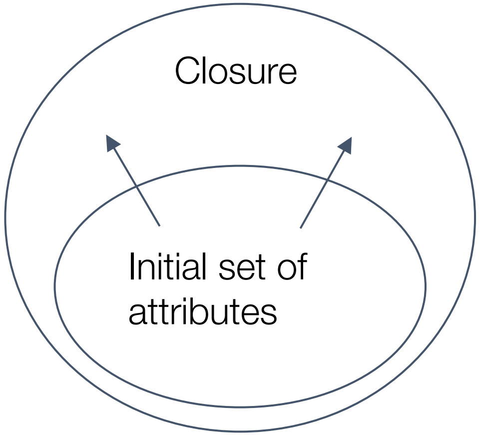
A, B, C, D, E

Cannot be expanded further, so this is a closure

Closure algorithm

- Input: $\{A_1, A_2, \ldots, A_n\}$ and a set of FD's S
- Output: the closure $\{A_1, A_2, \ldots, A_n\}+$

1. If necessary, split the FD's of S so each FD has a single attribute on the right
2. Initialize $X = \{A_1, A_2, \ldots, A_n\}$
3. Repeatedly search an FD $B_1B_2 \ldots B_m \rightarrow C$
   where the B's are in X, but C is not, and add C to X
4. Return X



Proof of correctness in textbook

Why computing closure?

- Test if FD $A_1A_2 \ldots A_n \rightarrow B$ follows from a set of FDs S
  - Compute $\{A_1, A_2, \ldots, A_n\}+$ and checking if it contains B
    - In the previous closure example, $AB \rightarrow D$ follows from the FD's because $\{A, B\}+ = \{A, B, C, D, E\}$

Closures and keys

- $A_1, A_2, \ldots, A_n$ is a superkey if and only if $\{A_1, A_2, \ldots, A_n\}+$ is the set of all attributes

Armstrong's axioms

You can derive any FDs that follows from a given set using these axioms:

1. Reflexivity: If $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$
   then $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$
2. Augmentation: If $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$
   then $A_1 A_2 \ldots A_n C_1 C_2 \ldots C_k \rightarrow B_1 B_2 \ldots B_m C_1 C_2 \ldots C_k$
   (remove any duplicates on left and right hand sides)
3. Transitivity: If $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ and $B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k$
   then $A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k$

These three inference rules are sound and complete

- Sound: only produces FDs in the closure
- Complete: produces all the FDs in the closure

Armstrong's axioms

- Does AB → D follow from the FDs below?

AB → C

BC → AD

D → E

CF → B

1. AB → C (given)
2. BC → AD (given)

Armstrong's axioms

- Does AB → D follow from the FDs below?

AB → C

BC → AD

D → E

CF → B

1. AB → C (given)
2. BC → AD (given)
3. AB → BC (Augmentation on 1)

Armstrong’s axioms

- Does AB → D follow from the FDs below?

AB → C

BC → AD

D → E

CF → B

1. AB → C (given)
2. BC → AD (given)
3. AB → BC (Augmentation on 1)
4. AB → AD (Transitivity on 2,3)

Armstrong’s axioms

- Does AB → D follow from the FDs below?

AB → C

BC → AD

D → E

CF → B

1. AB → C (given)
2. BC → AD (given)
3. AB → BC (Augmentation on 1)
4. AB → AD (Transitivity on 2,3)
5. AD → D (Reflexivity)

Armstrong’s axioms

- Does AB → D follow from the FDs below?

AB → C

BC → AD

D → E

CF → B

1. AB → C (given)
2. BC → AD (given)
3. AB → BC (Augmentation on 1)
4. AB → AD (Transitivity on 2,3)
5. AD → D (Reflexivity)
6. AB → D (Transitivity on 4,5)

Exercise #1

- Given R(A, B, C, D) and FD's AB $\rightarrow$ C, C $\rightarrow$ D, D $\rightarrow$ A
  - Can you show that AB is a key of R?
  - Can you show that BD is a key of R?

Minimal basis

- Sometimes we want to choose which FD's represent the full set of FD's of a relation
  - E.g., when computing keys
- Given a set of FD's S, any set of FD's equivalent to S is a basis for S
- A minimal basis of S is a basis M such that
  - All the FD's in M have singleton right sides
  - If any FD is removed, M is no longer a basis
  - If for any FD in M we remove one or more attributes from the left side, M is no longer a basis
- Suppose S = {A → AB, AB → C}
  - Then the minimal basis is {A → B, A → C}
  - In general, there can be multiple minimal bases

Minimal basis generation

Input: $S = \{A \rightarrow AB, AB \rightarrow C\}$

1. Split FD's so that they have singleton right sides
   $M = \{A \rightarrow B, A \rightarrow A, AB \rightarrow C\}$
2. Remove trivial FDs
   $M = \{A \rightarrow B, AB \rightarrow C\}$
3. Minimize the left sides of each FD
   $M = \{A \rightarrow B, A \rightarrow C\}$
4. Remove redundant FDs
   $M = \{A \rightarrow B, A \rightarrow C\}$

Projection of functional dependencies

- When designing a schema, sometimes need to answer the following question: Given a relation R with a set of FD's S, what FD's hold for $R_1 = \pi_L(R)$?
- Compute all the FD's that
  - follow from S and
  - involve only attributes in R_1
- Example
  - Suppose R(A, B, C, D) has FD's $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$
  - Then the FD's for R_1(A, C, D) are $A \rightarrow C$, $C \rightarrow D$

Recap

- Design theory
  - Functional dependency (FD)
  - Trivial FDs
  - Splitting/combining rule
  - Closure of attributes
  - Armstrong's axioms
  - Minimal basis
  - Projection of FDs

title, year → length, genre, studioName

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

AB → C

BC → AD

D → E

CF → B

1. AB → C (given)
2. BC → AD (given)
3. AB → BC (Augmentation on 1)
4. AB → AD (Transitivity on 2,3)
5. AD → D (Reflexivity)

Design of relational database schemas

- Careless schema selection may lead to redundancies and anomalies

- We will discuss
  - Redundancy and related anomalies
  - Relation decomposition
  - Boyce-Codd normal form (BCNF)
  - 3NF

Anomalies

- Occurs when we try to cram too much information into a single relation

  1. Redundancy: information is repeated unnecessarily
  2. Update anomaly: only updating the first tuple may leave the second tuple incorrect

Movies1

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Anomalies

- Occurs when we try to cram too much information into a single relation

Movies1

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

3. Deletion anomaly: removing the movie star Choi Min-Sik will also remove the movie information of Oldboy

Decomposing relations

- The accepted way to eliminate anomalies is to decompose relations

Movies2 No redundancy or update anomalies

title	year	length	genre	studioName
Ponyo	2008	103	anime	Ghibli
Oldboy	2003	120	mystery	Show East

Movies3 No deletion anomalies

title	year	starName
Ponyo	2008	Yuria Nara
Ponyo	2008	Hiroki Doi
Oldboy	2003	Choi Min-Sik

Decomposing relations

- The accepted way to eliminate anomalies is to decompose relations

Movies2

title	year	length	genre	studioName
Ponyo	2008	103	anime	Ghibli
Oldboy	2003	120	mystery	Show East

Movies3

title	year	starName
Ponyo	2008	Yuria Nara
Ponyo	2008	Hiroki Doi
Oldboy	2003	Choi Min-Sik

This is OK because title and year form a key of a movie and cannot be more succinct; if one of the year changes, the movie is a different one