Lecture 5. Functional Dependencies and Normalization (Chapter 14)

alisa.lincke@lnu.se

Linnæus University

Connection to previous lectures

ER

Relational Model

Lecture 3 ER and EER Conceptual models
- High –level data abstraction using Entity Relation (ER)

Lecture 4 Relation Model
- Medium-level data abstraction using Relational Models
- From ER to Relational Model

Lecture 5 SQL
- SQL standard language to create and manipulate the database

Lecture 6
- Functional Dependencies
- Normalization: 1NF, 2NF,3 NF
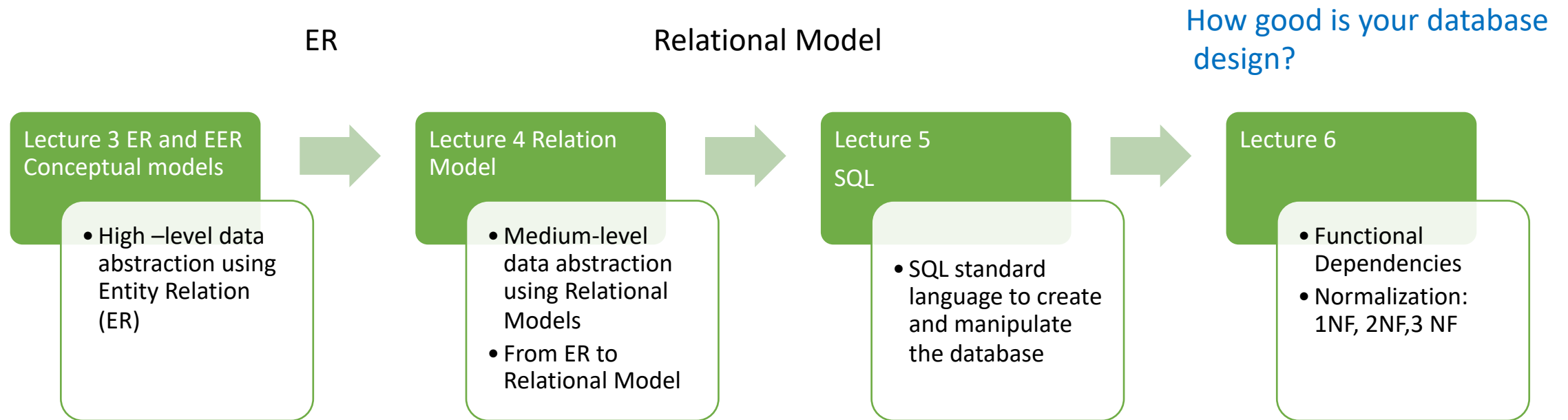
Outline

- Functional Dependencies
- Normalization:
    - 1NF,
    - 2NF,
    - 3NF,
    - Boyce-Codd NF
    - Multivalued Dependencies and 4NF, 5NF
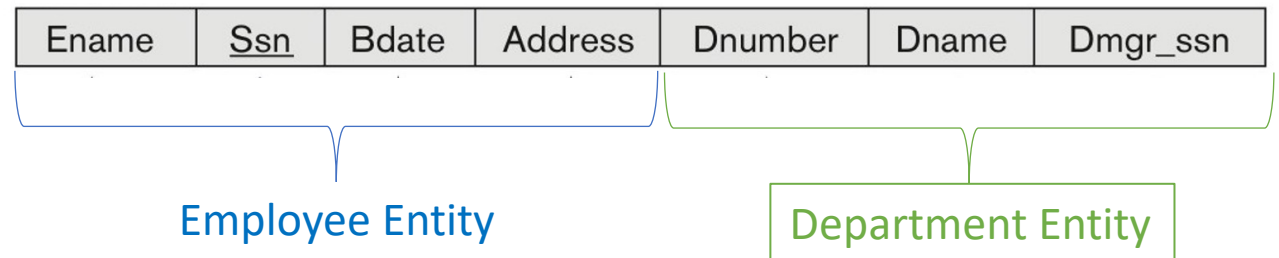
Informal Guidelines for Relation Databases

- **Guideline1. Making sure that the semantics of attributes is clear in the schema**: Do not combine attributes from multiple entities and relationships into a single table. If a relation schema corresponds to one entity or one relation, it is straightforward to explain its meaning. But if the relationship corresponds to a mixture of multiple entities (non-binary relationships), the relation can not be easily explained.
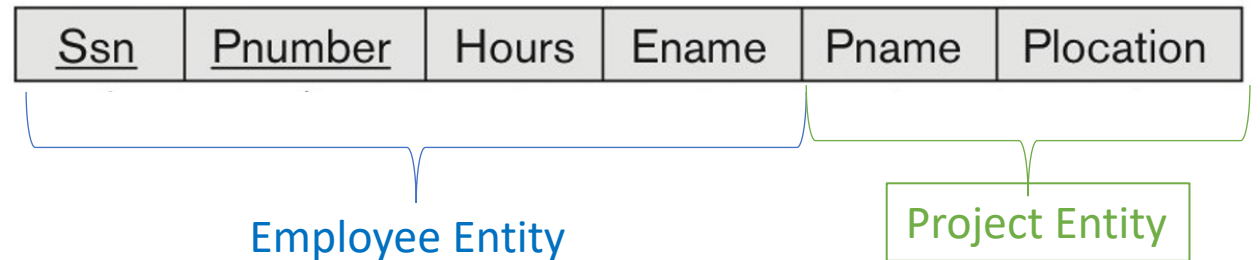
Bad design

Example:

(a)

Employee_Department

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

Employee Entity

Department Entity

(b)

Employee_Project

Ssn	Pnumber	Hours	Ename	Pname	Plocation

Employee Entity

Project Entity

A simplified COMPANY relational database schema.

Better design according the Guideline 1

EMPLOYEE   F.K.

Ename	Ssn	Bdate	Address	Dnumber

P.K.

DEPARTMENT   F.K.

Dname	Dnumber	Dmgr_ssn

P.K.

DEPT_LOCATIONS
    F.K.

Dnumber	Dlocation

P.K.

PROJECT   F.K.

Pname	Pnumber	Plocation	Dnum

P.K.

WORKS_ON
    F.K.   F.K.

Ssn	Pnumber	Hours

P.K.

Informal Guidelines for Relation Databases

• **Guideline 2. Reducing the redundant information in tuples:** design the relation schema/database so that *no insertion, deletion, or modification anomalies* are present in the entities/tables. If anomalies are present, note them clearly and make sure that the programs (e.g., Python program) that updates the database will operate correctly.

Example

Employee_Department

Redundancy

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

Employee_Project

Redundancy   Redundancy

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

Identifying Anomalies

Looking at the example (Employee_Department) shown on slide 6 we can identify the following anomalies

| Operation	Anomaly
Insert new department into Employee_Department Table.	If not employee currently worked in the new department, we can't insert the new department information without violating the database schema. This is an *insertion anomaly* because it restricts the ability to add a certain information without having related data already present.
The employee with ssn '888665555' stopped working in this company and we want to delete him.	Deleting this employee will lose the information about "Headquarters" department. This is a *deletion anomaly* because removing data about employee also removes information about department and department manager.
We need to update the department name from "Research" to "Research and Innovation".	To update this change consistently, we would need to modify each row where "Research" department is listed. This is an *update anomaly* because it requires multiple records updates instead of once to keep our database consistent.

Informal Guidelines for Relation Databases

- **Guidelines 3: Reducing NULL values in tuples/rows:** As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULL are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples/rows in the table.

- For example:
  - if only 15% of employees have individual offices, there is little justification for including an attribute Office_number in the EMPLOYEE table; an additional table Employee_Offices (Essn,Office_number) can be created to include rows/tuples for only the employees with individual offices.

Informal Guidelines for Relation Databases

- **Guideline 4. Disallowing the possibility of generating spurious tuples/rows in table:**
  - Spurious tuples/rows are those rows in a table, which occur as a result of natural joining two tables in wrong manner (more rows than the original set of rows). They are extra (rows) which might not be required. Natural join is implicit join statement on the common columns with the same names in the two tables being joined.
  - Design relation tables so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious rows are generated
  - Avoid tables that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious rows.
- Solution: the equality condition on primary key or foreign keys should be applied.

Example 1 No Spurious Rows

Original Entity   After join operation

Geeks=Geeks (primary key)
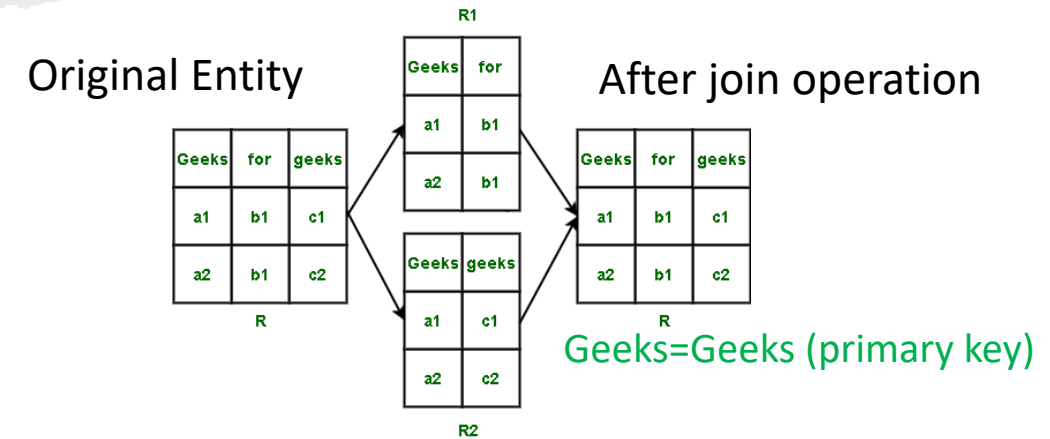
Example 2 Generated Spurious Rows

Spurious Tuples
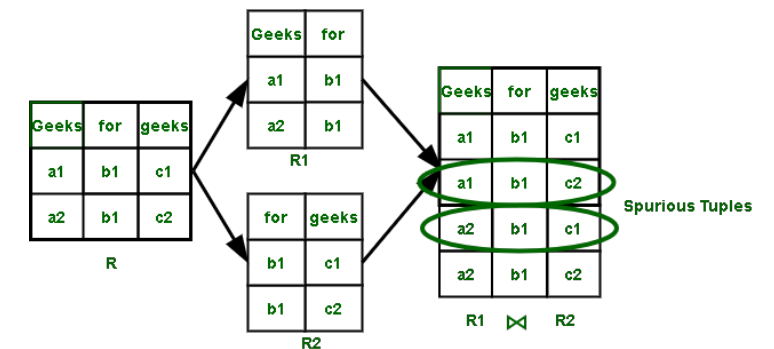
for=for (not primary key/foreign key)

Functional Dependencies (1)

- A *functional dependency* (FD) is a relationship between two attributes, typically between the primary key (PK) and other non-key attributes within a table.

- For any relation **R (or table)**, attribute *Y* is functionally dependent on attribute *X* (usually the PK), if for every valid instance of *X*, that value of *X* uniquely determines the value of *Y*. This relationship is indicated by the representation below :

- $$X \rightarrow Y$$

- The left side (X) is called the *determinant*, and the right side (Y) is the *dependent*.

- Example using Employee_Project table (on slide 6):

Example	Explanation
Ssn → Ename	The value of an employee's social security number (Ssn) uniquely determines the employee name (Ename)
Pnumber →{Pname, Plocation}	The values of a project's number (Pnumber) uniquely determines the project name (Pname) and location (Plocation)
{Ssn,Pnumber} → Hours	A combination of Ssn and Pnumber values uniquely determines the number of hours the employee currently works on the project per week (Hours)

Functional Dependencies (2)

- A functional dependency is a **property of the table R**, not of a particular table state *r* of R.

- Therefore, an FD **cannot** be inferred automatically from a given relation state r but must be defined explicitly by someone who knows the semantics of the attributes of R. All we can say is that a certain FD **may** exist if it holds in that particular state *r*.

- Hence, we can identify a certain FD which **does not hold** if there are rows which show the violation of such an FD.

Example. A table R(A,B,S,D) at particular state r1

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

FD _may hold_:
  B → C; C →B; {A,B} → C; {A,B} → D; {C,D} → B
FD _do not hold_:
  A →B (row 2 violates this constrain);
  B →A (row 3 violates this constrain);
  D → C (row 4 violates it)

Functional Dependencies (2)

- Example 2 A state of TEACH table

Teacher	Course	Book
Erik	Programming	Bartam
Erik	Database	Martin
Alice	Networks	Hoffman
Amanda	Machine Learning	Horowitz

Which FDs do not hold?

Functional Dependencies (3)

- FDs can be displayed on the diagram as follow:
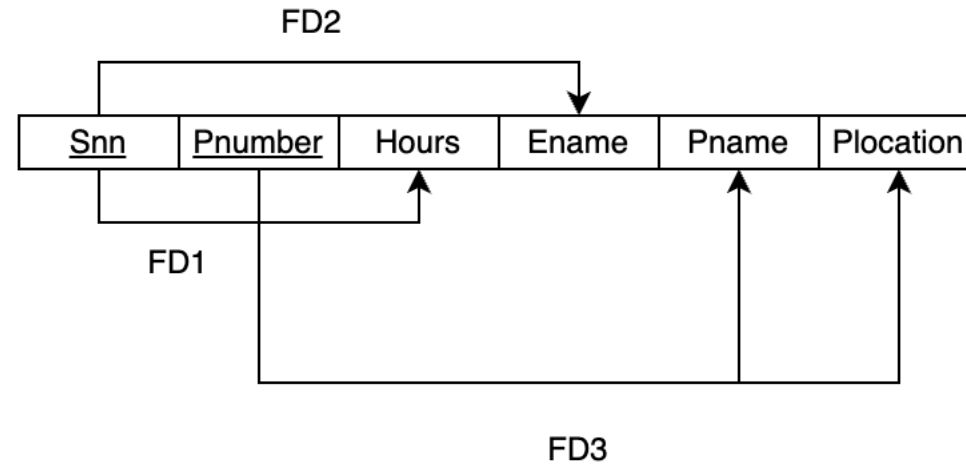
FD1: {Snn, Pnumber} → Hours
FD2: Snn → Ename
FD3: Pnumber → {Pname,Plocation}

Using Armstrong's Axioms rules to identify FDs

- **Reflexive Rule**:
  - A set of attributes always determines itself or any of its subsets
  - $X \supseteq Y, then\ X \rightarrow Y$
  - Example: if {A} → {A}, then {A} → {A,B,C}. Also called _trivial_ dependencies

  If {fname} ⊆ {fname,lname} then {fname,lname} → {fname}

    StudentId -> Name
    Studentid,Name - >Name. This dependency is trivial, because adding Name attribute does not bring a new value

- **Augmentation Rule**:
  - If X → Y holds, then XZ → YZ also holds for any set of attributes Z
  - Example: if {A} → {B} then {A,C} → {B,C}

  If {Dno} → {Dname} then {Dno, Lname} → {Dname, Lname} That is adding attributes to dependencies, does not change the basic dependencies

Using Armstrong's Axioms rules to identify FDs

- **Transitive Rule**:
  - If X → Y and Y → Z, then X → Z
  - Example: If {A} →{B} and {B} →{C} then {A} →{C}

{SSN} → {Dno}
{Dno} →{Dname}   ➡   {SSN} → {Dname}

- **Decomposition Rule**:
  - If X → {Y,Z}, then X →Y and X →Z
  - Example: {A,B} →{C}, then {A,B} → {A} and {A,B} → {B}
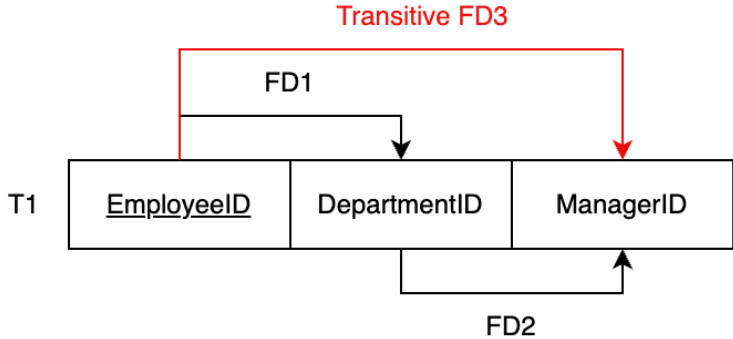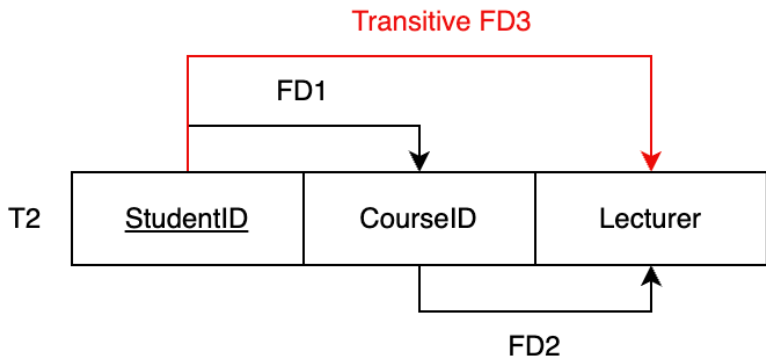
How to Apply Armstrong's Axioms

- Start with the given dependencies.
- Apply Armstrong's Axioms
- Repeat until no new FDs can be inferred
- Verify Dependencies against the data to ensure they hold true

Transitive FDs

- Transitive dependencies occur when an attribute depends on another attribute that is not its primary key.

| Example	Transitive FDs	
1		Attribute MangerID depends on DepartmentID (not primary key) and EmployeeID (primary key)
2		Attribute Lecturer depends on CourseID (not primary key) and StudentID (primary key)

Methodology for testing and improving Relation Database

- Using the Normalization of Relations
- The normalization process take a relation database (or schema) through a series of tests to *certify* whether it satisfies **a certain normal form**.
- All the tables in a database can be in one of the normal forms.
- There are six normal forms, but we will look at the first three, which are:
  - First normal norm (1NF)
  - Second normal norm (2NF)
  - Third normal norm (3NF)
- The **normal form** of a table refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized

First Normal Form(1NF)

- 1NF disallows *relations within relations*, or *relations as attribute values within rows*. The only attribute values permitted by 1NF are **single atomic (or indivisible) values**.

- Example:
    - The domain of Dlocation attrobite contains atomic values or single-valued (e.g., Houston), but some rows can have a set o these values (when one department has several locations). In this case, Dlocation is not functionally dependent on the primary key Dnumber.
    - ***Normalization to 1NF***: remove the attribute Dlocations that violates 1NF and place it in a separate table called DEPT_LOCATIONS with primary key Dnumber of DEPARTMENT. The primary key of this newly formed table is the combination {Dnumber,Dlocation}

(a) Department table is not in 1NF

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
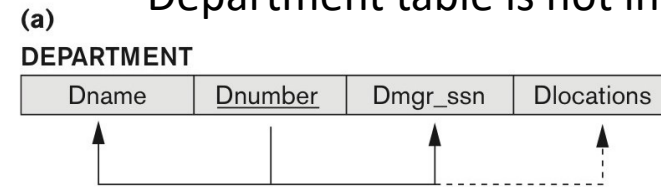
(b) Sample state of table DEPARTMENT

DEPARTMENT

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

(c) DEPARTMENT is not in 1NF version with redundancy

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

DEPARTMENT   F.K.

Dname	Dnumber	Dmgr_ssn

P.K.

DEPT_LOCATIONS
F.K.

Dnumber	Dlocation
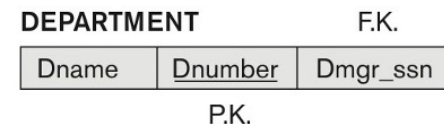
P.K.
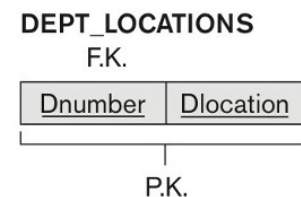
DEPARTMENT and DEPT_LOCATIONS in 1NF form

Normalizing nested relations to 1NF

Normalizing nested relations into 1NF.
(a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS.
(b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple.
(c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

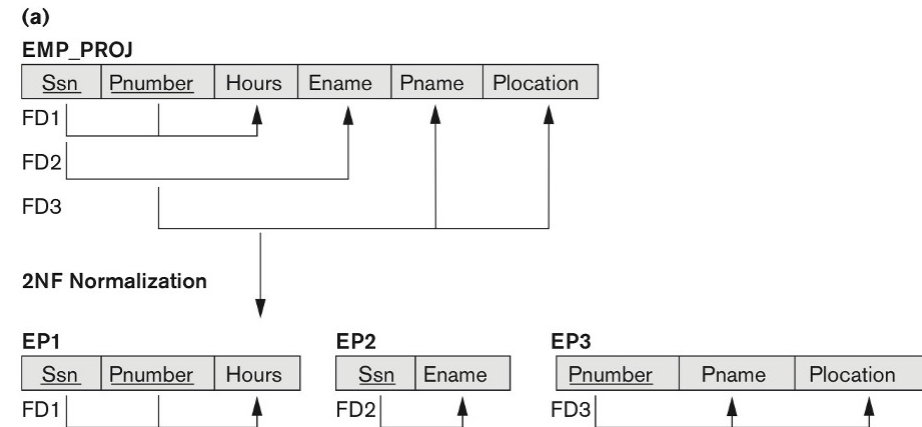| Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321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

Second Normal Form(2NF)

- A database (relation schema) *R* is in 2NF if every nonprime attribute *A* in *R* is ***fully functionally dependent*** on the primary key of *R*.

- ***Fully functionally dependent*** means PK are single-valued attribute or is a composite PK, then each non-key attribute must be fully dependent on the entire PK and not on a subset of the PK (i.e., there must be no partial dependency).

- **Normalization to 2NF**:
    - The relation must first be in 1NF. We can not jump from non 1NF to 2NF without first to normalize to 1NF. So, non NF → 1NF →2NF
    - Decompose and set up a new table for each partial key with its dependent attribute(s).

Normalizing EMP_PROJ into 2NF relations

(a)

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

FD1, FD2, FD3

2NF Normalization

EP1

Ssn	Pnumber	Hours

FD1

EP2

Ssn	Ename

FD2

EP3

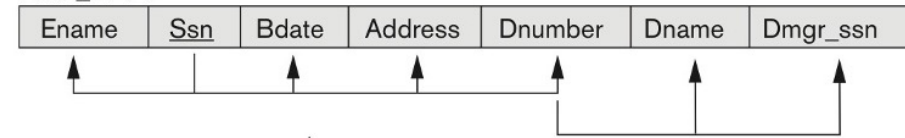| Pnumber	Pname	Plocation

FD3

Third Normal Form (3NF)

- A database (relation schema) R is in 3NF if it satisfies 2NF and no noprime attribute of R is transitively dependent on the primary key.

- More precisely: a non-key attribute may not be functionally dependent on another non-key attribute.

- **Normalization to 3NF**:
  - The relation must first be in 2NF. We can not jump from non 1NF to 3NF without first to normalize to 2NF. So, non NF → 1NF →2NF →3NF
  - Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Example: EMP_DEPT in 2NF form.
Hence it has non-key attributes FDs. Non-key attribute Dnumber uniquely identifies other non-key attributes Dname and Dmgr_snn. Thus violate the 3NF constrain
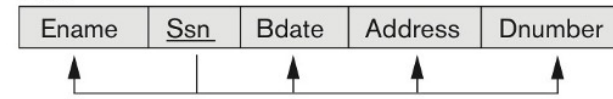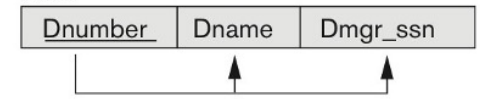
(b)

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

3NF Normalization

ED1

Ename	Ssn	Bdate	Address	Dnumber

ED2

Dnumber	Dname	Dmgr_ssn

ED1 table have single-valued primary key Ssn which uniquely identifies all other attributes. No FDs between non-key Attributes
Same for ED2

Summary of Three Normal Forms

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).

Example 1 : Book Store

- Consider the following relation for published books:

FD1

| Book_title | Author_name | Book_type | List_price | Auth_affil | Publisher |

FD2

FD3
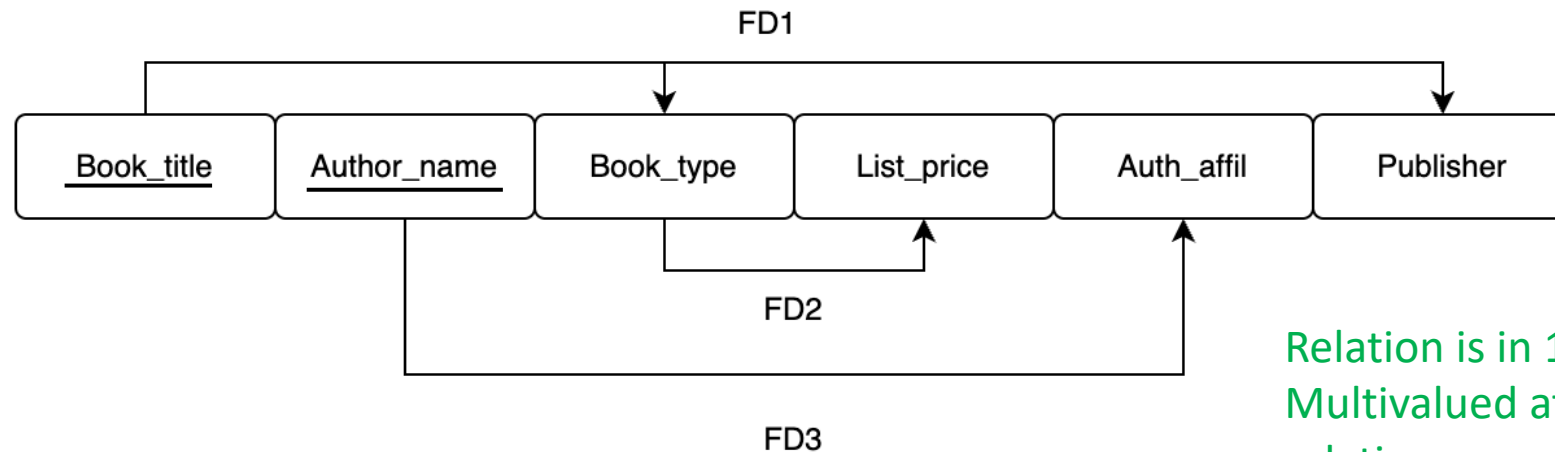
Relation is in 1NF, because it is no Multivalued attributes, and nested relations

What normal form is the relation?

Check if it is in 1NF:
- Is there multivalued attribute?   May be List_price, but we do not see the data and thus can not confirm that this attribute is multivalued.

- Is there nested relations?   No, we do not see the data, thus we assume that there is no nested relations

Example 1 : Book Store

- Consider the following relation for published books:

FD1

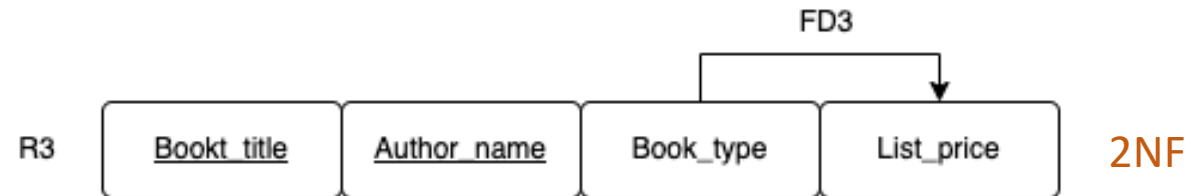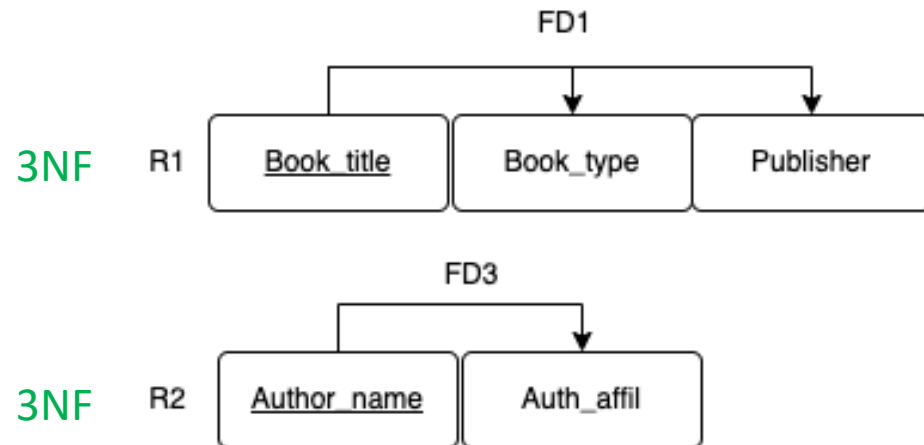| Book_title	Author_name	Book_type	List_price	Auth_affil	Publisher

FD2

FD3
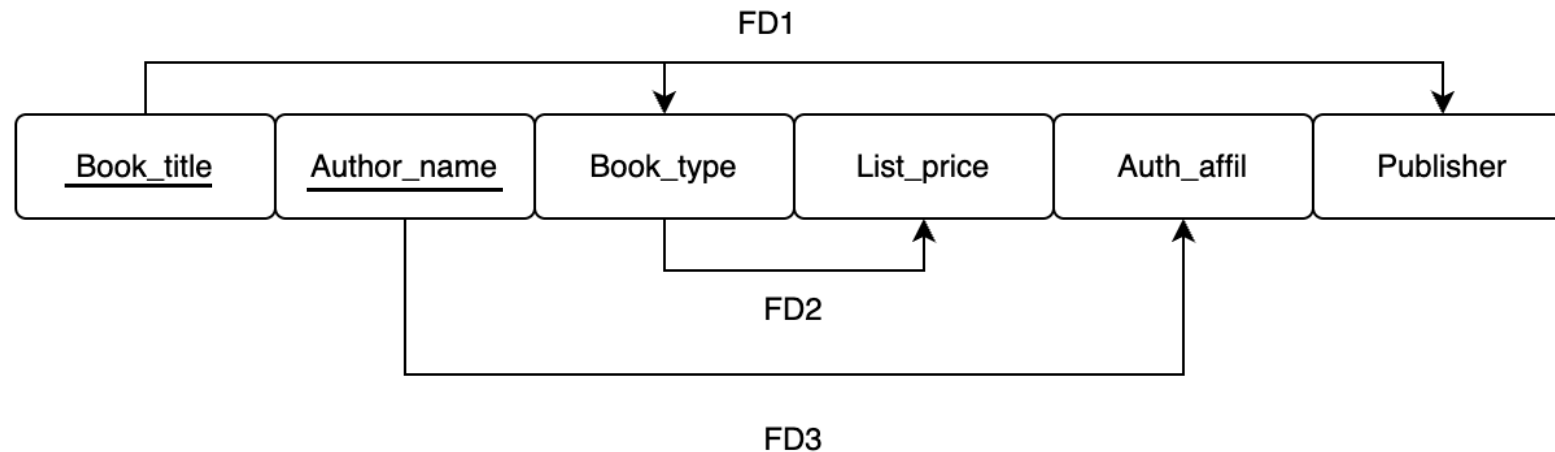
Check if it is in 2NF:

- Is relation contains a composite key?   Yes: Book_title and Author_name

- Is there dependencies exist between non-key attribute and part of key attribute?   Yes, FD1, FD3

Relation is **not in 2NF**, because it has dependencies between non-key attribute and part of the primary key, such as FD1 and FD3.
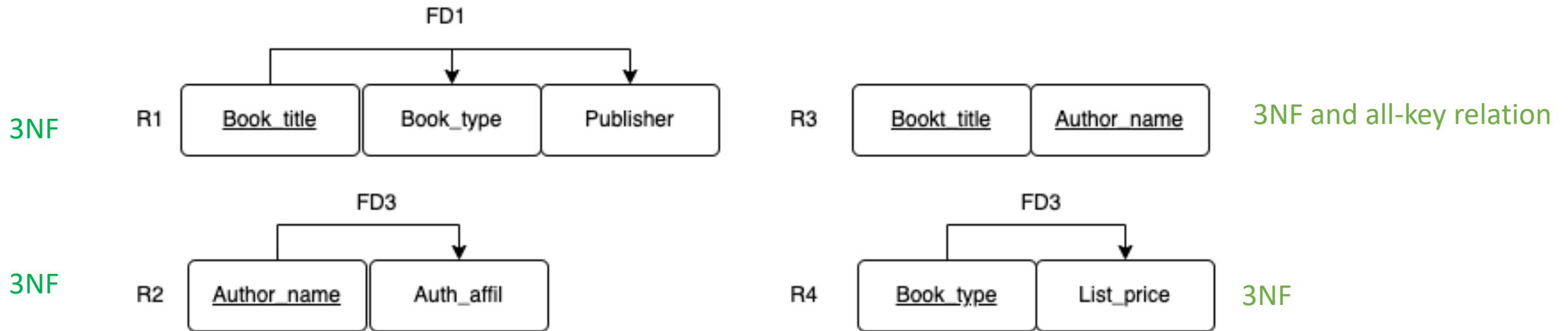
Decompose to 2NF

FD1

3NF | R1

| Book_title | Book_type | Publisher |

FD3

R3

| Bookt_title | Author_name | Book_type | List_price | 2NF

FD3

3NF | R2

| Author_name | Auth_affil |

R3 relation is in 2NF, and not in 3NF because it has Dependencies between non-key attribute and another non-key attribute which if FD3

Relations R1 –R2 are in 3NF, because all non-key attributes depend on the entire primary key, and not transitive dependencies.
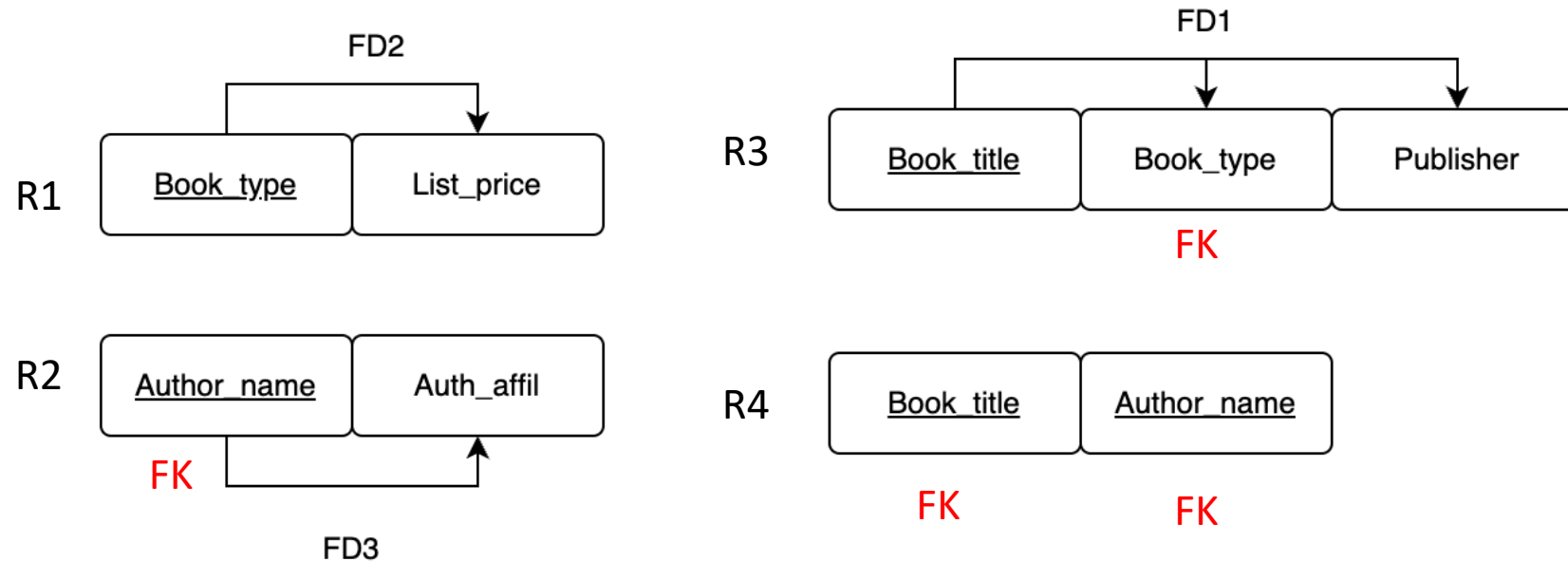
Decompose to R3 to 3NF

FD1

3NF — R1 — | Book_title | Book_type | Publisher |

R3 — | Bookt_title | Author_name | — 3NF and all-key relation

FD3

3NF — R2 — | Author_name | Auth_affil |

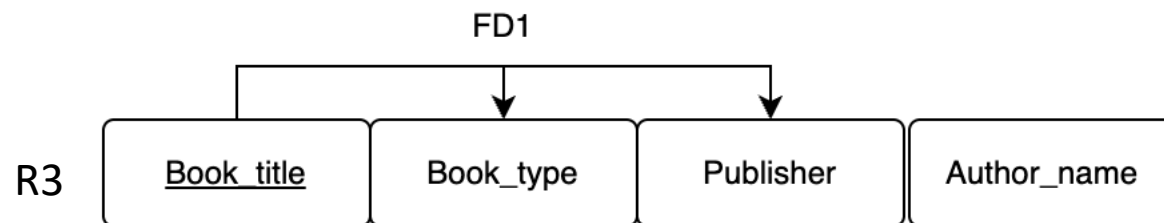**FD3**

R4 — | Book_type | List_price | — 3NF

Relations R1 – R4 are in 3NF, because all non-key attributes depend on the entire primary key, and not transitive dependencies.

How to check if the decomposition is correct?

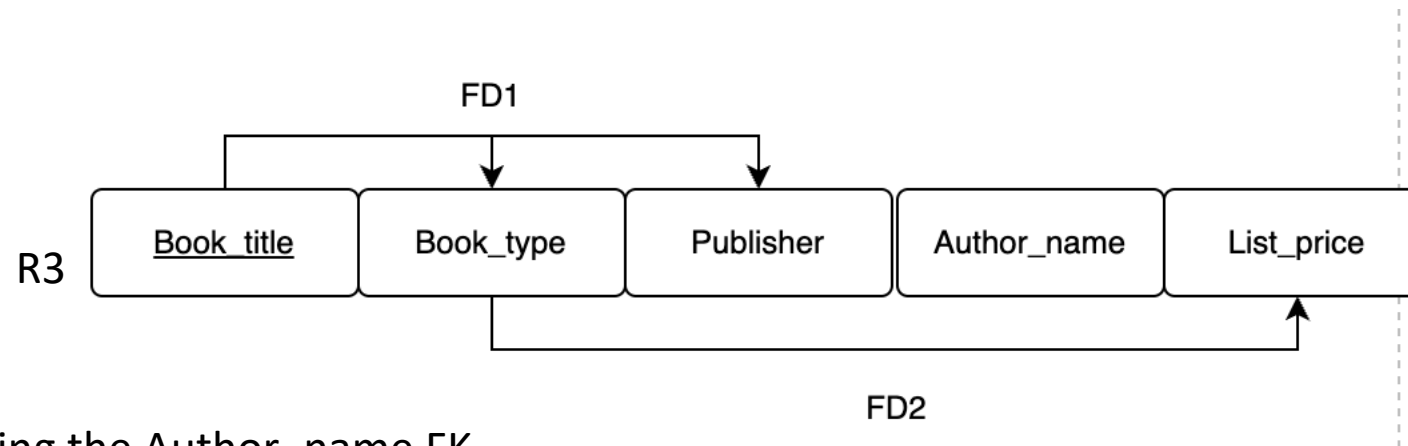You need to compose back to the original table in 0NF using inner join

FD2

R1
| Book_type | List_price |

R3
FD1
| Book_title | Book_type | Publisher |

FK

R2
| Author_name | Auth_affil |

FK

FD3

R4
| Book_title | Author_name |

FK   FK

Step 1: Join R3 and R4 using the Book_title FK

FD1

R3
| Book_title | Book_type | Publisher | Author_name |

Compose back to 0NF

Step 2: Join R1 and R3 using the Book_title FK

R3

FD1

| Book_title | Book_type | Publisher | Author_name | List_price |

FD2

Step 3: Join R2 and R3 using the Author_name FK

FD1   FD3

| Book_title | Book_type | Publisher | Author_name | List_price | Auth_affil |

FD2
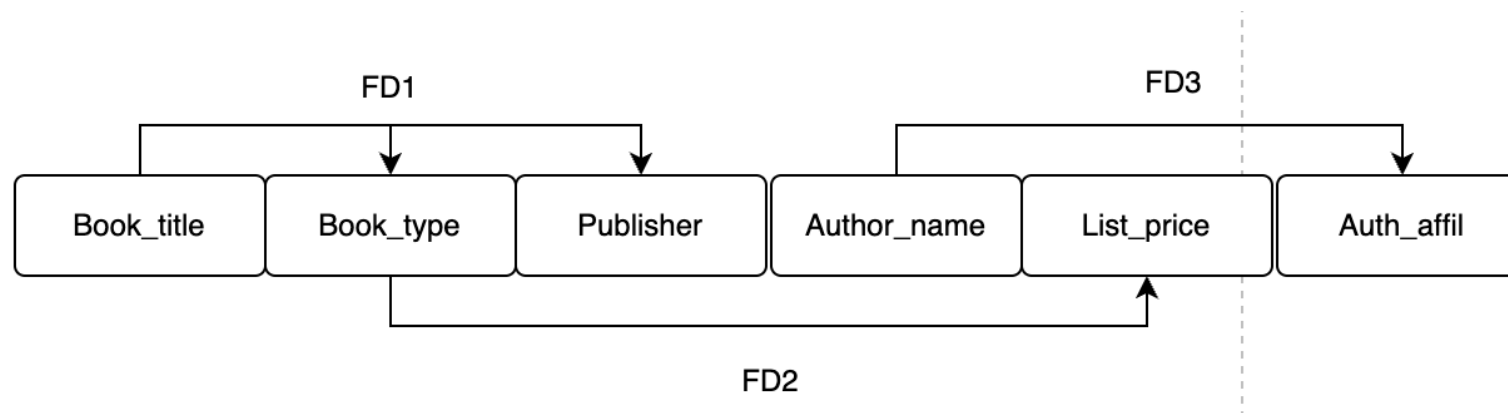
This is original
relation in 0NF
(slide 24)

29(44)

Example 2: Hospital visit

The relation is in 1NF because there is no nested relations, and all attributes are atomic

FD1

FD2

| Doctor | Patient | Date | Diagnosis | Treat_code | Charge |

FD3

Composite primary key: Doctor, Patient,Date
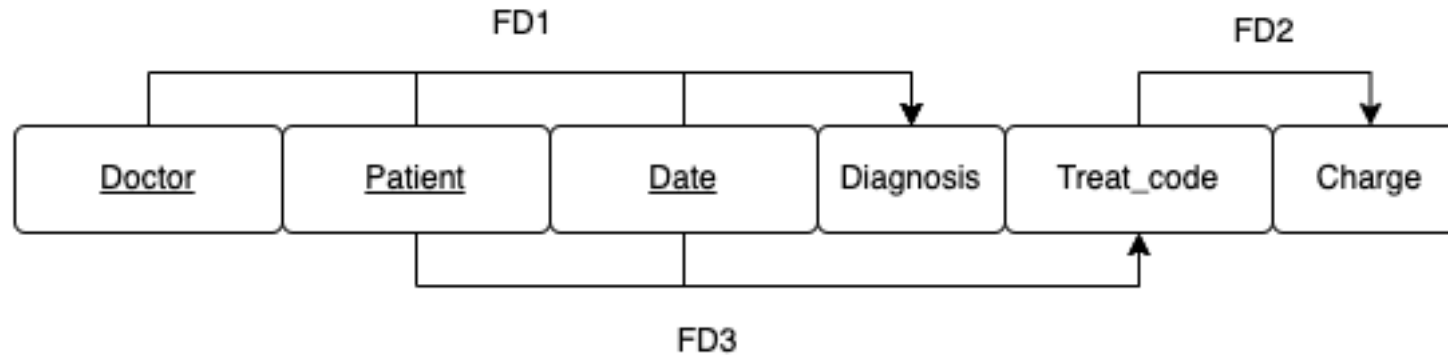
What normal form is the relation?

Check if it is in 1NF:
- Is there multivalued attribute?

- Is there nested relations?

Perhaps Diagnosis, hence we do not see the data therefore is not possible to confirm that there are multivalued attributes. We assume that all attributes are atomic

No, there is no data to confirm this

Example 2: Hospital visit
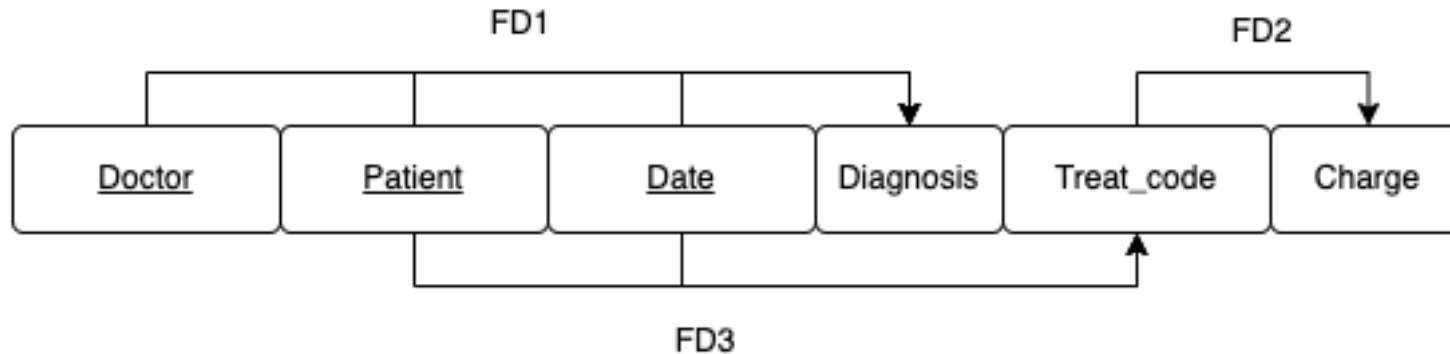
Composite primary key: Doctor, Patient, Date

Check if it is in 2NF:
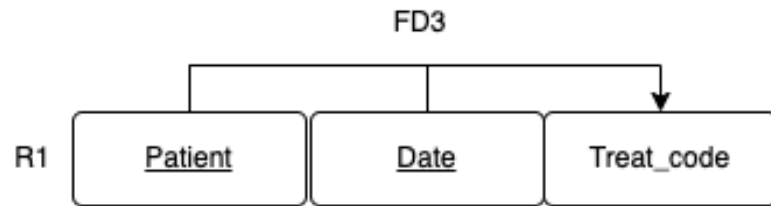• Is there dependencies exist between non-key attribute and part of key attribute??

Yes, FD3

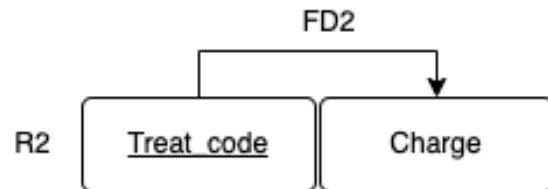The relation **is not in 2NF, because of FD3**

Normalize to 2NF

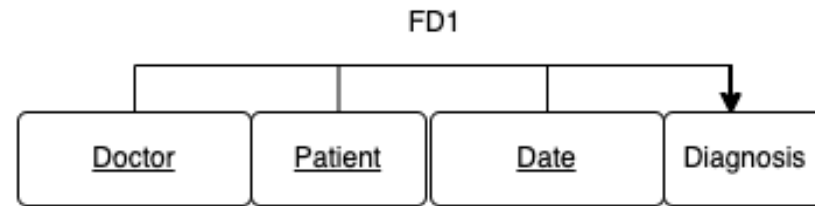Create a new relation for FD3 and FD2

3NF

R1

FD3

Patient	Date	Treat_code

3NF

R3

FD1

Doctor	Patient	Date	Diagnosis

3NF

3NF

R2

FD2

Treat_code	Charge

Check if it is in 3NF: Yes,
• Is there dependencies exist
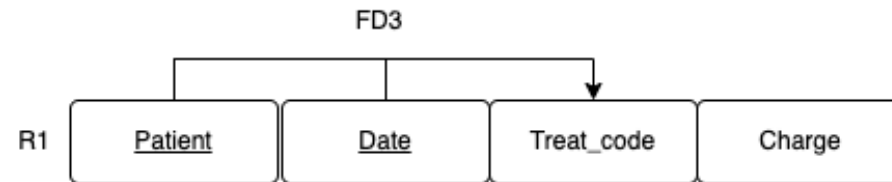between non-key attribute and
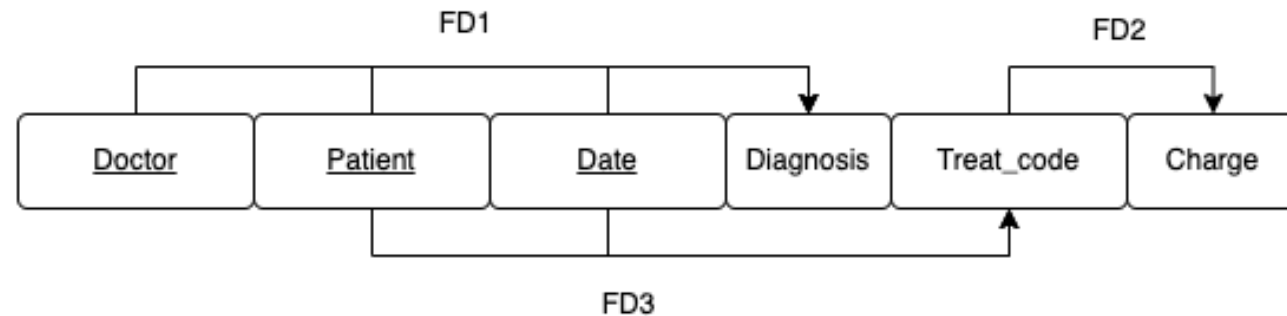another non-key attribute?

All relations in 3NF

Compose back to original table in 1NF

Step1: Join R1 and R2 on key: Treat_code

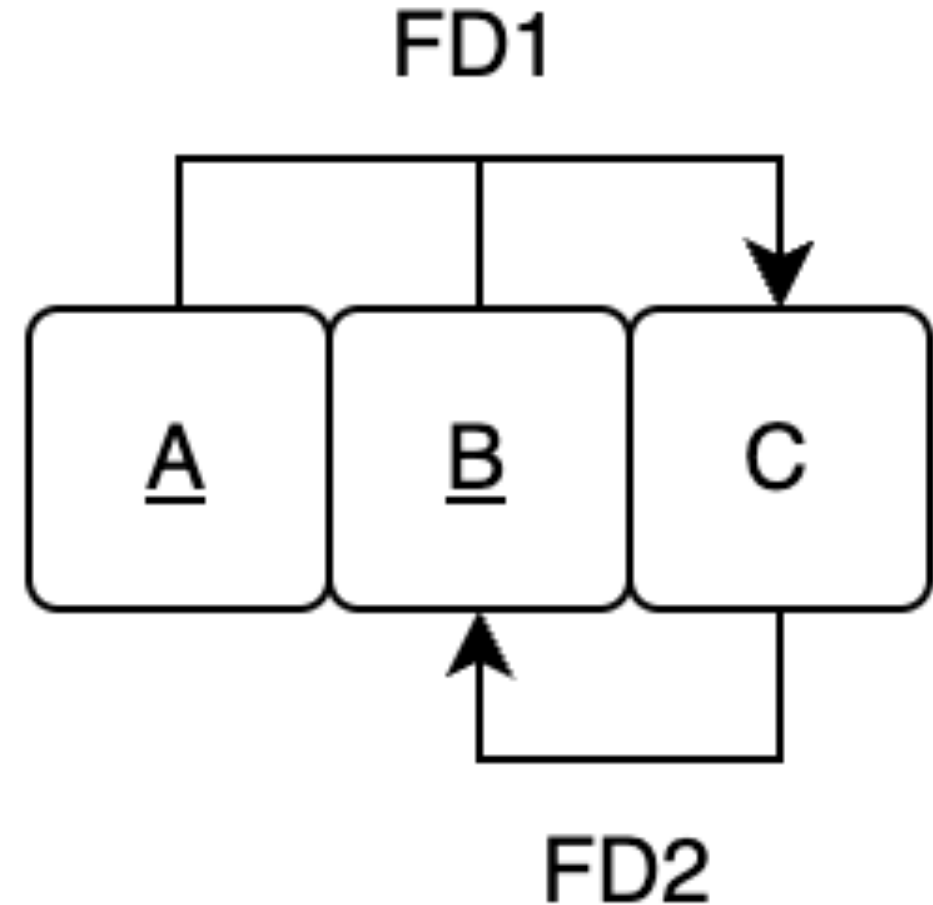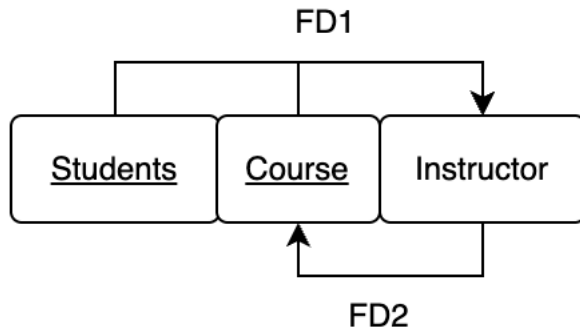Step 2: Join the result with R3 on key Patient, Date

Break 10 min

Relation is in BCNF if whenever a *nontirival functional dependencies* X->A holds in R, then X is a superkey of R

Boyce-Codd Normal Form (BCNF)

In 3NF, but not in BCNF

FD1

A B C

FD2

Example

FD1

N	Student	Course	Instructor
1	Narayan	Database	Mark
2	Smith	Database	Navathe
3	Smith	Informatics	Ammar
4	Smith	Math	Schulman
5	Wallace	Database	Mark
6	Wallace	Informatics	Ahamad
7	Wong	Database	Omiecinski
8	Zelaya	Database	Navathe
9	Narayan	Informatics	Ammar

Possible decompositions:
Option 1: R1 (Student, Instructor) and R2 (Student, Course)
Option 2: R1 (Course, Instructor) and R2 (Course, Student))
Option 3: R1 (Instructor, Course) and R2 (Instructor, Student)

Two properties of decomposition:
- Functional dependencies preserving property
- Nonadditive join property

Possible decompositions:
Option 1: R1 ∩ R2 ↦ (R1-R2): Student ↦ Instructor
Option 2: R1 ∩ R2 ↦ (R1-R2): Course ↦ Instructor
Option 3: R1 ∩ R2 ↦ (R1-R2): Instructor ↦ Course   ➡ FD2

Multivalued Dependency (MVD) and Fourth Normal Form

- MVD describes relationships between sets of attributes

All-key relation is always in BCNF, it has no FDs

Employee_name ↠ Project_Name | Dependent_Name

Employee_Name	Project_Name	Dependent_Name
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

Employee_Name	Project_Name
Smith	X
Smith	Y

4NF

Employee_Name	Dependent_Name
Smith	John
Smith	Anna

4NF

MVD Example

Manufacturer	Color	Fuel
Volvo	Green	diesel
Audi	Grey	Bensin
Volvo	White	diesel
Toyota	White	Bensin
Toyota	White	diesel
Toyota	Grey	Bensin
Ford	White	Etanol
Ford	Red	Etanol

Relation is in 3NF and BCNF, all-key relation

Adding a new color means we need to repeat
Manufacturer and Color attribute three times for each fuel type:
Bensin, Diesel, etanol.

Available colors must be independent from available fuel types.

Manufacturer has a set of colors and set of fuel which cause
the multivalued dependencies.

{Manufacturer} ↠ {Color}

{Manufacturer} ↠ {Fuel}

Manufacturer attribute is not a key, it is a part of the key.
Key is composite: Manufacturer, Color, Fuel

MVD Example

R1

Manufacturer	Color
Volvo	Green
Audi	Grey
Volvo	White
Toyota	White
Toyota	Grey
Ford	White
Ford	Red

R2

Manufacturer	Fuel
Volvo	diesel
Audi	Bensin
Toyota	Bensin
Toyota	diesel
Ford	Etanol

We can add a new color without violating insert constraint

Fifth Normal Form

Person	Brand	Flavor
Adam	Sia Glass	Vanilla
Adam	Sia Glass	Chocolate
Adam	Magnum	Vanilla
Jessica	GB Glace	Strawberry
Jessica	GB Glace	Orange
Amanda	GB Glace	Chocolate
Amanda	ICA glass	Strawberry
Amanda	ICA glass	Mint
Amanda	Sia Glass	Chocolate
Amanda	Sia Glass	Strawberry
Amanda	Sia Glass	Mint

5NF

Person N:M Brand
Brand N:M Flavor
Person N:M Flavor

Amanda likes now Sia Glass

Person	Brand
Adam	Sia Glass
Adam	Magnum
….	……

Brand	Flavor
Sia Glass	Vanilla
Magnum	Vanilla
….	…..

Person	Flavor
Amanda	Chocolate
Amanda	Strawberry
….	…..

Key terms (for Exam)

- A ***functional dependency(FD)*** is a constraint between two sets of attributes from the database.

- A ***normalization*** is a process to determining how much redundancy exists in a database. This process involves analysis of relational schema based on their functional dependencies and primary key to minimizing redundancy, insertions, edition, and update anomalies.

- There are six normal forms, but only three most commonly used are: 1NF, 2NF, and 3NF

- The understanding of table 14.1 (slide 20)

Video Tutorial

- https://www.youtube.com/watch?v=VFuAwSoMatw&list=PLyp-1WQY32mN6U6PUq9c977X9vyHxtohD&index=2