HALMSTAD UNIVERSITY

DS4001 Databases (7.5 credits)

Lecture 10 – Functional Dependencies

Yuantao Fan

yuantao.fan@hh.se

Halmstad University

Overview

- Functional Dependencies (FDs)
- An alternative approach to modeling and design databases
  - Generate schema based on the domain description given
- Inferring FDs, computing closure, mininal cover
- Database Normalisation
  - Design based on attributes and formal statements extracted from the domain desc.

Domain descriptioin → ER Model or Functional Dependencies → Schema → SQL code Relational DBMS

Functional Dependencies (FD)

- Recall that a relation R is a subset of the cartesian product of two or more sets
  - $R \subseteq A_1 \times A_2 \times \ldots \times A_n$
  - Equivalent notations
    - $R(a_1, a_2, \ldots, a_n)$
    - $(a_1, a_2, \ldots, a_n) \in R$

- Let $R(a_1, a_2, \ldots, a_n)$ be a relation schema, $S = \{a_1, a_2, \ldots, a_n\}$ is the attributes set of R
  - And $X, A \subseteq S$

- Functional Dependency X determines A, denoted X –> A
  - If and only if all rows t, u $\in R(\ldots)$, if t.X = u.X then t.A = u.A
  - Written as <set of attributes> -> <attribute>
    - e.g. room and time determine course, i.e., time room -> course

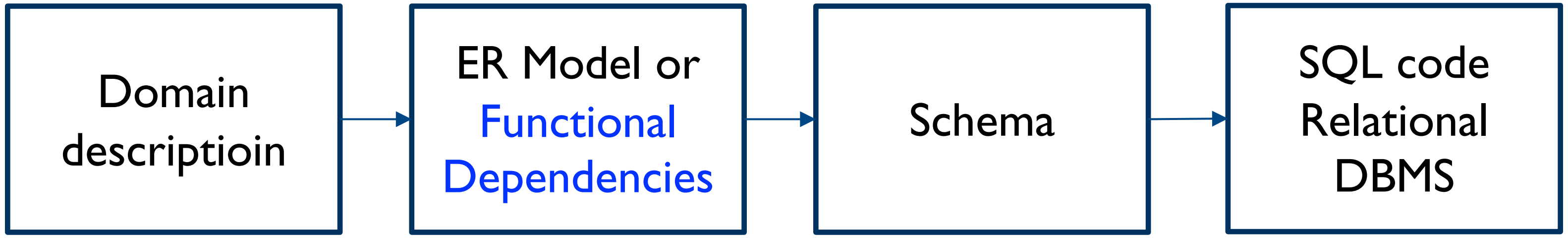
- Example: a b -> c
  - If two rows agree on the values of the attributes a and b, then they should also agree on the values of the attributes c
  - a and b uniquely determine the values of c

Inferring FDs

- Convention
  - Capital letters – sets of attributes
  - Lower case letters – single attributes
- a -> b c
  - a determines both b and c
  - Equivalent to a->b, a->c
- a b -> c
  - a and b togeter determind c
  - Not the same as a -> c, b-> c

Solution via FD?

- Decomposing tables to removed redundancies
  - No functional dependenies between attributes in the same table
    - i.e. FD (X –> Y) does not connect columns in the same table
    - Avoid anomalies previously mentioned
- Note that data loss must be avoided via decomposition
  - Reconstruction based on FD
  - Recompose by looking up X for Y values
- Decomposing table may require additional joins for querying the data



Finding Functional Dependencies (FDs)

- Common mistake
  - Try finding FDs by looking at data
- Data only captures current state of the database
  - Not all functional dependencies may appear
  - Data may suggest misleading “pseudo FDs”
- Two valid sources for mining FDs:
  - Domain knowledge
  - Inferring new FDs from given FDs

Properties of FDs

- Let $R(a_{1,}\ a_{2,}\ \ldots, a_n)$ be a relation schema, $S = \{a_{1,}\ a_{2,}\ \ldots, a_n\}$ is the attributes set of R, and X, Y, Z, A ⊆ S
  - Augmentation
    - If X -> Y then XZ -> YZ for any Z
  - Trasitivity
    - If X→ Y and Y→ Z then X -> Z
  - Reflexivity
    - (trivial) X –> X or {X, Y} -> X
    - (Reflixivity + augmentation) if B is a subset of A then A –> B is implied
  - Closure: X+ = {a|X->a} the set of all attributes determine by X
  - Superkey: X such that S ⊆ X+
  - (Minimal) Key: Minimal superkey, removing any attribute to the set will make it a non-superkey
    - Good candidate for primary key

Properties of FDs

- Let R(a_1, a_2, …, a_n) be a relation schema, S = {a_1, a_2, …, a_n} is the attributes set of R, and X, Y, Z, A ⊆ S
  - Augmentation
    - If X -> Y, then XZ -> YZ for any Z
  - Trasitivity
    - If X -> Y and Y -> Z, then X -> Z
  - Reflexivity
    - (trivial) X –> X or {X, Y} -> X
    - (Reflixivity + augmentation) if B is a subset of A then A –> B is implied
  - Decomposition (Derived)
    - If X -> YZ, then X -> Y and X –> Z
  - Composition (Derived)
    - If X-> Y AND A -> B, then XA -> YB
  - Union (Derived)
    - If X -> Y and X -> Z, then X -> YZ
  - Pseudo-transitivity (Derived)
    - If X -> Y and WY -> Z, then WX -> Z
  - Closure: X+ = {a|X->a} the set of all attributes determine by X
  - Superkey: X such that S ⊆ X+
  - (Minimal) Key: Minimal superkey, removing any attribute to the set will make it a non-superkey
    - Good candidate for primary key

Example: Inferring FDs

- F = {
  {Course} -> {Lecture},
  {Course} -> {Department},
  {Lecture, Department} -> {Office}
}

- FDs inferred from F:
  - {Course, Department} -> {Department}
  - {Course, Lecturer} -> {Department, Lecturer}
  - {Course} -> {Office}

Example: Inferring FDs

- Given the FD:
  - x -> y
  - z -> w
  - y w -> q
- Derive FD: x z -> q

Example: Inferring FDs

- Given the FD:
  - x -> y
  - z -> w
  - y w -> q

- Derive FD: x z -> q

- xz -> y : derive from x -> y using augmentation
- xz -> w : derive from z -> w using augmentation
- xz -> yw : merging the above two FDs
- xz -> q : xz -> yw and y w -> q using transitivity

Example: Compute the Closure

- Given the FDs
  - x -> y
  - z -> w
  - y w -> q
  - q -> x
  - m -> n
- Compute {x, z}+

Example: Computing the Closure

- Given the FDs
  - x -> y
  - z -> w
  - y w -> q
  - q -> x
  - m -> n
- Compute {x, z}+
  - {x, z} ⊆ {x, z}+
  - {x, z, y, w} ⊆ {x, z}+ : adding y and w based on the first two FDs
  - {x, z, y, w, q} ⊆ {x, z}+ : adding q, since y w -> q (the thrid FD)
  - Done
- Non-trivial FD in this closure: x z -> y w q or
  - x z -> y
  - x z -> w
  - x z -> q

Functional Dependencies Usage

- Check if a specific dataset hold a set of FDs
- Check if a specific design or schema ensures the FDs hold for all data set that follows the schema
- Express desired properties a schema / database design should obtain

Identifying FDs in the following table

cid	course_name	day	time	room	nn_seats
4	Databases	Monday	13:15 - 15:00	D415	50
4	Databases	Wednesday	13:15 - 15:00	D415	50
3	Linear Algebra	Monday	10:15 - 12:00	D208	30
3	Linear Algebra	Tuesday	13:15 - 15:00	D208	30
4	Databases	Thursday	15:15 - 17:00	D315	30

- cid -> course_name
- day -> time
- day time room -> cid
- room -> nn_seats
- cid name day -> time room seats

Identifying FDs in the following table

cid	course_name	day	time	room	nn_seats
4	Databases	Monday	13:15 - 15:00	D415	50
4	Databases	Wednesday	13:15 - 15:00	D415	50
3	Linear Algebra	Monday	10:15 - 12:00	D208	30
3	Linear Algebra	Tuesday	13:15 - 15:00	D208	30
4	Databases	Thursday	15:15 - 17:00	D315	30

- Yes - cid -> course_name
- No - day -> time
- Yes - day time room -> cid
- Yes - room -> nn_seats
- Yes - cid course_name day -> time room seats

Identifying FDs in the following Schema / database design

Teachers(tid, name)
Courses(cid, course_name, teacher)
company-> Teachers.tid

- cid -> course_name
- course_name -> cid
- cid course_name –> tid
- cid tid -> name

Identifying FDs in the following Schema / database design

Teachers(<u>tid</u>, name)
Courses(<u>cid</u>, course_name, teacher)
company-> Teachers.tid

- Yes: cid -> course_name
- No: course_name -> cid
- Yes: cid course_name –> tid
- Yes: cid tid -> name

Keys and superkeys

- The property of a Key of a relation using FDs can be defined
- A set of attributes is a superkey if it determines all other attributes
- Formal definition
  - The attribute set X is a superkey of R if X+ contains all attributes of R

- X is a minimal key if removing any attribute from X makes it a non-superkey

Example: Inferring FDs, Computing Closures, Keys and Superkeys

- Schema
  - R(a, b, c)
- Functional Dependencies
  - a -> b
  - a -> c
- Closures
  - {a}+ = {a, b, c}
  - {b}+ = {b}
  - {c}+ = {c}
- Superkeys
  - {a}, {a, b}, {a, c}, {a, b, c}
- Key
  - {a}

Example: Inferring FDs, Computing Closures, Keys and Superkeys

- Schema
  - R(<u>a</u>, b, c)
- Functional Dependencies
  - a -> b
  - a -> c
  - b -> a
- Closures
  - {a}+ = {a, b, c}
  - {b}+ = {b, a, c}
  - {c}+ = {c}
- Superkeys
  - {a}, {b}, {a, b}, {a, c}, {a, b, c}, {b, c}
- Key
  - {a}, {b}

Example: Inferring FDs, Computing Closures, Keys and Superkeys

- Schema
  - R(a, b, col)
- Functional Dependencies
  - a -> b
  - b -> c
    - Implies a -> c (transitivity)
- Closures
  - {a}+ = {a, b, c}
  - {b}+ = {b, c}
  - {c}+ = {c}
- Superkeys
  - {a}, {a, b}, {a, c}, {a, b, c}
- Key
  - {a}

Minimal Cover

- Let L be a set of FDs
- Def: the minimal cover F- is a simplified but equivalent set of FDs that satisfies
  - F- contains no trivial dependencies
    - If Y is a subset of X, then X -> Y is trivial
  - No dependencies in F- follow from other dependencies in F- through transitivity or augmentations

Example: Derive F-

- Given the FD
  - a -> b
  - b -> c
  - a d -> b c d

- Deriving F-

Example: Derive F-

- Given the FD
  - a -> b
  - b -> c
  - a d -> b c d
- Deriving F-
  - a d –> d (trivial)
  - a d -> b (augmentation, implied by a -> b)
  - a d -> c (transitivity + augmentation, implied by a -> b and b -> c)
- Minimal cover F-
  - a -> b
  - b -> c

Deriving Minimal Cover

- To check if a functional dependency X -> Y can be derived from other FDs is to check whether X+ is a superkey when X -> Y is not taking into consideration

- Given the FDs
  - a -> b
  - b c -> d
  - a c -> d : can we derive this FD from the above FDs?

- Compute {a, c}+ = {a, b, c, d}
  - {a, c}+ is a superkey and therefore a c -> d can be derived from other FDs

Property – does F- $\subseteq$ F?

- The minimal cover F- does not have to be a subset of F

- Given the FDs
  - a c -> b
  - a -> c

- F-
  - a -> b
  - a -> c

- F- is not a subset of the FDs given

Summary

- A Functional dependency X->Y means that any rows that agree on X also agree on Y
- We can extend a set of FDs using Armstrong's axioms, i.e. augmentation, reflexivity, and transitivity
- Minimal cover F- of a set of FDs can be computed with trivial and derived FDs removed
- The closure X+ is the set of all attributes that can be determined by X
- A superkey is a set of attributes that determines all others in the relation
- Keys are minimal superkeys
- To find a key: start with all attributes (a trivial superkey) and remove attributes until it is a key (there can be alternative ones)

Normal Forms

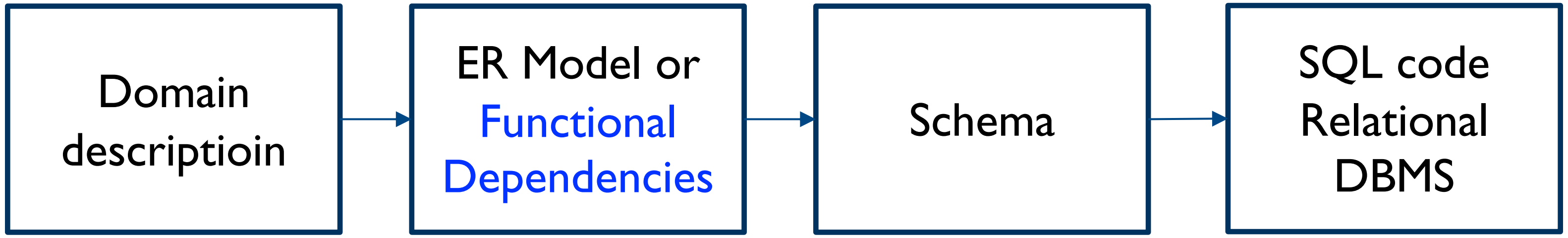
- 1NF
  - Each column has a single value
- 2NF
  - 1NF + has valid (single col. works) primary key
- 3NF
  - 2NF + no Functional Dependencies between attributes not in keys
- BCNF
  - 3NF + attributes depend only on keys
- 4NF:
  - 3NF + No violating Multiple Valued Dependencies

Constraint (informal description in parentheses)	UNF (1970)	1NF (1970)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Unique rows (no duplicate records)[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Scalar columns (columns cannot contain relations or composite values)[5]	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-prime attribute has a full functional dependency on a candidate key (attributes depend on the *complete* primary key)[5]	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (attributes depend *only* on the primary key)[5]	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute (a stricter form of 3NF)	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	–
Every non-trivial functional dependency begins with a superkey (a stricter form of 3NF)	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	–
Every non-trivial multivalued dependency begins with a superkey	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	–
Every join dependency has a superkey component[8]	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	–
Every join dependency has only superkey components	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	–
Every constraint is a consequence of domain constraints and key constraints	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Every join dependency is trivial	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Source: https://en.wikipedia.org/wiki/Database_normalization

Normalization

- Given the domain description, collect all attributes $R(a_1, a_2, \ldots, a_n)$ and a set of all FDs
- Nomalize R using F, or F-, to acquire a schema design
  - (Recursively) Decomposing R into R_1, R_2, …
- There are different normal forms and corresponding normalization algorithms
- BCNF (Boyce-Codd Normal Form)
  - Given a relational schema $R(a_1, a_2, \ldots, a_n)$, the non-trivial X -> Y (X is a subset of $\{a_1, a_2, \ldots, a_n\}$)is a BCNF violation if X is not a superkey ($\{a_1, a_2, \ldots, a_n\}$ is not a subset of X+)
  - A relational schema $R(a_1, a_2, \ldots, a_n)$ is in BCNF if for each non-trivial FD X -> Y, X is a superkey; if there is no BCNF violation



BCNF Algorithm

- Nomalizing R(S) with attribute set S= $\{a_{1,}\ a_{2,}\ \ldots, a_n\}$
  - Find a BCNF violation, i.e. a non-trivial FD X -> Y, such that X is not a superkey (X is a subset of S, but S is not a subset of X+)
  - If there is no such FD, then R is already in BCNF
  - Otherwise decompose R(S) into
    - R_1 (X+)
    - R_2 (X U (S – X+))
  - And normalize R_1 and R_2

Example: BCNF Normalization

- Given the FDs
  - cid -> course_name
  - room -> nn_seats
  - cid day time -> room
  - room day time -> cid
- Normalize R(cid, course_name, day, time, room, nn_seats)

cid	course_name	day	time	room	nn_seats
4	Databases	Monday	13:15 - 15:00	D415	50
4	Databases	Wednesday	13:15 - 15:00	D415	50
3	Linear Algebra	Monday	10:15 - 12:00	D208	30
3	Linear Algebra	Tuesday	13:15 - 15:00	D208	30
4	Databases	Thursday	15:15 - 17:00	D315	40

Example: BCNF Normalization

- Normalize R(cid, course_name, day, time, room, nn_seats)
- Start with cid -> course_name?

cid -> course_name
room -> nn_seats
cid day time -> room
room day time -> cid

Example: BCNF Normalization

- Normalize R(cid, course_name, day, time, room, nn_seats)
- Start with cid -> course_name?
  - cid -> course_name
  - X = {cid}
  - X+ = {cid, course_name}
  - R1(X+) = R1(cid, course_name)
  - R2(X U (S – X+)) = R2(cid, day, time, room, nn_seats)
- Continue with room -> nn_seat
  - X = {room}
  - X+ = {room, nn_seats}
  - R21(X+) = R(room, nn_seats)
  - R22 (X U (S – X+)) = R(cid, day, time, room)

cid -> course_name
room -> nn_seats
cid day time -> room
room day time -> cid

Example: BCNF Normalization

- Decomposed R into
  - R1(cid, course_name)
  - R21 (room, nn_seats)
  - R22 (cid, day, time, room)
- Can we further decompose R22?
  - Decompose with cid day time -> room ?
    - X = {cid, day, time}
    - X+ = {cid, day, time, room, nn_seats}
    - X is a subset of S and S is a subset of X+, so it is in BCNF form
  - Decompose with room day time -> cid?
    - X = {room, day, time}
    - X+ = {room, day, time, cid, nn_seats}
    - X is a subset of S and S is a subset of X+, so it is in BCNF form
- Therefore, R22 is in BCNF form

cid -> course_name
room -> nn_seats
cid day time -> room
room day time -> cid

Example: BCNF Normalization

- Given FDs
  - cid -> course_name
  - room -> nn_seats
  - cid day time -> room
  - room day time -> cid

- We obtained Schema
  - R1(cid, course_name)
  - R21(room, nn_seats)
  - R22(cid, day, time, room)

- Both {cid day time} and {room day time} are keys

Example: BCNF Normalization

- Given FDs
  - cid -> course_name
  - room -> nn_seats
  - cid day time -> room
  - room day time -> cid
- We obtained Schema
  - R1(cid, course_name)
  - R21(room, nn_seats)
  - R22(cid, day, time, room)
- Both {cid day time} and {room day time} are keys

R1 - Courses

cid	course_name
3	Linear Algebra
4	Databases

R21 - Rooms

room	nn_seats
D415	50
D208	30
D315	40

R22 - Schedules

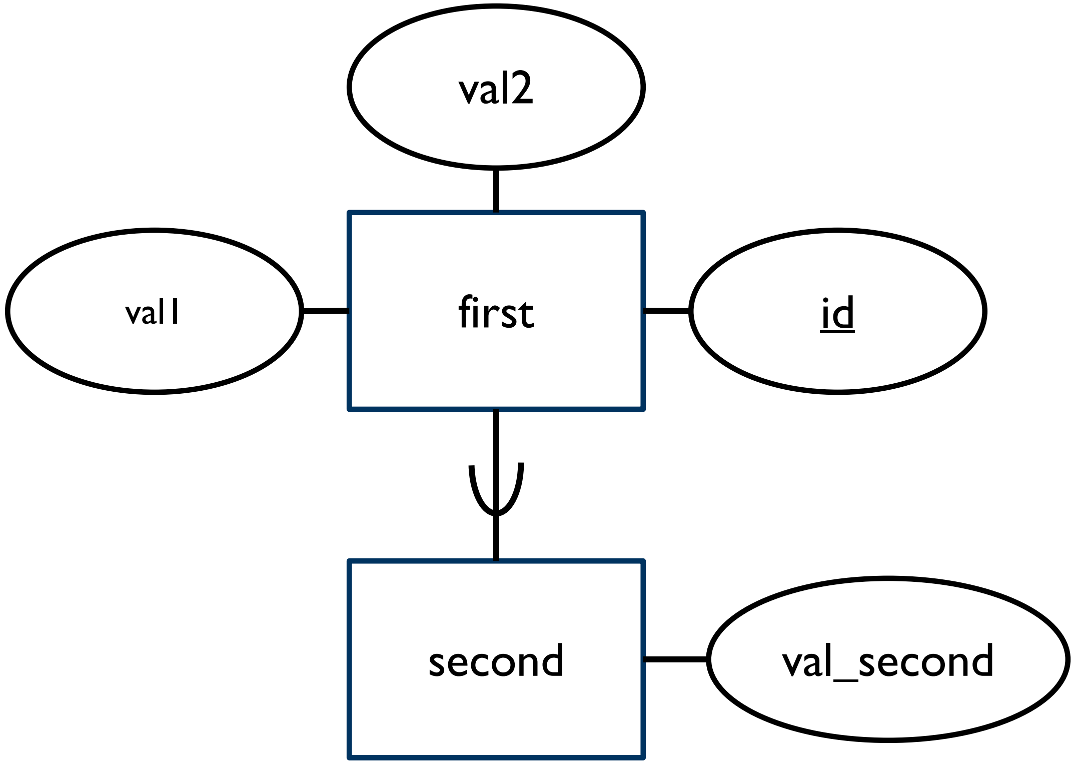
cid	day	time	room
4	Monday	13:15 - 15:00	D415
4	Wednesday	13:15 - 15:00	D415
3	Monday	10:15 - 12:00	D208
3	Tuesday	13:15 - 15:00	D208
4	Thursday	15:15 - 17:00	D315

“ISA” Inheritance in MySQL

```
CREATE TABLE first (
  id tinyint(4) NOT NULL,
  val1 real,
  val2 real,
  PRIMARY KEY (id)
);
```

```
CREATE TABLE second (
  parent integer REFERENCES First,
  val_second real,
  PRIMARY KEY (parent)
);
```

```
INSERT INTO first VALUES (1,11,22);
INSERT INTO second VALUES (1, 33);
```



first(<u>id</u>, val1, val2)
second(<u>sid</u>, val_second)
sid->first.id