Database Systems
(CS 355 / CE 373)

Dr. Umer Tariq
Assistant Professor,
Dhanani School of Science & Engineering,
Habib University

Acknowledgements

- Many slides have been borrowed from the official lecture slides accompanying the textbook:

  Database System Concepts, (2019), Seventh Edition,

  Avi Silberschatz, Henry F. Korth, S. Sudarshan

  McGraw-Hill, ISBN 9780078022159

  The original lecture slides are available at:

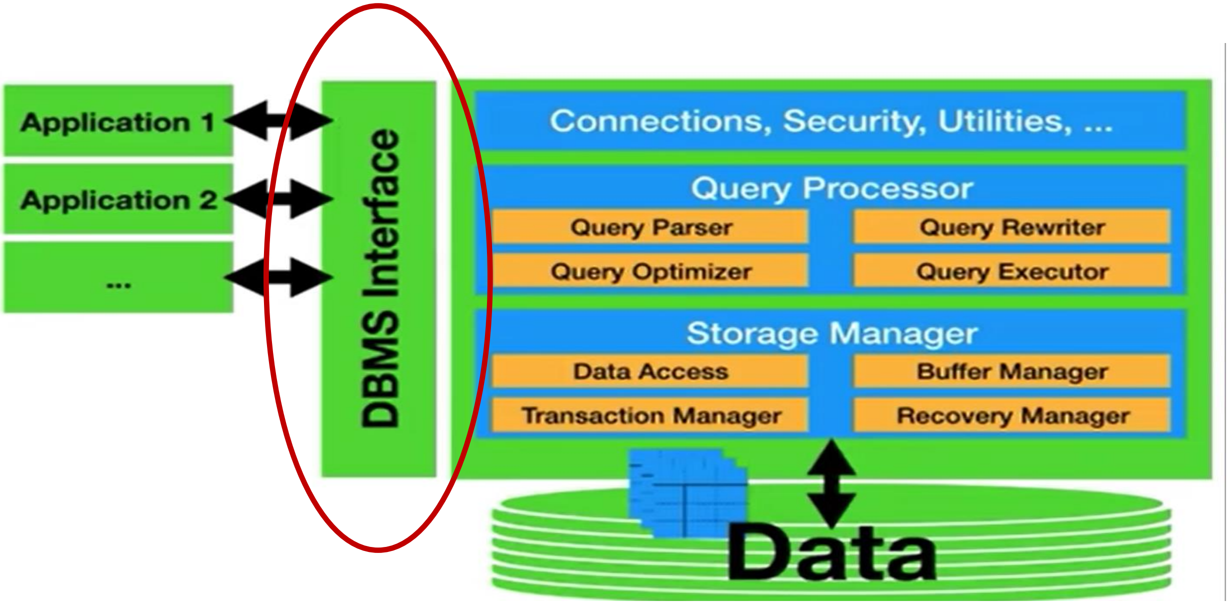
  https://www.db-book.com/

- Some of the slides have been borrowed from the lectures by Dr. Immanuel Trummer (Cornell University). Available at: (www.itrummer.org)

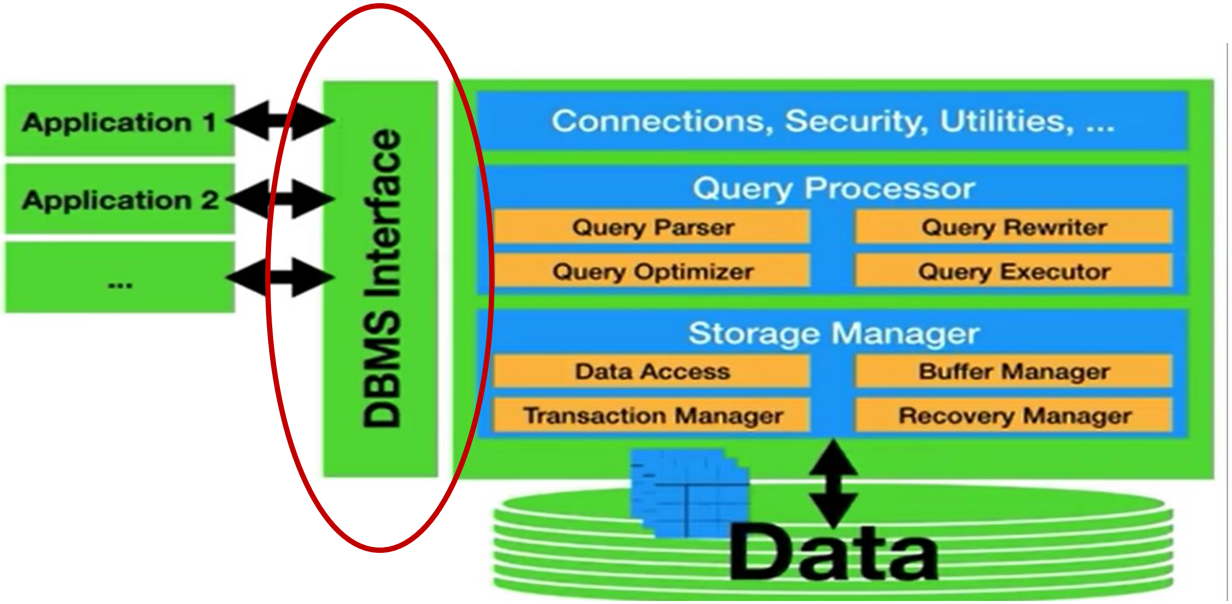
Outline: Week 4

- Elements of Good Relational Design
- 1NF
- 2NF
- 3NF

What should be the DBMS Interface?



What should be the DBMS Interface?



- Data Model
  - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

The Relational Model

- The relational model uses <u>a collection of tables</u> to represent both data and the relationships among those data.

Table/ Relation
instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Column / Attribute
dept_name

Row / Tuple
(83821, Brandt, Comp. Sci., 92000)

Figure 2.1 The *instructor* relation.

Schema Design 101

- Identify Tables
- Identify Columns/Attributes associated with each table
- Identify Primary Keys
- Identify relationships among tables through Foreign Keys

Database Design Process

- Specification of User Requirements
  - Involves extensive interaction with the users
- Conceptual Design
  - User requirements are translated into a conceptual schema of the database (such as an **E-R Diagram**)
- Specification of Functional Requirements
  - With the help of users, describe the kind of operations (modifying, searching, retrieving, updating) that will be performed on the data
  - At this stage, the designer can review the conceptual schema to ensure that it meets functional requirements
- Logical Design

  Translation

  - The designer maps the high-level conceptual schema (such as an **E-R Diagram**) into the implementation data model (such as the **relational data model**)

  Relational Schema
- Physical Design
  - The designer can specify the physical features of the database (such as the form of file organization) to optimize the performance of the database.

① Entity-Relationship Diagram vs Relational Schema Diagram

User Interviews (DESCRIPTION OF A SCENARIO)

② E-R Diagram

Translation

③ Relational Schema

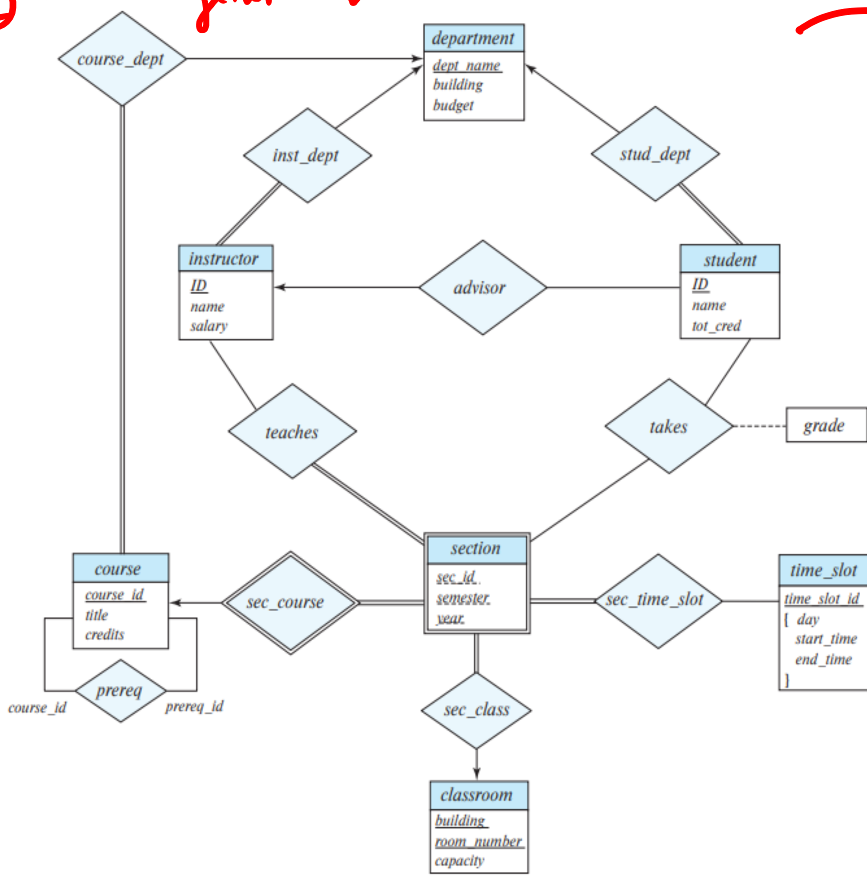
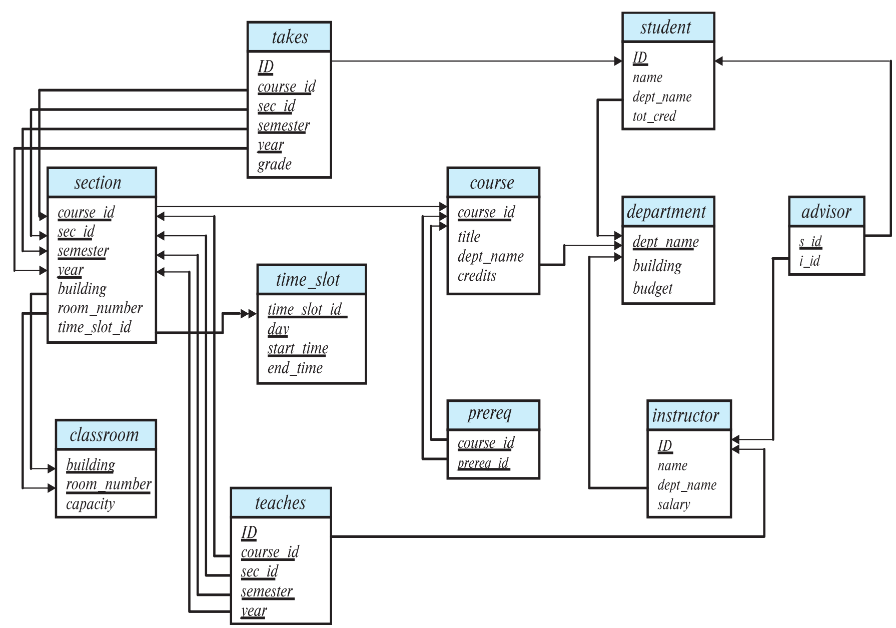


Figure 6.15 E-R diagram for a university enterprise.



Elements of Good Relational Design

- Ensure that the semantics of the attributes are clear in the schema
- Reduce redundant information in tuples
- Reduce null values in tuples
- Disallow the possibility of generating spurious tuples.

Example

- Consider this relation schema and instance.

- Do you see any issues/problems with this schema?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

Example

- Observe the redundancy here in the attributes *dept_name*, *building* and *budget*.

- In particular, think about the methodology if we want to update the *building* of a department and/or the *budget* of a department?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

Example

- Now consider the situation when we want to add a new department to the university.

- How can a department be added when currently there are no employees in it?

- Will the table have **NULL** values?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

Example

- Can we decompose the relation *in_dep* into two relations in order to minimize and/or eliminate these anomalies?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

Normalization

- A general methodology to define a relational database free of anomalies is called Normalization.
  - First Normal Form, (1NF)
  - Second Normal Form (2NF) (includes 1NF)
  - Third Normal Form, (3NF) (includes 2NF + 1NF)

First Normal Form (1NF)

- First normal form (1NF) states that the domain of an attribute must:
  - include only atomic (indivisible) values, and
  - that the value of any attribute in a tuple must be a single value from the domain of that attribute.

- Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple.

- The only attribute values permitted by 1NF are single atomic (or indivisible) values.

First Normal Form (1NF): Example

- Consider the following relation (not in 1NF):

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

- The following relation is normalized to 1NF:

First Normal Form (1NF): Example

- Consider the following relation (not in 1NF):

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

- The following relation is normalized to 1NF:

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Functional Dependency: Introduction

- Functional dependency from set of attributes **X** to set of attributes **Y (X->Y)**
  - Requires that the value for a certain set of attributes X determines uniquely the value for another set of attributes Y.
  - Whenever two tuples have the same value for X, they must have the same value for Y

$X \rightarrow Y$

X "uniquely determines" Y

Y is "functionally dependent" on X

$X \longrightarrow Y$

{dept-name} {building, budget}

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Full/Partial Functional Dependency

- Functional dependency from set of attributes **X** to set of attributes **Y (X->Y)**
  - Requires that the value for a certain set of attributes X determines uniquely the value for another set of attributes Y.
  - Whenever two tuples have the same value for X, they must have the same value for Y

- 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 anymore.

{building, room_number} → {capacity}

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Full/Partial Functional Dependency

- Functional dependency from set of attributes **X** to set of attributes **Y (X->Y)**
  - Requires that the value for a certain set of attributes X determines uniquely the value for another set of attributes Y.
  - Whenever two tuples have the same value for X, they must have the same value for Y

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

Nonprime Attributes

Non-Key

- An attribute of relation schema R is called a prime attribute of R if it is a member of some candidate key of R.
- An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key.

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

Second Normal Form (2NF)

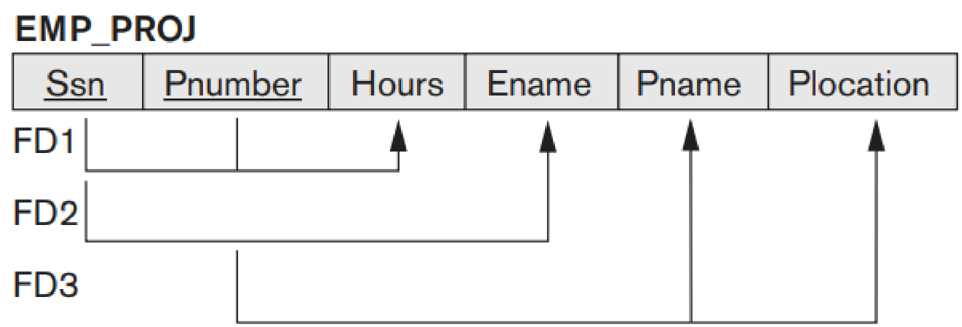
- A relation schema R is in 2NF if
  - (a) it is in 1NF, and
  - (b) every nonprime attribute A in R is fully functionally dependent on the primary key of R.
- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part (proper subset) of the primary key.
- If the primary key contains a single attribute, the test need not be applied at all.

{SSN, Pnumber} → {Pname, Plocation}

{SSN, Pnumber} → {Ename}

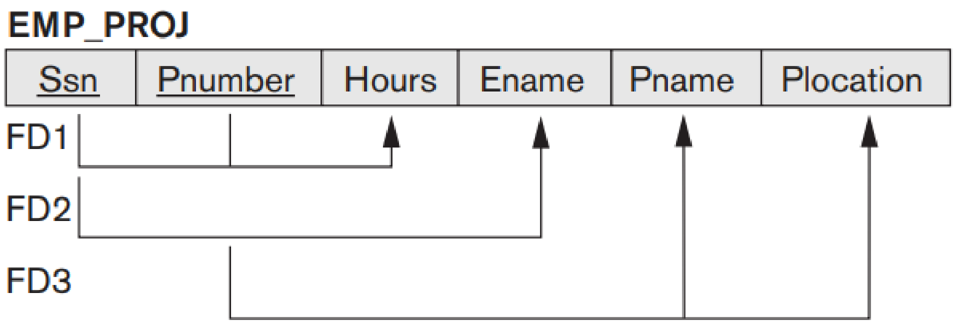
{SSN} → {Ename}

{Pnumber} → {Pname, Plocation}



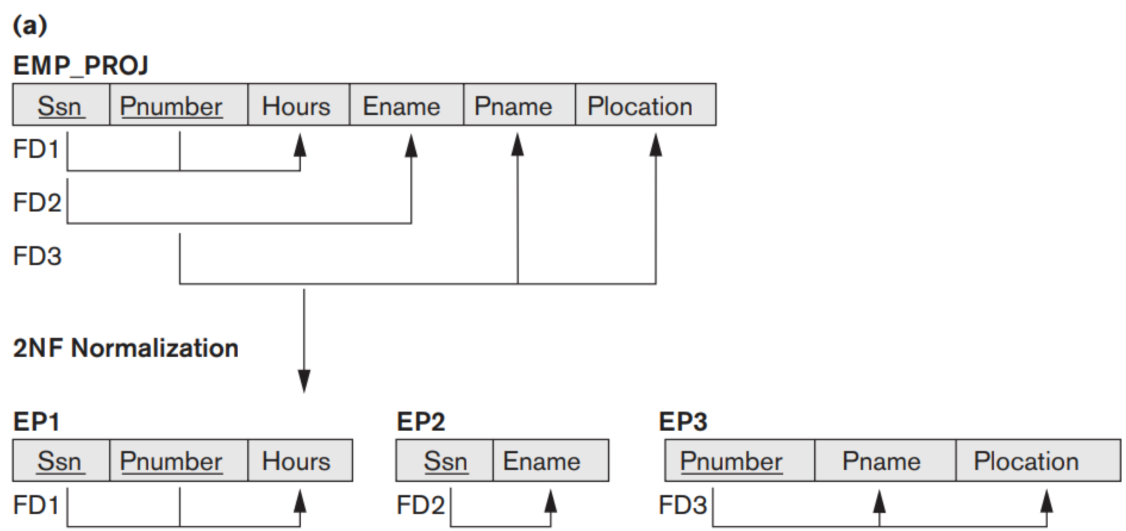
Second Normal Form (2NF)

- A relation schema R is in 2NF if
  - (a) it is in 1NF, and
  - (b) every nonprime attribute A in R is fully functionally dependent on the primary key of R.
- The nonprime attribute *Ename* violates 2NF because of FD2, as do the nonprime attributes *Pname* and *Plocation* because of FD3.



Second Normal Form (2NF): Normalization to 2NF

- A relation schema R is in 2NF if
  - (a) it is in 1NF, and
  - (b) every nonprime attribute A in R is fully functionally dependent on the primary key of R.
- If a relation schema is not in 2NF, it can be second normalized or 2NF normalized into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.



Functional Dependency: Introduction

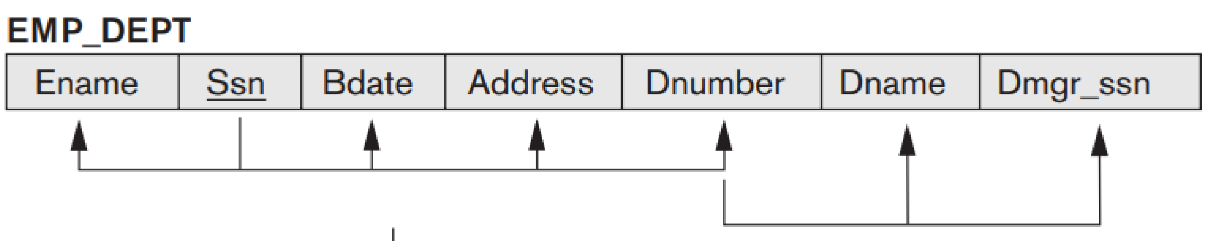
- Functional dependency from set of attributes **X** to set of attributes **Y (X->Y)**
  - Requires that the value for a certain set of attributes X determines uniquely the value for another set of attributes Y.
  - Whenever two tuples have the same value for X, they must have the same value for Y

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

Transitive Functional Dependency

- A functional dependency X → Y in a relation schema R is a **transitive dependency**
  - if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and
  - both X → Z and Z → Y hold.

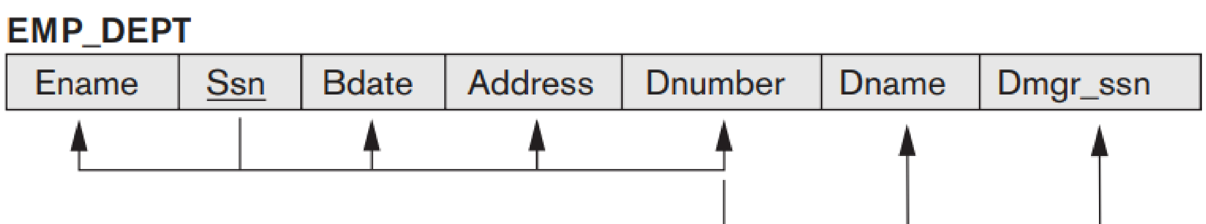


– $\overset{X}{\{SSN\}} \longrightarrow \overset{Y}{\{Dname\}}$

$\overset{X}{\{SSN\}} \longrightarrow \overset{Z}{\{Dnumber\}}$ $\overset{Z}{\{Dnumber\}} \longrightarrow \overset{Y}{\{Dname\}}$

Third Normal Form (3NF)

- A functional dependency X → Y in a relation schema R is a **transitive dependency**
  - if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and
  - both X → Z and Z → Y hold.
- A relation schema R is in 3NF if
  - (a) it is in 2NF, and
  - (b) no nonprime attribute of R is transitively dependent on the primary key.

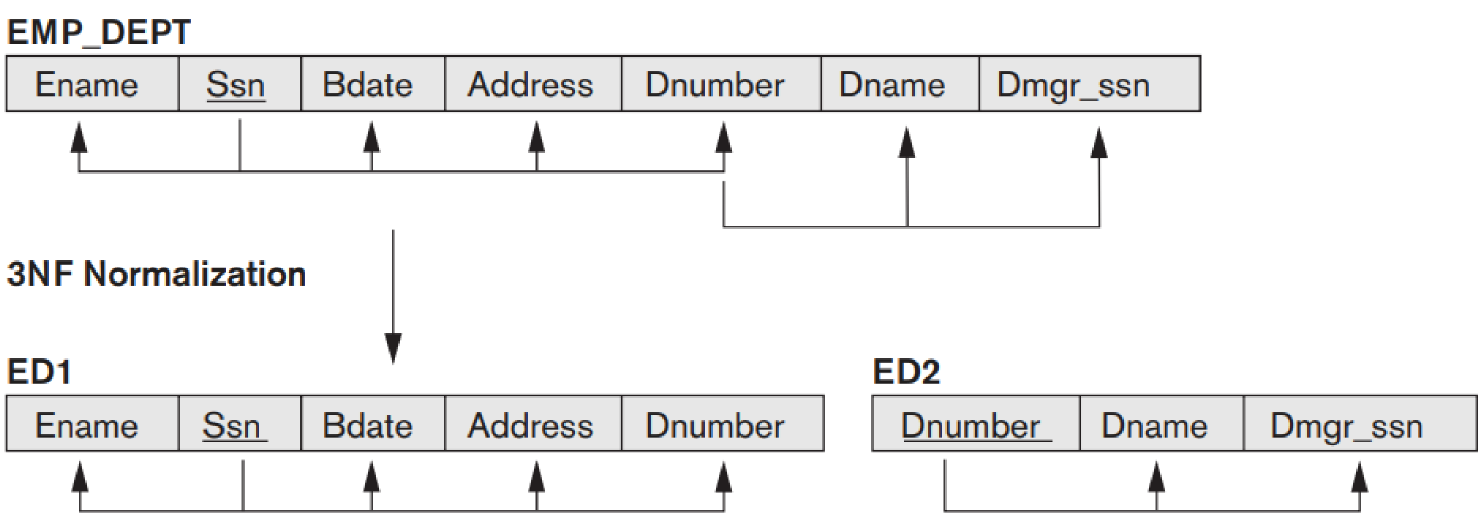


- $\overset{X}{\{SSN\}} \longrightarrow \overset{Y}{\{Dname\}}$

$\overset{X}{\{SSN\}} \longrightarrow \overset{Z}{\{Dnumber\}}$ $\quad$ $\overset{Z}{\{Dnumber\}} \longrightarrow \overset{Y}{\{Dname\}}$

Third Normal Form (3NF)

- A functional dependency X → Y in a relation schema R is a **transitive dependency**
  - if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and
  - both X → Z and Z → Y hold.
- A relation schema R is in 3NF if
  - (a) it is in 2NF, and
  - (b) no nonprime attribute of R is transitively dependent on the primary key.



Normalization: Intuition

- Intuitively, we can see that the following type of FDs are *problematic*
  - 2NF: Functional dependency in which the left-hand side is a proper subset of the primary key and the right-hand side is a set of non-key attributes
  - 3NF: Functional dependency in which the left-hand side is a set of non-key attributes and the right-hand side is also a set of non-key attributes

- 2NF and 3NF normalizations remove these problematic FDs by decomposing the original relations into new relations.

Normalization: Summary

Table 14.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Normalization: Example

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000