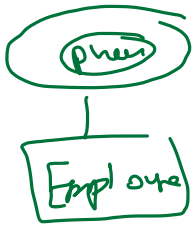
FUNCTIONAL DEPENDENCIES AND NORMALIZATION

CS 340: INTRODUCTION TO DATABASE SYSTEMS

Adapted from Dr. Omar Alomeir
2023

Course instructor:
Dr. Maram Alajlan.

① 1NF (Repeating Groups)



→ كل ما تطلعه من الجدول

Employee (ID, name)

Emp.phone (ID, phone)

2NF: Partial dependency

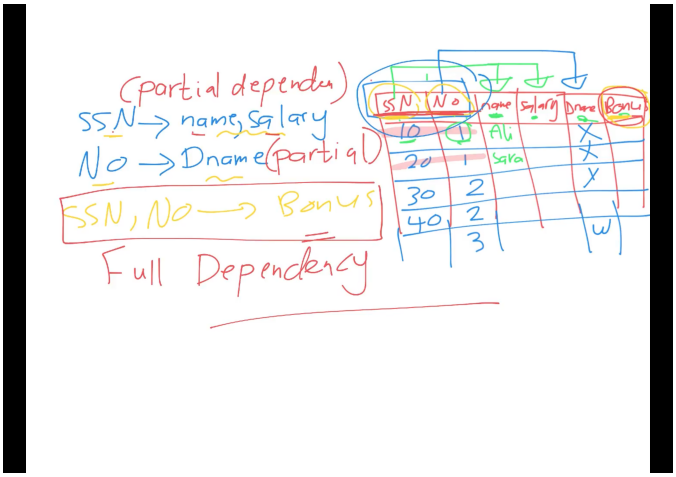
الاعتماد الجزئي

Functional Dependency

Employ non.p.K

ID	name	Sale	D
10	X		
20	X		
30	2		

لا استخدم اني استخدم الاسم مرتين أعوض بس الـ p.K اقدر اوصل



3NF transitive depenency

non.p.K

كلاهما non.PK ولكن واحد يعتمد على الثاني

LEARNING GOALS

CLO 4: Apply normalization forms (1st, 2nd, 3rd, BCNF forms) and functional dependencies to optimize relational model.

Chapter objectives:

- Debate the pros and cons of redundancy in a database.
- Provide examples of update, insertion, and deletion anomalies.
- Learn all the different types of Normal Forms (1st, 2nd, 3rd, and BCNF)
- Show that a table is/isn't in 3NF or BCNF.
- Decompose a table into a set of tables that are in 3NF, or BCNF

CHAPTER OUTLINE

1. Informal Design Guidelines for Relational Databases
   a. Semantics of the Relation Attributes
   b. Redundant Information in Tuples and Update Anomalies
   c. Null Values in Tuples
   d. Spurious Tuples
2. Functional Dependencies (FDs)
   a. Definition of Functional Dependency
3. Normal Forms Based on Primary Keys
   a. Normalization of Relations
   b. Practical Use of Normal Forms
   c. Definitions of Keys and Attributes Participating in Keys
   d. First Normal Form
   e. Second Normal Form
   f. Third Normal Form
4. General Normal Form Definitions for 2NF and 3NF (For Multiple Candidate Keys)
5. BCNF (Boyce-Codd Normal Form)

INTRODUCTION

Relational Schema consists of a number of attributes and Relational database Schema consists of a number of relational schemas.

Relational database design: is the grouping of attributes to form “good” relation schemas by using **common sense** of the database designer or by mapping a database schema design from a conceptual data model (ER or EER) which leads to a natural or logical grouping of the attributes into relations.

The designer needs some formal measure of appropriateness or goodness to measure the quality of design i.e. To measure formally why one set of grouping of relation schema is better than another.

INTRODUCTION

There are two levels where goodness of relation schemas can be discussed:

The logical or conceptual " user view" level: How users interpret the relation schemas and the meaning of their attributes of both the base relations and views (virtual relations).

The storage or implementation "base relation" level: How the tuples in a base relation are stored (as files) and updated. This level applies mainly to schemas of base relations, which will be physically stored as files.

Design is concerned mainly with base relations

INTRODUCTION

What are the criteria for "good" base relations?

The implicit goals of the design activity are

1. **Information preservation:** in terms of maintaining all concepts, including attributes types, entity types, and relationship types as well as generalization/ specialization relationships that are captured in the conceptual design.
2. **Minimum Redundancy:** implies minimizing redundant storage of the same information and reducing the need for multiple updates to maintain consistency across multiple copies of the same information in response to real world events.

Is redundancy always bad?

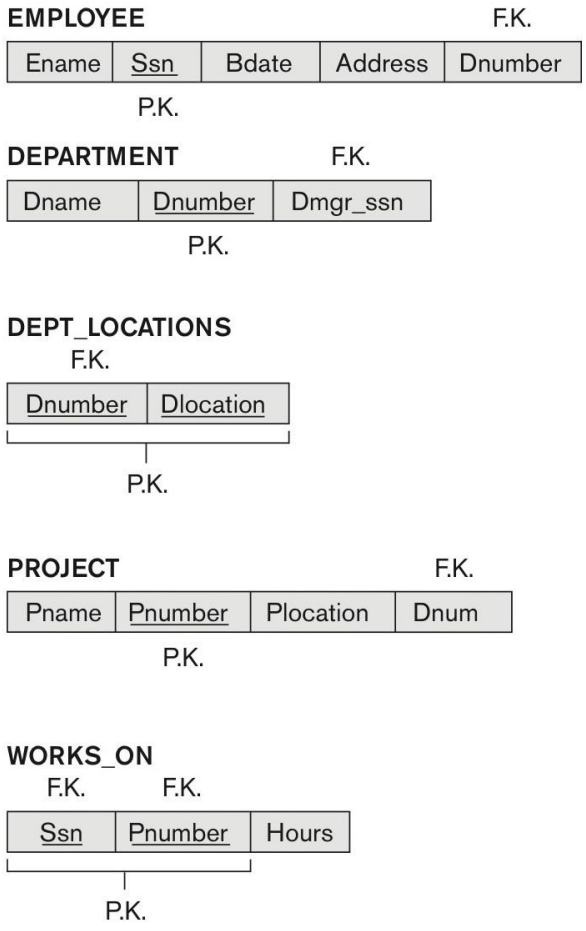
INFORMAL DESIGN GUIDELINES FOR RELATIONAL DATABASES

- We first discuss **informal guidelines** for good relational design
- Then we discuss **formal concepts** of functional dependencies and normal forms
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Normal Form)
  - BCNF (Boyce-Codd Normal Form)

GUIDELINE FOR REDUNDANT INFORMATION IN TUPLES AND UPDATE ANOMALIES

- **GUIDELINE 1:** Semantics of the Relational Attributes must be clear
- Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).
  - Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
  - Only foreign keys should be used to refer to other entities
  - Entity and relationship attributes should be kept apart as much as possible.
- **Bottom Line:** Design a schema that can be explained easily, relation by relation. The semantics of attributes should be easy to interpret.

COMPANY RELATIONAL DATABASE SCHEMA



COMPANY RELATIONAL DATABASE INSTANCE

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Ssn	Pnumber	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

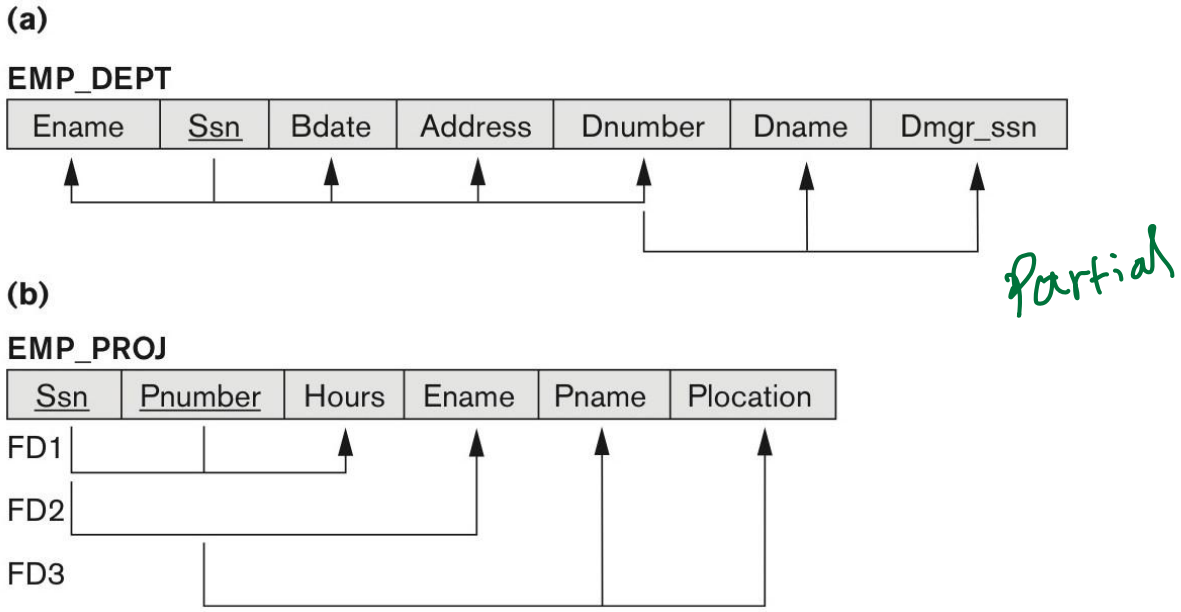
REDUNDANT INFORMATION IN TUPLES AND UPDATE ANOMALIES

Information if stored redundantly:

- Wastes storage
- Causes update anomalies:
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Anomalies are very simple concepts we will examine closely.

EXAMPLE: TWO RELATION SCHEMAS SUFFERING FROM UPDATE ANOMALIES



Redundancy

STATES FOR EMP_DEPT AND EMP_PROJ

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:

  EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:**

  Changing the name of project number P1 from “Billing” to “Customer-Accounting” means this update must be made for all 100 employees working on project P1.

EXAMPLE OF AN INSERT ANOMALY

- Consider the relation:

  EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

- **Insert Anomaly:**
  - Cannot insert a project unless an employee is assigned to it.

    Conversely
  - Cannot insert an employee unless he/she is assigned to a project.

EXAMPLE OF A DELETE ANOMALY

- Consider the relation:

  EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

- **Delete Anomaly:**
  - When a project is deleted, it will result in deleting all the employees who work on that project.
  - Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

GUIDELINE FOR REDUNDANT INFORMATION IN TUPLES AND UPDATE ANOMALIES

- GUIDELINE 2:
  - Design a schema that does not suffer from the insertion, deletion and update anomalies.
  - If there are any anomalies present, then note them so that applications can be made to take them into account.

1.3 NULL VALUES IN TUPLES

- GUIDELINE 3:
  - Relations should be designed such that their tuples will have as few NULL values as possible
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - Attribute not applicable or invalid
  - Attribute value unknown (may exist)
  - Value known to exist, but unavailable

1.3 NULL VALUES IN TUPLES

- In some schema designs many attributes may be grouped together into a “fat” relation.
- If many of the attributes do not apply to all tuples in the relation, many null values will appear in those tuples.

Problems:

- **Space is wasted** at the storage level and may lead to the problems with understanding the meaning of the attributes and with specifying JOIN operations at the logical level.
- The same word NULL is used for multiple interpretations (does not apply, unknown, known but absent) which compromises the different meanings they may have.

1.4 GENERATION OF SPURIOUS TUPLES – AVOID AT ANY COST

- Bad designs for a relational database may result in erroneous (wrong) results for certain JOIN operations.
- GUIDELINE 4:

Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys and foreign key in a way that guarantees that no spurious (false) tuples are generated.

JOINS?

The JOIN operation, is used to combine related tuples from two relations into single “longer” tuples.

If we are not careful, we can generate invalid tuples.

In this example, joining two tables on **Plocation** results in invalid data for employee 123456789. Can you tell what’s wrong?

Joins should be limited to attributes that match (lossless joins).

We will discuss joins in more detail in the next chapter.

(b)

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.

SUMMARY OF INFORMAL DESIGN GUIDELINES

Redundancy wastes space and can result in many anomalies.

Anomalies cause redundant work to be done during insertion into and modification of a relation. They may also cause accidental loss of information during a deletion from a relation.

Waste of storage space due to NULLs and difficulty interpreting the meaning of NULL.

Generation of invalid and spurious data during joins on base relations with matched attributes.

FUNCTIONAL DEPENDENCIES

A formal tool for analysis of relational schemas that enable us to detect and describe some of the problems discussed earlier in precise form.

Functional dependencies (FDs) are a form of constraint, finding them is part of database design. Used in normalization later.

Informally, if two tuples agree on attributes A1, A2, .., An then they must also agree on attributes B1, B2, .., Bn.
We say: A1, A2, .., An → B1, B2, .., Bn

FUNCTIONAL DEPENDENCIES

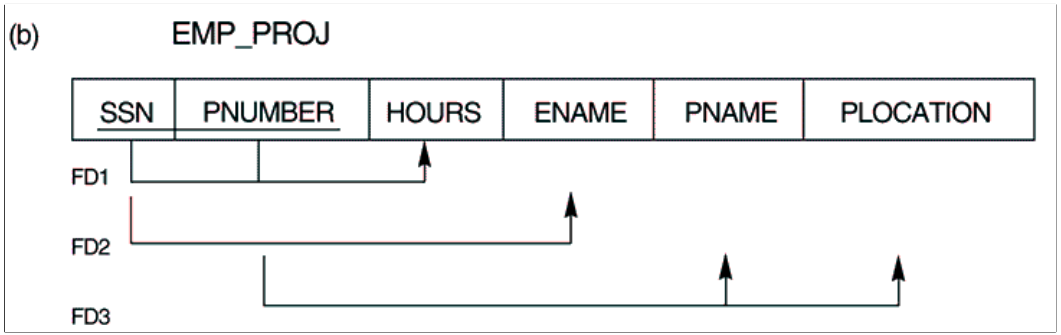
- Functional dependencies (FDs)
  - Are used to specify *formal measures* of the "goodness" of relational designs
  - With keys, they are used to define **normal forms** for relations
  - Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes
- A functional dependency is a property of the relation schema R, not of a particular legal relation state r of R.

A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

DEFINING FUNCTIONAL DEPENDENCIES

- X → Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
  - For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then they must have also* t1[Y]=t2[Y]
- X → Y in R specifies a *constraint* on all relation instances r(R)
- Written as X → Y; can be displayed graphically denoted by the arrow.
- FDs are derived from the real-world constraints on the attributes

FUNCTIONAL DEPENDENCIES



{SSN, PNUMBER} → HOURS

SSN → ENAME

PNUMBER → {PNAME, PLOCATION}

Each FD is displayed as a horizontal line. The left hand side attributes of the FD are connected by vertical lines to the line representing the FD, while the right hand side attributes are connected by arrows pointing towards the attributes.

EXAMPLES OF FD CONSTRAINTS

- Social security number determines employee name
  - SSN → ENAME
- Project number determines project name and location
  - PNUMBER → {PNAME, PLOCATION}
- Employee ssn and project number determines the hours per week that the employee works on the project
  - {SSN, PNUMBER} → HOURS

EXAMPLE BEFORE FORMAL DEFINITIONS

EmpID	Name	Phone	Position
E0045	Ahmad	1234	Clerk
E1847	Ali	9876	Salesrep
E1111	Saleh	9876	Salesrep
E9999	Omar	1234	Lawyer

EmpID → Name, Phone, Position

Position → Phone

but Phone ↛ Position

QUICK EXAMPLE

A	B	C	D
1	2	3	4
2	3	4	6
6	7	8	9
1	3	4	5

What FDs cannot be true, given the table?

a. B → C

b. B → D

c. D → B

d. All of the above can be true

e. None of the above can be true

DEFINING FUNCTIONAL DEPENDENCIES

non p.k non p.k

Student_ID	**Student_Name**	**Advisor_ID**	**Advisor_Name**
1234	Nouf	101	Dr. X
1256	Reem	106	Dr. Y
1890	Alaa	106	Dr. Y
1456	Nawal	101	Dr. X

transitive

Student_ID	**Student_Name**	**Advisor_ID**	**Advisor_Name**

Advisor_Id ? Advisor_Name

DEFINING FUNCTIONAL DEPENDENCIES

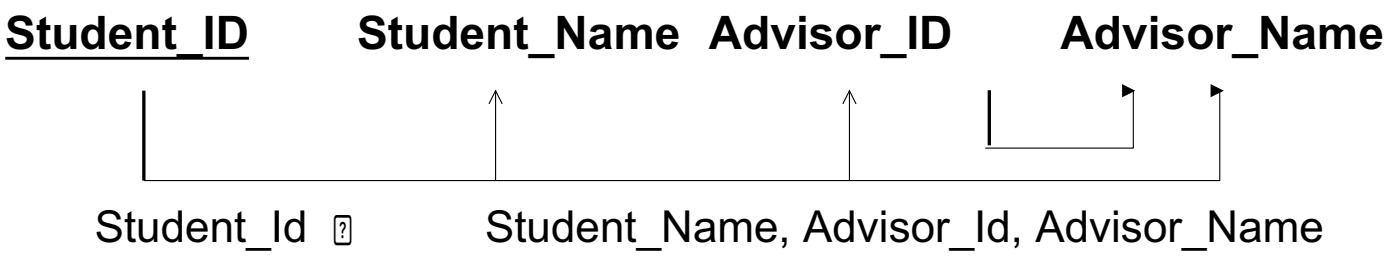
If a constraint on *R* states that there cannot be more than one tuple with a given *X*-value in any relation instance *r*(*R*)—that is, *X* is a **candidate key** of *R*—this implies that $X \rightarrow Y$ for any subset of attributes *Y* of *R* (because the key constraint implies that no two tuples in any legal state *r*(*R*) will have the same value of *X*).

If *X* is a candidate key of *R*, then $X \rightarrow R$.

since we never have two distinct tuples with t1[X]=t2[X].

DEFINING FUNCTIONAL DEPENDENCIES

Student_ID	**Student_Name**	**Advisor_ID**	**Advisor_Name**
1234	Nouf	101	Dr. X
1256	Reem	106	Dr. Y
1890	Alaa	106	Dr. Y
1456	Nawal	101	Dr. X



SOME FD PROPERTIES

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every* relation instance r(R).
- If $X \rightarrow Y$ in R, this does not say whether or not $Y \rightarrow X$ in R.
- In $X \rightarrow Y$, *we refer to X as t*he left-hand-side attributes of the FD and Y as the right-hand-side attributes.

DEFINING FDS FROM INSTANCES

- Note that in order to define the FDs, we need to understand the meaning of the attributes involved and the relationship between them.
- An FD is a property of the attributes in the schema R
- Given the instance (population) of a relation, all we can conclude is that an FD *may exist* between certain attributes.
- What we can definitely conclude is – that certain FDs *do not exist* because there are tuples that show a violation of those dependencies.

ACTIVITY: RULING OUT FDS

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Note that given the state of the TEACH relation, we can say that the FD: Text → Course may exist.
However, the FDs Teacher → Course, Teacher → Text and
Course → Text are ruled out.
Is this good design?

ACTIVITY 2: WHAT FDS MAY EXIST?

- A relation *R*(A, B, C, D) with its extension.
- Which FDs *may exist* in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

{A, B} → C

A → *B*? No
A → *C*? No
A → *D*? No
B → *A*? No
B → *C*? Yes
B → *D*? No
C → *A*? No
C → *B*? Yes
C → *D*? No
D → *A*? No
D → *B*? No
D → *C*? No

{*A*, *B*} → *C*? Yes
{*A*, *B*} → *D*? Yes
{*C*, *D*} → *B*? Yes
{*B*, *C*} → *D*? No

The following FDs *may hold* because the four tuples in the current extension have no violation of these constraints: *B* → *C*; *C* → *B*; {*A*, *B*} → *C*; {*A*, *B*} → *D*; and {*C*, *D*} → *B*. However, the following *do not* hold because we already have violations of them in the given extension:
A → *B* (tuples 1 and 2 violate this constraint);
B → *A* (tuples 2 and 3 violate this constraint);
D → *C* (tuples 3 and 4 violate it).

1, 2 → A → B ✗
2, 3 → B → A ✗

ACTIVITY 3: WHICH OF THESE FDS MAY EXIST?

A	B	C	TUPLE#
10	b1	c1	1
10	b2	c2	2
11	b4	c1	3
12	b3	c4	4
13	b1	c1	5
14	b3	c4	6

i. $A \rightarrow B$,

ii. $B \rightarrow C$,

iii. $C \rightarrow B$,

iv. $B \rightarrow A$,

v. $C \rightarrow A$

NORMAL FORMS BASED ON PRIMARY KEYS

3.1 Normalization of Relations
3.2 Practical Use of Normal Forms
3.3 Definitions of Keys and Attributes Participating in Keys
3.4 First Normal Form
3.5 Second Normal Form
3.6 Third Normal Form

3.1 NORMALIZATION OF RELATIONS

- **Normalization:**

  The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form:**

  Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

3.2 PRACTICAL USE OF NORMAL FORMS

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are *hard to understand* or to *detect*
- The database designers *need not* normalize to the highest possible normal form
  - (usually up to 3NF and BCNF. 4NF rarely used in practice.)
- **Denormalization**:
  - The process of storing higher normal form relations as a

3.3 DEFINITIONS OF KEYS AND ATTRIBUTES PARTICIPATING IN KEYS (1)

- A **superkey** of a relation schema
  R = {A1, A2, ..., An}
  is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]

- A **key** K is a **superkey** with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more. Remember that **Key** is "minimal" superkey

- Consider the following candidate keys:
  - ID Both
  - ID, First_name, Last_name Super Key
  - First_name, Last_name Both
  - ID, Order_date Super Key
- Are the key or superkey?

  Remember that **Key** is "minimal" superkey

ID	First_name	Last_name	Order_date	item
1	Adel	Ahmed	3/5/2024	PC
2	Reem	Saleh	3/5/2024	printer
3	Asma	Ahmed	4/5/2024	tablet
4	Reem	Naser	5/5/2024	PC

DEFINITIONS OF KEYS AND ATTRIBUTES PARTICIPATING IN KEYS (2)

- If a relation schema has more than one key, each is called a **candidate** key.
  - One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.
- A **Prime attribute** must be a member of *some* candidate key
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

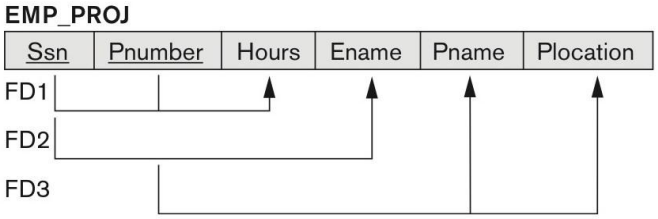
TYPES OF FUNCTIONAL DEPENDENCIES

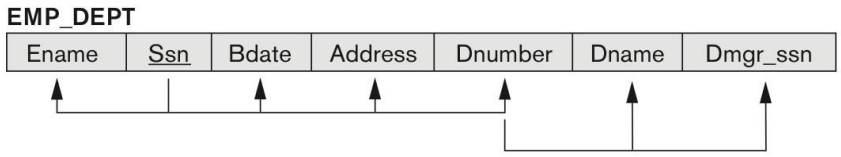
Full Dependency

Partial Dependency

Transitive Dependency

TYPES OF FUNCTIONAL DEPENDENCIES





Full Dependency *{SSN, PNUMBER}* ? *HOURS*

Partial Dependency *SSN* ? *ENAME*

Partial Dependency *PNUMBER* ? *{PNAME, PLOCATION}*

Transitive Dependency *DNUMBER* ? *DNAME, DMGR_SSN where DNUMBER is determined by the SSN i.e. SSN*? *DNUMBER thus Ssn → Dmgr_ssn is transitive*

3.4 FIRST NORMAL FORM

- Does not allow:
  - composite attributes
  - multivalued attributes
  - and their combinations which called **nested relations** or **repeated groups**.
- Considered to be part of the definition of a relation
- Most RDBMSs allow only those relations to be defined that are in First Normal Form

EXAMPLE: NORMALIZATIO N INTO 1NF

A relation with a design that includes multivalued attributes

An instance of the relation. Note that the key Dname → *Dlocations does not hold when we have multiple values.*

A 1NF relation with redundancy

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

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

(c)

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

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

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

A 1NF relation without redundancy

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

NORMALIZATION INTO 1NF

To achieve first normal form for multivalued attributes you can either:

1. Remove the attributes that violate 1NF and place them into a separate relation along with the primary key. The primary key of the new relation will be the combination of them.
2. Expand the key so that there will be a separate tuple in the original relation for each vale of the multivalued attributes. This solution has the disadvantage of introducing redundancy in the relation.

NORMALIZING NESTED RELATIONS INTO 1NF

A nested relation design

An instance of the relation

1NF form of the design: An employee table, and a project table

(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

NORMALIZATION INTO 1NF

To achieve first normal form for nested relations you should remove the nested relation attributes into a new relation along with the primary key. The primary key of the new relation will combine the primary key of the original relation with a candidate key of the nested relation.

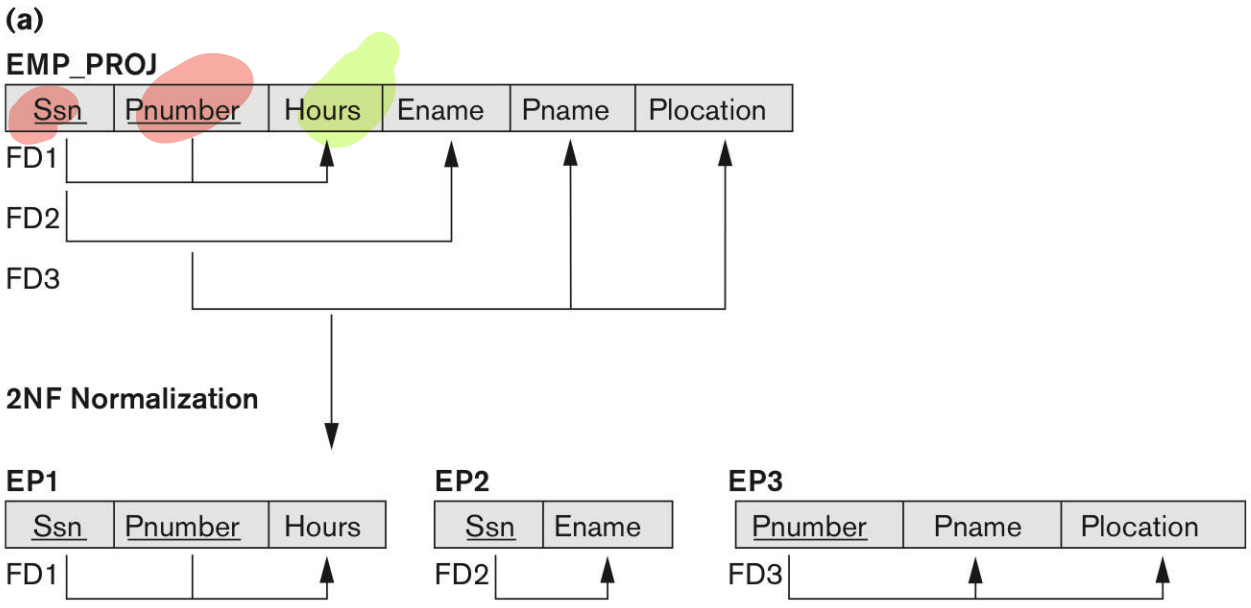
3.5 SECOND NORMAL FORM (1)

- Uses the concepts of **FDs, primary key**
- Definitions
  - **Prime attribute:** An attribute that is member of the primary key K
  - **Full functional dependency:** a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more
- Examples:
  - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold
  - {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) since SSN -> ENAME also holds

SECOND NORMAL FORM (2)

- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key.
- R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”

EXAMPLE: NORMALIZING INTO 2NF



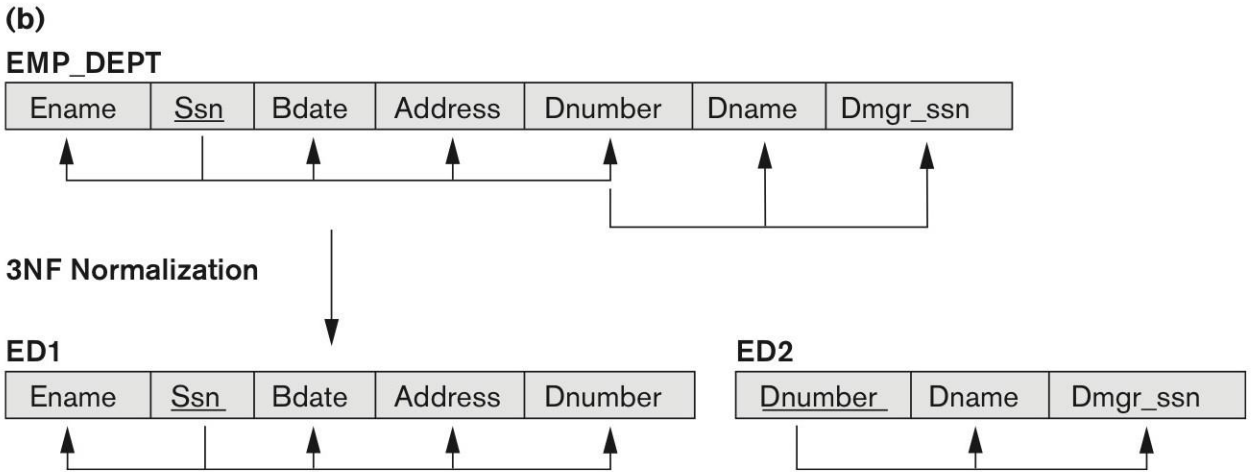
3.6 THIRD NORMAL FORM (1)

- Definition:
  - **Transitive functional dependency:** a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$
- Examples:
  - SSN → DMGRSSN is a **transitive** FD
    - Since SSN → DNUMBER and DNUMBER → DMGRSSN hold
  - SSN → ENAME is **non-transitive**
    - Since there is no set of attributes X where SSN → X and X → ENAME

THIRD NORMAL FORM (2)

- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key
- R can be decomposed into 3NF relations via the process of 3NF normalization
- NOTE:
  - In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key.
  - When Y is a candidate key, there is no problem with the transitive dependency
  - E.g., Consider EMP (SSN, Emp#, Salary).
    - Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

EXAMPLE: NORMALIZING INTO 3NF



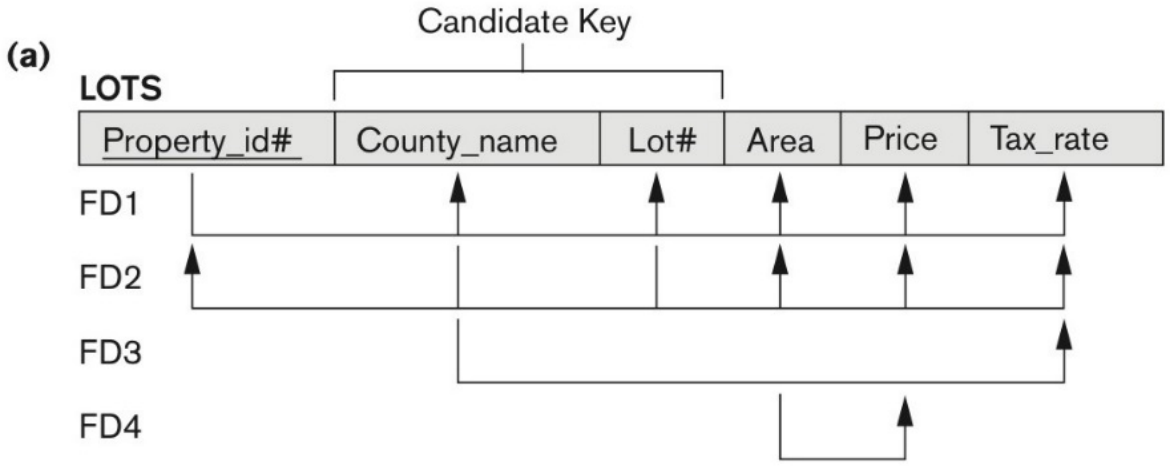
NORMAL FORMS DEFINED INFORMALLY

- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

4 GENERAL NORMAL FORM DEFINITIONS (FOR MULTIPLE KEYS) (1)

- The previous definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- Any attribute involved in a candidate key is a *prime attribute*
- All other attributes are called *non-prime attributes.*

RELATIOAN SCHEMA LOTS



GENERAL DEFINITIONS OF SECOND AND THIRD NORMAL FORMS

Consider the relation schema LOTS shown the previous slide, which describes parcels of land for sale in various counties of a state.

Suppose that there are two candidate keys : property_id# and (County_name, Lot#); i.e. lot numbers are unique only within each county but Property_id# numbers are unique across counties for the entire state.

FD1 and FD2 dependencies hold true based on the above two candidate keys.

FD3: County_name → Tax_rate

FD4 : Area → Price

FD3 indicates that the tax rate is fixed for a given county (i.e. does not vary lot by lot)

FD4 indicates that the price of a lot is determined by its area regardless of which county it is in.

4.1 GENERAL DEFINITION OF 2NF (FOR MULTIPLE CANDIDATE KEYS)

- A relation schema R is in second normal form (2NF) if every nonprime attribute A in R is not partially dependent on **any key** of R.
- In Figure 14.12 the FD

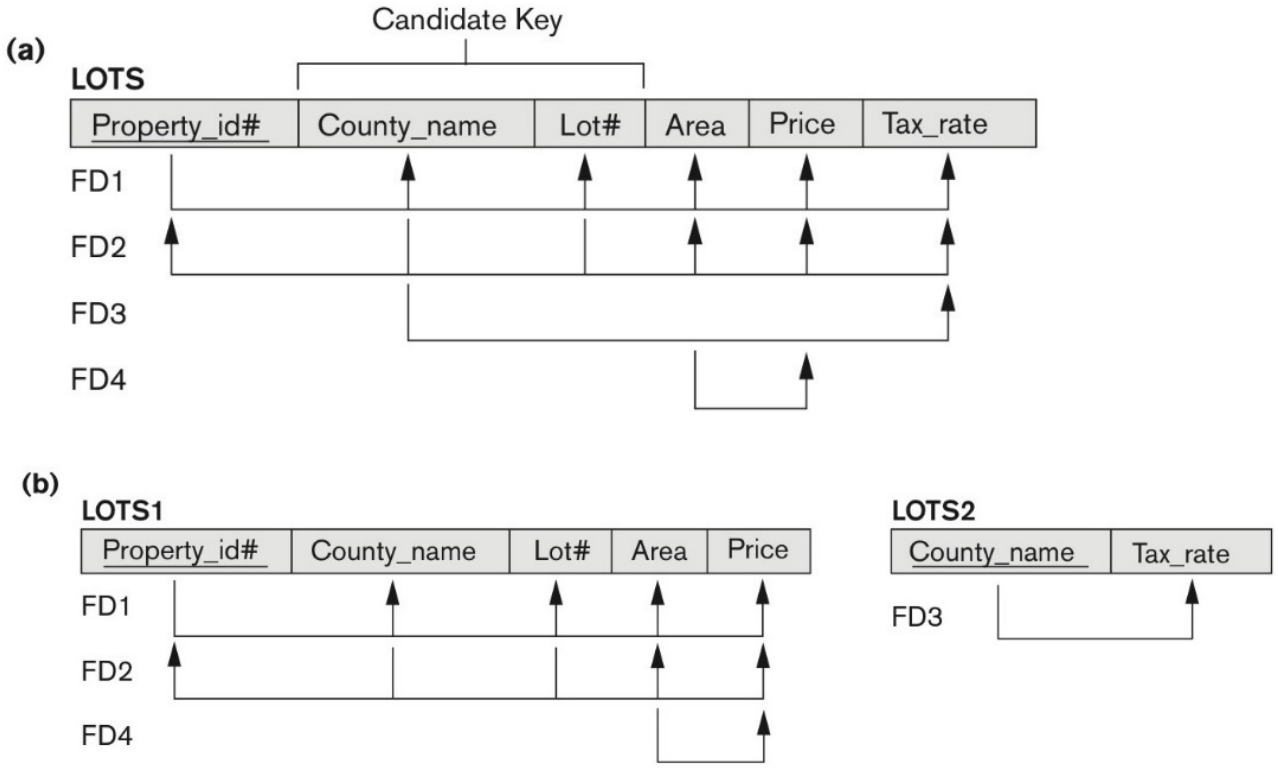
  County_name → **Tax_rate** violates 2NF.

So second normalization converts LOTS into

LOTS1 (Property_id#, County_name, Lot#, Area, Price)

LOTS2 (County_name, Tax_rate)

NORMALIZATION INTO 2NF



4.2 GENERAL DEFINITION OF THIRD NORMAL FORM

- Definition:

  NOTE: **Superkey** of relation schema R - a set of attributes S of R that contains a key of R

  - A relation schema R is in **third normal form (3NF)** if whenever a FD

    X → A holds in R, then either:

    - X is a superkey of R, **or**
    - A is a prime attribute of R

Example: LOTS1 relation violates 3NF because Area → Price

Area is not a superkey in LOTS1.

4.3 INTERPRETING THE GENERAL DEFINITION OF THIRD NORMAL FORM

From the definition: A relation schema R is in **third normal form (3NF)** if whenever a FD X → A holds in R, then either:

a) X is a superkey of R, or

b) A is a prime attribute of R

- Condition (a) catches two types of violations :
  - A nonprime attribute determines another nonprime attribute. Here we typically have a transitive dependency that violates 3NF.
  - A proper subset of a key of R functionally determines a nonprime attribute. Here we have a partial dependency that violates 2NF.

Thus, condition (a) alone addresses the problematic dependencies that were causes

for second and third normalization as we discussed.

4.3 INTERPRETING THE GENERAL DEFINITION OF THIRD NORMAL FORM (2)

- **ALTERNATIVE DEFINITION of 3NF:** We can restate the definition as: A relation schema R is in **third normal form (3NF)** if every non-prime attribute in R meets both of these conditions:
  - It is fully functionally dependent on every key of R.
  - It is non-transitively dependent on every key of R.

  Note that stated this way, a relation in 3NF also meets the requirements for 2NF.
- The condition ((b) A is a prime attribute of R) from the last slide takes care of the dependencies that "slip through" 3NF but are "caught by" BCNF which we discuss next.

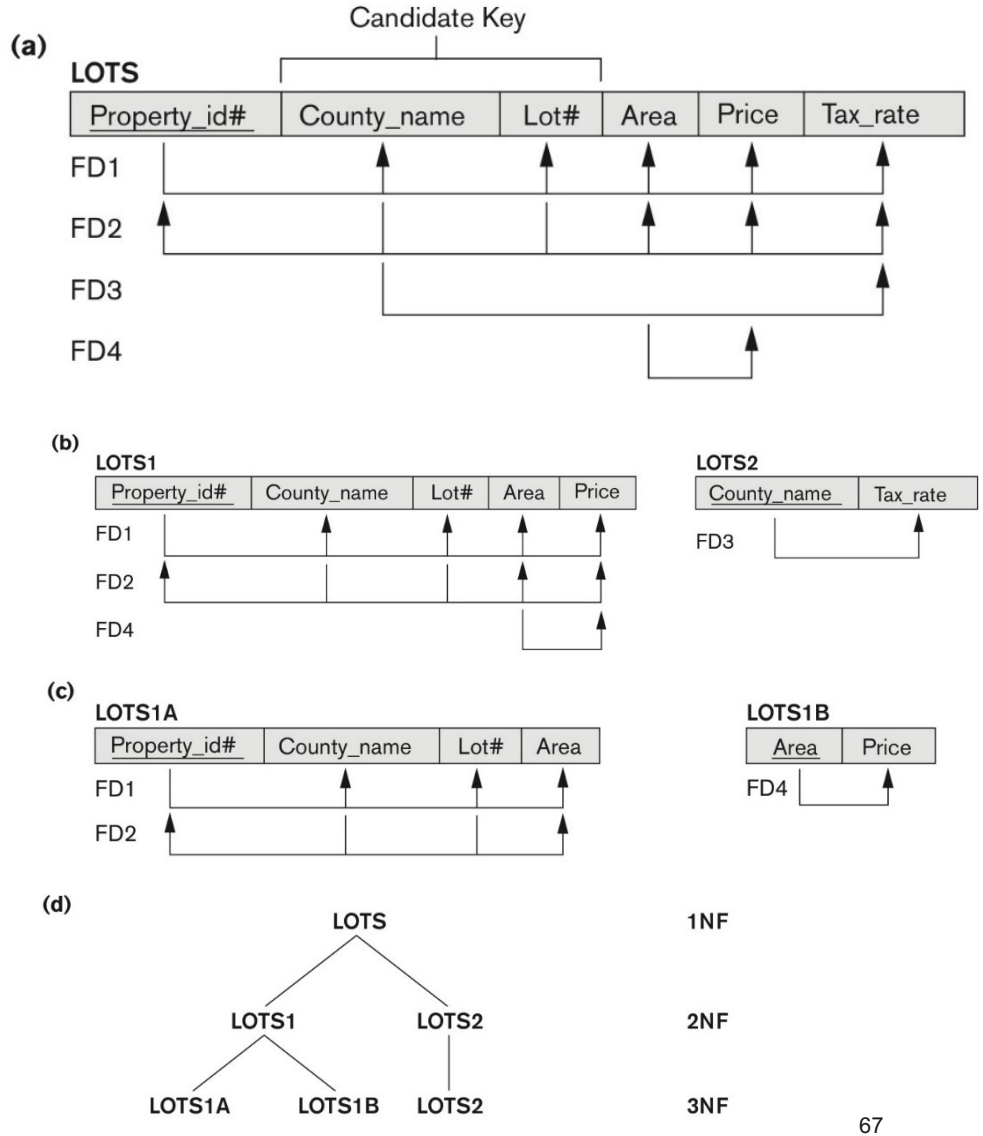
NORMALIZATION INTO 3NF

(a) The LOTS relation with its functional dependencies FD1 through FD4.

(b) Decomposing into the 2NF relations LOTS1 and LOTS2.

(c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B.

(d) Progressive normalization of LOTS into a 3NF design.



BOYCE-CODD NORMAL FORM (BCNF)

When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.

3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys i.e. composite candidate keys with at least one attribute in common.

BCNF is based on the concept of a determinant.

A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.

A relation is in BCNF, if and only if, every **determinant is a candidate key**.

5. BCNF (BOYCE-CODD NORMAL FORM)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X → A** holds in R, then **X is a superkey** of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- Hence BCNF is considered a stronger form of 3NF
- The goal is to have each relation in BCNF (or 3NF)

BOYCE-CODD NORMAL FORM (BCNF)

Suppose there are thousands of lots in the relation but the lots are from only two counties: **DeKalb** and **Fulton**.

Suppose also the lot sizes in **DelKab** County are only **0.5,0.6, 0.7, 0.8, 0.9 and 1.0** acres, whereas lot sizes in Fulton County are restricted to **1.1, 1.2, ….1.9, 2.0** acres.

In such as situation, the additional functional dependency

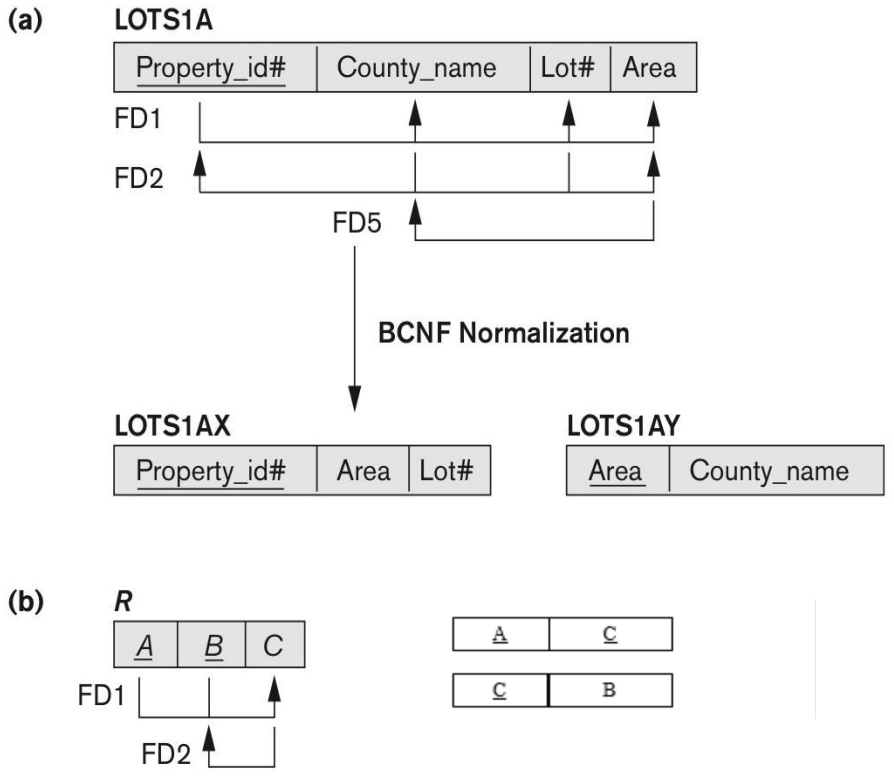
FD5: Area → County_name

By adding FD5 to LOTS1A, the relation is still in 3NF because County_name is a prime attribute.

EXAMPLE: BOYCE-CODD NORMAL FORM

BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition.

A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. C → B.



BOYCE-CODD NORMAL FORM (BCNF)

LOTS1A

PropertyID#	CountyName	Lot#	Area
P1000	Delkab	1	0.5
P1001	Fulton	1	1.2
P1002	DelKab	2	0.5
P1007	Fulton	10	1.4
P1089	DelKab	3	0.7
P1008	Fulton	2	1.2
P1245	Fulton	7	1.1
P2345	Delkab	4	0.7
P1235	Delkab	5	0.7
P6754	Fulton	5	1.1
P1245	Fulton	9	1.4

BOYCE-CODD NORMAL FORM (BCNF)

LOTS1AY

Area	County
0.5	Delkab
0.6	Delkab
0.7	Delkab
0.8	Delkab
0.9	Delkab
1.0	Fulton
1.1	Fulton
1.2	Fulton
1.3	Fulton
1.4	Fulton
1.5	Fulton
1.6	Fulton
1.7	Fulton
1.8	Fulton
1.9	Fulton
2.0	Fulton

LOTS1AX

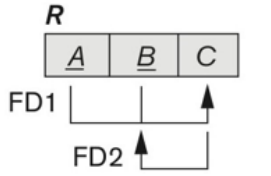
PropertyID#	Lot#	Area
P1000	1	0.5
P1001	1	1.2
P1002	2	0.5
P1007	10	1.4
P1089	3	0.7
P1008	2	1.2
P1245	7	1.1
P2345	4	0.7
P1235	5	0.7
P6754	5	1.1
P1245	9	1.4

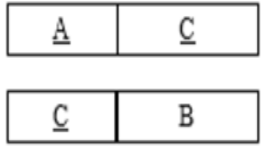
The area of a lot that determines the county, as specified by FD5, can be represented by 16 tuples in a separate relation ***LOTA1AY*(Area, County_name),** since there are only 16 possible Area values. This representation reduces the redundancy of repeating the same information in the thousands of LOTS1AX tuples.

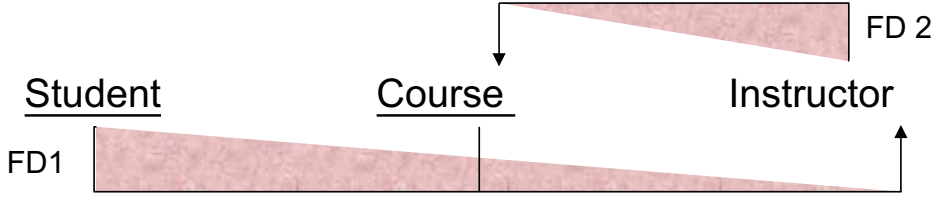
EXAMPLE: A RELATION TEACH THAT IS IN 3NF BUT NOT IN BCNF

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar







ACHIEVING THE BCNF BY DECOMPOSITION (1)

- Two FDs exist in the relation TEACH:
  - fd1: {student, course} -> instructor
  - fd2: instructor -> course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 14.13 (b).
  - So this relation is in 3NF *but not in* BCNF
- A relation **NOT** in BCNF should be decomposed to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations.

ACHIEVING THE BCNF BY DECOMPOSITION (1)

Decomposition of this relation schema into two schemas is not straightforward because it may be decomposed into one of the three following possible pairs:

1. D1: R1 (Student, Instructor) and R2(Student, Course)

2. D2: R1 (Course, Instructor) and R2(Course, Student)

3. D3: R1 (Instructor, Course) and R2(Instructor, Student)

All three decompositions *lose* the functional dependency FD1.

We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additive join property after decomposition.

We have to apply a non-additive test to the resulting relations.

We have to make sure that joining the two decomposed tables gives us the original table.

NON-ADDITIVE JOIN TEST

NJB (Non-additive Join Test for Binary Decompositions) A decomposition D = {R1, R2} of R has the lossless (nonadditive) join property with respect to a set of functional dependencies F on R if and only if either:

- The FD ((R1 ∩ R2) → (R1 − R2)) is in F^+, or
- The FD ((R1 ∩ R2) → (R2 − R1)) is in F^+

Briefly, the concept of F^+ refers to the cover of the set of functional dependencies and includes all f.d.'s implied by F. Here, we just need to make sure one fd holds:

	R1 ∩ R2	**R1 − R2**	**R2 − R1**
D1	Student	Instructor	Course
D2	Course	Instructor	Student
D3	Instructor	Course	Student

F+ = {Instructor →

Remember we lose FD2.

GENERAL METHOD FOR BCNF DECOMPOSITION

A relation R not in BCNF can be decomposed to meet the nonadditive join property by the following procedure. It decomposes R successively into a set of relations in BCNF:

Let R be the relation not in BCNF, let $X \subseteq R$, and let $X \rightarrow A$ be the FD that causes a violation of BCNF. R may be decomposed into two relations:

R1 = R – A

R2 = XA

If either R – A or XA. is not in BCNF, repeat the process.

We will not go any deeper into this methodology in this course.

BCNF OF TEACH RELATION

Instructor	Course
Mark	Database
Navathe	Database
Ammar	Operating Systems
Schulman	Theory
Ahamad	Operating Systems
Omiecinski	Database

Student	Instructor
Narayan	Mark
Smith	Navathe
Smith	Ammar
Smith	Schulman
Wallace	Ahamad
Wallace	Omiecinski

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

DETERMINING FD'S CLOSURE

A trivial FD determines what you already have, eg., AB -> B.

A key is a minimal set of attributes determining the rest of the attributes of a relation
A superkey is a set of attributes determining the rest of the attributes in the relation, but does NOT have to be minimal (e.g., the key above, or adding in either of City and Province)

Given a set of (explicit) functional dependencies, we can determine other FDs:

Remember this? X → Y , Y → Z implies X → Z

An FD *fd* is *implied by* a set of FDs *F* if *fd* holds whenever all FDs in *F* hold.

Closure of F: the set of all FDs implied by F. Denoted F^+

FUNCTIONAL DEPENDENCY INFERENCE RULES

We have a set of rules for determining FDs and finding closures.

Armstrong's Axioms (X, Y, Z are sets of attributes):

- *Reflexivity*: If $Y \subseteq X$, then $X \rightarrow Y$ e.g. $AB \rightarrow B$
- *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
- *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

These are *proven* and *complete* inference rules for FDs.

From these axioms, we also have:

- *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

PROOFS

Proof 1: Prove the *Reflexivity* property that when Y ⊆ X, then X → Y:

1. For any two tuples in a relation R such that t1[X] = t2[X]
2. Then t1[Y] = t2[Y] given that Y is a subset of X

Proof 2: Prove that when X → Y and X → Z, then X → YZ (Union rule) using the axioms.

1. Given: X → Y, X → Z
2. X → Y means X(X) → XY (augmentation)
3. Augment X → Z with Y: XY → ZY
4. X → XY → ZY means X → ZY (transitivity) end of proof

This shows that union rule can be proven using the axioms.

CLOSURE OF ATTRIBUTES

Definition: For each such set of attributes X, we determine the set X^+ of attributes that are functionally determined by X based on F; X^+ is called the closure of X under F.

A key contains a full set of attributes in its closure.

We use an algorithm to find the closure set of an attribute.

CLOSURE ALGORITHM

Algorithm: Determining X^+, the Closure of X under F

```
Input: A set F of FDs on a relation schema R, and a set of attributes
X, which isa subset of R.

X+ := X; //Set closure of X as all attributes of X

repeat
   old X+ := X+; // Set old X as current X
   for each functional dependency Y → Z in F do
      if X+ ⊇ Y then X+ := X+ ∪ Z; //Add attributes to the closure
                                   // using transitive and decomp rules
   until (X+ = old X+); //Stop when the set no longer changes
```

FINDING THE KEY

Given the relation: SupplierPart(sname, city, status, p#, pname, qty)

With the following functional dependencies:

1. fd1: sname → city
2. fd2: city → status
3. fd3: p# → pname
4. fd4: sname, p# → qty

Show that (sname, p#) is a key!

FINDING THE KEY

1. fd1: sname → city
2. fd2: city → status
3. fd3: p# → pname
4. fd4: sname, p# → qty

First step: Show that (sname, p#) is a super key.

1. sname, p# → sname, p# (trivial)
2. sname → city (fd1)
3. sname → status (transitive, fd2)
4. sname, p# → city, p# (augmentation 2)
5. sname, p# → status, p# (augmentation 3)
6. sname, p# → sname, status, p# (union 1 and 5)
7. sname, p# → sname, status, city, p# (union 4 and 6)
8. sname, p# → sname, status, city, p#, qty (union 7 and fd4)
9. sname, p# → sname, pname (augmentation fd3)
10. sname, p# → sname, status, city, p#, qty, pname (union 8 and 9)

FINDING THE KEY

1. fd1: sname → city
2. fd2: city → status
3. fd3: p# → pname
4. fd4: sname, p# → qty

Second step: Show that (sname, p#) is minimal. Which means showing sname isn't a key and p# isn't a key.

p# isn't on the right hand side of any FD, nothing determines p#.

sname → p# does not hold so sname is not a key.

The same goes for p#

So (sname, p#) is minimal and thus it is a key

A FULL EXAMPLE

You are tasked with improving the design of a database for a clinic. They really hate redundancy. They have the following relation:

Patient_Appointment

PatientNo	PatientName	AppointmentNo	Time	Doctor
1	John	0	09:00	Zorro
1	John	1	10:00	Zorro
1	John	2	15:00	Ather
2	Kerr	0	09:00	Killer
2	Kerr	1	14:00	Mary
3	Adam	1	10:00	Zorro
4	Robert	0	13:00	Killer
5	Zane	1	14:00	Zorro

At first look, thoug s (Insert, update, delete)?

What Normal form is this?

A FULL EXAMPLE (2)

The following describes how the clinic works:

This is a special dieting clinic where each patient has 4 appointments. On the first they take their weight, the second they measure fitness, the third are scheduled for any procedures, and on the fourth their mouth is stitched closed. Not all patients need all four appointments!

If the Patient Name begins with a letter before “P” they get a morning appointment, otherwise they get an afternoon appointment. Appointment 0 is either 09:00 or 13:00, Appointment 1 10:00 or 14:00, and Appointment 2 is either 11:00 or 15:00 and Appointment 4 is either 12:00 or 16:00.

From this make-believe scenario we can extract some FDs.

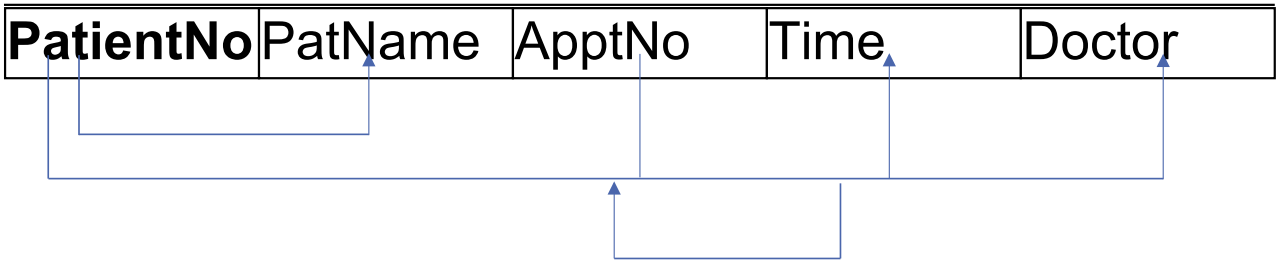
Patient_Appointment

PatientNo	PatName	ApptNo	Time	Doctor
1	John	0	09:00	Zorro
1	John	1	10:00	Zorro
1	John	2	15:00	Ather
2	Kerr	0	09:00	Killer
2	Kerr	1	14:00	Mary
3	Adam	1	10:00	Zorro
4	Robert	0	13:00	Killer
5	Zane	1	14:00	Zorro

FD1: PatNo → PatName
FD2: Patno, ApptNo → Time, Doctor
FD3: Time → AppNo

A FULL EXAMPLE (3)

Patient_Appointment



FD1: PatNo → PatName
FD2: Patno, ApptNo → Time, Doctor
FD3: Time → AppNo

A FULL EXAMPLE (4): KEYS

Finding the closure sets of left-hand side attributes:

1. PatNo+ ={PatNo, PatName}
2. (PatNo, AppNo)+ = {PatNo, AppNo, Time, Doctor} Since PatNo → PatName (transitive)

   {PatNo, PatName, AppNo, Time, Doctor}

   PatNo, AppNo is a candidate key because all attributes are in its closure set.

3. (PatNo, Time)+ = {PatNo, Time}

   Since PatNo → PatName and Time → AppNo

   (PatNo, Time)+ = {PatNo, PatName, Time, AppNo}

   Since (PatNo,AppNo) → Doctor

   (PatNo, Time)+ = {PatNo, PatName, Time, AppNo, Doctor}

   (PatNo, Time) is also a candidate key

A FULL EXAMPLE (5): 1NF AND 2NF

2 keys: {PatNo, AppNo}, and {PatNo, Time}

Is it in 1NF?

Any composite or multivalued attributes? No.

Is it in 2NF?

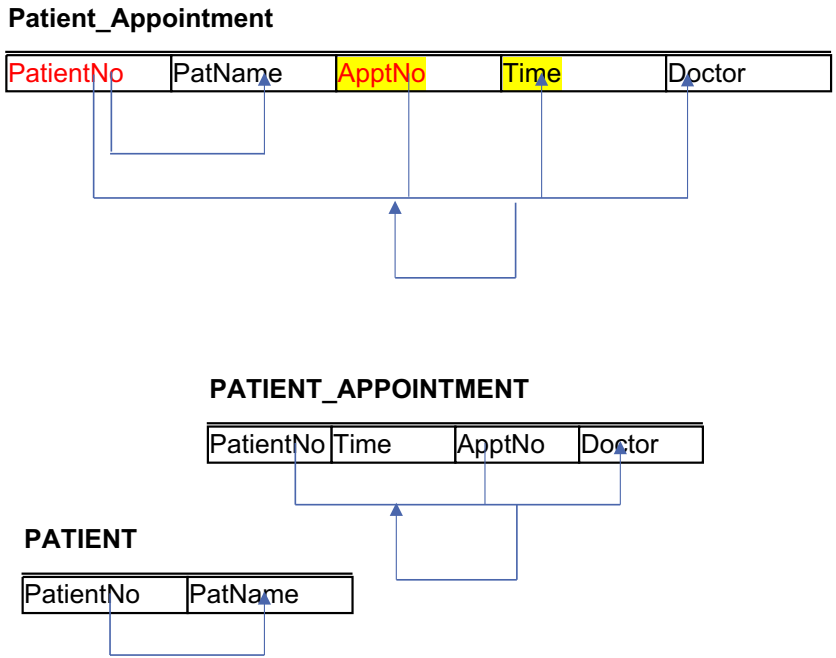
No. Partial dependencies: PatNo → PatName AND Time → ApptNo

However, ApptNo is a prime attribute, so it does not apply.

Decompose Patient_Appointment into two relations:

PATIENT_APPOINTMENT1
(Patno,Time,Doctor,ApptNo)

PATIENT(**Patno**,PatName)



A FULL EXAMPLE (5): 3NF AND BCNF

Is it in 3NF?

No transitive dependencies?

Every determinant should be a candidate key or the dependent attribute(s) should be a prime attribute. Again, Time is prime, no issues.

Is it in BCNF?

Every determinant is a key/superkey? No.

Time ? ApptNo and Time is not a key.

Decompose. Now in BCNF.