# Databases

# TDA357/DIT621– LP3 2023

Lecture 6

Ana Bove

(much of the material is based on material from both Thomas Hallgren and Jonas Duregård)

January 30th 2023

# Recall Last Lecture

- Entity-Relationship modelling:
    - Entities and attributes;
    - Many-to-many relationships;
    - Many-to-exactly-one relationships;
    - Many-to-at-most-one relationships;
    - Multiway relationships;
    - Self-relationships;
    - Weak entities;
    - ISA relationships;
- (ER-example/exercise.)

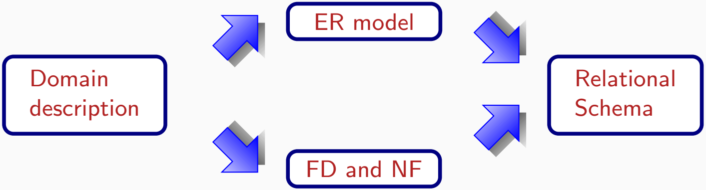
# Overview of Today's Lecture

- Functional dependencies:
  - Reflexivity, transitivity and augmentation;
  - Closures;
  - Superkeys and keys;
  - Minimal basis/cover;
- Boyce-Codd Normal form (BCNF) ..
- ... and its normalisation algorithm.

# Functional Dependencies and Normal Forms

This is an alternative, and in some way complementary, approach to model a database.

We start with a domain description and end up with a database schema.




Here we work as follows:

# Relations, Relation Schemas and Tables

**Recall:** A relation $S$ is a subset of the cartesian product of two or more sets $T_1, T_2, \ldots, T_n$:

$$S \subseteq T_1 \times T_2 \times \cdots \times T_n$$

Given a relation schema $R(a_1, \ldots, a_n)$, consider the *domain/type* of each attribute $a_1 : T_1, \ldots, a_n : T_n$.

The *relation signature* of the relation $R$ is the corresponding cartesian product $T_1 \times \cdots \times T_n$.

So, given a relation schema $R(a_1, \ldots, a_n)$ with signature $T_1 \times \cdots \times T_n$:

- A *table* for the schema $R(a_1, \ldots, a_n)$ is a subset of the cartesian product $T_1 \times \cdots \times T_n$;
- A *row* in the table is an element of the cartesian product $t \in T_1 \times \cdots \times T_n$.

# Attribute Names vs Tuple Components

**Recall:** The elements of a cartesian product $T_1 \times \cdots \times T_n$ are tuples $t = \langle t_1, \ldots, t_n \rangle$.

The *i*th projection $\pi_i\, t$ gives us the element $t_i$ (of type $T_i$).

If $t$ is a row in the table for the schema $R(\ldots, a, \ldots)$ (containing attribute $a$), we will:

- Assume an *indexing* function $i$ giving us the index/position of each attribute;
- Denote $t.a = t_{i(a)}$ the *projection* $\pi_{i(a)}\, t$;
- If $A = \{a_1, \ldots, a_n\}$ is a set of attributes in $R(\ldots)$, then $t.A = \langle t.a_1, \ldots, t.a_n \rangle$ is the *simultaneous projection*.

# Small Example

```
CREATE TABLE Countries (
  name TEXT ...,
  abbr CHAR(2) ...,
  area FLOAT ... );
```

*Relation schema:*
Countries (name, abbr, area)

*Relation signature:*
TEXT $\times$ CHAR(2) $\times$ FLOAT

| **name** | **abbr** | **area** |
|---|---|---|
| Denmark | DK | 43094 |
| Sweden | SE | 449964 |
| Norway | NO | 324220 |
| ... | ... | ... |

Let $t = \langle$ 'Denmark', 'DK', 43094 $\rangle$

$t$.name = 'Denmark'

If $A = \{$abbr, area$\}$ then
$t.A = \langle$ 'DK', 43094 $\rangle$

# Functional Dependencies (FD)

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

**Definition:** (*Functional Dependency*) $X$ *determines* $A$, denoted $X \to A$, iff for all rows $t, u \in R(\ldots)$, if $t.X = u.X$ then $t.A = u.A$.

**Example:** Suppose we have $a\,b \to c$.
What does this mean?

It means that if two rows agree on the values of the attributes $a$ and $b$, then they should also agree on the values of the attributes $c$.

Hence, the values of $a$ and $b$ *uniquely determine* the values of $c$.

# Some Words on Notation

Capital vs small letters: We will use capital letters $A, \ldots, X, Y, Z$ to denote sets of attributes and small letters $a, b, c, \ldots, z$ to denote single attributes (unless otherwise stated).

$a \rightarrow b\,c$: means that $a$ *determines both* $b$ and $c$:

$$\boxed{a \rightarrow b\,c} \quad \text{is the same as} \quad \boxed{\begin{array}{c} a \rightarrow b \\ a \rightarrow c \end{array}}$$

$a\,b \rightarrow c$: means that $a$ and $b$ *together determine* $c$:

$$\boxed{a\,b \rightarrow c} \quad \text{is NOT the same as} \quad \boxed{\begin{array}{c} a \rightarrow c \\ b \rightarrow c \end{array}}$$

# Properties of Functional Dependencies

Let $R(a_1, \ldots, a_n)$ be a relation schema, $S = \{a_1, \ldots, a_n\}$ the set of attributes of $R$, and $X, Y, Z, A \subseteq S$.

**Transitivity:** If $X \to Y$ and $Y \to Z$ then $X \to Z$;

**Augmentation:** If $X \to A$ and $a$ is an attribute, then $X\,a \to A$ and $a\,X \to A$;

**Reflexivity:** (trivial dependency) $X \to X$;

**Reflexivity + augmentation** (trivial dependency) If $Y \subseteq X$ then $X \to Y$.

**Closure:** $X^+ = \{a \mid X \to a\}$, the set of all attributes determined by $X$;

**Superkey:** $X$ such that $S \subseteq X^+$;
($X$ is a superkey if $X^+$ includes all the attributes in $R(\ldots)$)

**(Minimal) Key:** Minimal superkey (and good candidate for primary key!); removing an attribute to the set will make it a non-superkey.

# Example: Deriving FD

Given the FD:

$$\boxed{\begin{array}{l} x \rightarrow y \\ z \rightarrow w \\ y\,w \rightarrow q \end{array}}$$

we can derive the FD: $\boxed{x\,z \rightarrow q}$

- $x\,z \rightarrow y$: from $x \rightarrow y$ by augmentation;
- $x\,z \rightarrow w$: from $z \rightarrow w$ by augmentation;
- $x\,z \rightarrow y\,w$: by merging left-hand side of FD (see words on notation);
- $x\,z \rightarrow q$: by transitivity of $x\,z \rightarrow y\,w$ and $y\,w \rightarrow q$.

# Example: Computing the Closure

Given the FD: $\boxed{\begin{array}{l} x \to y \\ z \to w \\ y\,w \to q \\ q \to x \\ r \to s \end{array}}$ compute: $\boxed{\{x, z\}^+}$

- $\{x, z\} \subseteq \{x, z\}^+$: we start from trivial FD;
- $\{x, z, y\} \subseteq \{x, z\}^+$: we add $y$ because $x \to y$;
- $\{x, z, y, w\} \subseteq \{x, z\}^+$: we add $w$ because $z \to w$;
- $\{x, z, y, w, q\} \subseteq \{x, z\}^+$: we add $q$ because $y\,w \to q$;
- $\{x, z, y, w, q\} = \{x, z\}^+$: nothing else to add.

The closure gives us the non-trivial FD: $\boxed{\begin{array}{l} x\,z \to y \\ x\,z \to w \\ x\,z \to q \end{array}}$ or $\boxed{x\,z \to y\,w\,q}$

(Recall: if $Y \subseteq X$ then $X \to Y$ trivial)

# Uses of Functional Dependencies

There are three ways we can use functional dependencies:

- Check if they hold for a specific data set;
- Check if a specific design/relational schema ensures the FD hold for all data set that follows the schema;
- Express desired properties a design/relational schema should have.

  (This use is what makes FD a design tool, and what we will concentrate on in (most of) the rest of this lecture.)

# Example: Which FD Hold for this Data?

| code | name | day | time | room | seats |
|---|---|---|---|---|---|
| TMV028 | Automata | Monday | 10 | HB1 | 108 |
| TDA357 | Databases | Monday | 15 | HC4 | 104 |
| TMV028 | Automata | Tuesday | 13 | HB1 | 108 |
| TDA357 | Databases | Wednesday | 10 | HC2 | 115 |
| TDA357 | Databases | Thursday | 10 | HC4 | 104 |

code $\rightarrow$ name: YES

day $\rightarrow$ time: NO

day time room $\rightarrow$ code: YES

room $\rightarrow$ seats: YES

code name day $\rightarrow$ time room seats: YES

# Example: Which FD Hold for this Design?

Teachers (<u>name</u>, email)
Courses (<u>code</u>, cname, teacher)
teacher $\rightarrow$ Teachers.name

Does the relational schema guarantee:

code $\rightarrow$ cname: YES

cname $\rightarrow$ code: NO

code cname $\rightarrow$ teacher: YES

code teacher $\rightarrow$ email: YES

# Example: Deriving Closures, Keys and Superkeys (I)

Relation schema: Countries (country, capital, currency)

Functional dependencies:

country $\rightarrow$ capital
country $\rightarrow$ currency

Closures:

$\{\text{country}\}^+ = \{\text{country, capital, currency}\}$
$\{\text{capital}\}^+ = \{\text{capital}\}$
$\{\text{currency}\}^+ = \{\text{currency}\}$

Superkeys:

{country}, {country, capital}, {country, currency},
{country, capital, currency}

Key: {country}

# Example: Deriving Closures, Keys and Superkeys (II)

Relation schema: Countries (country, capital, currency)

Functional dependencies:

country $\rightarrow$ capital
country $\rightarrow$ currency
capital $\rightarrow$ country
(by transitivity: capital $\rightarrow$ currency)

Closures:

$\{\text{country}\}^+ = \{\text{country, capital, currency}\}$
$\{\text{capital}\}^+ = \{\text{capital, country, currency}\}$
$\{\text{currency}\}^+ = \{\text{currency}\}$

Superkeys: {country}, {capital}, {country, capital}, {country, currency}, {capital, currency}, {country, capital, currency}

Key: {country}, {capital}

# Example: Deriving Closures, Keys and Superkeys (III)

Relation schema: Countries (country, currency, value)

Functional dependencies:

country $\rightarrow$ currency
currency $\rightarrow$ value
(by transitivity: country $\rightarrow$ value)

Closures:

$\{$country$\}^+ =$ $\{$country, currency, value$\}$
$\{$currency$\}^+ =$ $\{$currency, value$\}$
$\{$value$\}^+ =$ $\{$value$\}$

Superkeys:

$\{$country$\}$, $\{$country, currency$\}$, $\{$country, value$\}$, $\{$country, currency, value$\}$

Key: $\{$country$\}$

# Example: Deriving Closure, Keys and Superkeys (III, Cont.)

If <u>country</u> is the primary key, then the dependency

$$\boxed{\text{country} \to \text{currency}}$$

will be trivially satisfied.

But how to ensure that

$$\boxed{\text{currency} \to \text{value}}$$

is satisfied?
This is a non-trivial FD and {currency, value} is not a superkey!

**Note:** FD help us identify a problematic schema!

# Minimal Basis/Cover $F^-$

Let $F$ be a set of functional dependencies.

**Definition:** The *minimal basis or minimal cover $F^-$* is a simplified but equivalent set of functional dependencies such that:

- $F^-$ contains no trivial dependencies (if $Y \subseteq X$ then $X \to Y$ trivial);
- No dependencies in $F^-$ follow from other dependencies in $F^-$ through transitivity or augmentations.

# Example: Deriving Minimal Basis/Cover

Given the FD:

$$\boxed{\begin{array}{l} a \rightarrow b \\ b \rightarrow c \\ a\,d \rightarrow b\,c\,d \end{array}}$$

we can compute the minimal basis by removing:

- $a\,d \rightarrow d$: because it is trivial;
- $a\,d \rightarrow b$: because it can be computed from $a \rightarrow b$ by augmentation;
- $a\,d \rightarrow c$: because it can be computed from $a \rightarrow b$ and $b \rightarrow c$ by transitivity, and then augmentation.

Minimal basis/cover:

$$\boxed{\begin{array}{l} a \rightarrow b \\ b \rightarrow c \end{array}}$$

# Deriving Minimal Basis/Cover

A way to see if a FD $X \to Y$ can be derived from the other FD is to check whether $X^+$ is a superkey when $X \to Y$ is not taken into account.

**Example:** Consider the FD:

$$\boxed{\begin{array}{l} a \to b \\ b\,c \to d \\ a\,c \to d \end{array}}$$

Is $a\,c \to d$ derived?

Let us compute $\{a, c\}^+$ from

$$\boxed{\begin{array}{l} a \to b \\ b\,c \to d \end{array}}$$

$$\{a, c\}^+ = \{a, b, c, d\}$$

That is, $\{a, c\}^+$ is a superkey and hence it should be possible to derived $a\,c \to d$ from the other FD.

# Minimal basis/cover: is $F^- \subseteq F$?

Given a set $F$ of FD, the minimal basis/cover $F^-$ does not have to be a subset of $F$.

Consider the FD: $\boxed{\begin{array}{c} a\,c \to b \\ a \to c \end{array}}$

Minimal basis/cover: $\boxed{\begin{array}{c} a \to b \\ a \to c \end{array}}$

since $a \to c$, then $a\,c \to b$ can actually be derived from $a \to b$ by augmentation.
Otherwise $a\,c \to b$ might hold but not $a \to b$!

The minimal basis is not included in the original set of FD.

# FD: Summary so Far

- A FD $X \to Y$ means that any rows that agree on $X$ also agree on $Y$;
- We can extend a set of FD with additional derived FD using transitivity, augmentation, and reflexivity;
- Conversely, we can reduce a set of FD to its minimal basis/cover by removing all trivial and derived FD;
- 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 (finding all keys is more work though).

# Normal Forms and Normalisation

- We work like this:
  - We start by collecting all the attributes in the domain and place them in one big relation schema $R(a_1, ..., a_n)$;
  - We collect the set $F$ with all the FD;
  - (We compute the minimal basis/cover $F^-$;)
  - We *normalise* $R(\ldots)$ using $F$ (alt. $F^-$) to get a good design.

- *Normalisation* is a recursive procedure.
  To normalise $R(\ldots)$:
  - We check if $R(\ldots)$ is already in *normal form*;
  - If not, we decompose $R(\ldots)$ into $R_1(\ldots)$ and $R_2(\ldots)$ and normalise them.

- There are several *normal form* definitions ...

- ... and several *normalisation* algorithms (depending on the normal form definitions).

# BCNF (Boyce-Codd Normal Form) and BCNF-violation

**Definition:** Given a relation schema $R(a_1, ..., a_n)$,
the non-trivial FD $X \to Y$ (with $X \subseteq \{a_1, ..., a_n\}$)
is a *BCNF-violation* if $X$ is not a superkey ($\{a_1, ..., a_n\} \not\subseteq X^+$).

**Definition:** A relation schema $R(a_1, ..., a_n)$ is in *BCNF* if for each non-trivial FD $X \to Y$, $X$ is a superkey ($X \subseteq \{a_1, ..., a_n\} \subseteq X^+$).

That is, if there are no BCNF-violations.

# BCNF Normalisation Algorithm

To *normalise* a relation schema $R(S)$ with $S = \{a_1, ..., a_n\}$:

- Find a *BCNF-violation*: that is, a non-trivial FD $X \rightarrow Y$ such that $X$ is not a superkey ($X \subseteq S \not\subseteq X^+$);
- If there is no such FD then $R$ is already in BCNF;
- Otherwise decompose $R(S)$ into $R_1(X^+)$ and $R_2(X \cup (S - X^+))$ and normalise them both.

  **Note:** $R(S)$ is of no interest anymore and has been replaced by $R_1(S_1)$ and $R_2(S_2)$!

# Example: BCNF Normalisation

Given the FD:

$$\boxed{\begin{array}{l}\text{code} \to \text{name} \\ \text{room} \to \text{seats} \\ \text{day time code} \to \text{room} \\ \text{day time room} \to \text{code}\end{array}}$$

normalise $\boxed{R(\text{code, name, day, time, room, seats})}$

Decompose using code $\to$ name?

$$X = \{\text{code}\},\ X^+ = \{\text{code}\}^+ = \{\text{code, name}\}$$

$$R_1(X^+) = \boxed{R_1(\text{code, name})} \text{ (in BCNF!)}$$

$$R_2(X \cup (S - X^+)) = R_2(\text{code, day, time, room, seats})$$

Decompose using room $\to$ seats?

$$X = \{\text{room}\},\ X^+ = \{\text{room}\}^+ = \{\text{room, seats}\}$$

$$R_{21}(X^+) = \boxed{R_{21}(\text{room, seats})} \text{ (in BCNF!)}$$

$$R_{22}(X \cup (S - X^+)) = R_{22}(\text{code, day, time, room})$$

# Example: BCNF Normalisation (Cont.)

Can $R_{22}$(code, day, time, room) be decomposed further?

We look at the remaining FD:

Decompose using day time room $\rightarrow$ code?

$$X = \{\text{day, time, room}\}$$
$$X^+ = \{\text{day, time, room}\}^+ = \{\text{day, time, room, code, seats}\}$$

Decompose using day time code $\rightarrow$ room?

$$X = \{\text{day, time, code}\}$$
$$X^+ = \{\text{day, time, code}\}^+ = \{\text{day, time, room, code, name}\}$$

So $\boxed{R_{22}\text{(code, day, time, room)}}$ is in BCNF!

Both {day, time, room} and {day, time, code} are keys!

# Identifying the Keys, Uniqueness Constrains and References

FD:

code → name
room → seats
day time code → room
day time room → code

Relational Schema:

$R_1$(code, name)
$R_{21}$(room, seats)
$R_{22}$(code, day, time, room)

Keys can be determined using the FD and the normalisation algorithm.

(A (minimal) $X$ that determines all the attributes in the relation schema is a key!)

We get 2 possible solutions:

$R_1$(<u>code</u>, name)
$R_{21}$(<u>room</u>, seats)
$R_{22}$(<u>code</u>, <u>day</u>, <u>time</u>, room)
  code → $R_1$.code
  room → $R_{21}$.room
  Unique (day, time, room)

$R_1$(<u>code</u>, name)
$R_{21}$(<u>room</u>, seats)
$R_{22}$(code, <u>day</u>, <u>time</u>, <u>room</u>)
  code → $R_1$.code
  room → $R_{21}$.room
  Unique (day, time, code)

**Note:** We need to keep track of the splitting for the references!

# BCNF Decomposition Avoids Data Redundancy (based on FD)

$R$: CourseBookings

| code | name | day | time | room | seats |
|---|---|---|---|---|---|
| TMV028 | Automata | Monday | 10 | HB1 | 108 |
| TDA357 | Databases | Monday | 15 | HC4 | 104 |
| TMV028 | Automata | Tuesday | 13 | HB1 | 108 |
| TDA357 | Databases | Wednesday | 10 | HC2 | 115 |
| TDA357 | Databases | Thursday | 10 | HC4 | 104 |

$R_1$: Courses

| code | name |
|---|---|
| TMV028 | Automata |
| TDA357 | Databases |

$R_{21}$: Rooms

| room | seats |
|---|---|
| HB1 | 108 |
| HC2 | 115 |
| HC4 | 104 |

$R_{22}$: Bookings

| code | day | time | room |
|---|---|---|---|
| TMV028 | Monday | 10 | HB1 |
| TDA357 | Monday | 15 | HC4 |
| TMV028 | Tuesday | 13 | HB1 |
| TDA357 | Wednesday | 10 | HC2 |
| TDA357 | Thursday | 10 | HC4 |

# Lossless Join: All Data Back!

$R_1$: Courses

| code | name |
|---|---|
| TMV028 | Automata |
| TDA357 | Databases |

$R_{21}$: Rooms

| room | seats |
|---|---|
| HB1 | 108 |
| HC2 | 115 |
| HC4 | 104 |

$R_{22}$: Bookings

| code | day | time | room |
|---|---|---|---|
| TMV028 | Monday | 10 | HB1 |
| TDA357 | Monday | 15 | HC4 |
| TMV028 | Tuesday | 13 | HB1 |
| TDA357 | Wednesday | 10 | HC2 |
| TDA357 | Thursday | 10 | HC4 |

Query: $R_1$ NATURAL JOIN $R_{21}$ NATURAL JOIN $R_{22}$

| code | name | day | time | room | seats |
|---|---|---|---|---|---|
| TMV028 | Automata | Monday | 10 | HB1 | 108 |
| TDA357 | Databases | Monday | 15 | HC4 | 104 |
| TMV028 | Automata | Tuesday | 13 | HB1 | 108 |
| TDA357 | Databases | Wednesday | 10 | HC2 | 115 |
| TDA357 | Databases | Thursday | 10 | HC4 | 104 |

# Overview of Next Lecture

- FD examples;
- BCNF decomposition and dependency preservation;
- Multivalued dependencies (MVD);
- 4NF and its normalisation algorithm;
- MVD examples.

**Reading:**

Book: chapter 3

Notes: chapters 4.1–4.3 and 5