Course Code: **IS201**

Course Title : **Database**

Dr. Mahmoud Zaher

Egyptian Russian University (ERU)

Faculty of Artificial Intelligence

Text Book

The concepts and presentation of this course are drawn from:

- R. ElMasri & S. Navathe, “Fundamentals of Database Systems”, Addison Wesley, Fifth Edition, 20011.
- Carlos M. Coronel - Database Systems_ Design, Implementation, & Management-Cengage Learning (2018).
- Learn SQL Database Programming_ Query and manipulate databases from popular relational database servers using SQL-Packt Publishing (2020)

Functional Dependencies and Normalization for Relational Databases

Functional Dependency (FD)

- A functional dependency is a constraint between two sets of attributes, say X and Y, from the database.
- A functional dependency X → Y is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more,

  that is for any attribute $A \in X$,

  (X-|A|) does not functionally determine Y.

Functional Dependency

- A FD $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds;

that is for some $A \in X$, $(X-|A|) \rightarrow Y$.

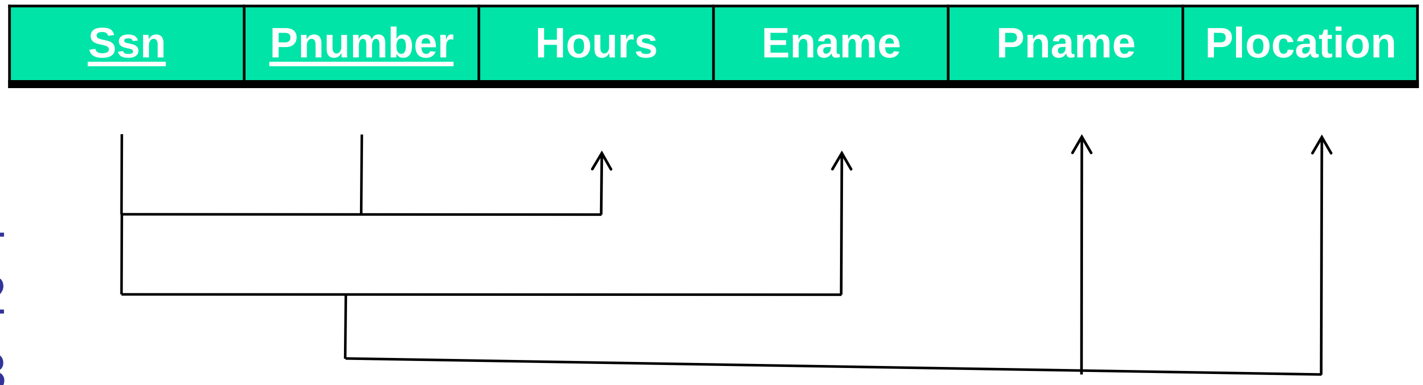
Functional Dependency

- For example in the relation EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

- {Ssn, Pnumber} → Hours is a **full dependency** (neither Ssn → Hours nor Pnumber → Hours holds).
- The dependency {Ssn, Pnumber} → Ename is **partial** because Ssn → Ename holds

Functional Dependency



Ssn	Pnumber	Hours	Ename	Pname	Plocation

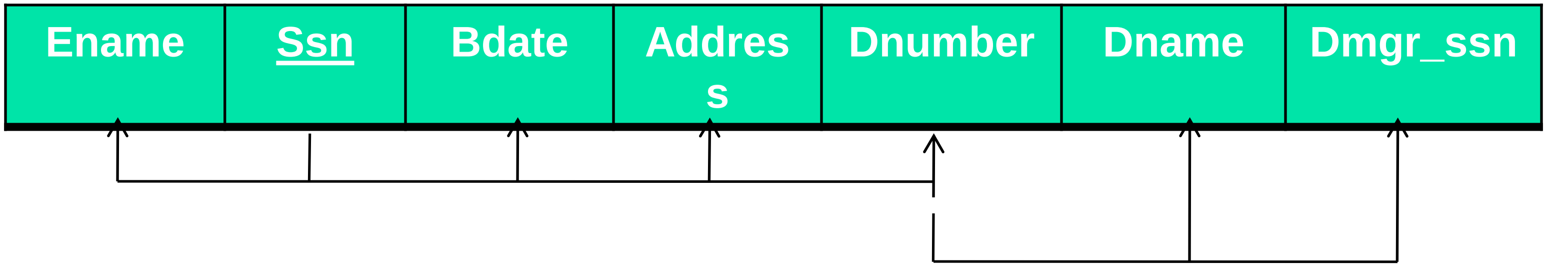
- FD1
- FD2
- FD3

Functional Dependency

- A FD X → Y is a **transitive dependency** if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both X → Y and Z → Y hold.
- The dependency Ssn → Dmgr_ssn is transitive through Dnumber in EMP_DEPT because both the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn hold and dnumber is neither a key itself nor a subset of the key of EMP_DEPT.

Functional Dependency

- EMP_DEPT



Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

Normalization

- **The normalization** process takes a relation schema through a series of tests to certify whether it satisfies a certain normal form.
- **The process**, which proceed in a top down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary,
- It is considered as relational design by analysis.

Normal Forms

Normal form	Traditional definition	algorithm
First normal form (1NF)	• All attributes must be atomic, and • No repeating groups	• **Eliminate** multi-valued attributes, and • Eliminate repeated attributes
Second normal form (2NF)	• First normal form, and • No partial functional dependencies	• **Eliminate** subkeys **(**where the subkey is part of a composite primary key**)**
Third normal form (3NF)	• Second normal form, and • No transitive functional dependencies	• **Eliminate** subkeys **(**where the subkey is not part of the primary key**)**

First Normal Form (1NF)

- 1NF states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute.

First Normal Form (1NF)

A relation schema that is not in 1NF

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	{ **Bellaire, Sugarland, Houston** }
Administration	4	987654321	{ **Stafford** }
Headquarter	1	888665555	{ **Houston** }

1NF version of the same relation with redundancy

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	**5**	**333445555**	**Bellaire**
Research	**5**	**333445555**	**Sugarland,**
Research	**5**	**333445555**	**Houston**
Administration	4	987654321	**Stafford}**
Headquarter	1	888665555	**Houston**

First Normal Form (1NF)

- Remove the attribute Dlocations and place it in a separate relation with a primary key { Dnumber, Dlocation }1NF

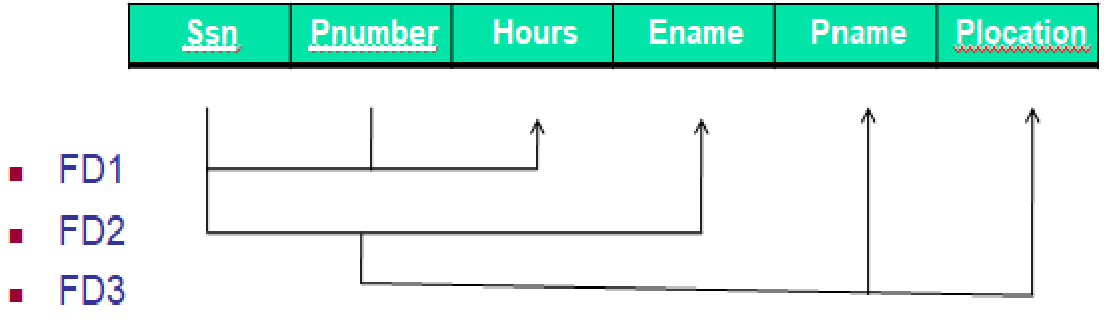
Dname	Dnumber	Dmgr_ssn

Dnumber	Dlocation

Second Normal Form (2NF)

- 2NF is based on the concept of **full functional dependency** (FD)
- A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

Second Normal Form (2NF)



Ssn	Pnumber	Hours	Ename	Pname	Plocation

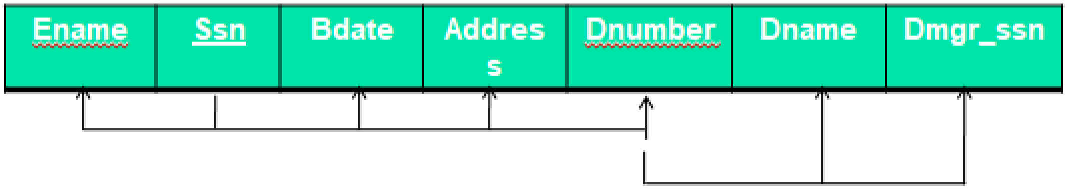
Normalizing EMP_PROJ relation into 2NF

Ssn	Pnumber	Hours

Ssn	Ename

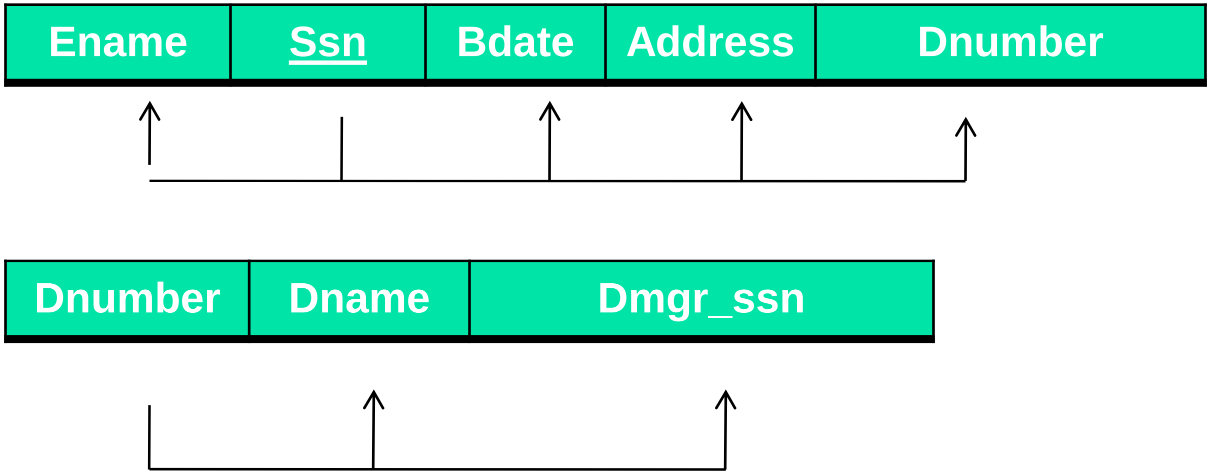
Pnumber	Pname	Plocation

Third Normal Form (3NF)



Ename	Ssn	Bdate	Addres s	Dnumber	Dname	Dmgr_ssn

- A relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.
- Normalizing EMP_DEPT relation into 3NF



Ename	Ssn	Bdate	Address	Dnumber

Dnumber	Dname	Dmgr_ssn

Normal Forms

- Normalization is usually thought of as a process of applying a **set of rules to your database design**, mostly to **achieve minimum redundancy in the data**.
- Most textbooks present this as a three-step process, with correspondingly labeled "**normal forms**," which could be done in an almost algorithmic sequence.

Normal Forms

- • In theory, you could start with a single relation scheme (sometimes called the universal scheme, or *U*) that contains all of the attributes in the database—then apply these rules recursively to develop a set of increasingly-normalized sub-relation schemes.
- **When all of the schemes are in third normal form, then the whole database is properly normalized**.

Normal Forms

- In practice, you will more likely apply the rules gradually, refining each relation scheme as you develop it from the UML class diagram or ER model diagram.
- The final table structures should be the same no matter which method (or combination of methods) you've used.

Normal Forms

Normal form	Traditional definition	algorithm
First normal form (1NF)	• All attributes must be atomic, and • No repeating groups	• **Eliminate** multi-valued attributes, and • Eliminate repeated attributes
Second normal form (2NF)	• First normal form, and • No partial functional dependencies	• **Eliminate** subkeys **(**where the subkey is part of a composite primary key**)**
Third normal form (3NF)	• Second normal form, and • No transitive functional dependencies	• **Eliminate** subkeys **(**where the subkey is not part of the primary key**)**

Example:

Consider the following relation:
CAR_SALE(Car#, Date_sold, Salesman#, Commision%, Discount_amt)

Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the primary key.
Additional dependencies are:
Date_sold → Discount_amt
and
Salesman# → commission%

Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?

Given the relation schema

CAR_SALE(Car#, Date_sold, Salesman#, Commission%, Discount_amt)

Answer:

with the functional dependencies

Date_sold → Discount_amt

Salesman# → Commission%

This relation is not satisfies 1NF , 2NF or 3NF

To normalize,

1NF:

CAR_SALE1(Car#, Date_sold, Discount_amt)

CAR_SALE2(Car#, Salesman#, Commission%)

2NF:

Car_Sale1(Car#, Date_sold, Discount_amt)

Car_Sale2(Car#, Salesman#)

Car_Sale3(Salesman#, Commission%)

3NF:

Car_Sale1-1(Car#, Date_sold)

Car_Sale1-2(Date_sold, Discount_amt)

Car_Sale2(Car#, Salesman#)

Car_Sale3(Salesman#, Commission%)

CAR_SALE(Car#, Date_sold, Salesman#, Commission%, Discount_amt)

Assume that a car may be sold by multiple salesmen and hence
{CAR#, SALESMAN#} is the primary key.

Additional dependencies are:

Date_sold → Discount_amt and Salesman# → commission%

Car#	Date_sold	Salesman#	Commission%	Discount_amt

1NF:

CAR_SALE1(Car#, Date_sold, Discount_amt)
CAR_SALE2(Car#, Salesman#, Commision%)

Car#	Date_sold	Discount_amt

Car#	Salesman#	Commission%

2NF:

Car_Sale1(Car#, Date_sold, Discount_amt)
Car_Sale2(Car#, Salesman#)
Car_Sale3(Salesman#, Commission%)

Car#	Date_sold	Discount_amt

Car#	Salesman#

Salesman#	Commission%

3NF:

Car_Sale1-1(Car#, Date_sold)
Car_Sale1-2(Date_sold, Discount_amt)
Car_Sale2(Car#, Salesman#)
Car_Sale3(Salesman#, Commission%)

Car#	Date_sold

Date_sold	Discount_amt

Car#	Salesman#

Salesman#	Commission%

Quiz1

Consider the following relation:
CAR_SALE(Car#, Date_sold, Salesman#, Commission%, Discount_amt)

Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the primary key.
Additional dependencies are:
Date_sold → Discount_amt
and
Salesman# → commission%

- **Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?**
- **Draw the Schema Diagram for the given relation in the 3NF?**

Quiz2

Consider the following relation:
CAR_SALE(Car#, Date_sold, Salesman#, Commission%, Discount_amt)

Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the primary key.
Additional dependencies are:
Date_sold → Discount_amt
and
Salesman# → commission%

- **Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?**
- **Draw the ERD for the given relation in the 3NF?**

Summary

- Functional Dependencies (FDs)
  - Definition, Inference Rules, Equivalence of Sets of FDs, Minimal Sets of FDs
- Normal Forms Based on Primary Keys