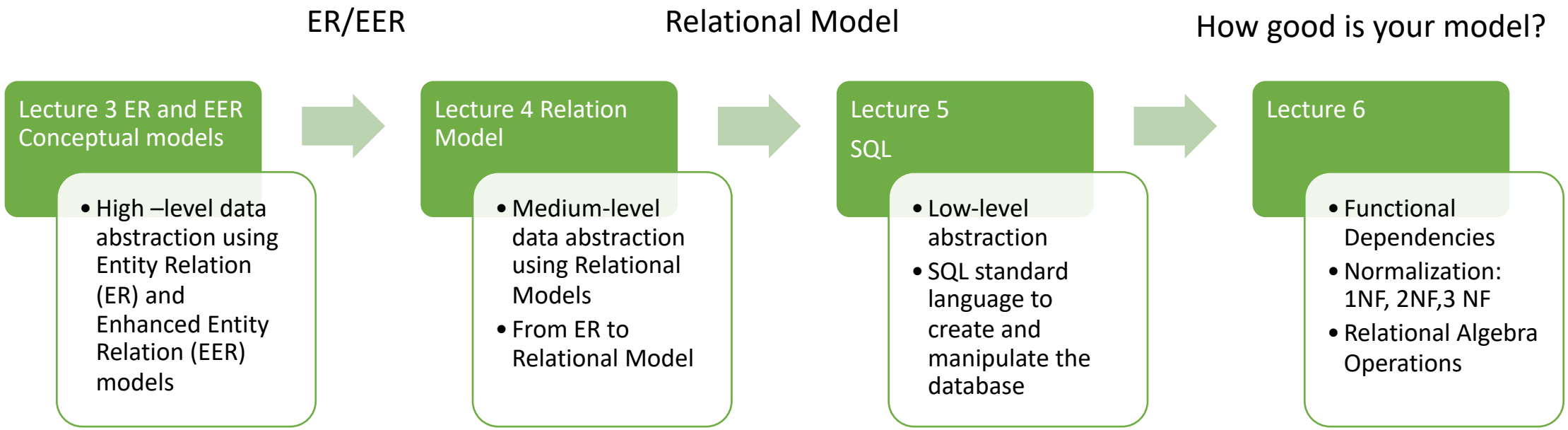
Lecture 6. Relational Algebra, Functional Dependencies and Normalization (Chapter 8 and 14)

alisa.lincke@lnu.se

Connection to previous lectures



Outline

- Functional Dependencies
- Normalization: 1NF,2NF,3NF
- Break 10 min
- Relational Algebra Operations

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

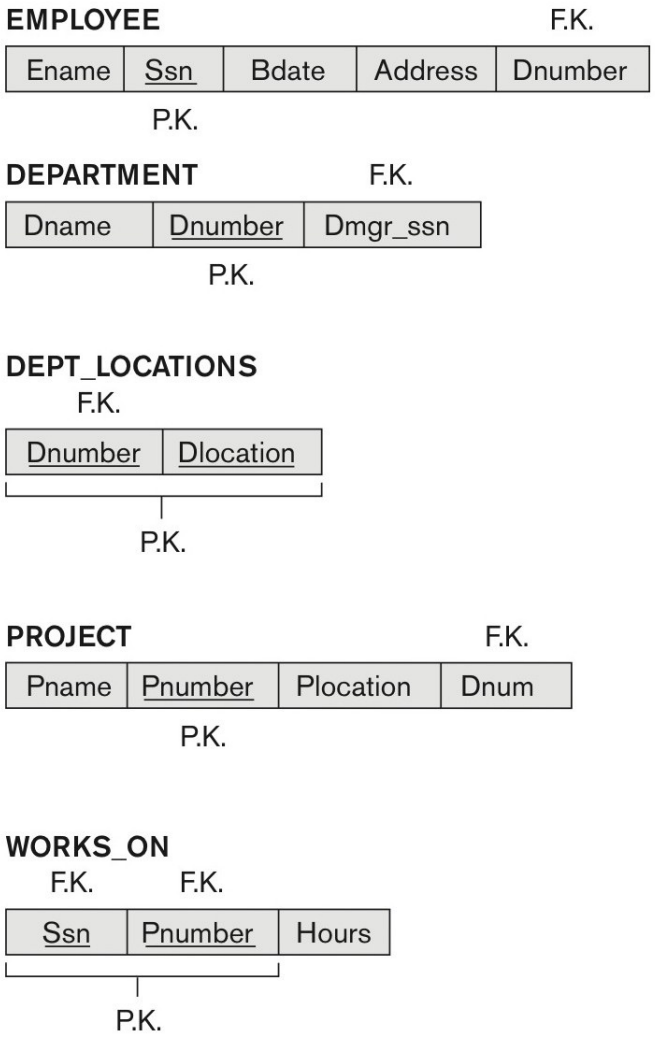
Ssn	Pnumber	Hours	Ename	Pname	Plocation

Employee Entity

Project Entity

A simplified COMPANY relational database schema.

Better design according the Guideline 1



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

Redundancy

Employee_Department

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

Employee_Project

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

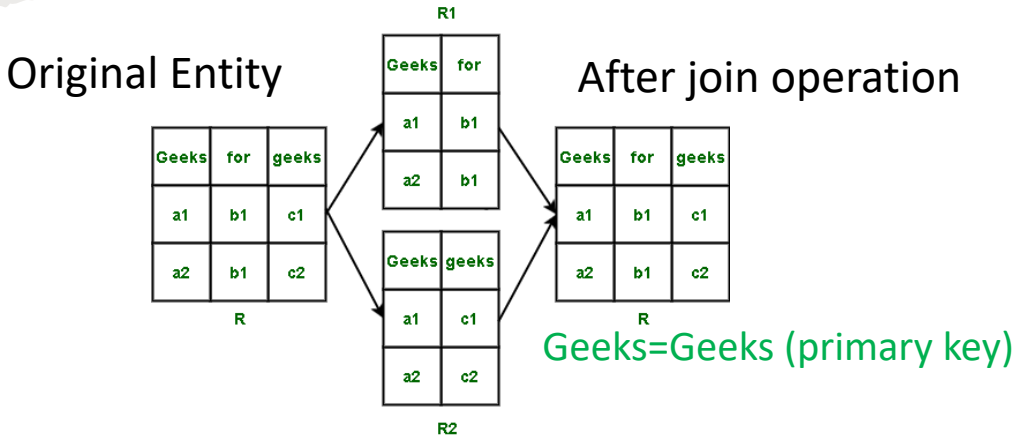
Informal Guidelines for Relation Databases

- **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.

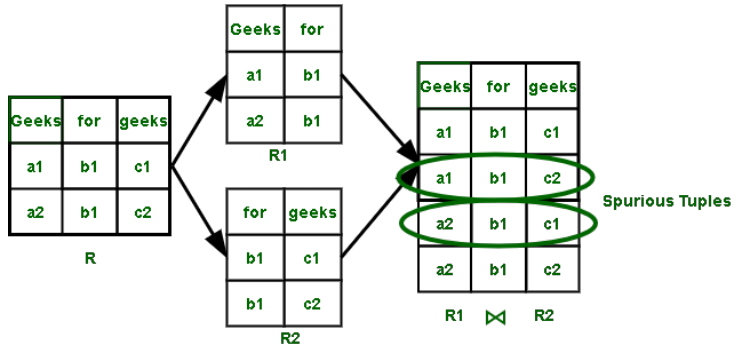
Informal Guidelines for Relation Databases

- **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 sat 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



Example 2 Generated Spurious Rows



for=for (not primary key/foreign key)

[Image taken from: https://www.geeksforgeeks.org/spurious-tuples-in-dbms/](https://www.geeksforgeeks.org/spurious-tuples-in-dbms/)

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 relation 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:

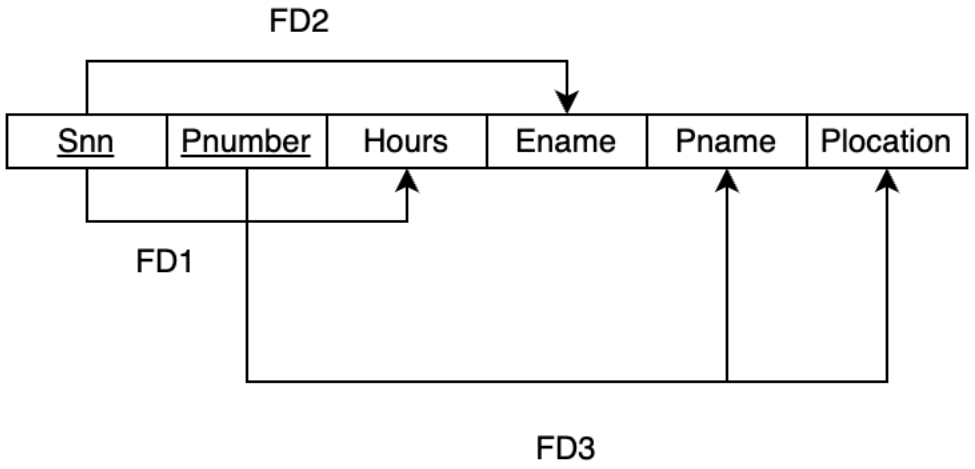
B → C; C →B; {A,B} → C; {A,B} → D

FD do not hold:

A →B (rows 1 and 1 violates this constrain);
B →A (rows 2 and 3 violate this constrain);
D → C (roes 3 and 4 violate it)

Functional Dependencies (3)

- FD can be displayed on the diagram as follow:



FD1: {Snn, Pnumber} → Hours
FD2: Snn → Ename
FD3: Pnumber → {Pname,Plocation}

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}

Department table is not in 1NF

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

Sample state of table DEPARTMENT

(b)

DEPARTMENT

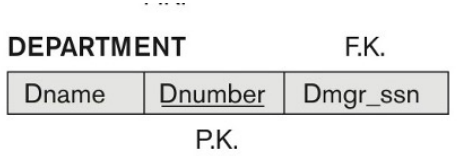
Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

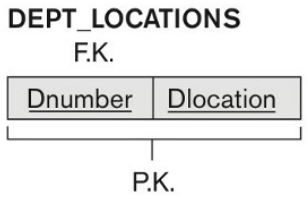
DEPARTMENT is not in 1NF version with redundancy

(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





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

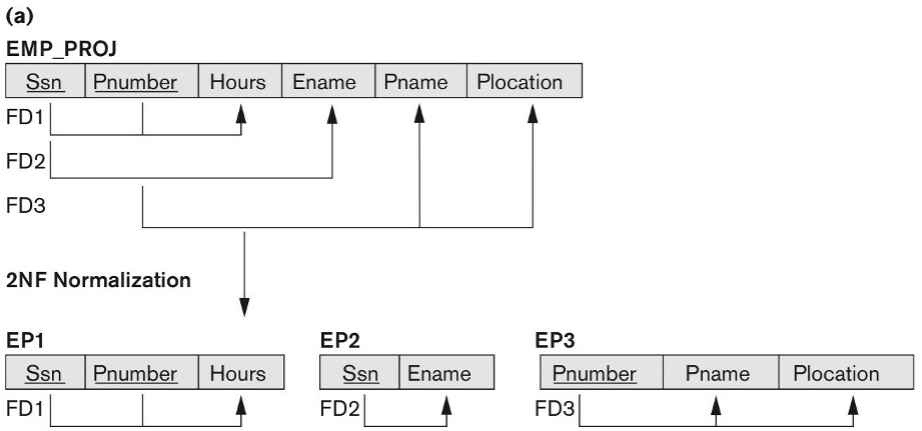
EMP_PROJ2

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).

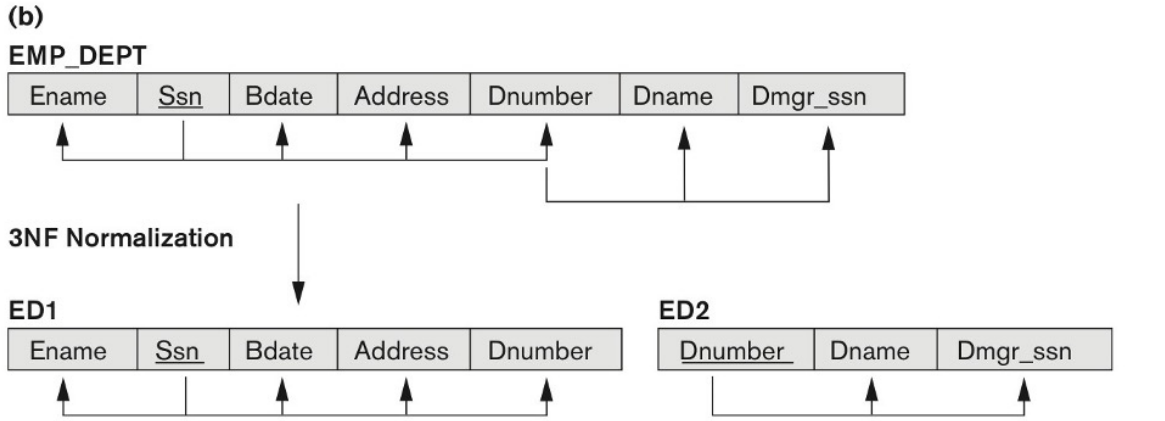
Normalizing EMP_PROJ into 2NF relations



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



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).

Break 10 min

Relational Algebra

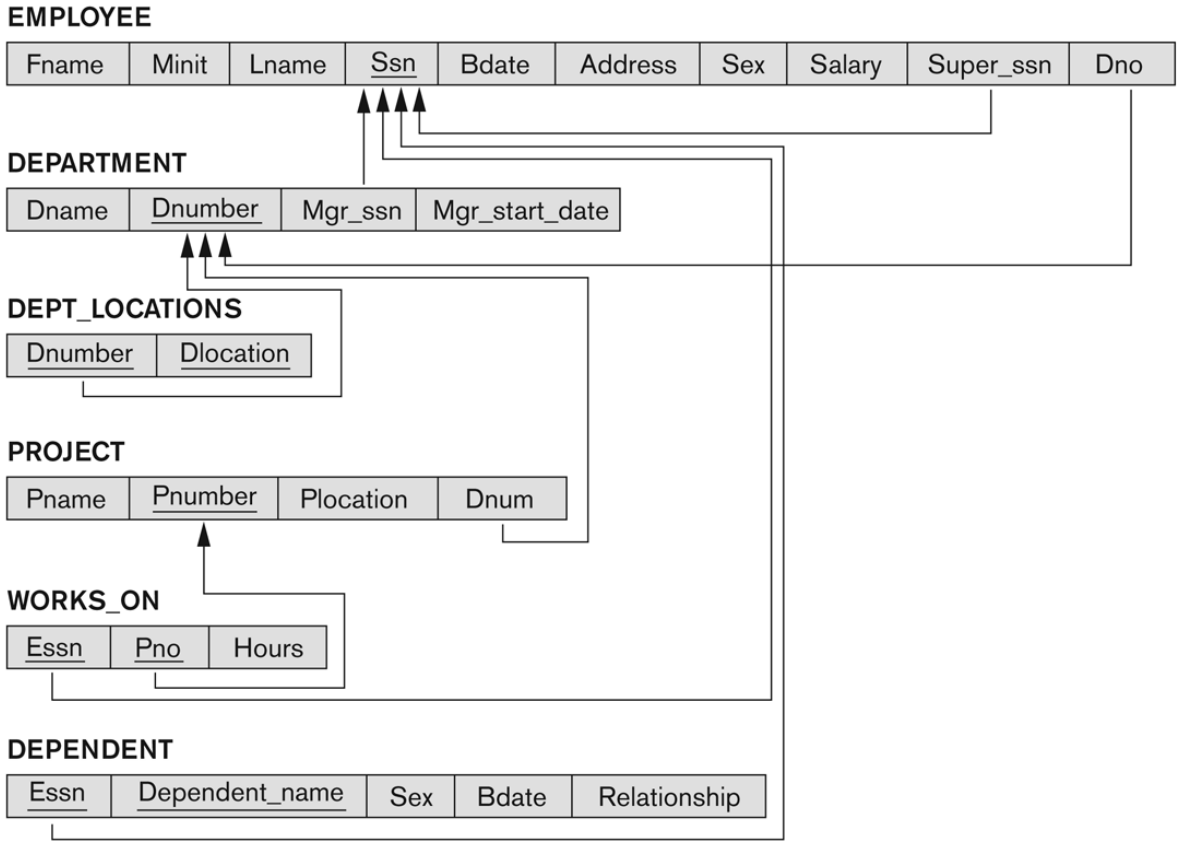
- Relational algebra is the basic set of operations for the relational model
- These operations enable a user to specify **basic retrieval requests** (or **queries**)
- The result of an operation is a *new relation*, which may have been formed from one or more *input* relations
- The **algebra operations** thus produce new relations
  - These can be further manipulated using operations of the same algebra
- A sequence of relational algebra operations forms a **relational algebra expression**
  - The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)
- Relation (formal name) means table (informal name)

Relational Algebra Overview

- Relational Algebra consists of several groups of operations
  - Unary Relational Operations
    - SELECT (symbol: σ (sigma))
    - PROJECT (symbol: π (pi))
    - RENAME (symbol: ρ (rho))
  - Relational Algebra Operations From Set Theory
    - UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE (or MINUS, **–**)
    - CARTESIAN PRODUCT (**x**)
  - Binary Relational Operations
    - JOIN (several variations of JOIN exist)
    - DIVISION
  - Additional Relational Operations
    - OUTER JOINS, OUTER UNION
    - AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

Company Schema

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Unary Relational Operations: SELECT

- The SELECT operation (denoted by σ (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**.
  - The selection condition acts as a **filter**
  - Keeps only those tuples that satisfy the qualifying condition
  - Tuples satisfying the condition are *selected* whereas the other tuples are discarded (*filtered out*)
- Examples:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{DNO = 4} (EMPLOYEE)$$

  - Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{SALARY > 30,000} (EMPLOYEE)$$

Unary Relational Operations: SELECT

- In general, the *select* operation is denoted by $\sigma_{\text{<selection condition>}}(R)$ where
  - the symbol σ (sigma) is used to denote the *select* operator
  - the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
  - tuples that make the condition **true** are selected
    - appear in the result of the operation
  - tuples that make the condition **false** are filtered out
    - discarded from the result of the operation

Unary Relational Operations: PROJECT

- PROJECT Operation is denoted by π (pi)
- This operation keeps certain *columns* (attributes) from a relation and discards the other columns.
  - PROJECT creates a vertical partitioning
    - The list of specified columns (attributes) is kept in each tuple
    - The other attributes in each tuple are discarded
- Example: To list each employee's first and last name and salary, the following is used:

$$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

Unary Relational Operations: PROJECT (cont.)

- The general form of the *project* operation is:

$$\pi_{<\text{attribute list}>}(R)$$

  - π (pi) is the symbol used to represent the *project* operation
  - <attribute list> is the desired list of attributes from relation R.
- The project operation *removes any duplicate tuples*
  - This is because the result of the *project* operation must be a *set of tuples*
    - Mathematical sets *do not allow* duplicate elements.
- PROJECT Operation Properties
  - The number of tuples in the result of projection $\pi_{<\text{list}>}(R)$ is always less or equal to the number of tuples in R
    - If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R
  - PROJECT is *not* commutative
    - $\pi_{<\text{list1}>}(\pi_{<\text{list2}>}(R)) = \pi_{<\text{list1}>}(R)$ as long as <list2> contains the attributes in <list1>

Examples of applying SELECT and PROJECT operations

Figure 8.1 Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4\ AND\ Salary>25000)\ OR\ (Dno=5\ AND\ Salary>30000)}$ (EMPLOYEE). (b) $\pi_{Lname,\ Fname,\ Salary}$(EMPLOYEE). (c) $\pi_{Sex,\ Salary}$(EMPLOYEE).

(a)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

(b)

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Relational Algebra Expressions

- We may want to apply several relational algebra operations one after the other
  - Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
  - We can apply one operation at a time and create **intermediate result relations**.
- In the latter case, we must give names to the relations that hold the intermediate results.

Single expression versus sequence of relational operations (Example)

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

Unary Relational Operations: RENAME

- The RENAME operator is denoted by ρ (rho)
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
  - Useful when a query requires multiple operations
  - Necessary in some cases (see JOIN operation later)
- The general RENAME operation ρ can be expressed by any of the following forms:
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn

Unary Relational Operations: RENAME (continued)

- For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation:
  - If we write:
    - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)
    - RESULT will have the *same attribute names* as DEP5_EMPS (same attributes as EMPLOYEE)
  - If we write:
    - RESULT (F, M, L, S, B, A, SX, SAL, SU, DNO)$\leftarrow$ $\rho_{\text{RESULT (F.M.L.S.B,A,SX,SAL,SU, DNO)}}$(DEP5_EMPS)
    - The 10 attributes of DEP5_EMPS are *renamed* to F, M, L, S, B, A, SX, SAL, SU, DNO, respectively

  Note: the $\leftarrow$ symbol is an assignment operator

Relational Algebra Operations from Set Theory: UNION

- UNION Operation
  - Binary operation, denoted by $\cup$
  - The result of R $\cup$ S, is a relation that includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples are eliminated
  - The two operand relations R and S must be “type compatible” (or UNION compatible)
    - R and S must have same number of attributes
    - Each pair of corresponding attributes must be type compatible (have same or compatible domains)

Relational Algebra Operations from Set Theory: UNION

- Example:
  - To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)
  - We can use the UNION operation as follows:

DEP5_EMPS $\leftarrow \sigma_{DNO=5}$ (EMPLOYEE)

RESULT1 $\leftarrow \pi_{SSN}$(DEP5_EMPS)

RESULT2(SSN) $\leftarrow \pi_{SUPERSSN}$(DEP5_EMPS)

RESULT $\leftarrow$ RESULT1 $\cup$ RESULT2

  - The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by $\cap$
- The result of the operation R $\cap$ S, is a relation that includes all tuples that are in both R and S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be “type compatible”

Example to illustrate the result of UNION, INTERSECT, and DIFFERENCE

Figure 8.4 The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT − INSTRUCTOR. (e) INSTRUCTOR − STUDENT.

(a) **STUDENT**

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

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.
- The normal form of a relation/table refers to the highest normal form condition that is meets, and indicates the degree to which it has been normalized.
- There are six normal forms, but only three most commonly used are: 1NF, 2NF, and 3NF
- Th content and understanding of table 14.1 (slide 17)