Announcements

Make-up midterms

- Saturday: 1300-1500
- Utown LT51
- If you have MC, send your valid MC to your tutor during tutorial
- If you're from CS2040, I will get your data from Prof. Chong

Remedial

- Friday: 1500-1800
- SR10

Useful websites

- https://github.com/thisisadiyoga/cs2102_ay1819_s2
- http://cs2102-ay1819-s2.herokuapp.com/
- http://cs2102-fd-nf.herokuapp.com/

CS2102
Database Systems

Slides adapted from Prof. Chan Chee Yong

LECTURE 08

FUNCTIONAL DEPENDENCIES

PREVIOUS LECTURE

Transaction, procedures, triggers

- A transaction starts with `BEGINS` ends with either `COMMIT` or `ROLLBACK`
  - We assume ACID property is maintained
- Stored function
  - `CREATE [ OR REPLACE ] FUNCTION func_name ...`
  - `SELECT func_name (...);`
- Stored procedure
  - `CREATE [ OR REPLACE ] PROCEDURE proc_name ...`
  - `CALL proc_name (...);`
- Triggers

```
CREATE TRIGGER trigger_name
{ BEFORE | AFTER | INSTEAD OF }
{ event [ OR event [ ... ] ] } ON table
[ FOR [ EACH ] { ROW | STATEMENT } ]
[ WHEN cond ] EXECUTE PROCEDURE func_name();
```

PREVIOUS LECTURE

Application development

- Server & libraries
  - Accept connection
  - Route if necessary
    - What to do when client request `/page`?
  - Connect to database
    - How to construct SQL queries?
  - Create and return response
    - What to give to client when `/page` is requested?
- Security
  - SQL injection attack
  - Placeholder for sanitization

- Motivation
  - ER model
  - Schema
- Functional dependencies
  - Definition
- Asking questions
  - Armstrong's axiom
  - Closure
  - Minimal cover

Overview

- Motivation
  - ER model
  - Schema
- Functional dependencies
  - Definition
- Asking questions
  - Armstrong's axiom
  - Closure
  - Minimal cover

Motivation

Database design process

ER diagram

1. **Requirement analysis**
   - Find out the data/application/performance requirement of the enterprise
2. **Conceptual database design**
   - Capture data requirements using a conceptual schema
3. **Logical database design**
   - Map conceptual schema to logical schema supported by DBMS
4. **Schema refinement**
   - Improve logical schema design using data constraints
5. **Physical database design**
   - Use performance requirements to design physical schema
6. **Application & security design**
   - Specify access control policies

Database design process

Next step

1. **Requirement analysis**
   - Find out the data/application/performance requirement of the enterprise
2. **Conceptual database design**
   - Capture data requirements using a conceptual schema
3. **Logical database design**
   - Map conceptual schema to logical schema supported by DBMS
4. **Schema refinement**
   - Improve logical schema design using data constraints
5. **Physical database design**
   - Use performance requirements to design physical schema
6. **Application & security design**
   - Specify access control policies

Database design process

Requirement analysis

I would like my customers to be able to browse my catalog of books and place orders over the Internet. Currently, I take orders over the phone. I have mostly corporate customers who call me and give me the ISBN number of a book and a quantity; they often pay by credit card. If I don't have enough copies in stock, I order additional copies and delay the shipment until the new copies arrive; I want to ship a customer's entire order together. My catalog includes all the books I sell. For each book, the catalog contains its ISBN number, title, author, purchase price, sales price, and the year the book was published. Most of my customers are regulars, and I have records with their names and addresses. New customers have to call me first and establish an account before they can use my website. On my new website, customers should first identify themselves by their unique customer identification number. Then they should be able to browse my catalog and to place orders online.

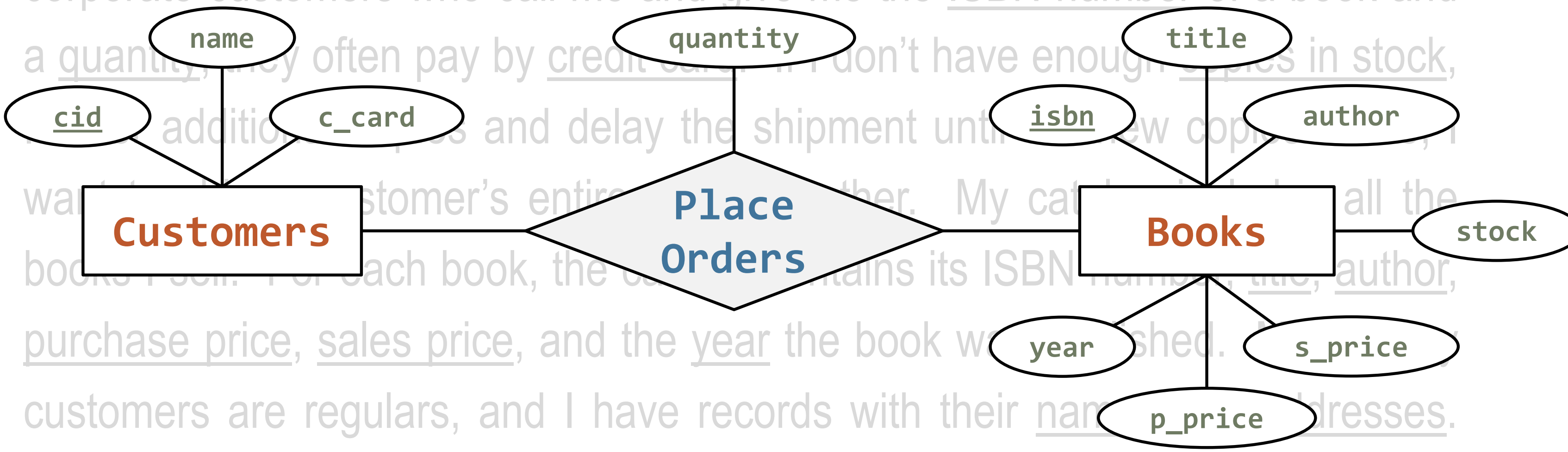
ER model

Entities, relationships, and attributes

I would like my customers to be able to browse my catalog of books and place orders over the Internet. Currently, I take orders over the phone. I have mostly corporate customers who call me and give me the ISBN number of a book and a quantity; they often pay by credit card. If I don't have enough copies in stock, I order additional copies and delay the shipment until the new copies arrive; I want to ship a customer's entire order together. My catalog includes all the books I sell. For each book, the catalog contains its ISBN number, title, author, purchase price, sales price, and the year the book was published. Most of my customers are regulars, and I have records with their names and addresses. New customers have to call me first and establish an account before they can use my website. On my new website, customers should first identify themselves by their unique customer identification number. Then they should be able to browse my catalog and to place orders online.

ER model

Entities, relationships, and attributes



Constraints:

- Each customer is identified by their cid, they have a name and a credit card
- Each book is identified by their isbn, they have title, author, etc
- Each customer buys at most one book, the quantity is to be recorded

Schema

Entities, relationships, and attributes

I would like my customers to be able to browse my catalog of books and place orders over the Internet. Currently, I take orders over the phone. I have mostly

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

books I sell. For each book, the catalog contains its ISBN number, title, author, purchase price, sales price, and the year the book was published. Most of my customers are regulars, and I have records with their names and addresses. New customers have to call me first and establish an account before they can use my website. On my new website, customers should first identify themselves by their unique customer identification number. Then they should be able to browse my catalog and to place orders online.

Constraints:

- Each customer is identified by their cid, they have a name and a credit card
- Each book is identified by their isbn, they have title, author, etc
- Each customer buys at most one book, the quantity is to be recorded

Schema

Schema refinement issues

- How do we know if the schema design is good or bad?
- How to transform a bad schema design into a good design?

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

- **Possible problems**
  - Insertion anomaly: we cannot store any customer if they have never made a purchase
  - Deletion anomaly: if we delete all books with isbn 1234, we lose information about Alice
  - Update anomaly: if Trudy credit card number change, we have to be careful of consistent updates (*update on all places*)

Schema

Schema refinement issues

- How do we know if the schema design is good or bad?
- How to transform a bad schema design into a good design?

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

- **ER diagram solution to the problem**

cid	*name*	*c_card*
3118	Alice	5243 ****
1423	Trudy	1234 ****
5609	Carol	5243 ****

cid	*quantity*	*isbn*
3118	100	1234
1423	100	1234
1423	200	2102
5609	200	2102

isbn	*title*	*author*	*year*	*omitted*
1234	DB	Adi	2019	
2102	PSQL	Yoga	2016	

Schema

Schema refinement issues

- How do we know if the schema design is good or bad?
- How to transform a bad schema design into a good design?

cid	*name*	*c_card*
3118	Alice	5243 ****
1423	Trudy	1234 ****
5609	Carol	5243 ****

cid	*quantity*	*isbn*
3118	100	1234
1423	100	1234
1423	200	2102
5609	200	2102

isbn	*title*	*author*	*year*	*omitted*		
1234	DB	Adi	2019			
2102	PSQL	Yoga	2016			

- **Possible problems**
  - Insertion anomaly: we cannot store any customer if they have never made a purchase
  - Deletion anomaly: if we delete all books with isbn 1234, we lose information about Alice
  - Update anomaly: if Trudy credit card number change, we have to be careful of consistent updates (*update on all places*)

Schema

Schema refinement issues

- How do we know if the schema design is good or bad?
- How to transform a bad schema design into a good design?

cid	***name***	***c_card***
3118	Alice	5243 ****
1423	Trudy	1234 ****
5609	Carol	5243 ****

cid	***quantity***	***isbn***
3118	100	1234
1423	100	1234
1423	200	2102
5609	200	2102

isbn	***title***	***author***	***year***	***omitted***		
1234	DB	Adi	2019			
2102	PSQL	Yoga	2016			

- **Possible problems**
  - Insertion anomaly: ~~we cannot store any customer if they have never made a purchase~~
  - Deletion anomaly: ~~if we delete all books with isbn 1234, we lose information about Alice~~
  - Update anomaly: ~~if Trudy credit card number change, we have to be careful of consistent updates (*update on all places*)~~

- Motivation
  - ER model
  - Schema
- Functional dependencies
  - Definition
- Asking questions
  - Armstrong’s axiom
  - Closure
  - Minimal cover

Functional dependencies

Functional dependencies

Notations

- We use $R(A_1, A_2, \dots, A_n)$ to denote relation schema with n attributes
- We use R to denote the set of attributes in R (i.e., $attr(R)$)
  - Example: $R = \{A_1, A_2, \dots, A_n\}$
- We use lowercase letter $a, b, \dots$ except r to denote subsets of attributes in R
  - Let $a, b \subseteq R$ and $A_i, A_j \in R$
    - We use ab to denote $a \cup b$ union
    - We use A_iA_j to denote $\{A_i, A_j\}$ set
    - We use A_ib to denote $\{A_i\} \cup b$ union
    - We use $b - A_i$ to denote $b - \{A_i\}$ set difference

Functional dependencies

Definition

- Functional dependencies (FD) are constraints on schemas that specify that the values for certain set of attributes determine unique values for another set of attributes (i.e., *uniquely identifies*)
- Let a and b denote subsets of attributes of a relational schema R
  - We use $a \rightarrow b$ to denote that a functionally determines b
  - We also say that b functionally depends on a
- The FD $a \rightarrow b$ holds on R *if and only if* for any relation instance r of R, whenever two tuples t_1 and t_2 of r agree on the attributes a, they also agree on the attributes b
  - $\pi_a(t_1) = \pi_a(t_2) \quad \Rightarrow \quad \pi_b(t_1) = \pi_b(t_2)$
  - Example
    - `cid` → `name`
    - `cid` → `c_card`
    - `name` → `c_card`
    - `(cid,name)` → `name`

cid	*name*	*c_card*
3118	Alice	5243 ****
1423	Trudy	1234 ****
5609	Carol	5243 ****

Functional dependencies

Definition

- Functional dependencies (FD) are constraints on schemas that specify that the values for certain set of attributes determine unique values for another set of attributes (i.e., *uniquely identifies*)
- Let a and b denote subsets of attributes of a relational schema R
    - We use $a \to b$ to denote that a functionally determines b
    - We also say that b functionally depends on a
- The FD $a \to b$ holds on R *if and only if* for any relation instance r of R, whenever two tuples t_1 and t_2 of r agree on the attributes a, they also agree on the attributes b
    - $\pi_a(t_1) = \pi_a(t_2) \quad \Rightarrow \quad \pi_b(t_1) = \pi_b(t_2)$
    - Example
        - cid → name
        - cid → c_card
        - ~~name → c_card~~
        - (cid,name) → name

cid	***name***	***c_card***
3118	Alice	5243 ****
1423	Trudy	1234 ****
5609	Carol	5243 ****
3119	Alice	1234 ****

Functional dependencies

Example

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

Constraints:

- Each customer is identified by their cid, they have a name and a credit card
- Each book is identified by their isbn, they have title, author, etc
- Each customer buys at most one book, the quantity is to be recorded

Functional dependencies

Example

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

Constraints:

- Each customer is identified by their cid, they have a name and a credit card
  - `cid → (name, c_card)`
- Each book is identified by their isbn, they have title, author, etc
  - `isbn → (title, author, year, ...)`
- Each customer buys at most one book, the quantity is to be recorded
  - `(cid, isbn) → quantity`

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \to b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

cid→(name, c_card)

isbn→(title, ...)

(cid,isbn)→quantity

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \rightarrow b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
3118	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

cid→(name, c_card)

isbn→(title, ...)

(cid,isbn)→quantity

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \rightarrow b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
3118	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

~~cid→(name, c_card)~~

isbn→(title, ...)

(cid,isbn)→quantity

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \rightarrow b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	3456	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

cid→(name, c_card)

isbn→(title, ...)

(cid,isbn)→quantity

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \rightarrow b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	3456	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

cid→(name, c_card)

~~isbn→(title, ...)~~

(cid,isbn)→quantity

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \rightarrow b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	name	c_card	quantity	isbn	title	author	year	omitted
1423	Alice	5243 ****	100	2102	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

cid→(name, c_card)

isbn→(title, ...)

(cid,isbn)→quantity

Functional dependencies

Definition

- Let r be a relation instance of relation schema R
  - r satisfies FD $a \rightarrow b$ *if* for every pair of tuples t_1 and t_2 in r such that $\pi_a(t_1) = \pi_a(t_2)$, it is also true that $\pi_b(t_1) = \pi_b(t_2)$
  - An FD f holds on R *if and only if* for any relation instance r of R, r satisfies R
  - r violates an FD f *if* r does not satisfy f
  - r is a legal instance of R *if* r satisfies all FDs that holds on R

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
1423	Alice	5243 ****	100	2102	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

~~cid→(name, c_card)~~

~~isbn→(title, ...)~~

~~(cid,isbn)→quantity~~

Functional dependencies

Trivial and non-trivial

- An FD $a \rightarrow b$ is a trivial FD *if* $b \subseteq a$
  - b is a subset of a
- An FD $a \rightarrow b$ is a non-trivial FD *if* $b \nsubseteq a$
  - b is NOT a subset of a (i.e., $a \rightarrow b$ is NOT a trivial FD)
- An FD $a \rightarrow b$ is a completely non-trivial FD *if* $a \cap b = \emptyset$
  - $a \rightarrow b$ is non-trivial, and
  - a and b have completely different attributes

FD	*is trivial?*	*is non-trivial?*	*is completely non-trivial?*
$AB \rightarrow B$			
$AB \rightarrow AC$			
$AB \rightarrow C$			
$C \rightarrow \emptyset$			
$\emptyset \rightarrow \emptyset$			

Functional dependencies

Trivial and non-trivial

- An FD $a \rightarrow b$ is a trivial FD *if* $b \subseteq a$
  - b is a subset of a
- An FD $a \rightarrow b$ is a non-trivial FD *if* $b \nsubseteq a$
  - b is NOT a subset of a (i.e., $a \rightarrow b$ is NOT a trivial FD)
- An FD $a \rightarrow b$ is a completely non-trivial FD *if* $a \cap b = \emptyset$
  - $a \rightarrow b$ is non-trivial, and
  - a and b have completely different attributes

Evaluate the following claim:

- There are trivial FDs that are also completely non-trivial
- There are completely non-trivial FDs that are trivial
- There are non-trivial FDs that are not completely non-trivial

Functional dependencies

Trivial and non-trivial

- An FD $a \rightarrow b$ is a trivial FD *if* $b \subseteq a$
  - b is a subset of a
- An FD $a \rightarrow b$ is a non-trivial FD *if* $b \nsubseteq a$
  - b is NOT a subset of a (i.e., $a \rightarrow b$ is NOT a trivial FD)
- An FD $a \rightarrow b$ is a completely non-trivial FD *if* $a \cap b = \emptyset$
  - $a \rightarrow b$ is non-trivial, and
  - a and b have completely different attributes

Evaluate the following claim:

- ~~There are trivial FDs that are also completely non-trivial~~
- ~~There are completely non-trivial FDs that are trivial~~
- There are non-trivial FDs that are not completely non-trivial

- Motivation
  - ER model
  - Schema
- Functional dependencies
  - Definition
- Asking questions
  - Armstrong's axiom
  - Closure
  - Minimal cover

Asking questions

Introduction

What kind of questions can we ask about FDs?

- Given a set of FDs F (that hold on R) and an FD f
  - Does f also hold on R
- Example:
  - $F =$ { cid → (name, c_card), isbn→(title, ...), (cid,isbn)→quantity}
  - Which of the following FDs also hold?

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

  - (cid,isbn)→name
  - isbn→quantity

❖ How do we know?

Asking question

Closure

- Definitions:
  - Let F and G denote sets of FDs, and f denote an FD
  - F logically implies (or simply implies) *if* every relation instance r of R that satisfies the FDs F also satisfies the FD f
    - Denoted as $F \vDash f$
    - More generally, $F \vDash G$ *if* $F \vDash g$ for all $g \in G$
  - The closure of F is the set of all FDs implied by F
    - Denoted as $F^+ = \{ f \mid F \vDash f \}$
  - F is equivalent to G *if* $F^+ = G^+$
    - Denotes as $F \equiv G$
    - In other words, $F \vDash G$ and $G \vDash F$
    - Or simply, their closure are the same

Asking question

Finding closure

- Let $R = (A, B)$ with FDs $F = \{A \rightarrow B\}$
  - In other words, we have relation *R(<u>A</u>,B)*

FD	*is implied?*	*FD*	*is implied?*
$\emptyset \rightarrow \emptyset$	☺	$A \rightarrow \emptyset$	☺
$\emptyset \rightarrow A$	☹	$A \rightarrow A$	☺
$\emptyset \rightarrow B$	☹	$A \rightarrow B$	☺
$\emptyset \rightarrow AB$	☹	$A \rightarrow AB$	☺
$B \rightarrow \emptyset$	☺	$AB \rightarrow \emptyset$	☺
$B \rightarrow A$	☹	$AB \rightarrow A$	☺
$B \rightarrow B$	☺	$AB \rightarrow B$	☺
$B \rightarrow AB$	☹	$AB \rightarrow AB$	☺

In general, enumeration works, but you will need to go through $2^n \times 2^n$ FDs

- Closure:

$$F^+ = \left\{ \begin{matrix} \emptyset \rightarrow \emptyset, B \rightarrow \emptyset, B \rightarrow B, A \rightarrow \emptyset, A \rightarrow A, A \rightarrow B, A \rightarrow AB \\ AB \rightarrow \emptyset, A \rightarrow AB, AB \rightarrow B, AB \rightarrow AB \end{matrix} \right\}$$

Asking question

Finding closure

- Let $R = (A, B)$ with FDs $F = \{A \rightarrow B\}$
  - In other words, we have relation *R(<u>A</u>,B)*

FD	is implied?	FD	is implied?
$\emptyset \rightarrow \emptyset$	☺	$A \rightarrow \emptyset$	☺
$\emptyset \rightarrow A$	☹	$A \rightarrow A$	☺
$\emptyset \rightarrow B$	☹	$A \rightarrow B$	☺
$\emptyset \rightarrow AB$	☹	$A \rightarrow AB$	☺
$B \rightarrow \emptyset$	☺	$AB \rightarrow \emptyset$	☺
$B \rightarrow A$	☹	$AB \rightarrow A$	☺
$B \rightarrow B$	☺	$AB \rightarrow B$	☺
$B \rightarrow AB$	☹	$AB \rightarrow AB$	☺

Is there a simpler way?

- Consider the function way of writing things
- $A \rightarrow B \equiv (F(A) = B)$
- Given $F(A)$ can you find the value of A?
- Given A and C such that $F(A) = B$, can you find the value of B and C?
- Given A such that $F(A) = B$ and $F(B) = C$, can you find the value of C?

  - Closure:

$$F^+ = \left\{ \begin{matrix} \emptyset \rightarrow \emptyset, B \rightarrow \emptyset, B \rightarrow B, A \rightarrow \emptyset, A \rightarrow A, A \rightarrow B, A \rightarrow AB \\ AB \rightarrow \emptyset, A \rightarrow AB, AB \rightarrow B, AB \rightarrow AB \end{matrix} \right\}$$

Asking question

Finding closure

- Let $R = (A, B)$ with FDs $F = \{A \rightarrow B\}$
  - In other words, we have relation *R(<u>A</u>,B)*

Rewrite the consideration into $A \rightarrow B$ form again

- Given $F(A)$ can you find the value of A?
  - $A \rightarrow A$ for any A
  - Any attribute uniquely identify itself
  - More generally $a \rightarrow b$ for any $b \subseteq a$
- Given A and C such that $F(A) = B$, can you find the value of B and C?
  - $AC \rightarrow BC$
  - Adding new information does not make us lose information
- Given A such that $F(A) = B$ and $F(B) = C$, can you find the value of C?
  - $A \rightarrow B$ and $B \rightarrow C$ implies $A \rightarrow C$
  - If A uniquely identifies B and B uniquely identified C
    - Surely A uniquely identifies C

- Closure:

$$F^+ = \left\{ \begin{matrix} \emptyset \rightarrow \emptyset, B \rightarrow \emptyset, B \rightarrow B, A \rightarrow \emptyset, A \rightarrow A, A \rightarrow B, A \rightarrow AB \\ AB \rightarrow \emptyset, A \rightarrow AB, AB \rightarrow B, AB \rightarrow AB \end{matrix} \right\}$$

Asking question

Armstrong's axioms

- Let $R = (A, B)$ with FDs $F = \{A \rightarrow B\}$
  - In other words, we have relation *R(<u>A</u>,B)*

Set of Armstrong's axioms

- **Reflexivity**
  - $A \rightarrow A$ for any A
  - Any attribute uniquely identify itself
  - More generally $a \rightarrow b$ for any $b \subseteq a$
- **Augmentation**
  - $AC \rightarrow BC$
  - Adding new information does not make us lose information
- **Transitivity**
  - $A \rightarrow B$ and $B \rightarrow C$ implies $A \rightarrow C$
  - If A uniquely identifies B and B uniquely identified C
    - Surely A uniquely identifies C

- Properties
  - Sound any derived FD is implied by F
  - Complete all FDs in F^+ can be derived

Asking question

Using Armstrong's axioms

- Consider $R(A, B, C, D, E)$ with the following FDs
  - $A \to C \quad B \to C \quad CD \to E$
  - Show that $F \vDash AD \to E$
  - Steps

1.	$A \to C$	Given
2.	$AD \to CD$	Augmentation of (1) with D
3.	$CD \to E$	Given
4.	$AD \to E$	Transitivity with (2) and (3)

Asking question

Extended Armstrong's axioms

- Recap
  - Reflexivity — if $b \subseteq a$ then $a \rightarrow b$
  - Augmentation — if $a \rightarrow b$ then $ac \rightarrow bc$
  - Transitivity — if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$
- Extension
  - Union — if $a \rightarrow b$ and $a \rightarrow c$ then $a \rightarrow bc$
  - Decomposition — if $a \rightarrow b$ then $a \rightarrow b'$ where $b' \subseteq b$
    - Specific case — if $a \rightarrow bc$ then $a \rightarrow b$ and $a \rightarrow c$

- Unless specified, we will only use the non-extended version
  - The extended Armstrong's axiom can be derived from the first three axioms

Asking question

Using extended Armstrong's axioms

- Consider the following sets of FDs
  - $F = \{A \rightarrow BCD\}$
  - $G = \{A \rightarrow B,\ A \rightarrow C,\ A \rightarrow D\}$
  - Show that $F \equiv G$
  - Steps
    - Proof $F \vDash G$
      - By decomposition, we have $A \rightarrow B, A \rightarrow C, A \rightarrow D$
      - Therefore $F \vDash \{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$
    - Proof $G \vDash F$
      - By union, we have $A \rightarrow BCD$
      - Therefore $G \vDash A \rightarrow BCD$

Revisiting the keys

Superkeys, keys, and prime attributes

- A set of attributes a is a superkey of schema R (with FDs F) *if* $F \vDash a \rightarrow R$
  - If we know the values of attributes a, we can uniquely identify any tuple in R
- A set of attributes a is a key of schema R *if*
  - a is a superkey
  - No proper subset of a is a superkey (*minimal*, *for each* $b \subset a$, $F \nvDash b \rightarrow R$)
    - This is the candidate key
- An attribute $A \in R$ is a prime attribute *if* A is contained in some key of R

❖ *Can you find the prime attributes of the table below?* *(in blue)*

- $F = \{$ cid → (name, c_card), isbn→(title, ...), (cid,isbn)→quantity $\}$

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

Attribute closure

Interpreting information

- FDs allow us to figure out *what we can know* given a piece of information
  - Consider knowing `cid`, can you know the name and credit card number?
- However, there are *limits to what we can know* if the information is limited
  - Even if we know `cid`, we still cannot know the title of a book
  - To know the title of a book, we need to know the `isbn`
    - What we want to know is the limit of what we can know
- Given a set of attribute a, other attributes that we can know is called *attribute closure of a*
  - $F = \{$ cid → (name, c_card), isbn→(title, ...), (cid,isbn)→quantity $\}$

cid	*name*	*c_card*	*quantity*	*isbn*	*title*	*author*	*year*	*omitted*
3118	Alice	5243 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	100	1234	DB	Adi	2019	
1423	Trudy	1234 ****	200	2102	PSQL	Yoga	2016	
5609	Carol	5243 ****	200	2102	PSQL	Yoga	2016	

Attribute closure

Interpreting information

- FDs allow us to figure out *what we can know* given a piece of information
  - Consider knowing `cid`, can you know the name and credit card number?
- However, there are *limits to what we can know* if the information is limited
  - Even if we know `cid`, we still cannot know the title of a book
  - To know the title of a book, we need to know the `isbn`
    - What we want to know is the limit of what we can know
- Given a set of attribute a, other attributes that we can know is called *attribute closure of* a
  - More formally, given $a \subseteq R$ and the set of FDs F on R
    - The closure of a (w.r.t. F) is $a^+ = \{A \in R \mid F \vDash a \to A\}$
    - In other words, all the attributes that functionally depends on a
  - A nice attribute that we have is $F \vDash a \to b$ *if and only if* $b \subseteq a^+$
  - We can then also compute F^+ if we can compute a^+

Attribute closure

Computing attribute closure

- Algorithm #1
  - **Input** A set of attributes $a \subseteq R$ and a set of FDs F on R
  - **Output** a^+ (w.r.t. F)

1. initialize $\theta = a$
2. while (there exists some FD $b \rightarrow c \in F$
   such that $b \subseteq \theta$ and $c \nsubseteq \theta$)
3. $\theta = \theta \cup c$
4. return θ

Attribute closure

Computing attribute closure

- Algorithm #1
  - **Input** A set of attributes $a \subseteq R$ and a set of FDs F on R
  - **Output** a^+ (w.r.t. F)

1. initialize $\theta = a$
2. while (there exists some FD $b \rightarrow c \in F$ such that $b \subseteq \theta$ and $c \nsubseteq \theta$)
3. $\theta = \theta \cup c$
4. return θ

- Example: let $F = \{A \rightarrow C, B \rightarrow C, CD \rightarrow E\}$
  - Show that $F \vDash AD \rightarrow E$

1. initialize $\Rightarrow \theta = AD$
2. with $A \rightarrow C$ $\Rightarrow \theta = ACD$
3. with $CD \rightarrow E$ $\Rightarrow \theta = ACDE$
4. therefore $AD^+ = ACDE$
   - thus $F \vDash AD \rightarrow E$

Minimal covers

Motivation

- We can compute F^+, we don't need to store all FDs
- Therefore, some FDs are redundant $\Rightarrow$ can be removed
- The smallest set of FDs is called minimal cover
  - This is simpler than the original F, can be enforced more efficiently

Definition

- Given an FD $a \to b$, an attribute $A \in a$ is a redundant attribute in FD *if*
  - $(F - \{a \to B\}) \cup \{(a - A) \to B\}$ is equivalent to F
  - In other words, having $(a - A) \to B$ instead of $a \to B$ does not change F^+
- How to compute?
  - $\left((F - \{a \to B\}) \cup \{(a - A) \to B\}\right)^+ = F^+$
  - Or simply if $F \vDash (a - A) \to B$
    - then $(F - \{a \to B\}) \cup (a - A) \to B \vDash a \to B$

Minimal covers

Motivation

- We can compute F^+, we don't need to store all FDs
- Therefore, some FDs are redundant $\Rightarrow$ can be removed
- The smallest set of FDs is called minimal cover
  - This is simpler than the original F, can be enforced more efficiently

Definition

- Given an FD $a \to b$, an attribute $A \in a$ is a redundant attribute in FD *<u>if</u>*
  - $(F - \{a \to B\}) \cup \{(a - A) \to B\}$ is equivalent to F
  - In other words, having $(a - A) \to B$ instead of $a \to B$ does not change F^+
- Example
  - $F = \{AB \to C,\ A \to B,\ B \to A\}$
    - Is A in $AB \to C$ redundant? yes, since $B^+ = ABC$ w.r.t. F
    - Is B in $AB \to C$ redundant? yes, since $A^+ = ABC$ w.r.t. F
      - *<u>But can only remove one, cannot both!</u>*

Minimal covers

Motivation

- We can compute F^+, we don't need to store all FDs
- Therefore, some FDs are redundant $\Rightarrow$ can be removed
- The smallest set of FDs is called minimal cover
  - This is simpler than the original F, can be enforced more efficiently

Definition

- Given an FD $f \in F$, f is a redundant FD *if*
  - $F - f$ is equivalent to F
  - In other words, not using f is fine, since we can derive f
- How to compute?
  - $(F - f)^+ = F^+$
  - Or simply, $(F - f) \vDash f$
    - If $f = a \rightarrow B$, we can simply compute a^+ w.r.t. $(F - f)$
    - If it contains B, then f is redundant

Minimal covers

Motivation

- We can compute F^+, we don't need to store all FDs
- Therefore, some FDs are redundant $\Rightarrow$ can be removed
- The smallest set of FDs is called minimal cover
  - This is simpler than the original F, can be enforced more efficiently

Definition

- Given an FD $f \in F$, f is a redundant FD *if*
  - $F - f$ is equivalent to F
  - In other words, not using f is fine, since we can derive f
- Example
  - $F = \{A \rightarrow B,\ A \rightarrow C,\ B \rightarrow A,\ B \rightarrow C,\ C \rightarrow A\}$
  - Which FDs are redundant?
    - $A \rightarrow C$ can be derived from $A \rightarrow B \wedge B \rightarrow C$
    - $B \rightarrow A$ can be derived from $B \rightarrow C \wedge C \rightarrow A$
    - $B \rightarrow C$ can be derived from $B \rightarrow A \wedge A \rightarrow C$

Minimal covers

Motivation

- We can compute F^+, we don't need to store all FDs
- Therefore, some FDs are redundant $\Rightarrow$ can be removed
- The smallest set of FDs is called minimal cover
  - This is simpler than the original F, can be enforced more efficiently

Definition

- A minimal cover for a set F of FDs is a set G of FDs such that
  - Every FD in G is of the form $\alpha \rightarrow A$ (*i.e., single attribute on the right*)
  - For each FD $\alpha \rightarrow A$ in G, α has no redundant attributes
  - There are no redundant FDs in G
  - G and F are equivalent

- Each set of FDs has at least one minimal cover
  - *Trivially, it is itself!*

Minimal covers

Definition

- A minimal cover for a set F of FDs is a set G of FDs such that
  - Every FD in G is of the form $a \rightarrow A$ (*i.e., single attribute on the right*)
  - For each FD $a \rightarrow A$ in G, a has no redundant attributes
  - There are no redundant FDs in G
  - G and F are equivalent

Computing a minimal cover

- Algorithm #2
  - **Input** A set of FDs F
  - **Output** A minimal cover for F

A. Break down any $a \rightarrow B_1, \ldots, B_n$ to $\{a \rightarrow B_1, \ldots, a \rightarrow B_n\}$

B. Remove redundant attribute: while preserving F^+

C. Remove redundant FD: while preserving F^+

Minimal covers

Computing a minimal cover

- Algorithm #2
  - **Input** A set of FDs F
  - **Output** A minimal cover for F

initialize $G = \emptyset$

for each (FD $a \rightarrow B_1 \dots B_n$ in F)
  $G = G \cup \{ a \rightarrow B_i \mid i \in [1, n] \}$

Decompose

for each (FD $a \rightarrow B$ in G)
  initialize $a' = a$
  for each $(A \in a)$ do
    if (B in $(a' - A)^+$ w.r.t G) then
      replace $a' \rightarrow B$ in G by $(a' - A) \rightarrow B$
      $a' = a' - A$

Remove redundant attribute

for each (FD $a \rightarrow B$ in G)
  if (B in a^+ w.r.t. $G - \{a \rightarrow B\}$) then
    remove $a \rightarrow B$ from G

Remove redundant FDs

return G

Minimal covers

Computing a minimal cover

- Example: $F = \{ABCD \to E, E \to D, A \to B, AC \to D\}$
  - Find a minimal cover of F
  - Steps
  - Decompose FDs: already decomposed
  - Remove redundant attributes:
    start with $G = \{ABCD \to E, E \to D, A \to B, AC \to D\}$
    1. A in $ABCD \to E$ is non-redundant $BCD^+ = BCD$ w.r.t. G
    2. B in $ABCD \to E$ is redundant $ACD^+ = ABCDE$ w.r.t. G
       - $G = \{ACD \to E, E \to D, A \to B, AC \to D\}$
    3. C in $ABCD \to E$ is non-redundant $AD^+ = ABD$ w.r.t. G
    4. D in $ABCD \to E$ is redundant $AC^+ = ABCDE$ w.r.t. G
       - $G = \{AC \to E, E \to D, A \to B, AC \to D\}$
    5. A in $AC \to D$ is non-redundant $C^+ = C$ w.r.t. G
    6. C in $AC \to D$ is non-redundant $A^+ = AB$ w.r.t. G

Minimal covers

Computing a minimal cover

- Example: $F = \{ABCD \to E, E \to D, A \to B, AC \to D\}$
  - Find a minimal cover of F
  - Steps
  - Remove redundant FDs:
    start with $G = \{AC \to E, E \to D, A \to B, AC \to D\}$
    1. $AC \to E$ is non-redundant $AC^+ = ABCD$ w.r.t. $G - \{AC \to E\}$
    2. $E \to D$ is non-redundant $E^+ = E$ w.r.t. $G - \{E \to D\}$
    3. $A \to B$ is non-redundant $A^+ = A$ w.r.t. $G - \{A \to B\}$
    4. $AC \to D$ is redundant $AC^+ = ABCDE$ w.r.t. $G - \{AC \to D\}$
  - Minimal cover is $G = \{AC \to E, E \to D, A \to B\}$

Summary

- Functional dependencies
  - Abstraction of "*uniquely identifies*"
  - Superkey $a \rightarrow R$
  - Key (candidate key) a is also minimal
  - Prime attribute $A \in a$ is in some key $a \rightarrow R$
- Armstrong's axiom
  - Reflexivity, Augmentation, Transitivity
  - Repeated application of axiom ⇒ FD closure and attribute closure
- Attribute closure
  - Given an attribute, what other attributes can we uniquely identify?
  - Algorithm #1
- Minimal cover
  - Find a simpler set G such that $G \equiv F$ and no redundancy
  - Algorithm #2