Database Technology

Teaching Session 1: Functional Dependencies and Normalization

Valentina Ivanova

Slides from Olaf Hartig

LINKÖPING UNIVERSITY

Motivation

- How can we be sure that the translation of an EER diagram into a relational schema results in a good database design?
- Given a deployed database, how can we be sure that it is well-designed?
- **What *is* a good database design?**
  - Informal measures
  - Formal measure: *normal forms*
    - Definition based on functional dependencies

Example of Bad Design

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

- Every tuple contains employee data and department data
- Redundancy
  - Dname and Dmgr_ssn repeated for every employee in a department
- Potential for too many NULL values
  - Employees not in any department need to pad tuples with NULLs
- Update anomalies
  - Deleting the last employee in a department will result in deleting the department
  - Changing the department name or manager requires many tuples to be updated
  - Inserting employees requires checking for consistency of its department name and manager

Informal Measures

- Easy-to-explain meaning for each relation schema
  - Each relation schema should be about only one type of entities or relationships
  - Natural result of good ER design
- Minimal redundant information in tuples
  - Avoids update anomalies
  - Avoids wasted space
- Minimal number of NULL values in tuples
  - Avoids inefficient use of space
  - Avoids costly outer joins
  - Avoids ambiguous interpretation (e.g., unknown vs. does not apply)

Functional Dependencies (FDs) – Idea

- Assume that no two actors have the same name, i.e., name is unique
- Each actor has one year and city of birth
- Thus, given an actor's name, there is only one possible value for birth year and city
  - name → yearOfBirth
  - name → cityOfBirth

Actor		
name	**yearOfBirth**	**cityOfBirth**
Ben Affleck	1972	Berkeley
Alan Arkin	1934	New York
Tommy Lee Jones	1946	San Saba
John Wells	1957	Alexandria
Steven Spielberg	1946	Cincinnati
Daniel Day-Lewis	1957	Greenwich

- However, given a birth year, we do not have a unique corresponding name or city
  - ~~yearOfBirth → name~~
  - ~~yearOfBirth → city~~
- Cannot tell from the example whether city determines name or birth year

Functional Dependencies (FDs) – Definition

- Constraint between two sets of attributes from a relation

Let R be a relational schema with the attributes $A_1, A_2, \ldots, A_n$ and let X and Y be subsets of $\{A_1, A_2, \ldots, A_n\}$.

Then, the functional dependency $X \rightarrow Y$ specifies the following constraint on *any* valid relation state r of R.

For *any* two tuples t_1 and t_2 in state r we have that:

$$\text{if } t_1[X] = t_2[X], \text{ then } t_1[Y] = t_2[Y] .$$

  - where $t[X]$ is the sequence of values that the tuple t has for the attributes in set X
  - We say "X determines Y" or "Y depends on X"

Trivial Functional Dependencies

- Some dependencies must always hold
  - {name, yearOfBirth} → {name, yearOfBirth}
  - {name, yearOfBirth} → {name}
  - {name, yearOfBirth} → {yearOfBirth}
- Formally:
  - Let R be a relation schema, and
  - let X and Y be subsets of attributes in R.
  - If Y is a subset of X, then $X \rightarrow Y$ holds trivially.

Reasoning About FDs

- Logical implications can be derived by using inference rules called **Armstrong's rules**:
  - *Reflexivity:* If Y is a subset of X, then $X \rightarrow Y$
  - *Augmentation:* If $X \rightarrow Y$, then $XZ \rightarrow YZ$
    (we use XY as a short form for $X \cup Y$)
  - *Transitivity:* If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- Additional rules can be derived:
  - *Decomposition:* If $X \rightarrow YZ$, then $X \rightarrow Y$
  - *Union:* If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Pseudo-transitivity:* If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

Warmup

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**
- Use the Armstrong rules to derive the following FD: **AC** → **D**
- Use the Armstrong rules to derive the following FD: **A** → **D**

Warmup – Solutions

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:

  FD1: **A** → **BC**

  FD2: **C** → **AD**

  FD3: **DE** → **F**

- Use the Armstrong rules to derive the following FD: **AC** → **D**
  - FD4: **AC** → **AD** (Augmentation of FD2 with A)
  - FD5: **AC** → **D** (Decomposition of FD4)

- Use the Armstrong rules to derive the following FD: **A** → **D**
  - FD6: **A** → **C** (Decomposition of FD1)
  - FD7: **A** → **AD** (Transitivity of FD6 and FD2)
  - FD8: **A** → **D** (Decomposition of FD7)

Warmup (cont'd)

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: $\mathbf{A} \rightarrow \mathbf{BC}$
  FD2: $\mathbf{C} \rightarrow \mathbf{AD}$
  FD3: $\mathbf{DE} \rightarrow \mathbf{F}$
- Use the Armstrong rules to derive the following FD: $\mathbf{AE} \rightarrow \mathbf{ABCDEF}$

Warmup (cont'd)

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:

  FD1: **A → BC**

  FD2: **C → AD**

  FD3: **DE → F**

- Use the Armstrong rules to derive the following FD: **AE → ABCDEF**
  - FD9: **AE → BCE** (Augmentation of FD1 with E)
  - FD10: **AE → C** (Decomposition of FD9)
  - FD11: **AE → AD** (Transitivity of FD10 and FD2)
  - FD12: **AE → ADE** (Augmentation of FD11 with E)
  - FD13: **AE → DE** (Decomposition of FD12)
  - FD14: **AE → F** (Transitivity of FD13 and FD3)
  - FD15: **AE → ABCDEF** (Union of FD9, FD11, and FD14)

Revisiting Keys

- Given a relation schema R with attributes $A_1, A_2, \ldots, A_n$ and X a subset of these attributes
- X is a **superkey** of R if $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ or, $X \rightarrow \{A_1, A_2, \ldots, A_n\} \setminus X$
  - Often written as $X \rightarrow R$
- Given a set of FDs, how can we easily test whether $X \rightarrow R$?

> Let Σ be a set of FDs over the attributes of a relation R and let X be a subsets of these attributes.
>
> The **attribute closure** of X w.r.t. Σ is the maximum set of attributes functionally determined by X.

  - If the attribute closure of X contains all attributes, we have $X \rightarrow R$
  - The attribute closure can be computed in polynomial time ...

Computing (Super)Keys

X – set with attributes

Σ – set with functional dependencies

function *ComputeAttrClosure*(X, Σ)
begin
    $X^+ := X$;
    while Σ contains an FD $Y \rightarrow Z$ such that
        (i) Y is a subset of X^+, and
        (ii) Z is not a subset of X^+ **do**
    $X^+ := X^+ \cup Z$;
    end while
    return X^+;
end

Revisiting Keys (cont'd)

- Given a relation schema R with attributes $A_1, A_2, \ldots, A_n$ and X a subset of these attributes
- X is a **superkey** of R if $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ or, $X \rightarrow \{A_1, A_2, \ldots, A_n\} \setminus X$
  - Often written as $X \rightarrow R$
  - Can be tested easily by computing the attribute closure of X
- However, not every superkey is a candidate key
- To determine that X is a **candidate key** of R, we also need to show that no proper subset of X determines R
  - i.e., there does not exist a Y such that $Y \subsetneq X$ and $Y \rightarrow R$
- Hence, identifying *all* candidate keys is a matter of testing increasingly smaller subsets of $\{ A_1, A_2, \ldots, A_n \}$

Warmup (cont'd)

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**
- Determine **all candidate keys** of **R**
  - Compute the **attribute closure** of X = { **A,E,C,D** } w.r.t. Σ = {FD1, FD2, FD3}
    to show that we have: **AECD** → **ABCDEF**

Warmup (cont'd)

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**
- Determine **all candidate keys** of **R**
  - Compute the **attribute closure** of X = { **A,E,C,D** } w.r.t. Σ = {FD1, FD2, FD3}
    to show that we have: **AECD** → **ABCDEF**
  - Initially: X^+ = { **A,E,C,D** }
  - For FD1: X^+ = { **A,E,C,D,B** } A in X^+ thus add B (C already included)
  - For FD2: X^+ = { **A,E,C,D,B** } C in X^+, A and D already in the set thus do nothing
  - For FD3: X^+ = { **A,E,B,C,D,F** } D and E in X^+ thus add F

Warmup (cont'd)

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**
- Determine **all candidate keys** of **R**
  - **AECD is a superkey** → find the minimal superkeys (i.e. the candidate keys) by computing the attribute closure for all of the subsets
  - **AE and CE** are **all candidate keys**

Decomposition into 2NF

2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

- Let $X \rightarrow Y$ be the FD that violates 2NF in a relation schema *R*
- Create a new relation schema *R2* with the attributes in *X* and in *Y*
- Remove attributes *Y* from *R*
- New relational database schema consists now of *R1* and *R2*

R$_{\textbf{non2NF}}$

EmpID	***Dept***	**Work%**	**EmpName**
100	*Dev*	50	Baker
100	*Support*	50	Baker
200	*Dev*	80	Miller

FD1: EmpID → EmpName

FD2: { EmpID, Dept } → { Work%, EmpNar

Decompose

R1$_{\textbf{2NF}}$

EmpID	**EmpName**
100	Baker
200	Miller

R2$_{\textbf{2NF}}$

EmpID	***Dept***	**Work%**
100	*Dev*	50
100	*Support*	50
200	*Dev*	80

Decomposition into 3NF

3NF: Non-prime attributes must not be functionally dependent on a set of attributes that is not a superkey.

- Let $X \rightarrow Y$ be the FD that violates 3NF in a relation schema *R*
- Create a new relation schema *R2* with the attributes in *X* and in *Y*
- Remove attributes *Y* from *R*
- New relational database schema consists now of *R1* and *R2*

R_{non3NF}

ID	**Name**	**Zip**	**City**
100	Andersson	58214	Linköping
101	Björk	10223	Stockholm
102	Carlsson	58214	Linköping

FD1: Zip $\rightarrow$ City

FD2: ID $\rightarrow$ { Name, Zip, City }

Decompose

$R1_{3NF}$

ID	**Name**	**Zip**
100	Andersson	58214
101	Björk	10223
102	Carlsson	58214

$R2_{3NF}$

Zip	**City**
58214	Linköping
10223	Stockholm

Decomposition into BCNF

BCNF: For every FD $X \rightarrow Y$ it holds that X is a superkey of the relation that the FD is associated with.

- Let $X \rightarrow Y$ be the FD that violates BCNF in a relation schema *R*
- Create a new relation schema *R2* with the attributes in *X* and in *Y*
- Remove attributes *Y* from *R*
- New relational database schema consists now of *R1* and *R2*
- We may have to find a new primary key for *R*

- Example
  - Let *R*(<u>*A*</u>,<u>*B*</u>,*C*,*D*) be a relation schema with $AB \rightarrow CD$ and $C \rightarrow B$
  - *AB* is a candidate key, and so is *AC* !
  - *R* is in 3NF (*D* is the only non-prime attribute and AB)
  - *R* is not in BCNF (because *C* is not a candidate key)
  - By using $C \rightarrow B$, we decompose *R* into
    R1(<u>*A*</u>,<u>*C*</u>,*D*) with $AC \rightarrow D$ and
    R2(<u>*C*</u>,*B*) with $C \rightarrow B$

Normalization Algorithm

- Identify **all candidate keys** based on the given FDs
  - By using the algorithm to compute superkeys and then finding the minimal superkeys
- Given all the candidate keys, mark all **prime** and all **non-prime** attributes
- Choose one of the candidate keys as the **primary key**
- **2NF – Question: Is there any FD that violates the condition of 2NF? Try to derive FDs that violates 2NF (*prime → (all) non-primes)*** by applying the Armstrong's rules!
  - Yes: Decompose the relation based on the violating FD; this gives you two new relations; *restart* the algorithm for both of them
  - No: Continue to 3NF
- **3NF** – Question: Is there any FD that violates the condition of 3NF? Try to derive FDs that violates 3NF!
  - Yes: Decompose the relation based on the violating FD; this gives you two new relations; *restart* the algorithm for both of them
  - No: Continue to BCNF
- **BCNF** – Question: Is there any FD that violates the condition of BCNF? Try to derive such!
  - Yes: Decompose the relation based on the violating FD; this gives you two new relations; *restart* the algorithm for both of them
  - No: Done

Running Example

- Recall *R*(<u>PID</u>, <u>Country</u>, PersonName, Continent,
  ContinentArea, NumberVisitsCountry) with:

  FD1: PID → PersonName

  FD2: PID, Country → NumberVisitsCountry

  FD3: Country → Continent

  FD4: Continent → ContinentArea

- Is *R* in BCNF?
  If not, normalize R into BCNF relations.

> 2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

> 3NF: Non-prime attributes must not be functionally dependent on a set of attributes that is not a superkey.

> BCNF: For every FD $X \rightarrow Y$ it holds that X is a superkey of the relation that the FD is associated with.

Running Example (cont'd)

In order to identify all candidate keys we first need to find a superkey:

function *ComputeAttrClosure*(X, Σ)
begin
    $X^+ := X$;
    while Σ contains an FD $Y \rightarrow Z$ such that
        (i) Y is a subset of X^+, and
        (ii) Z is not a subset of X^+ **do**
      $X^+ := X^+ \cup Z$;
    end while
    return X^+;
end

R(PID, PersonName, Country, Continent, ContinentArea, NumberVisitsCountry):

FD1: PID → PersonName

FD2: PID, Country → NumberVisitsCountry

FD3: Country → Continent

FD4: Continent → ContinentArea

- The attribute closure of X = { PID, Country } w.r.t. FD1–FD4 is { PID, Country, PersonName, NumberVisitsCountry, Continent, ContinentArea }

Running Example (cont'd)

- The only candidate key: (PID, Country)
- Prime attributes: PID, Country
- Non-prime: PersonName, Continent, ContinentArea, NumberVisitsCountry

Running Example (cont'd)

- Recall *R*(<u>PID</u>, <u>Country</u>, PersonName, Continent, ContinentArea, NumberVisitsCountry) with:

  FD1: PID → PersonName

  FD2: PID, Country → NumberVisitsCountry

  FD3: Country → Continent

  FD4: Continent → ContinentArea

2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

- Is *R* in 2NF?
  - No, non-prime attribute PersonName depends on PID (part of a candidate key), similarly for Continent and Country
  - Try to derive PID → all non-primes
    - Only PID → PersonName
  - Try to derive Country → all non-primes
    - Country → Continent and Continent → ContinentArea imply Country→ContinentArea (using transitivity rule)
    - Country → Continent and Country→ContinentArea imply Country→Continent, ContinentArea (using additive rule)
  - Decompose *R* into *R1*(<u>PID</u>, PersonName) and *R2*(<u>PID</u>, <u>Country</u>, NumberVisitsCountry) and R3(<u>Country</u>, Continent, ContinentArea)

Running Example (cont'd)

- Recall *R*(<u>PID</u>, <u>Country</u>, PersonName, Continent, ContinentArea, NumberVisitsCountry) with:

  FD1: PID → PersonName

  FD2: PID, Country → NumberVisitsCountry

  FD3: Country → Continent

  FD4: Continent → ContinentArea

> 2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

- Is *R* in 2NF (cont'd)?
  - Decompose *R* into *R1*(<u>PID</u>, PersonName) and *R2*(<u>PID</u>, <u>Country</u>, NumberVisitsCountry) and R3(<u>Country</u>, Continent, ContinentArea)
  - Don't forget to write down the FDs for each new relation:
    - *R1:* <u>PID</u> → PersonName
    - *R2:* <u>PID</u>, <u>Country</u> → NumberVisitsCountry
    - R3: <u>Country</u> → Continent and Continent → ContinentArea
  - All are in 2NF

Running Example (cont'd)

- Recall *R*(<u>PID</u>, <u>Country</u>, PersonName, Continent, ContinentArea, NumberVisitsCountry) with:
  - *FD1:* PID → PersonName
  - *FD2:* PID, Country → NumberVisitsCountry
  - *FD3:* Country → Continent
  - *FD4:* Continent → ContinentArea

> 3NF: Non-prime attributes must not be functionally dependent on a set of attributes that is not a superkey.

- Is *R* in 3NF?
  - No, R3 is not in 3NF due to Continent → ContinentArea
  - Can we derive other FDs that violate 3NF? No.
  - Decompose to R31(<u>Country,</u> Continent) and R32(<u>Continent,</u> ContinentArea)
  - Don't forget to write down the FDs for each new relation:
    - R31: Country → Continent
    - R32: Continent → ContinentArea

Running Example (cont'd)

- Now we have *R1*(<u>PID</u>, PersonName),
  R2(<u>PID</u>, <u>Country</u>, NumberVisitsCountry),
  R31(<u>Country</u>, Continent), and
  R32(<u>Continent</u>, ContinentArea) and initial FDs:

  FD1: PID → PersonName

  FD2: PID, Country → NumberVisitsCountry

  FD3: Country → Continent

  FD4: Continent → ContinentArea

- Are *R1*, *R2*, *R31* and *R32* in 3NF, respectively?
  - Yes.

> 3NF: Non-prime attributes must not be functionally dependent on a set of attributes that is not a superkey.

Running Example (cont'd)

- Now we have *R1*(<u>PID</u>, PersonName),
  R2(<u>PID</u>, <u>Country</u>, NumberVisitsCountry),
  R31(<u>Country</u>, Continent), and
  R32(<u>Continent</u>, ContinentArea) and initial FDs:

  FD1: PID → PersonName
  FD2: PID, Country → NumberVisitsCountry
  FD3: Country → Continent
  FD4: Continent → ContinentArea

> BCNF: For every FD *X*→*Y* it holds that *X* is a superkey of the relation that the FD is associated with.

- Are *R1*, *R2*, *R31* and *R32* in BCNF, respectively?
  - Yes.

Normalization Example

2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**
- Candidate keys: {A,E} and {C,E}
- Prime attributes: A,C,E
- Non-prime attributes: B,D,F
- In which normal form is R? Normalize it into BCNF relations.

Normalization Example

2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**

- Is R in 2NF? Try to derive one of these:
  A → BDF
  C → BDF
  E → BDF

- Let's go for A → BDF:
  - A → BC implies A → B and A → C (decomposition)
  - C → AD implies C → D and C → A (decomposition)
  - A → C and C → D imply A → D (transitivity)
  - A → B and A → D imply **A** → **BD** (additive)
  - What about A → F, can we derive it? No.
    - OBS: A → D and DE → F imply AE → F (by transitivity) but this is not the same as A → F!!!

Normalization Example

2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A → BC**
  FD2: **C → AD**
  FD3: **DE → F**
- Is R in 2NF? Try to derive one of these:
  A → BDF
  C → BDF
  E → BDF
- Similarly we can show C → BDF:
  - C → A and A → BD implies **C → BD** (transitivity)
  - What about C → F, can we derive it? No.
    - OBS: C → D and DE → F imply CE → F (by transitivity) but this is not the same as C → F!!!
- Can we derive E → BDF? No!

2NF: Non-prime attributes must not be functionally dependent on a part of a candidate key.

Normalization Example

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: $\mathbf{A} \rightarrow \mathbf{BC}$
  FD2: $\mathbf{C} \rightarrow \mathbf{AD}$
  FD3: $\mathbf{DE} \rightarrow \mathbf{F}$
- Is R in 2NF? $\mathbf{A} \rightarrow \mathbf{BD}$ and $\mathbf{C} \rightarrow \mathbf{BD}$ violate 2NF
- By using $\mathbf{A} \rightarrow \mathbf{BD}$ we decompose R into R1 and R2:
  - R1(A,B,D) with $A \rightarrow BD$
    - Candidate keys: {A}; non-prime attributes: B, D
  - R2(A,C,E,F) $A \rightarrow C$, $C \rightarrow A$, $\mathbf{AE} \rightarrow \mathbf{F}$ and $\mathbf{CE} \rightarrow \mathbf{F}$
    - What happened with $DE \rightarrow F$?
      Recall we derived $\mathbf{AE} \rightarrow \mathbf{F}$ and $\mathbf{CE} \rightarrow \mathbf{F}$
    - Candidate keys: {A,E} and {C,E}; non-prime attributes: F
- R1 and R2 are in 2NF: in R2, C and A are both prime attributes

Normalization Example

3NF: Non-prime attributes must not be functionally dependent on a set of attributes that is not a superkey.

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A** → **BC**
  FD2: **C** → **AD**
  FD3: **DE** → **F**

- Is R in 3NF?
  - R1(A,B,D) with A → BD
    - Candidate keys: {A}; non-prime attributes: B, D
    - R1 is in 3NF
  - R2(A,C,E,F) A → C, C → A, AE→ F and CE → F
    - Candidate keys: {A,E} and {C,E}; non-prime attributes: F
    - R2 is in 3NF since the only non-prime attribute F is functionally determined only by superkeys (AE and CE)

- R1 and R2 are in 3NF: in R2, C and A are both prime attributes

Normalization Example

BCNF: For every FD $X \rightarrow Y$ it holds that X is a superkey of the relation that the FD is associated with.

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:

  FD1: **A** → **BC**

  FD2: **C** → **AD**

  FD3: **DE** → **F**

- Is R in BCNF?
  - R1(A,B,D) with A → BD
    - Candidate keys: {A}; non-prime attributes: B, D
    - R1 is in BCNF
  - R2(A,C,E,F) A → C, C → A, AE → F and CE → F
    - Candidate keys: {A,E} and {C,E}; non-prime attributes: F
    - R2 is not in BCNF due to A → C, C → A

Normalization Example

BCNF: For every FD $X \rightarrow Y$ it holds that X is a superkey of the relation that the FD is associated with.

- Consider the relation **R(A,B,C,D,E,F)** with the following FDs:
  FD1: **A → BC**
  FD2: **C → AD**
  FD3: **DE → F**
- Is R in BCNF?
- R2(A,C,E,F) A → C, C → A, AE → F and CE → F
  - Candidate keys: {A,E} and {C,E}; non-prime attributes: F
  - R2 is not in BCNF due to A → C, C → A
  - Decompose using A → C into R21 (A,E,F) and R22 (A,C)
    - R21: AE → F
    - R22: A → C and C → A
    -
- Now R1, R21 and R22 are in BCNF

www.liu.se

LINKÖPING
UNIVERSITY