Database Design & Applications

Normalization

Objectives

- Informal Design Guidelines for Relational Schemas
- What is Normalization
- Normal Forms
- Functional Dependency
- Normal Forms
- De-Normalization

Introduction

- What is relational database design?

  The grouping of attributes to form "good" relation schemas
- Two levels of relation schemas
  - The logical "user view" level
  - The storage "base relation" level
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Informal Design Guidelines for Relational Schemas

- Semantics of the attributes
- Reducing the redundant values in tuples
- Reducing the null values in tuples
- Disallowing the possibility of generating spurious tuples

Semantics of the Relation Attributes

- 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
- **Bottom Line:** Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

Reducing the redundant values in tuples

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

UPDATE ANOMALY (1)

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” may cause this update to be made for all 100 employees working on project P1.

UPDATE ANOMALY (2)

- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .
- **Inversely:** Cannot insert an employee unless he/she is assigned to a project.
- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project.
- If an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Null Values in Tuples

- 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

Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The **"lossless join"** property is used to guarantee meaningful results for join operations
- The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.

Normalization

Many problems in databases are caused by redundancy. Normalization aims at reducing the level of redundancy in a database.

Redundancy can lead to :

- **Inconsistency :** Errors are more likely to occur when facts are repeated.
- **Update anomalies:** Inserting modifying and deleting data may cause inconsistencies.

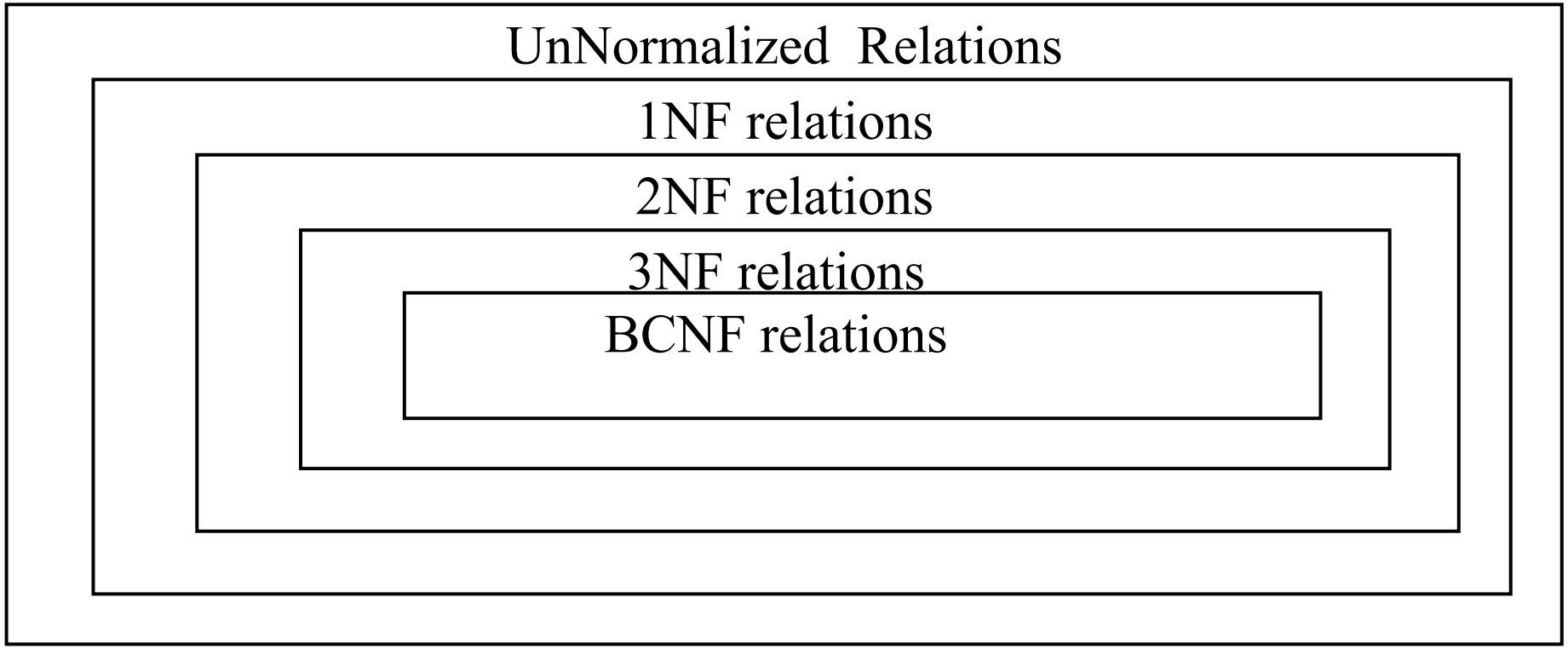
A fully normalized record consist of :

- A primary key that identifies that entity.
- A set of attributes that describes that entity

Normal Forms

- Normalization results in formation of tables that satisfy certain specified constraints, and represents certain normal forms.

Normalization



Functional Dependencies (1)

- Functional dependencies (FDs) are used to specify *formal measures* of the "goodness" of relational designs
- FDs and keys are used to define normal forms for relations
- FDs are constraints that are derived from the *meaning* and *interrelationships* of the data attributes
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

Functional Dependencies (2)

- 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* 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 on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Functional Dependencies (3)

- 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

Functional Dependencies (4)

- A functional dependency is a *1-n* relation of a set of attributes within a relation to another set of attributes in the same relation.
- Given a relation R, attribute A is functionally dependent on B, if each value of A in R is associated with precisely one value of B.

Functional dependency(5)

Customer Entity with following attributes:

- Code
- Name
- Address
- Phone
- Given a particular value of Code, there is precisely one corresponding value of name, address, phone. Therefore name, address, phone are functionally dependent on the attribute Code.
- In Customer entity the attribute Code will be unique for each tuple. Therefore it is candidate key. All attributes must be functionally dependent on the key.

UNF → 1NF

1st normal form: A relation is in the first normal form when each cell of the table contains precisely one value.

Procedure: Put all double values on a separate line.

UNF

JobRef	*JobTitle*	*DNo*	*DName*	*Interview Date*	*Interview Time*	*CandNo*	*CandName*
J101	PC Support	D5	Finance	12/9/95	10:00	C0023	Tom Burns
				12/9/95	11:00	C0024	Ellen McPhee
				12/9/95	12:00	C0030	Tracy Jones
J102	Programmer	D11	IT	13/9/95	10:00	C0034	Allan Watt
				13/9/95	11:00	C0024	Ellen McPhee
				14/9/95	14:00	C0035	Rorry Stone
J103	Analyst	D11	IT	14/5/95	15:00	C0023	Tom Burns
				14/5/95	16:00	C0041	Clare Craig

1NF

JobRef	*JobTitle*	*DNo*	*DName*	*Interview Date*	*Interview Time*	*CandNo*	*CandName*
J101	PC Support	D5	Finance	12/9/95	10:00	C0023	Tom Burns
J101	PC Support	D5	Finance	12/9/95	11:00	C0024	Ellen McPhee
J101	PC Support	D5	Finance	12/9/95	12:00	C0030	Tracy Jones
J102	Programmer	D11	IT	13/9/95	10:00	C0034	Allan Watt
J102	Programmer	D11	IT	13/9/95	11:00	C0024	Ellen McPhee
J102	Programmer	D11	IT	14/9/95	14:00	C0035	Rorry Stone
J103	Analyst	D11	IT	14/5/95	15:00	C0023	Tom Burns
J103	Analyst	D11	IT	14/5/95	16:00	C0041	Clare Craig

1NF Anomalies

INSERT: It is not possible to add jobs without adding applicants

DELETE: As soon as the only application for a department is removed, the department is gone too.

UPDATE: the same information is listed more than once in the database. If IT gets another DNO, we have to change that for all

lines!

2NF

A Relation is in 2NF if the relation is in 1NF and if all non-key attributes are fully functionally dependant on the primary key.

If a non-key attribute functionally dependent on a part of primary key(Composite key) this is termed as partial dependency.

Full Functional Dependency

Full functional dependency $A \rightarrow B$ is a functional dependency on B of A, so that B is not dependant on a part of A

Two attributes (A and B) are fully functional dependant on each other if:

- $A \rightarrow B$
- A is a composite determinant
- B is not dependant on a part of A

Procedure for 2NF

1. Identify the primary key for the 1NF relation
2. Identify the functional dependencies in the relation
3. Remove partial dependencies by placing them in new relations, together with a copy of the determinant.
4. The table should be decomposed without any loss of information

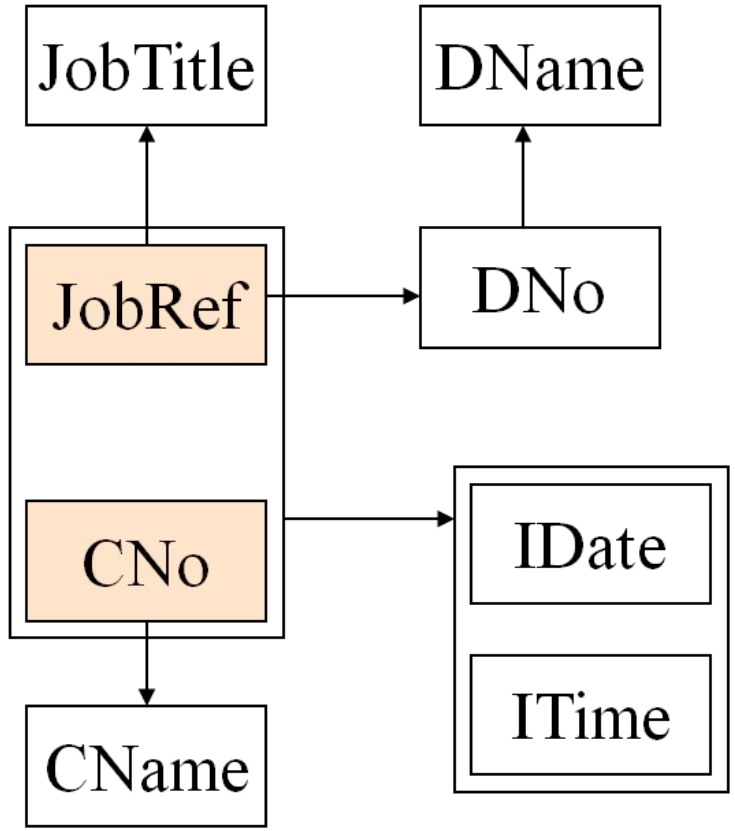
2NF example

1. Identify the primary key

JobRef	*JobTitle*	*DNo*	*DName*	*Interview Date*	*Interview Time*	*CandNo*	*CandName*
J101	PC Support	D5	Finance	12/9/95	10:00	C0023	Tom Burns
J101	PC Support	D5	Finance	12/9/95	11:00	C0024	Ellen McPhee
J101	PC Support	D5	Finance	12/9/95	12:00	C0030	Tracy Jones
J102	Programmer	D11	IT	13/9/95	10:00	C0034	Allan Watt
J102	Programmer	D11	IT	13/9/95	11:00	C0024	Ellen McPhee
J102	Programmer	D11	IT	14/9/95	14:00	C0035	Rorry Stone
J103	Analyst	D11	IT	14/5/95	15:00	C0023	Tom Burns
J103	Analyst	D11	IT	14/5/95	16:00	C0041	Clare Craig

2NF example

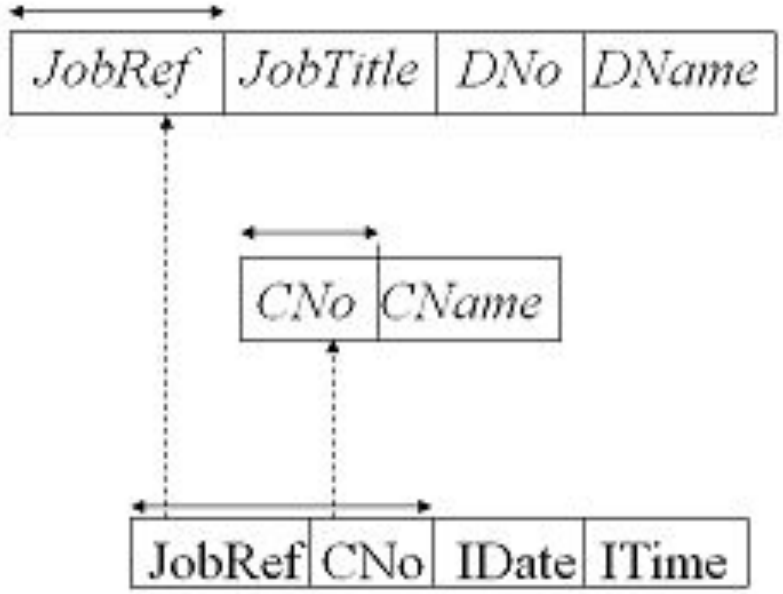
2. Identify functional dependencies:



2NF Example

3. Remove partial dependencies

- All domains contain atomic values
- All non-key attributes are FFD of the PK



2NF problems

INSERT: If a department has no job openings, the department cannot be entered.

DELETE: When all job openings of a certain department are removed, the department itself is lost.

UPDATE: When IT is assigned a different DepNo, all job openings for the department have to be updated.

3NF

Third normal form:

A relation is in 3NF, if it is in 2NF and if all non-key attributes are non-transitively dependant on the primary key.

Transitive dependancy:

C is transitively dependant of A, when:

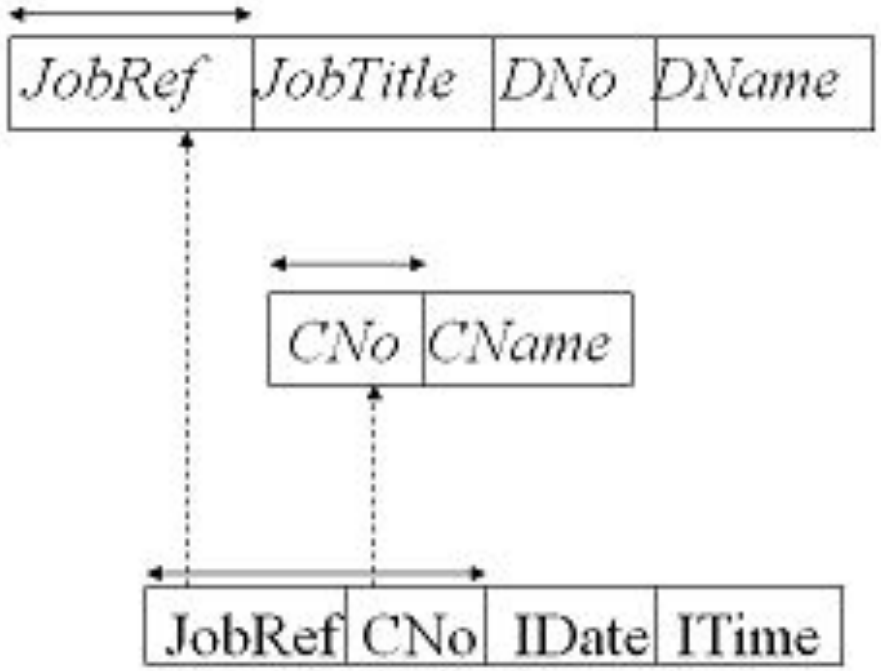
$A \rightarrow B; B \rightarrow C$

3NF procedure

1. Identify the primary key in each relation
2. Identify functional dependencies in each relation
3. Remove transitive dependencies by moving them to other tables, with a copy of the determinant

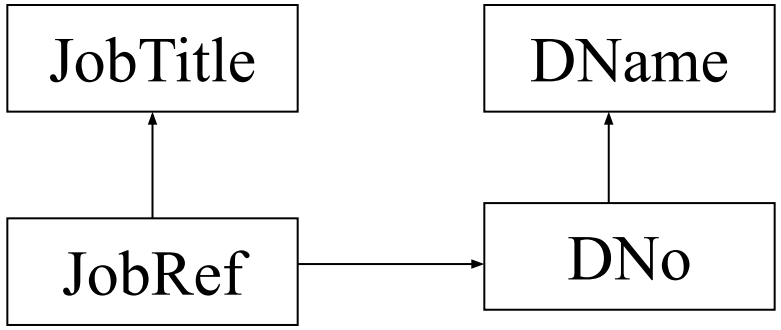
3NF Example

1. Identify primary keys

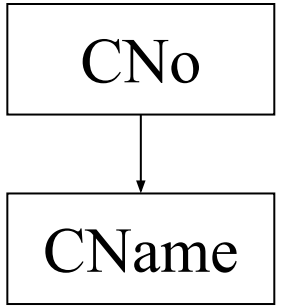


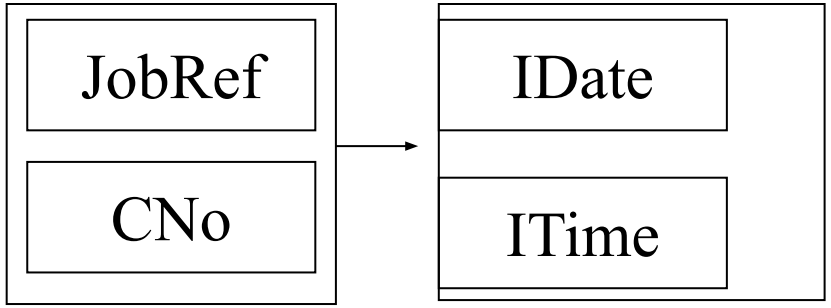
3NF Example

2. Identify functional dependencies



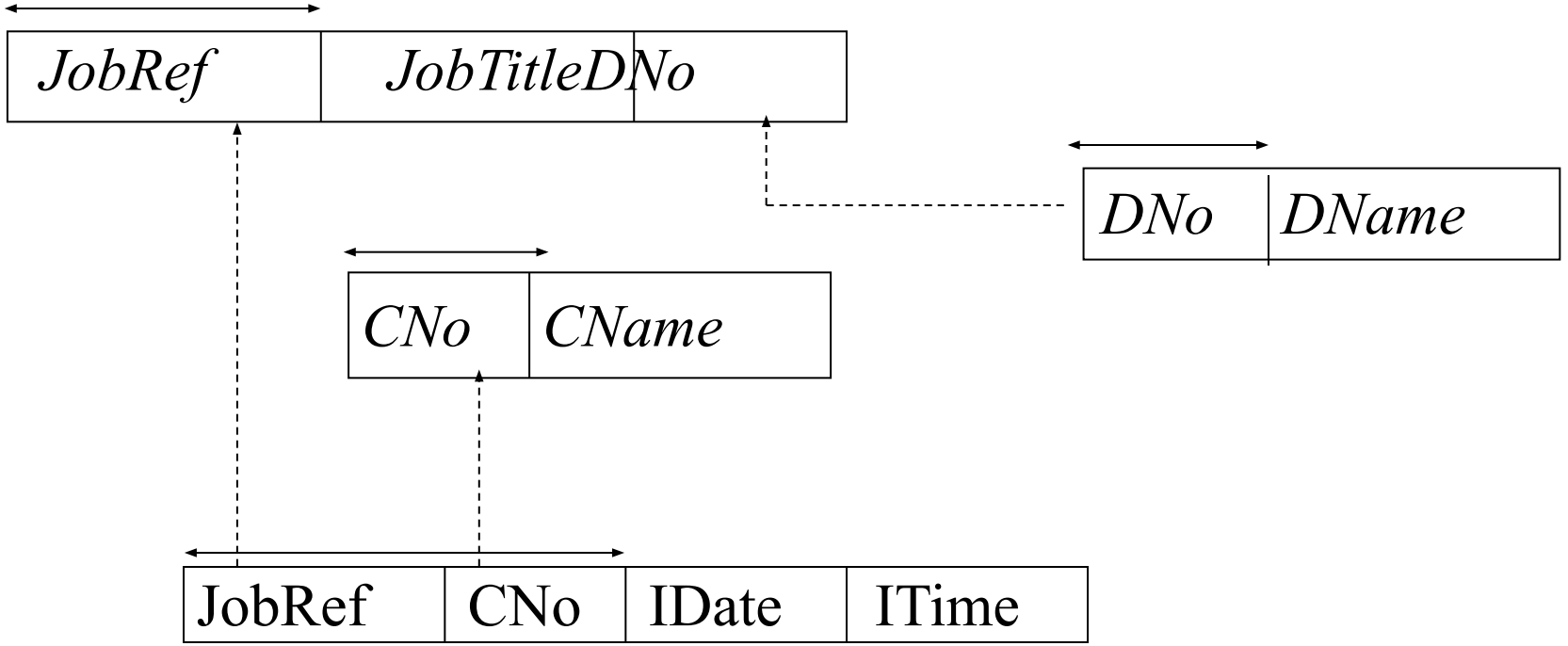
Transitive dependency!





3NF Example

3. Remove the transitive dependencies



BCNF

Relations in 3NF can still contain redundancy when:

1. A relation has two or more candidate keys, *and*
2. Candidate keys are composite keys, *and*
3. Candidate keys overlap.

Boyce-Codd Normal form:
A relation is in BCNF if all determinants are candidate keys.
Usually, a relation that is in 3NF is in BCNF too.

BCNF procedure

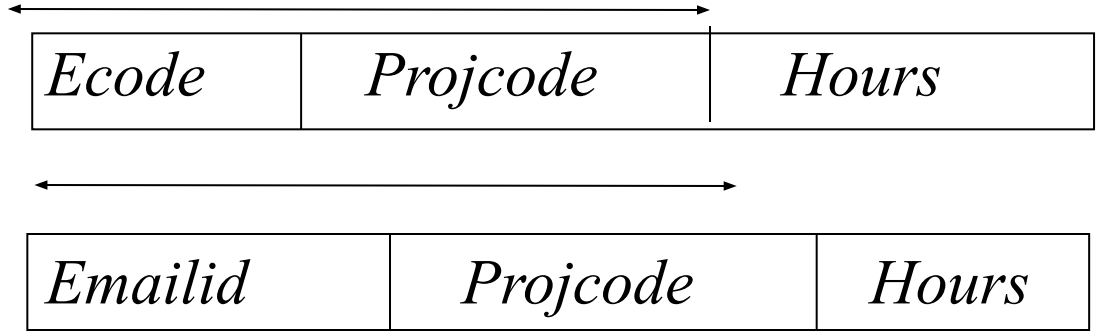
1. Identify all candidate keys
2. Identify all functional dependencies
3. If functional dependencies exist in the relation where the determinant is not candidate key, remove that FD and place it in a new relation.

BCNF Example

ECODE	EMAILID	PROJCODE	HOURS
E1	vinodjha@jecrc.com	P2	48
E1	vinodjha@jecrc.com	P5	100
E1	vinodjha@jecrc.com	P6	15
E4	atultak@jecrc.com	P6	250
E4	atultak@jecrc.com	P5	75

BCNF Example

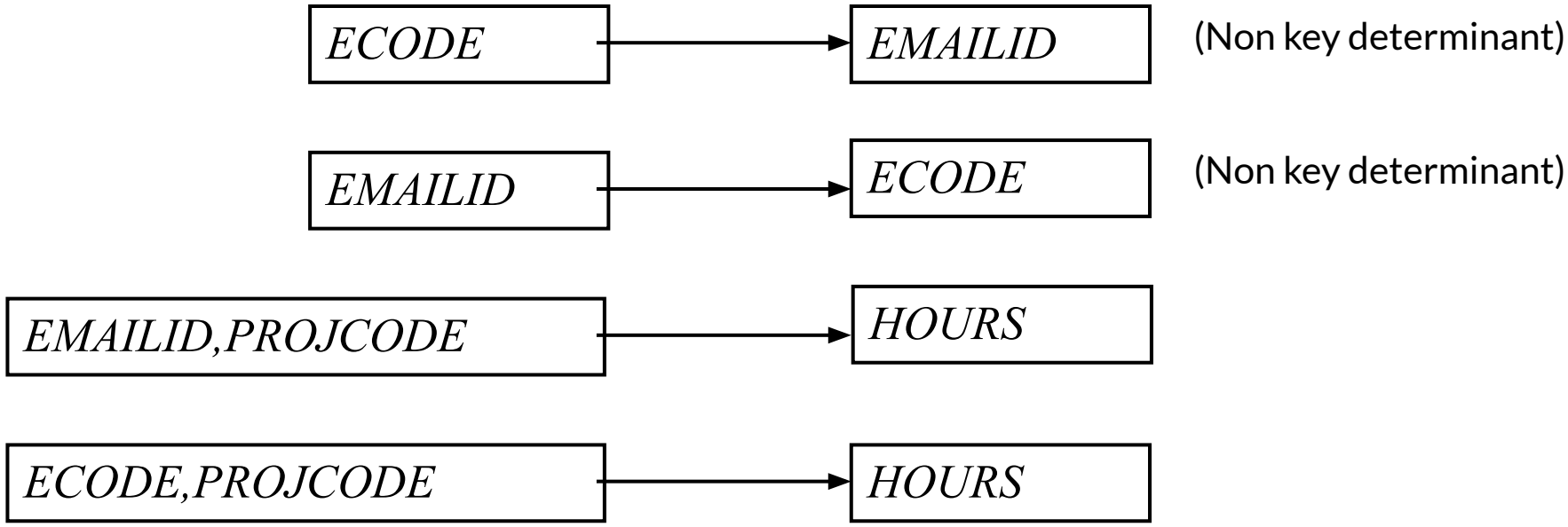
1. Identify all candidate keys



Projcode is overlapping candidate key

BCNF Example

- Valid functional dependencies are



BCNF Tables

ECODE	PROJCODE	HOURS
E1	P2	48
E1	P5	100
E1	P6	15
E4	P6	250
E4	P5	75

ECODE	EMAILID
E1	vinodjha@jecrc.com
E4	atultak@jecrc.com

4NF

- A relational table is in the Fourth Normal Form (4NF) if it is in BCNF and all multivalued dependencies are also functional dependencies.
- Multivalued dependencies arise when a relation R having a nonatomic attribute is converted to a normalized form.
- For Each X value in such a relation, there will be a set of Y values associated with it. This association between the X and Y values does not depend on the values of the other attributes in the relation

4 NF

- A relation is in 4NF when it is in BCNF and has no more than one multi-valued dependency.
- If A↠B and A↠C but B and C are unrelated, i.e. A↠(B,C) is false, then we have more than one multi-valued dependency.

4 NF

- This table is difficult to maintain since adding a new hobby requires multiple new rows corresponding to each skill
- Two multi-valued dependencies
- **EMPLOYEE# ↠ SKILLS**
- **EMPLOYEE# ↠ HOBBIES**

Emp#	Skills	Hobbies
121	Programming	Golf
121	Programming	Bowling
121	Analyst	Golf
121	Analyst	Bowling
122	Management	Golf
1211	Management	Gardening

4 NF

- Removing Multivalued Dependency from the table

Emp#	Hobbies
121	Golf
121	Bowling
122	Golf
122	Gardening

Emp#	Skills
121	Programming
121	Analyst
122	Management

5 NF
Project-Join Normal Form (PJNF)

- A table is in Fifth Normal Form(5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and there are no pairwise cyclical dependencies in the primary key comprised of three or more attributes.

5 NF

- Pairwise cyclical dependency means that:
  - You always need to know two values (pairwise).
  - For any one you must know the other two (cyclical).
- 5NF is based on the concept of join dependencey .

5 NF

- Join dependencies
- In some cases it may not be possible to find a lossless decomposition of a relation schema into two relational schemas
- The same relation schema can be decomposed losslessly into three relation schemes. This property is referred to as the join dependency (JD).

Example 5 NF

- Consider the relation
- STUDENT_ADVISOR (Name, Department, Advisor)
- Functional dependencies
  - Name->Department
  - Name->Advisor
  - Advisor->Department
- There is a cyclic dependency between Name and Advisor

Example 5 NF

Name	Department	Advisor
Jones	Comp Sci	Smith
Ng	Chemistry	Turner
Martin	Physics	Bosky
Dulles	Decision Sci	Hall
Duke	Mathematics	James
James	Comp Sci`	Clark
Evan	Comp Sci	Smith
Baxter	English	Bronte

Example 5 NF

- The decomposition of STUDENT_ADVISOR into
  - STUDENT_DEPARTMENT(Name, Department)
  - DEPARTMENT_ADVISOR(Department, Advisor)
- Lossy decomposition of STUDENT_ADVISOR Table

Name	Department
Jones	Comp Sci
Ng	Chemistry
Martin	Physics
Dulles	Decision Sci
Duke	Mathematics
James	Comp Sci`
Evan	Comp Sci
Baxter	English

Department	Advisor
Comp Sci	Smith
Chemistry	Turner
Physics	Bosky
Decision Sci	Hall
Mathematics	James
Comp Sci`	Clark
English	Bronte

5 NF Example

- The join of these decomposed relations contains tuples that did not exist in the original relation(Shown as highlighted rows). The decomposition is called lossy.

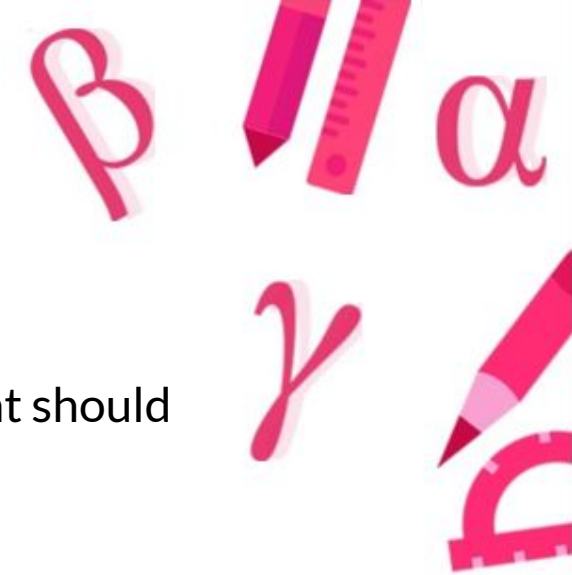
Name	Department	Advisor
Jones	Comp Sci	Smith
Jones	Comp Sci	Clark
Ng	Chemistry	Turner
Martin	Physics	Bosky
Dulles	Decision Sci	Hall
Duke	Mathematics	James
James	Comp Sci	Smith
James	Comp Sci	Clark
Evan	Comp Sci	Smith
Evan	Comp Sci	Clark
Baxter	English	Bronte

5 NF Example

Therefore we need to divide the original relation STUDENT_ADVISOR into three relations:

- Relation1 : Name , Department
- Relation 2: Name, Advisor
- Relation 3 : Department , Advisor

6 NF
Domain Key Normal Form (DKNF)



- A relation is said to be in DKNF if all constraints and dependencies that should hold on the relation
- Key" means Primary Key
- "Domain" means the set of definitions of the contents of attributes
- "Constraint" means any rule dealing with attributes

De-Normalization

What Is Denormalization?

- Denormalization is a database optimization technique in which we add redundant data to one or more tables.
- This can help us avoid costly joins in a relational database.
- Denormalization does not mean not doing normalization.
- It is an optimization technique that is applied after doing normalization.
- In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data.

De-Normalization

Why To Denormalize A Database ?

- To enhance query performance
- To make a database more convenient to manage
- To facilitate and accelerate reporting

De-Normalization

Pros of Denormalization:-

- Retrieving data is faster since we do fewer joins
- Queries to retrieve can be simpler (since we need to look at fewer tables)

Cons of Denormalization:-

- Updates and inserts are more expensive.
- Denormalization can make update and insert code harder to write.
- Data may be inconsistent . Which is the “correct” value for a piece of data?
- Data redundancy necessitates more storage.

Normalization Vs De-Normalization

Normalization	De-Normalization
• Normalization is the process of dividing the data into multiple tables	• De-Normalization is the process , where the data from multiple tables are combined into one table, for faster data retrieval
• It removes data redundancy	• It creates data redundancy i.e. duplicate data may be found in the same table.
• It maintains data integrity	• It may not retain the data integrity.
• It increases the number of tables in the database, therefore joins required to get the result.	• It reduces the number of tables, therefore less joins are required and query performance is faster.
• Inserts, updates and deletes are more efficient , reduces data loss and maintains data integrity.	• In this case all the duplicate data are at single table , extra effort required to maintain data integrity.
• Normalized database is required when the application needs more number of insert/update/delete operations.	• Use de-normalization database when application frequently execute queries based on joins

THANK YOU!