MDA104: Tutorial 3
Functional Dependencies

Vlastislav Dohnal

Functional dependency: Definition

- What does functional dependence express?

  faculty $\rightarrow$ address

- Is this dependency satisfied in the following relations?

faculty	address
FI	Botanická 68a, 602 00 Brno
PřF	Kotlářská 267/2, 611 37 Brno

department	faculty	address	area
Geogr	PřF	Kotlářská 267/2, 611 37 Brno	250
KIT	FI	Botanická 68a, 602 00 Brno	200
KPGD	FI	Botanická 68a, 602 00 Brno	180
KTP	FI	Botanická 68a, 602 00 Brno	210

- If the dependency holds on to the relational schema, then each instance satisfies that dependency.
  - i.e., the definition of integrity constraint.

Creation of func. deps and their validity

- Consider the registry of enrolled courses with the following requirements:
  - The student has a name and his or her own unique ID;
  - The course has a title, number of credits obtained when passed, and its own unique code;
  - The student can enroll in more courses;
  - More students can enroll in one course;
  - The student has a certain completion selected for the course he or she has signed up for.

- Task: Formulate functional dependencies.

Creation of func. deps and their validity (solution)

- Consider the registry of enrolled courses with the following requirements:
  - The student has a name and his or her own unique ID;
  - The course has a title, number of credits obtained when passed, and its own unique code;
  - The student can enroll in more courses;
  - More students can enroll in one course;
  - The student has a certain completion selected for the course he or she has signed up for.

- Functional dependencies:
  - student_id $\rightarrow$ first_name, last_name
  - course_code $\rightarrow$ title, credits
  - student_id, course_code $\rightarrow$ completion_type
- Commentary:
  - There is many-to-many relationship between students and courses.
  - student_id $\rightarrow$ course_code
    - It models "one course can have multiple students", but a student cannot enroll into multiple courses.
  - course_code $\rightarrow$ student_id
    - vice versa ("one students can enroll more courses, but just one student can join a course!)

Creation of func. deps and … (solution) (cont.)

- Consider the registry of enrolled courses with the following requirements:
  - The student has a name and his or her own unique ID;
  - The course has a title, number of credits obtained when passed, and its own unique code;
  - The student can enroll in more courses;
  - More students can enroll in one course;
  - The student has a certain completion selected for the course he or she has signed up for.
- The issue is how to handle many-to-many relationship using functional dependencies:
  - student_id, course_code $\rightarrow$ completion_type
  - If there is no “completion_type”, then we cannot have the left-hand side only,
    - i.e. “student_id, course_code $\rightarrow$ Ø”
  - Instead, we may not write any dep.
    - and the process of decomposition will leave “student_id, course_code” in a table.

Inference of additional dependencies

- Armstrong axioms
  - *Reflexivity*: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
  - *Augmentation*: if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$
  - *Transitivity*: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$
- Consider the following dependencies

  F = { student_id → first_name
  student_id → last_name
  student_id → avg_grade
  course_code → title, credits
  student_id, course_code → completion_type
  study_program, avg_grade → scholarship_amount }
- Task: Prove that the following dependencies are satisfied:
  - student_id → first_name, last_name
  - course_code → title
  - student_id, study_program → scholarship_amount

Inference of add…: Solution

- Armstrong axioms
  - *Reflexivity*: if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
  - *Augmentation*: if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$
  - *Transitivity*: if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$
- Consider the following dependencies

  F = { student_id → first_name
  student_id → last_name
  student_id → avg_grade
  course_code → title, credits
  student_id, course_code → completion_type
  study_program, avg_grade → scholarship_amount }

- Solution:
  - student_id → first_name, last_name
    - augmentation of student_id → first_name with last_name: student_id, last_name → first_name, last_name
    - augmentation of student_id → last_name with student_id: student_id → last_name, student_id
    - transitivity of both these gets the needed.
  - course_code → title
    - reflexivity: title, credits → title
    - transitivity: course_code → title, credits and title, credits → title gets the needed.
  - student_id, study_program → scholarship_amount
    - augmentation of student_id → avg_grade with study_program: student_id, study_program → avg_grade, study_program
    - transitivity of the augmented f.d. and study_program, avg_grade → scholarship_amount gets the needed.

Schema keys

- Revision:
  - *K is the super-key for the relation schema R*, if and only if $K \rightarrow R$
- *K is the candidate key R if and only if it holds*
  - $K \rightarrow R$, and at the same time
  - for none $\alpha \subset K$, *the functional dep.* $\alpha \rightarrow R$ *holds*

- Consider the following relation schema and functional dependencies:
  r(student_id, first_name, last_name, course_code, title, credits, completion_type)

  F = {
  student_id → first_name, last_name
  course_code → title, credits
  student_id, course_code → completion_type
  }
  - Task: Resolve the primary key.

Revision: Attribute set closure algorithm

- For a set of attributes α, we define the **closure** of it, denoted as α^+, as a set of attributes that are functionally dependent according to the set of functional dependencies F.
- Algorithm to get α^+:

```
result := α;
previousResult := null;
while (result != previousResult ) do {
    previousResult  := result;
    for each β → γ in F do {
        if β ⊆ result then result := result ∪ γ ;
    }
}
```

- Use this algorithm to verify the choice of keys from the previous task.

Schema keys: Solution

- Consider the following relation schema and functional dependencies:
  r(student_id, first_name, last_name, course_code, title, credits, completion_type)

  F = { student_id $\rightarrow$ *first_name, last_name*
  course_code $\rightarrow$ title, credits
  student_id, course_code $\rightarrow$ completion_type
  }
  - Task: Resolve the primary key.

- Solution to "Resolve the primary key".
  - We need to identify all candidate keys, so then we may pick one as the primary key.
    - Candidate key is the minimal super key, so we start with individual attributes.
      - No individual attribute "implies" R.
    - So, a pair of attributes… Possible pairs can be formed by the attributes on the left-hand sides of F.D.s. -> There is only one:
      - {student_id, course_code}$^+$ = R, so this holds:
      - student_id, course_code $\rightarrow$ student_id, first_name, last_name, course_code, title, credits, completion_type)

First normal form (1NF)

- Definition:
  - The relation schema R is in first normal form if the domains of all attributes are atomic.

- Task: Are the following relations in 1NF?
  - If not, then convert them to 1NF.

address

employee	address	phone_numbers
Zdeněk	Plotní 30, Brno	{549 49 1810, 736 09 1809}
Petr	Horní 12, Brno	{727 49 1911, 549 49 1911}

course

course_code	title	credits
FI:PB154	Základy databázových systémů	3
PřF:Bi8150	Evoluční biologie	3

First normal form (1NF)

- Solution:
  - Attributes may not contain more than “one” value.
- Address(employee, street, house_no, city)
  - address decomposed
- Phone(employee, phone_number)
  - phone_numbers (originally an array) took into a new table (one row per phone number)
- Course(faculty_code, cource_code, title, credits)
  - the composed value of course_code decomposed and faculty code extracted into a new attribute.

address

employee	address	phone_numbers
Zdeněk	Plotní 30, Brno	{549 49 1810, 736 09 1809}
Petr	Horní 12, Brno	{727 49 1911, 549 49 1911}

course

course_code	title	credits
FI:PB154	Základy databázových systémů	3
PřF:Bi8150	Evoluční biologie	3

Second normal form (2NF)

- Definition: A relational schema R is in 2NF if it is in 1NF, and for each attribute A of R, any of the following statements are true:
  - *A* is part of a candidate key (CK) (so trivially CK $\rightarrow$ A), or
  - *A* is not partially dependent on any candidate key.
    - $\alpha \rightarrow \beta$ is *partial dependency*, if $\exists\ \gamma \subset \alpha$ such that $\gamma \rightarrow \beta$
- Task: Decide if the relation **thesis** is in 2NF

thesis

student_id	first_name	last_name	program	spec	thesis_title	supervisor	supervisor_dept	guarantor
12345	Jan	Novák	Inf.	Mat. inf.	Teorie sčítání	Brázdil	KTP	Hliněný
67890	Lenka	Šťastná	Apl. inf.	Bioinf.	Život brouka	Lexa	KTP	Brim

F = { student_id, program $\rightarrow$ spec, supervisor, thesis_title
spec $\rightarrow$ guarantor, program
supervisor $\rightarrow$ supervisor_dept
student_id $\rightarrow$ first_name, last_name }

This relation records data necessary for the application for the defense of the final thesis. A student can study more than one program, then he or she has their own final thesis for each study.

Second normal form (2NF)

thesis

student_id	first_name	last_name	program	spec	thesis_title	supervisor	supervisor_dept	guarantor
12345	Jan	Novák	Inf.	Mat. inf.	Teorie sčítání	Brázdil	KTP	Hliněný
67890	Lenka	Šťastná	Apl. inf.	Bioinf.	Život brouka	Lexa	KTP	Brim

F = { student_id, program → spec, supervisor, thesis_title
spec → guarantor, program
supervisor → supervisor_dept
student_id → first_name, last_name }

- Solution: Decompose into 2NF
  - State candidate keys:
    - A. *student_id, program*
    - B. *student_id, spec*
  - first_name, last_name depend on a part of a candidate key, so break 2NF
    - since student_id → first_name, last_name
  - guarantor depends on a part of a candidate key, so breaks 2NF.
    - since spec → guarantor, program
  - supervisor_dept is OK
    - since supervisor → supervisor_dept and supervisor depends on a whole candidate key.
    - by analogy thesis_title
  - the others are part of a candidate key
- Resulting relations:
  - **student**(student_id, first_name, last_name)
  - **specialization**(spec, program, guarantor)
  - **thesis**(student_id, program, spec, thesis_title, supervisor, supervisor_dept)

Third normal form (3NF)

- Definition: The relation schema *R* is in 3NF, if it is in 2NF and for each *A* in *R,* any of the statements hold:
  - *A* is part of a candidate key, or
  - *A* is not transitively dependent on any candidate key.
    - *A* is *transitively dependent on* α, if $\alpha \rightarrow \beta$, $\beta \rightarrow$ A, and $\beta \not\rightarrow \alpha$, where *A* is not part of β either α.

Third normal form (3NF)

- Task: Decide if the following relations are in 3NF

thesis

student_id	program	spec	thesis_title	supervisor	supervisor_dept
12345	Inf.	Mat. inf.	Teorie sčítání	Brázdil	KTP
67890	Apl. Inf.	Bioinf.	Život brouka	Lexa	KTP

F_1 = { student_id, program → spec, supervisor, thesis_title
spec → program
supervisor → supervisor_dept }

student

student_id	first_name	last_name
12345	Jan	Novák
67890	Lenka	Šťastná

F_2 = { student_id → first_name, last_name }

specialization

spec	program	guarantor
Mat. inf.	Inf.	Hliněný
Bioinf.	Apl. Inf.	Brim

F_3 = { spec → program, guarantor }

Third normal form (3NF)

- Solution: Decide if the following relations are in 3NF

- **student**(student_id, first_name, last_name)
  - trivially in 3NF, since student_id $\rightarrow$ first_name, last_name

- **specialization**(spec, program, guarantor)
  - trivially in 3NF, since spec $\rightarrow$ program, guarantor

- thesis(student_id, program, spec, thesis_title, supervisor, supervisor_dept)
  - not in 3NF because of supervisor_dept, so decompose into:

  - **thesis**(student_id, program, spec, thesis_title, supervisor)
  - **supervisor**(supervisor, supervisor_dept)

* final relations are in bold

Boyce-Codd normal form (BCNF)

- Definition: The relation schema *R* is in BCNF w.r.t. *F*, if it is in 1NF and for each f. dep. $\alpha \rightarrow \beta$ in ***F*⁺** the following holds:
  - $\alpha \rightarrow \beta$ is a trivial dependency (i.e., $\beta \subseteq \alpha$), or
  - α is a super-key in *R*.

  Usage: If we verify that a relation is in BCNF, <u>it is sufficient to verify</u> the listed properties only for $\alpha \rightarrow \beta$ in F. After the decomposition, it is necessary to consider F^+ to verify the resulting relations.
- Task: Decide whether the following relation is in BCNF

thesis

student_id	program	spec	thesis_title	supervisor
12345	Inf.	Mat. inf.	Teorie sčítání	Brázdil
67890	Apl. Inf.	Bioinf.	Život brouka	Lexa

F = { student_id, program $\rightarrow$ spec, supervisor, thesis_title
spec $\rightarrow$ program }

Decomposition to BCNF

- Algorithm: Let be *R* not in BCNF. Then there is at least one non-trivial functional dependence $\alpha \rightarrow \beta$ such that α is not a super-key of *R*. Relation *R* gets replaced by:
  - $R_1 = (\alpha \cup \beta)$
  - $R_2 = (R - (\beta - \alpha))$

  Caution: After executing the algorithm, it is necessary to check whether R_1, R_2 already meet BCNF – see the previous procedure, this time it is necessary to verify for all dependencies in F^+!
- Task: Convert the following relation to BCNF.
  Does the result preserve all functional dependencies?

thesis

student_id	program	spec	thesis_title	supervisor
12345	Inf.	Mat. inf.	Teorie sčítání	Brázdil
67890	Apl. Inf.	Bioinf.	Život brouka	Lexa

F = { student_id, program $\rightarrow$ spec, supervisor, thesis_title
spec $\rightarrow$ program }

Decomposition to BCNF

thesis

student_id	program	spec	thesis_title	supervisor
12345	Inf.	Mat. inf.	Teorie sčítání	Brázdil
67890	Apl. Inf.	Bioinf.	Život brouka	Lexa

F = { student_id, program → spec, supervisor, thesis_title
spec → program }

- Solution: Convert the following relation to BCNF.
  - Check each f.d.
  - student_id, program → spec, supervisor, thesis_title
    - ok, since student_id, program is a candidate key.
  - spec → program
    - break BNCF, since spec is not a super-key.
    - Extract program for the relation and create a new table.
- **thesis**(student_id, spec, thesis_title, supervisor)
  - F={student_id, spec → thesis_title, supervisor}
- **spec_program**(spec, program) (Ensures no redundancy in storing spec and program pair, as was in original thesis.)
  - F={spec → program}
- Solution: Does the result preserve all functional dependencies?
  - No, student_id, program → spec cannot be verify unless join of thesis and spec_program is done.

Final result in BCNF

- **student**(student_id, first_name, last_name)
  - F={student_id $\rightarrow$ first_name, last_name}
- **supervisor**(supervisor, supervisor_dept)
  - F={supervisor $\rightarrow$ supervisor_dept}
- **thesis**(student_id, spec, thesis_title, supervisor)
  - F={student_id, spec $\rightarrow$ thesis_title, supervisor}
- **specialization**(spec, program, guarantor) (It removes redundancy by factoring out dependencies related to spec.)
  - F={spec $\rightarrow$ program, guarantor)
- **spec_program**(spec, program) (Ensures no redundancy in storing spec and program pair, as was in original thesis.)
  - F={spec $\rightarrow$ program}

- Mind student_id, program $\rightarrow$ spec is not preserved! (Attributes are not part of one relation.)

Example (voluntary)

- Design a database in BCNF/3NF according to the following requirements:
  - We keep records of books and their physical copies in the library.
  - Each book has its own unique ISBN, title, one or more authors (the order of authors matters), publication number and category.
  - We keep records of the first and last names of the authors of the books.
  - Each book can be present in the library in the form of several copies. For each copy, the library keeps track of its condition ("new", "damaged", "under repair", ...).
  - Books in the "new" category can be borrowed for a maximum of a week, books in the "archival materials" category are not borrowed, others can be borrowed for a month.
    - i.e., there are special categories of books that have a loan period limited to a specific period.