



Dependencies, normal forms and normalization

GRA 4142 Data Management and Python Programming

Jan Kudlicka (jan.kudlicka@bi.no)

Department of Data Science and Analytics

November 4, 2021

- ▶ Each table should represent a single entity or relationship type.
- ▶ Interpretability: it should be easy to understand the meaning of tables and attributes from their names (and types).
- ▶ Each attribute should have an appropriate data type.
- ▶ Each attribute should have a NOT NULL constraint if the value cannot be NULL.
- ▶ Primary key for each table.
- ▶ Constraining references to other entities with foreign keys.

Company database with a single relation: EmpProj(eid, lname, fname, pid, project, hours)

Note: eid = employee ID, pid = project ID

eid	lname	fname	pid	project	hours
1	Alnes	Bernt	1	Alfa	100
2	Fjelldal	Mads	3	Charlie	140
4	Longva	Victor	2	Bravo	80
7	Bakke	Alfred	1	Alfa	80
8	Vie	Tor	1	Alfa	910
8	Vie	Tor	3	Charlie	720
9	Westgaard	Sten	3	Charlie	310
10	Liseth	Rakel	3	Charlie	480
11	Norman	Emil	3	Charlie	460
12	Dyrhaug	Atle	2	Bravo	810
15	Kvien	Amalie	2	Bravo	15
15	Kvien	Amalie	3	Charlie	75
16	Tveten	Thomas	2	Bravo	20
17	Lende	Marita	1	Alfa	40

What's "wrong" with this database?

- ▶ **Insertion anomaly:** Cannot insert a project unless an employee is working on it. Cannot insert an employee unless assigned to a project.
- ▶ **Deletion anomaly:** If we remove all employees from a given project, the information about the project is lost. (And similarly if we remove all projects from a given employee.)
- ▶ **Update anomaly:** When some information changes, we need to update all occurrences, otherwise we will introduce **data inconsistency**.
- ▶ **Redundancy:** Same information is stored multiple times (a waste of storage and a risk of inconsistency).

Normalization is a step-by-step process to decompose a relation schema into a set of several smaller schemas to avoid redundancy and anomaly problems.

First, we need to learn about:

- ▶ Functional dependencies
- ▶ Normal forms (we will cover only 1NF, 2NF and 3NF)

Let R denote a relation and X and Y two subsets of its attributes.

X **functionally determines** Y if for any two tuples in any legal instance having equal values for each of the attributes in X means that they have equal values for each of the attributes in Y as well.

Notation:

$$X \rightarrow Y$$

We will also say that Y is (functionally) dependent on X , and that Y depends on X .

E.g. $\{ssn\} \rightarrow \{last\ name, first\ name, dob\}$, $\{dob\} \rightarrow \{age\}$, $\{project\ id\} \rightarrow \{project\ title\}$.

Functional dependencies are given by real-world constraints and they show the interrelationships of the attributes.

This means we need to know and understand the meaning of the attributes and relationships between them.

We can rule out an FD based on the current instance but not vice versa (ref. “for any legal instance” in the definition)!

$X \rightarrow Y$ is

- ▶ **trivial** if $Y \subseteq X$
- ▶ **non-trivial** if $Y \not\subseteq X$
- ▶ **completely non-trivial** if $X \cap Y = \emptyset$

E.g.:

$\{eid, lname\} \rightarrow \{lname\}$ is trivial.

$\{eid, lname\} \rightarrow \{lname, fname\}$ is non-trivial but not completely non-trivial.

$\{eid, lname\} \rightarrow \{fname\}$ is completely non-trivial.

A functional dependency $X \rightarrow Y$ is called a **full functional dependency** (FFD) if we cannot remove any attribute from X so that for the remaining set still determines Y .

We will also say that Y is fully (functionally) dependent on X .

Closure of X is a set of all attributes that are functionally dependent on X :

$$X^+ = \{A_i \mid X \rightarrow A_i\}$$

Algorithm to determine the closure X^+ :

1. Start with an empty set.
2. Add the attributes of X to the set.
3. Loop through the relation's FFDs $U_i \rightarrow V_i$:
If all attributes of U_i are in the set, add the attributes of V_i to it.
4. If the set changed during the last step, repeat it.

Consider a relation $R(A, B, C, D, E)$ with the following FFDs:

- ▶ $A, B \rightarrow C, D, E$
- ▶ $C \rightarrow A, B, D, E$
- ▶ $E \rightarrow D$

Determine the closures $\{A\}^+$, $\{B\}^+$, $\{A, B\}^+$, $\{C\}^+$, $\{D\}^+$, $\{C, D\}^+$ and $\{E\}^+$.

Let $R(A_1, A_2, \dots, A_n)$ denote a relation schema.

A set X is

- ▶ superkey of R if $X \rightarrow \{A_1, A_2, \dots, A_n\}$,
- ▶ superkey of R if $X^+ = \{A_1, A_2, \dots, A_n\}$,
- ▶ candidate key of R if $X \rightarrow \{A_1, A_2, \dots, A_n\}$ is a full functional dependency.

An attribute which is a member of any candidate key is called a **prime** attribute.

An attribute which is not a member of any candidate key is called a **non-prime** attribute.

Find all full functional dependencies and identify candidate keys, prime and non-prime attributes.
(Make your own assumptions if necessary.)

eid	lname	fname	did	dept	pid	project	hours
1	Alnes	Bernt	1	Planning	1	Alfa	100
2	Fjelldal	Mads	1	Planning	3	Charlie	140
4	Longva	Victor	1	Planning	2	Bravo	80
7	Bakke	Alfred	2	Production A	1	Alfa	80
8	Vie	Tor	2	Production A	1	Alfa	910
8	Vie	Tor	2	Production A	3	Charlie	720
9	Westgaard	Sten	2	Production A	3	Charlie	310
10	Liseth	Rakel	3	Production B	3	Charlie	480
11	Norman	Emil	3	Production B	3	Charlie	460
12	Dyrhaug	Atle	3	Production B	2	Bravo	810
15	Kvien	Amalie	4	Sales	2	Bravo	15
15	Kvien	Amalie	4	Sales	3	Charlie	75
16	Tveten	Thomas	4	Sales	2	Bravo	20
17	Lende	Marita	4	Sales	1	Alfa	40

Note: eid = employee ID, did = department ID, pid = project ID

A relation R is in 1NF if

- ▶ the domain of each attribute contains only atomic (indivisible) values, and
- ▶ each attribute value is a single value from the domain of that attribute.

1NF disallows:

- ▶ composite attributes
- ▶ multivalued attributes
- ▶ nested relations

A relation R is in 2NF if

- ▶ R is in 1NF, and
- ▶ no non-prime attribute is functionally dependent on any proper subset of any candidate key.

Equivalent definition: A relation R is in 2NF if

- ▶ R is in 1NF, and
- ▶ every non-prime attribute is fully functionally dependent on the whole of every candidate key.

EmpProj(eid, lname, fname, pid, project, hours)

eid	lname	fname	pid	project	hours
1	Alnes	Bernt	1	Alfa	100
2	Fjelldal	Mads	3	Charlie	140
4	Longva	Victor	2	Bravo	80
7	Bakke	Alfred	1	Alfa	80
8	Vie	Tor	1	Alfa	910
8	Vie	Tor	3	Charlie	720
9	Westgaard	Sten	3	Charlie	310
10	Liseth	Rakel	3	Charlie	480
11	Norman	Emil	3	Charlie	460
12	Dyrhaug	Atle	2	Bravo	810
15	Kvien	Amalie	2	Bravo	15
15	Kvien	Amalie	3	Charlie	75
16	Tveten	Thomas	2	Bravo	20
17	Lende	Marita	1	Alfa	40

$\{eid\} \rightarrow \{lname, fname\}$ and eid is a proper subset of the candidate key $\{eid, pid\}$.

A relation R is in 3NF if

- ▶ R is in 2NF, and
- ▶ no non-prime attribute is transitively dependent on any candidate key.

$X \rightarrow Y$ is a transitive dependency if there exists Z such that $X \rightarrow Z$, $Z \not\rightarrow X$, and $Z \rightarrow Y$ hold.

3NF is ofttest violated by having a non-prime attribute that is functionally determined by another non-prime attribute (or a set of non-prime attributes).

EmpDept(eid, fname, lname, did, dept)

eid	fname	lname	did	dept
1	Bernt	Alnes	1	Planning
2	Mads	Fjelldal	1	Planning
3	Karoline	Lekve	1	Planning
4	Victor	Longva	1	Planning
5	Ingvar	Nymo	2	Production A
6	Runar	Bodin	2	Production A
7	Alfred	Bakke	2	Production A
8	Tor	Vie	2	Production A
9	Sten	Westgaard	2	Production A
10	Rakel	Liseth	3	Production B
11	Emil	Norman	3	Production B
12	Atle	Dyrhaug	3	Production B
13	Jens	Kvistad	3	Production B
14	Lucas	Ulset	3	Production B
15	Amalie	Kvien	4	Sales
16	Thomas	Tveten	4	Sales
17	Marita	Lende	4	Sales

{did} → {dept}, and both did and dept are non-prime attributes.

Each NF implies the previous ones:

- ▶ If a relation is in 2NF, it is also in 1NF.
- ▶ If a relation is in 3NF, it is also in 2NF (and 1NF).

There are also higher normal forms (EKNF, BCNF, 4NF, ETNF, 5NF, DKNF, 6NF).

In practice it is usually enough to stop at 3NF or BCNF.

<u>eid</u>	lname	fname	email
1	Alnes	Bernt	alnes@example.com
2	Fjelldal	Mads	fjelldal@example.com, mads@fjelldal.info
3	Lekve	Karoline	lekve@example.com
4	Longva	Victor	longva@example.com
5	Nymo	Ingvar	nymo@example.com, boss@nymo.com



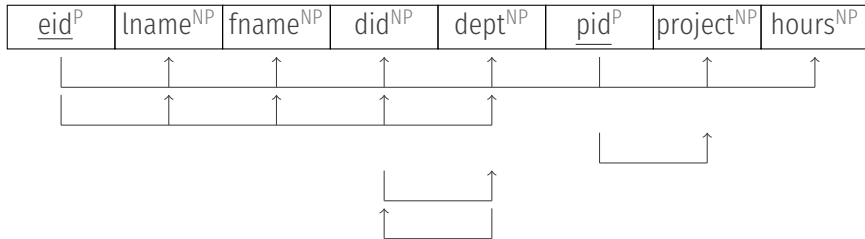
<u>eid</u>	lname	fname
1	Alnes	Bernt
2	Fjelldal	Mads
3	Lekve	Karoline
4	Longva	Victor
5	Nymo	Ingvar

<u>eid</u>	<u>email</u>
1	alnes@example.com
2	fjelldal@example.com
2	mads@fjelldal.info
3	lekve@example.com
4	longva@example.com
5	nymo@example.com
5	boss@nymo.com

UNF = “unnormalized” form, sometimes denoted by ONF.

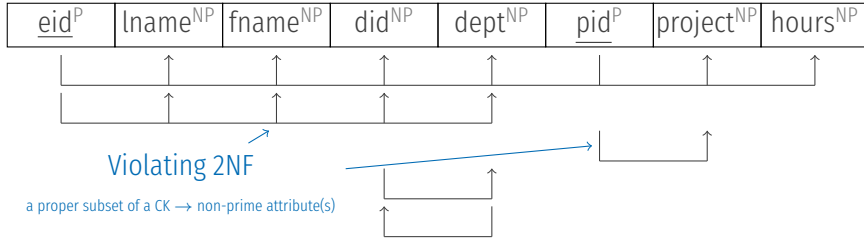
- ▶ Assumption: The relation is in the previous normal form.
- ▶ Find a non-trivial FFD $X \rightarrow Y$ that violates the given normal form.
 - ▶ Create a new relation with attributes X^+ and the primary key X .
 - ▶ Copy the distinct sets of values for these attributes to the new relation.
 - ▶ Remove the X^+ attributes except X from the original relation.
 - ▶ Add a FK for X in the original relation (referencing X in the new relation).
- ▶ Repeat until the relation conforms to the normal form.

Example from Exercise 3 with the following assumptions: Different people might have the same name, the department names are unique and there might be several projects with the same name. There is only one candidate key: {eid, pid}.



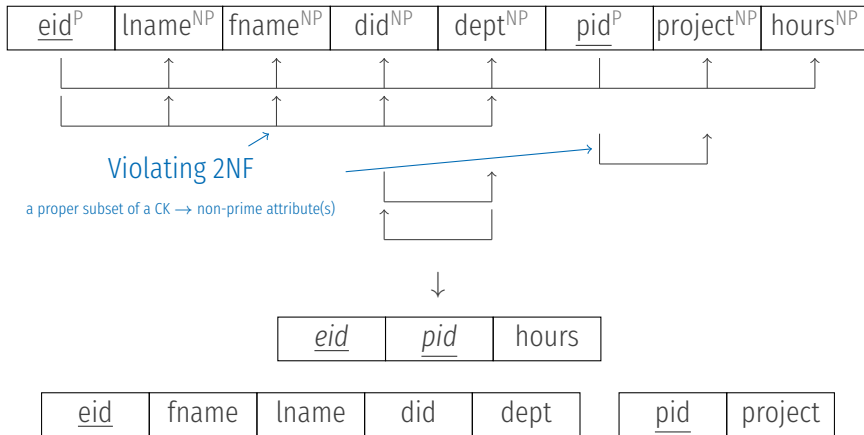
EXAMPLE: NORMALIZATION 1NF $\rightarrow$ 2NF

Example from Exercise 3 with the following assumptions: Different people might have the same name, the department names are unique and there might be several projects with the same name. There is only one candidate key: {eid, pid}.



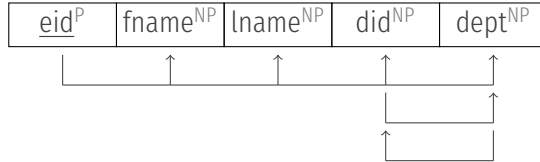
EXAMPLE: NORMALIZATION 1NF $\rightarrow$ 2NF

Example from Exercise 3 with the following assumptions: Different people might have the same name, the department names are unique and there might be several projects with the same name. There is only one candidate key: {eid, pid}.



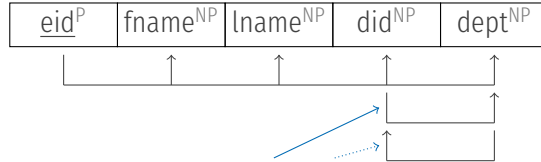
EXAMPLE: NORMALIZATION 2NF $\rightarrow$ 3NF

The relation with the data about employees is not in 3NF. There is only one CK: {eid}.



EXAMPLE: NORMALIZATION 2NF $\rightarrow$ 3NF

The relation with the data about employees is not in 3NF. There is only one CK: {eid}.

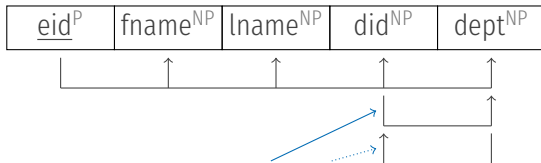


Violating 3NF

non-prime attribute $\rightarrow$ non-prime attribute

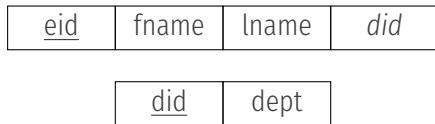
EXAMPLE: NORMALIZATION 2NF $\rightarrow$ 3NF

The relation with the data about employees is not in 3NF. There is only one CK: {eid}.



Violating 3NF

non-prime attribute $\rightarrow$ non-prime attribute



Booking system at a dental clinic:

Booking(DentistNo, DentistFullName, PatientSSN, PatientFullName, Date, Time, SurgeryNo)

- ▶ Are there some attributes violating 1NF? How can we fix that?
- ▶ Identify all FFD.
- ▶ Identify all CKs, prime and non-prime attributes.
- ▶ In which NF is the table?
- ▶ Normalize the database to 3NF.

Make your own assumptions if necessary.