

CSCI235 Database Systems

Database Normalization

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Database normalization

Outline

First Normal Form (1NF)

Keys

Functional dependencies and keys

Attributes

Full and partial functional dependencies

Second Normal Form (2NF)

Transitive functional dependencies

Third Normal Form (3NF)

Boyce-Codd Normal Form (BCNF)

Normalization of relational schemas

First Normal Form (1NF)

A relational schema is in the **First Normal Form (1NF)** if all occurrences of rows in the respective relational table contain the same number of fields and include the atomic values only, i.e. there are no repeating fields and groups

1NF relational table								
cnumber	fname	lname	onumber	odate	lnumber	item	price	total
7	James	Bond	7	2017-01-01	1	bolt	23.04	5
7	James	Bond	7	2017-01-01	2	nut	29.01	3
7	James	Bond	8	2017-01-02	1	nut	4.55	2
7	James	Bond	8	2017-01-02	2	pin	14.25	2

A relational schema which is not in the **First Normal Form (1NF)** is in **Zerth Normal Form (0NF)** or it is called as a **nested relational schema**

A relational table built over a **nested relational schema** is called as a **nested relational table** or **0NF relational table**.

First Normal Form (1NF)

1NF relational table

cnumber	fname	lname	orders			
7	James	Bond	onumber	odate	parts	
			lnumber	item	price	total
			7	2017-01-01	1	bolt 23.04 5
					2	nut 29.01 3
			8	2017-01-02	1	nut 4.55 2
					2	pin 14.25 2

In HTML view press 'p' to see the lecture notes

[TOP](#)

Created by Janusz R. Getta, CSCI235 Database Systems, Autumn 2024

4/43

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Keys

A **superkey** is a nonempty subset X of relational schema $R = (A_1, \dots, A_n)$ such that for any two rows $t_i, t_j, t_i \neq t_j$ in a relational table created over a relational schema R $t_i[X] \neq t_j[X]$

If X is a **superkey** in R then $X \rightarrow A_1, \dots, A_n$

A **minimal key** is a **superkey** K with an additional property such that removal of any attribute from K causes K not to be a **superkey**

For example, a relational schema `TRIP(rego#, licence#, tdate)` has one **minimal key** `(rego#, licence#, tdate)`

For example, a relational schema `DRIVER(licence#, employee#, first-name, last-name)` has two **minimal keys** `(licence#)` and `(employee#)`

Keys

For example, relational schema `DRIVER(licence#, employee#, first-name, last-name)` has many **superkeys**: `(licence#)`, `(employee#)`, `(licence#, employee#)`, `(licence#, first-name)`, `(licence#, last-name)`, `(licence#, first-name, last-name)`, and so on ...

A **primary key** is an arbitrarily selected **minimal key**

A **candidate key** is any other **minimal key** which is not a **primary key**

For example, a relational schema

`TRIP(rego#, licence#, tdate)`

has a **primary key** `(rego#, licence#, tdate)`

For example, a relational schema

`DRIVER(licence#, employee#, first-name, last-name)`

has a **primary key** `(licence#)` and a **candidate key** `(employee#)` or the opposite

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Functional dependencies and keys

Let $R = (A_1, \dots, A_n)$ be a relational schema and let X, Y be nonempty subsets of R such that $X \cup Y = R$

If a functional dependency $X \rightarrow Y$ is valid in R then X is a **superkey**

If X is a **superkey** then a functional dependency $X \rightarrow Y$ is valid in R

For example, if a functional dependency **licence#, employee#** \rightarrow **first-name, last-name** is valid in a relational schema

`DRIVER(licence#, employee#, first-name, last-name)`

then **(licence#, employee#)** is a **superkey**

For example, if **(licence#)** is a superkey in a relational schema

`DRIVER(licence#, employee#, first-name, last-name)`

then a functional dependency **licence#** \rightarrow **employee#, first-name, last-name** is valid in `DRIVER`

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Attributes

A **prime attribute** is an attribute from relational schema **R** which is a member of at least one minimal key in **R**

A **nonprime attribute** is an attribute which is not **prime**

For example, the attributes **licence#** and **employee#** are **prime attributes** in a relational schema

```
DRIVER(licence#, employee#, first-name, last-name)
```

For example, the attributes **first-name** and **last-name** are **nonprime attributes** in a relational schema

```
DRIVER(licence#, employee#, first-name, last-name)
```

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Full and partial functional dependencies

A **full functional dependency** is a functional dependency $X \rightarrow Y$ such that removal of any attribute A from X causes that $(X-A) \not\rightarrow Y$

A **partial functional dependency** is a functional dependency which is not full functional dependency

For example, a functional dependency
 $\text{student\#, subject\#, edate} \rightarrow \text{grade}$
valid in a relational schema

`ENROLMENT(student#, subject#, edate, grade)` is a **full functional dependency** because none of the attributes student\# , subject\# , edate can be removed from the left hand side of the functional dependency such that it is still valid

For example, a functional dependency
 $\text{licence\#, employee\#} \rightarrow \text{first-name, last-name}$ valid in a relational schema `DRIVER(licence#, employee#, first-name, last-name)` is a **partial functional dependency** because if either licence\# or employee\# attributes are removed from the left hand side of the functional dependency then it is still valid

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Second Normal Form (2NF)

A relational schema R is in the **Second Normal Form (2NF)** if every nonprime attribute A in R is fully functionally dependent on all minimal keys of schema R

A relational table based on a relational schema

`INVENTORY(part, quantity, warehouse, warehouse-address)`

contains information about parts stored in warehouses, quantities of parts, and addresses of warehouse

The following functional dependencies are valid in a relational schema

`INVENTORY(part, quantity, warehouse, warehouse-address)`

`warehouse → warehouse-address`

`part, warehouse → quantity`

If `warehouse → warehouse-address` then

`part, warehouse → warehouse-address`

If `part, warehouse → warehouse-address` and `part, warehouse →`

`quantity` then `part, warehouse → warehouse-address, quantity`

Second Normal Form (2NF)

If $\text{part, warehouse} \rightarrow \text{warehouse-address, quantity}$ then a minimal key is (part, warehouse)

A relational schema

$\text{INVENTORY}(\text{part, quantity, warehouse, warehouse-address})$

is not in 2NF because nonprime attribute warehouse-address depends on a part (warehouse) of a key (part, warehouse)

A functional dependency that violates 2NF is $\text{warehouse} \rightarrow \text{warehouse-address}$

If all minimal keys in a relational schema consist of only one attribute (single attribute keys) then such schema is always in 2NF

This is because any nonprime attribute in the schema does not depend on a part of a key because each key consists of one attribute only

Second Normal Form (2NF)

A relational schema

`INVENTORY(part, quantity, warehouse, warehouse address)`

must be decomposed into the relational schemas

`INVENTORY(part, quantity, warehouse)`

`WAREHOUSE(warehouse, warehouse-address)`

The following functional dependencies are valid in a relational schema

`INVENTORY(part, quantity, warehouse)`

`part, warehouse → quantity`

Hence `(part, warehouse)` is a minimal key

A relational schema `INVENTORY(part, quantity, warehouse)`

is in **2NF** because a nonprime attribute `quantity` does not depend on a part of key `(part, warehouse)`

Second Normal Form (2NF)

The following functional dependencies are valid in a relational schema

`WAREHOUSE(warehouse, warehouse-address)`

`warehouse → warehouse-address`

Hence `(warehouse)` is a minimal key

A relational schema `WAREHOUSE(warehouse, warehouse-address)`

is in **2NF** because a nonprime attribute `warehouse-address` does not depend on a part of key `(warehouse)`

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Transitive functional dependencies

A functional dependency $X \rightarrow Y$ valid in a relational schema R is a **transitive functional dependency** if there exists a nonempty subset Z of R , that is not a subset of any key in R and such that the functional dependencies $X \rightarrow Z$ and $Z \rightarrow Y$ are valid in R

For example, if the functional dependencies $\text{employee\#} \rightarrow \text{project-title}$ and $\text{project-title} \rightarrow \text{department-name}$ are valid in a relational schema

`DEPARTMENT(department-name, project-title, employee#)`

then a functional dependency $\text{employee\#} \rightarrow \text{department-name}$ is a **transitive functional dependency**

For example, if the functional dependencies $\text{licence\#} \rightarrow \text{employee\#}$ and $\text{employee\#} \rightarrow \text{first-name}$ are valid in a relational schema

`DRIVER(licence#, employee#, first-name, last-name)`

then a functional dependency $\text{licence\#} \rightarrow \text{first-name}$ is **not a transitive functional dependency**

It is because `(employee#)` is a key in a relational schema `DRIVER`

Transitive functional dependencies

We say that Y is **transitively dependent** on X in a schema R if $X \rightarrow Y$ is valid in R and $X \rightarrow Y$ is a **transitive functional dependency**

For example, if the functional dependencies $\text{trip\#} \rightarrow \text{licence\#}$, $\text{licence\#} \rightarrow \text{employee\#}$, and $\text{employee\#} \rightarrow \text{licence\#}$ are valid in a relational schema $\text{TRIP}(\text{trip\#}, \text{licence\#}, \text{employee\#})$ then an attribute employee\# is **transitively dependent** on an attribute trip\# and ...

... an attribute licence\# is **transitively dependent** on an attribute trip\#

Then, information about a licence\# of a driver with a given employee\# is listed as many times as the total number of trips performed by the driver

For example, if a driver performed 100 trips then his/her licence\# is listed together with his/her employee\# 100 times.

Transitive functional dependencies

For example, if the functional dependencies `textbook → subject` and `subject → lecturer` are valid in a relational schema `OUTLINE(lecturer, subject, textbook)` then an attribute `lecturer` is **transitively dependent** on an attribute `textbook`

Then, information about a `lecturer` assigned to a `subject` is repeated as many times as many `textbooks` are listed for the `subject`

For example, if a `subject` has 2 `textbooks` then information about a `lecturer` assigned to a `subject` is listed twice

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Third Normal Form (3NF)

A relational schema R is in the **Third Normal Form (3NF)** if it is in **2NF** and no **nonprime attribute** of R is **transitively dependent** on the primary key

A relational table based on a relational schema

`SUPPLIER(s#, sname, company-name, city)`

contains information about suppliers working for a company located in a given city

The following functional dependencies are valid in a relational schema

`SUPPLIER(s#, sname, company-name, city)`

$s\# \rightarrow sname$

$s\# \rightarrow company-name$

$s\# \rightarrow city$

$company-name \rightarrow city$

A primary key in a relational schema `SUPPLIER` is ($s\#$) because

$s\# \rightarrow sname, company-name, city$

Third Normal Form (3NF)

A relational schema **SUPPLIER** is **not** in the **Third Normal Form (3NF)** because an attribute **city** is transitively dependent on a primary key (**s#**)

An attribute **city** is transitively dependent on a primary key (**s#**) because

$s\# \rightarrow \text{company-name}$ and $\text{company-name} \rightarrow \text{city}$

A relational schema **SUPPLIER** is in the **Second Normal Form (2NF)** because each nonprime attribute **sname**, **company-name**, and **city** is fully functionally dependent on a primary key (**s#**)

$s\# \rightarrow \text{sname}$

$s\# \rightarrow \text{company-name}$

$s\# \rightarrow \text{city}$

Third Normal Form (3NF)

A relational schema `SUPPLIER(s#, sname, company-name, city)` should be decomposed into the relational schemas

`SUPPLIER(s#, sname, company-name)`

`COMPANY(company-name, city)`

A relational schema `SUPPLIER(s#, sname, company-name)` is in **3NF** because no attribute is transitively dependent on a primary key (`s#`)

`s# → sname`

`s# → company-name`

A relational schema `COMPANY(company-name, city)` is in **3NF** because no nonprime attribute is transitively dependent on a primary key (`company-name`)

`company-name → city`

Third Normal Form (3NF)

Any relational schema that consists of at most 2 attributes is always in 3NF

Let $R(a,b)$ be a relational schema such that no nontrivial functional dependencies are valid in R

Then (a, b) is a primary key in R and ...

... no nonprime attribute is transitively dependent on a primary key (a, b)

Let $R(a,b)$ be a relational schema such that a functional dependency $a \rightarrow b$ is valid in R

Then (a) is a primary key in R and ...

... no nonprime attribute is transitively dependent on a primary key (a)

Third Normal Form (3NF)

Let $R(a,b)$ be a relational schema such that the functional dependencies $a \rightarrow b$ and $b \rightarrow a$ are valid in R

Then either (a) is a primary key in R or (b) is a primary key in R and ...

... no nonprime attribute is transitively dependent on either (a) or (b)

Third Normal Form (3NF)

Alternative definition of the **Third Normal Form**

A relational schema **R** is in the **Third Normal Form (3NF)** if whenever a functional dependency $X \rightarrow A$ is valid in **R** then either ...

- **X** is a superkey in **R** or
- **A** is a prime attribute in **R**

For example, a relational schema

`SUPPLIER(s#, sname, company-name, city)`

$s\# \rightarrow sname$

$s\# \rightarrow company-name$

$s\# \rightarrow city$

$company-name \rightarrow city$

is **not** in **3NF** because ...

... if we consider a functional dependency $company-name \rightarrow city$ then ...

- an attribute `company-name` is not a superkey and
- an attribute `city` is not a prime attribute

[In HTML view press 'p' to see the lecture notes](#)

[TOP](#)

Created by Janusz R. Getta, CSCI235 Database Systems, Autumn 2024

29/43

Third Normal Form (3NF)

For example, if the functional dependencies

$city, street \rightarrow zipcode$

$zipcode \rightarrow city$

are valid in a relational schema `LOCATION(city, street, zipcode)`

then the schema has two minimal keys

`(city, street)`

Directly implied by a functional dependency $city, street \rightarrow zipcode$

`(zipcode, street)`

If $zipcode \rightarrow city$ then $zipcode, street \rightarrow city, street$

A relational schema `LOCATION(city, street, zipcode)` is in **3NF**

because

- the left hand side of $city, street \rightarrow zipcode$ is a superkey and
- the left side of $zipcode \rightarrow city$ is not a superkey but ... the right hand side (`city`) of $zipcode \rightarrow city$ is a prime attribute

Third Normal Form (3NF)

The following relational table created over a relational schema `LOCATION(city, street, zipcode)` is redundant

city	street	zipcode
NY	55	484
NY	56	484
LA	55	473
LA	56	473
LA	57	474

3NF relational table

because the repetitions of | LA ... 473 | and | NY ... 484 | are forced by a functional dependency `zip-code → city`

It means that **3NF** is not the highest normal form required !

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Boyce-Codd Normal Form (BCNF)

A relational schema R is in the **Boyce-Codd Normal Form (BCNF)** if whenever a functional dependency $X \rightarrow A$ is valid in R then

- X is a superkey in R

Boyce-Codd Normal Form is more restrictive because its definition does not give the "second chance"

- A is a prime attribute in R

For example, if the functional dependencies

$city, street \rightarrow zipcode$

$zipcode \rightarrow city$

are valid in a relational schema $LOCATION(city, street, zipcode)$

then the schema has two minimal keys

$(city, street)$ and $(zipcode, street)$

Then, a relational schema $LOCATION$ is not in **BCNF** because

- the left side of $zipcode \rightarrow city$ is not a superkey

Boyce-Codd Normal Form (BCNF)

A relational schema `LOCATION(city, street, zipcode)`

should be decomposed into the relational schemas

`SZ(street, zipcode)`

`CZ(city, zipcode)`

A relational schema `SZ(street, zipcode)` has no valid nontrivial functional dependencies

A minimal key in a relational schema `SZ(street, zipcode)` is

`(street, zipcode)`

A relational schema `SZ(street, zipcode)` is in **BCNF** because does not exist a functional dependency whose left hand side is not a superkey

Boyce-Codd Normal Form (BCNF)

A functional dependency $\text{zipcode} \rightarrow \text{city}$ is valid in a relational schema $\text{CZ}(\text{city}, \text{zipcode})$

A minimal key in a relational schema $\text{CZ}(\text{city}, \text{zipcode})$ is (zipcode)

A relational schema $\text{CZ}(\text{city}, \text{zipcode})$ is in **BCNF** because the left hand of functional dependency $\text{zipcode} \rightarrow \text{city}$ is a superkey (zipcode)

Every relational schema, which consists of at most 2 attributes is always in **BCNF**

Normalization to **BCNF** "costs" a functional dependency $\text{city}, \text{street} \rightarrow \text{zipcode}$

It means that it is impossible to enforce the functional dependency $\text{city}, \text{street} \rightarrow \text{zipcode}$ with a primary key or candidate key constraints of **CREATE TABLE** statement

Boyce-Codd Normal Form (BCNF)

The following relational tables are created through decomposition of a relational schema `LOCATION(city, street, zipcode)` into the relational schemas `SZ(street, zipcode)` and `CZ(city, zipcode)`

LOCATION			SZ		CZ	
NY	55	484	55	484	NY	484
NY	56	484	56	484	LA	473
LA	55	473	55	473	LA	474
LA	56	473	56	473		
LA	57	474	57	474		

Decomposition into BCNF tables

A functional dependency `city, street → zipcode` cannot be enforced in the relational tables `SZ` and `CZ`

Database normalization

Outline

[First Normal Form \(1NF\)](#)

[Keys](#)

[Functional dependencies and keys](#)

[Attributes](#)

[Full and partial functional dependencies](#)

[Second Normal Form \(2NF\)](#)

[Transitive functional dependencies](#)

[Third Normal Form \(3NF\)](#)

[Boyce-Codd Normal Form \(BCNF\)](#)

[Normalization of relational schemas](#)

Normalization of relational schemas

Let $R = (A_1, \dots, A_n)$ be a relational schema (a header of relational table)

Normalization of a relational schema R is performed in the following way

- Identify all functional dependencies valid in a relational schema R
- Use the functional dependencies to derive all minimal keys
- Use the functional dependencies and minimal keys to identify the highest normal form satisfied by a relational schema R
- Decompose a relational schema R into the relational schemas in **BCNF (3NF)**

Normalization of relational schemas

For example, consider a relational schema

`SHIPMENT(s#, city, status, p#, quantity)`

The following functional dependencies are valid in the schema

$s\# \rightarrow city$

$s\# \rightarrow status$

$city \rightarrow status$

$s\#, p\# \rightarrow quantity$

$s\#, p\# \rightarrow city$

$s\#, p\# \rightarrow status$

A minimal key is $(s\#, p\#)$

A relational schema `SHIPMENT(s#, city, status, p#, quantity)`

is **not** in **2NF** because a nonprime attribute `city` depends on a subset of a minimal key $(s\#, p\#)$, $s\# \rightarrow city$

Normalization of relational schemas

A relational schema `SHIPMENT(s#, city, status, p#, quantity)` should be decomposed into the relational schemas `SP(s#, p#, quantity)` with a minimal key `(s#, p#)` and `SUPPLIER(s#, city, status)` with a minimal key `(s#)`

A relational schema `SP(s#, p#, quantity)` is in **BCNF** because `s#, p# → quantity`, i.e. left hand side of the functional dependency is a superkey

A relational schema `SUPPLIER(s#, city, status)` is **not** in **3NF** because an attribute `status` is transitively dependent on an attribute `s#`, i.e. `s# → city` and `s# → status`

A relational schema `SUPPLIER(s#, city, status)` is **not** in **3NF** because left hand side of functional dependency `city → status` is not a superkey and right hand side is not prime attribute

Normalization of relational schemas

A relational schema `SUPPLIER(s#, city, status)` should be decomposed into the relational schemas

`SUPPLIERCITY(s#, city)` with a minimal key (`s#`)

`LOCATION(city, status)` with a minimal key (`city`)

A relational schema `SUPPLIERCITY(s#, city)` is in **BCNF** because $s\# \rightarrow city$, i.e. left hand side of the functional dependency is a superkey

A relational schema `LOCATION(city, status)` is in **BCNF** because $city \rightarrow status$, i.e. left hand side of the functional dependency is a superkey

Normalization of relational schemas

A relational schema `SUPPLIER(s#, city, status)` can be alternatively decomposed into the relational schemas `SUPPLIERCITY(s#, city)` with a minimal key `(s#)` and `SUPPLIERSTAT(s#, status)` with a minimal key `(s#)`

A relational schema `SUPPLIERCITY(s#, city)` is in **BCNF** because `s# → city`, i.e. left hand side of the functional dependency is a superkey

A relational schema `SUPPLIERSTAT(s#, status)` is in **BCNF** because `s# → status`, i.e. left hand side of the functional dependency is a superkey

References

T. Connolly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 14.5 The Process of Normalization, Chapter 14.6 First Normal Form (1NF), Chapter 14.7 Second Normal Form (2NF), Chapter 14.8 Third Normal Form (3NF), Chapter 14.9 General definitions of 2NF and 3NF, Chapter 15.2 Boyce-Codd Normal Form (BCNF) Pearson Education Ltd, 2015