# SWEN304: Normal Forms

Dr. Dionysios Athanasopoulos

Lecturer
dionysios.athanasopoulos@vuw.ac.nz
**Office**: EA111, Easterfield building, Kelburn Campus

# Agenda

1. Background - Terminology
2. Normal Forms

# Background

## Functional Dependency (FD)

Given a relation schema $R(A, F)$, where $A = \{..., X, Y, ...\}$ is a set of attributes and $F$ is a set of FDs,

- FD denoted by $X \rightarrow Y$, is that, for any two tuples $t_1$ and $t_2$ in $R$ that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

Relation schema: Grade

| **student_id** | **course_id** | **student_name** | **grade** |
|---|---|---|---|
| 1000 | SWEN304 | Joe M. | A+ |
| 1000 | SWEN222 | Joe M. | A- |

Functional dependency:

- $student\_id \rightarrow student\_name$

# Terminology

## Primary Key

- If there cannot be more than one tuple of relation schema with the same value for a subset $X$ of attributes, then $X$ is the **primary key** of the relation schema:
  - $X \rightarrow Y$ for any subset $Y$ of the attributes of the relation schema.

Relation schema: Grade

| student_id | course_id | student_name | grade |
|---|---|---|---|
| 1000 | SWEN304 | Joe M. | A+ |
| 1000 | SWEN222 | Joe M. | A- |

Functional dependencies:

- $student\_id + course\_id \rightarrow student\_name$
- $student\_id + course\_id \rightarrow grade$
- $student\_id + course\_id \rightarrow student\_name + grade$

# Terminology

## Candidate key

- If a relation schema has more than one keys, then each one of them is a **candidate key**.

## Prime attribute

- An attribute of relation schema is called **prime attribute** if it is a member of a candidate key.

## Superkey

- Any set of attributes that includes not only a primary key but also extra attributes is called **superkey**;
  - in contrast to a superkey, a key has to be **minimal**; i.e. if an attribute is removed from a key, then it is not a key.

# Algorithm for Determining a Key of a Relation Schema

**Input:** $R$, $F$
**Output:** $K$

1: $K \leftarrow R$
2: **for all** attributes $A$ in $K$ **do**
3:     **compute** the closure $(K - A)^+$ w.r.t. $F$
4:     **if** $(K - A)^+$ contains all the attributes in $R$ **then**
5:         $K \leftarrow K - \{A\}$

Relation schema: Faculty $R$

| StudID | StudName | Pts | CourID | CourName | LectID | LectName | Grade |
|---|---|---|---|---|---|---|---|

Functional dependencies:

- $F = \{StudID \rightarrow StudName + Pts,\ CourID \rightarrow CourName,\ LectID \rightarrow LectName + CourID,\ StudID + CourID \rightarrow Grade + LectID\}$

**1st step** of the algorithm for determining a key:

- $K = \{StudID,\ StudName,\ Pts,\ CourID,\ CourName,\ LectID,\ LectName,\ Grade\}$

# Algorithm for Determining the Closure of Attributes

**2nd - 5th steps** of the algorithm for determining a key:

- $(K - A)^+$ = ??
- e.g. $(K - StudID)^+$ = ??

**function** CALCULATECLOSURE($K$, $F$): $K^+$
    $K^+ \leftarrow K$
    **repeat**
        $oldK^+ \leftarrow K^+$
        **for all** $Y \rightarrow Z$ **do**
            **if** $K^+ \supseteq Y$ **then**
                $K^+ \leftarrow K^+ \cup Z$
    **until** $oldK^+ = K^+$
**end function**

**2nd - 5th steps** of the algorithm for determining a key:

- $(K - StudID)^+$ = $\{StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ **does not contain** all the attributes in $R$
  - $K$ = $\{StudID, StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ ($StudID$ is **not removed** from the $K$)

# Algorithm for Determining a Key of a Relation

**2nd - 5th steps** of the algorithm for determining a key:

- $(K - StudName)^+ = \{StudID, StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ **contains** all the attributes in $R$
  - $K = \{StudID, Pts, CourID, CourName, LectID, LectName, Grade\}$ ($StudName$ is **removed** from the $K$)
- $(K - Pts)^+ = \{StudID, StudName, Pts, CourID, CourName, LectID, LectNam$ $Grade\}$ **contains** all the attributes in $R$
  - $K = \{StudID, CourID, CourName, LectID, LectName, Grade\}$ ($Pts$ is **removed** from the $K$)
- $(K - CourName)^+ = \{StudID, StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ **contains** all the attributes in $R$
  - $K = \{StudID, CourID, LectID, LectName, Grade\}$ ($CourName$ is **removed** from the $K$)

# Algorithm for Determining a Key of a Relation

- $(K - CourID)^+ = \{StudID, StudName, Pts, LectID, LectName, Grade\}$ **does not contain** all the attributes in $R$
  - $K = \{StudID, CourID, LectID, LectName, Grade\}$ ($CourID$ is **not removed** from the $K$)
- $(K - LectName)^+ = \{StudID, StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ **contains** all the attributes in $R$
  - $K = \{StudID, CourID, LectID, Grade\}$ ($LectName$ is **removed** from the $K$)
- $(K - LectID)^+ = \{StudID, StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ **contains** all the attributes in $R$ (transitivity rule)
  - $K = \{StudID, CourID, Grade\}$ ($LectID$ is **removed** from the $K$)
- $(K - Grade)^+ = \{StudID, StudName, Pts, CourID, CourName, LectID, LectName, Grade\}$ **contains** all the attributes in $R$
  - $K = \{StudID, CourID\}$ ($Grade$ is **removed** from the $K$)

**Primary key:** $K = \{StudID, CourID\}$

# Candidate Keys & Inference Rules

Candidate keys can be **inferred** via changing the order that the attributes are examined.

- e.g. candidate keys for the relation *Faculty* are:
  - $K = \{StudID, CourID\}$ (concluded to it in the previous slides)
  - $K = \{StudID, LectID\}$

Inference rules for determining keys:

- (Reflexivity) $Y \subseteq X \vDash X \rightarrow Y$ (trivial FD)
- (Augmentation) $X \rightarrow Y \wedge Z \supseteq W \vDash XZ \rightarrow YW$ (partial FD)
- (Transitivity) $X \rightarrow Y \wedge Y \rightarrow Z \vDash X \rightarrow Z$ (transitive FD)
- (Pseudo transitivity) $X \rightarrow Y \wedge WY \rightarrow Z \vDash WX \rightarrow Z$
- (Decomposition) $X \rightarrow YZ \vDash X \rightarrow Y \wedge X \rightarrow Z$
- (Union) $X \rightarrow Y \wedge X \rightarrow Z \vDash X \rightarrow YZ$

# Exercises on Candidate Keys

Relation schema: Exercise 1

| C | Z | S |
|---|---|---|

Functional dependencies:

- $\{Z \rightarrow C, CS \rightarrow Z\}$

**Determine all the candidate keys.**

Relation schema: Exercise 2

| A | B | C | D |
|---|---|---|---|

Functional dependencies:

- $\{A \rightarrow B, B \rightarrow C, CD \rightarrow A, AC \rightarrow D\}$

**Determine all the candidate keys.**

# Agenda

# Normal Forms

## Normalization

- Normalization is a process of analyzing relation schemas based on **normal form tests** to minimize:
  - redundancy
  - insertion, deletion, and update anomalies.
- Relation schemas that do not meet the normal form tests are **decomposed** into smaller relation schemas.

Normal form tests:

1. Functional dependencies
   - First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), and Boyce-Codd Normal Form (BCNF)
2. Multi-valued and join dependencies
   - Fourth Normal Form (4NF) and Fifth Normal Form (5NF).

# First Normal Form

## 1NF

- It is part of the formal definition of a **flat** relation schema;
- it disallows **nested relation schemas.**

Relation schema: Student (nested relation schema)

| id | name | address | school | year |
|---|---|---|---|---|
| 1000 | Joe M. | Kelburn | ECS<br>SMS | 2005<br>2010 |

# First Normal Form (1NF)

To achieve 1NF for such a relation schema:

- **Unnest relation schemas**: remove the nested relation attributes into a new relation schema and propagate the primary key into it;

Relation schema: Student

| **id** | **name** | **address** |
|---|---|---|
| 1000 | Joe M. | Kelburn |

Relation schema: School

| **id** | **school** | **year** |
|---|---|---|
| 1000 | ECS | 2005 |
| 1000 | SMS | 2010 |

# First Normal Form

## 1NF

- It is part of the formal definition of a **flat** relation schema;
- it disallows **nested relation schemas**;
- it disallows **multivalued attributes.**

Relation schema: Student (multivalued attribute)

| id | name | address | school |
|---|---|---|---|
| 1000 | Joe M. | Kelburn | { ECS, SMS } |

# First Normal Form (1NF)

There are two ways to achieve 1NF for such a relation schema:

1. **Place the multi-valued attribute in a separate relation schema**, along with the primary key.

Relation schema: Student

| **<u>id</u>** | **name** | **address** |
|---|---|---|
| 1000 | Joe M. | Kelburn |

Relation schema: School

| **<u>id</u>** | **<u>school</u>** |
|---|---|
| 1000 | ECS |
| 1000 | SMS |

# First Normal Form (1NF)

There are two ways to achieve 1NF for such a relation schema:

1. Place the multivalued attribute in a separate relation schema, along with the primary key;
2. **Expand the key** so that there will be a separate tuple for value of the problematic attribute;
   - it introduces redundancy.

Relation schema: Student

| id | name | address | school |
|---|---|---|---|
| 1000 | Joe M. | Kelburn | ECS |
| 1000 | Joe M. | Kelburn | SMS |

# First Normal Form (1NF)

- Redundancy leads to **update** anomalies.

Relation schema: Student

| **id** | **name** | **address** | **school** |
|---|---|---|---|
| 1000 | Joe M. | Kelburn | ECS |
| 1000 | Joe M. | Kelburn | SMS |

- If we would correct a mistake in a student `name`, then we have to update every row that is related to a student.

# Second Normal Form

## 2NF

- It disallows **partial functional dependencies**;
  - $X \rightarrow Y$ is a full functional dependency if removal of any attribute from the set X means that the dependency does not hold any more.
- In other words, if primary key contains multiple attributes, **no non-key attribute should not be functionally dependent on a part of the primary key.**

Relation schema: Grade (partial functional dependency)

| student_id | course_id | student_name | grade |
|---|---|---|---|
| 1000 | SWEN304 | Joe M. | A+ |
| 1000 | SWEN222 | Joe M. | A- |

Partial functional dependency:

- *student_id* $\rightarrow$ *student_name*

# Second Normal Form (2NF)

To achieve 2NF for such a relation schema:

- **Decompose and set up a new relation schema** for each partial key with its dependent attribute(s).

Relation schema: Grade 1

| student_id | course_id | grade |
|---|---|---|
| 1000 | SWEN304 | A+ |
| 1000 | SWEN222 | A- |

Relation schema: Grade 2

| student_id | student_name |
|---|---|
| 1000 | Joe M. |

# 2NF Exercise 1

Relation schema: Student

| <u>**StudID**</u> | <u>**CourID**</u> | **StudName** | **NoOfPts** | **Grade** |
|---|---|---|---|---|

Functional dependencies:

- $StudID \rightarrow StudName + NoOfPts$
- $StudID + CourID \rightarrow Grade$

**Observation**: if the `Grade` attribute cannot have `null` values, a new student cannot be inserted until s/he passes a new exam.

**In what normal form is the above relation schema**?
Make the proper decompositions in order to bring the relation schema in 2NF.

# 2NF Exercise 2

Relation schema: Faculty

| <u>LectID</u> | LectName | <u>CourID</u> | CourName |
| --- | --- | --- | --- |

Functional dependencies:

- *LectID* → *LectName*
- *LectID* → *CourID*
- *LectID* → *CourName*
- *CourID* → *CourName*

**Observation**: new `Course` data cannot be inserted without knowing who is going to lecture it. Moreover, if a `Lecturer` resigns, `Course` data will be lost.

**In what normal form is the above relation schema**?
Make the proper decompositions in order to bring the relation schema in 2NF.

# Third Normal Form

## 3NF

- It disallows **transitive dependencies**;
  - $X \rightarrow Y$ is a transitive dependency if there exists a set of attributes $Z$ that is neither a candidate key nor a subset of a primary key and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.
- In other words, **a nonkey attribute should not be functionally dependent by another nonkey attribute.**

Relation schema: Lecturer (transitive dependency)

| lecturer_id | lecturer_name | university_shortcut | university_name |
|---|---|---|---|
| L1000 | Jack A. | VUW | Victoria University of Wellington |
| L1001 | Chris M. | UOA | University of Auckland |

Part of the transitive functional dependency:

- *university_shortcut* $\rightarrow$ *university_name*

# Third Normal Form (3NF)

To achieve 3NF for such a relation schema:

- **decompose and set up a new relation schema** that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Relation schema: Lecturer

| lecturer_id | lecturer_name | university_shortcut |
|---|---|---|
| L1000 | Jack A. | VUW |
| L1001 | Chris M. | UOA |

Relation schema: University

| university_shortcut | university_name |
|---|---|
| VUW | Victoria University of Wellington |
| UOA | University of Auckland |

# 3NF Exercise

Relation schema: Faculty

| **<u>LectID</u>** | **LectName** | **CourID** | **CourName** |
|---|---|---|---|

Functional dependencies:

- $LectID \rightarrow LectName$
- $LectID \rightarrow CourID$
- $LectID \rightarrow CourName$
- $CourID \rightarrow CourName$

**In what normal form is the above relation schema**?
Make the proper decompositions in order to bring the relation schema in 3NF.

# Boyce-Codd Normal Form (BCNF)

## BCNF

- Whenever a nontrivial functional dependency $X \to Y$ holds in a relation schema, X is a **superkey** of the relation schema.

Trivial functional dependency:

- if $X \to Y$ holds and $Y \subseteq X$, then it is a trivial functional dependency.

Relation schema: Example

| <u>**A**</u> | <u>**B**</u> | **C** |
|---|---|---|

Functional dependencies:

- $A + B \to C$
- $C \to B$

# BCNF Example

Relation schema: Teach

| <u>**student_id**</u> | <u>**course_id**</u> | **lecturer_name** |
|---|---|---|

Functional dependencies:

- $student\_id + course\_id \rightarrow lecturer\_name$
- $lecturer\_name \rightarrow course\_id$[1]

**In what normal form is the above relation schema**?
Make the proper decompositions to bring the relation schema in BCNF.

[1] Assuming that each lecturer teaches only one course.

# BCNF Example

To achieve BCNF for such a relation schema:

- **decompose and set up a new relation schema** that includes the attributes of the problematic dependency.

- (*lecturer_name*, *course_id*) and (*lecturer_name*, *student_id*)

Observations:

- all the decompositions **lose** the first functional dependency: *student_id* + *course_id* → *lecturer_name*
- the decomposition does not generate spurious tuples after a join (i.e. additional tuples that were not present in the original relation schema).

# BCNF Exercise

Relation schema: Faculty

| LectID | <u>StudID</u> | <u>CourID</u> | Grade |
|---|---|---|---|

Functional dependencies:

- $\{LectID \rightarrow CourID, StudID + CourID \rightarrow LectID, StudID + CourID \rightarrow Grade\}$

Observations:

- If a lecturer resigns and we delete a tuple, we will lose information about student enrollments.
- If a lecturer wants to start teaching a new course and we insert a tuple with the same lecturer and a different course, the functional dependency $LectID \rightarrow CourID$ will be violated.

**In what normal form is the above relation schema**?
Make the proper decompositions to bring the relation schema in BCNF.

# Conclusions in 1NF, 2NF, 3NF, and BCNF

- Only the first four normal forms are practically used
- 1NF, 2NF, and (partly) 3NF suffer from update anomalies
- BCNF relations are (practically) free of update anomalies
  - it represents a possible goal of normalization
  - however, it possibly loses functional dependencies.

# Normal Form of a Set of Relation Schemas

- The normal form of a set of relation schemas is the normal form of the relation schema being in the lowest normal form.

Relation schema: $R_1$

| A | B |
|---|---|

Relation schema: $R_2$

| B | C | D | E |
|---|---|---|---|

Functional dependencies:

- $\{A \rightarrow B, BC \rightarrow D, C \rightarrow E\}$

**Due to $R_2$, the whole set is in 1NF, even though $R_1$ is in BCNF.**

# References


- ElMasri, Navathe, Fundamentals of Database Systems, 6th Edition, Addison Wesley.
- Hui Ma & Pavle Mogin, SWEN304 Lecture Slides, 2016
  https://ecs.victoria.ac.nz/Courses/SWEN304_2016T2/LectureSchedule