



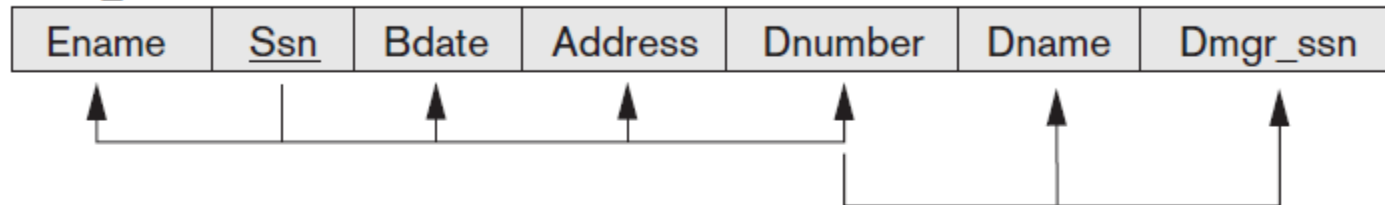
## **Basics of Functional Dependencies and Normalization for Relational Databases**

# Informal Design Guidelines for Relational Databases

- 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?

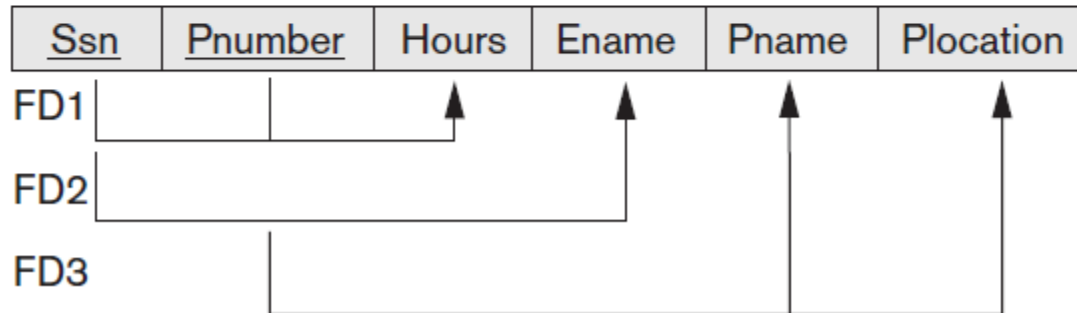
(a)

EMP\_DEPT



(b)

EMP\_PROJ



# Semantics of the Relational Attributes must be clear

- **GUIDELINE 1: 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.

# A simplified COMPANY relational database schema

EMPLOYEE

Ename	<u>Ssn</u>	Bdate	Address	Dnumber
-------	------------	-------	---------	---------

P.K.

F.K.

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn
-------	----------------	----------

P.K.

F.K.

DEPT\_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

P.K.

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

P.K.

F.K.

WORKS\_ON

<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------

P.K.

F.K.

F.K.

# Redundant Information in Tuples and Update Anomalies

- Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
    - Insertion anomalies
    - Deletion anomalies
    - Modification anomalies

## 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

## 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” may cause this update to 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 an he/she is assigned to a project.

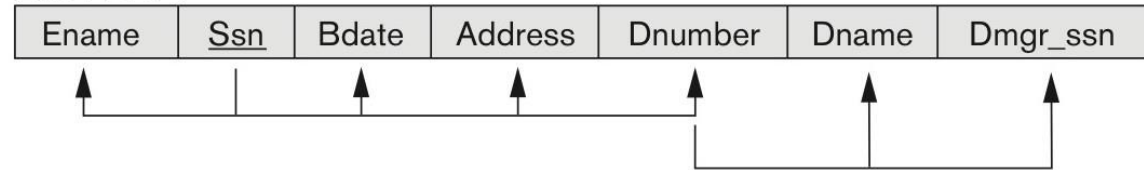
# EXAMPLE OF A DELETE ANOMALY

- 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.

# Two relation schemas suffering from update anomalies

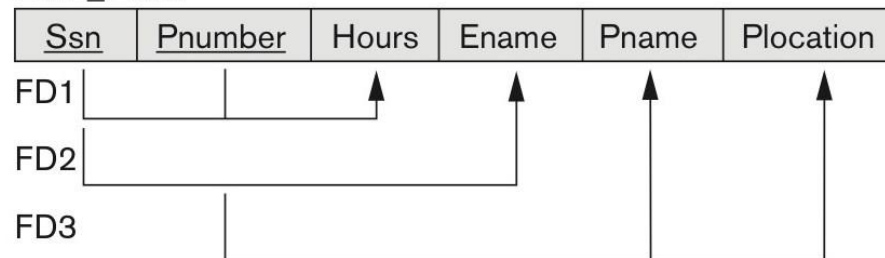
(a)

**EMP\_DEPT**



(b)

**EMP\_PROJ**



# 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.

# 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

# Generation of Spurious Tuples – avoid at any cost

- 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
- **GUIDELINE 4:**
  - 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.

(a)

EMP\_LOCS

<u>Ename</u>	<u>Plocation</u>
--------------	------------------

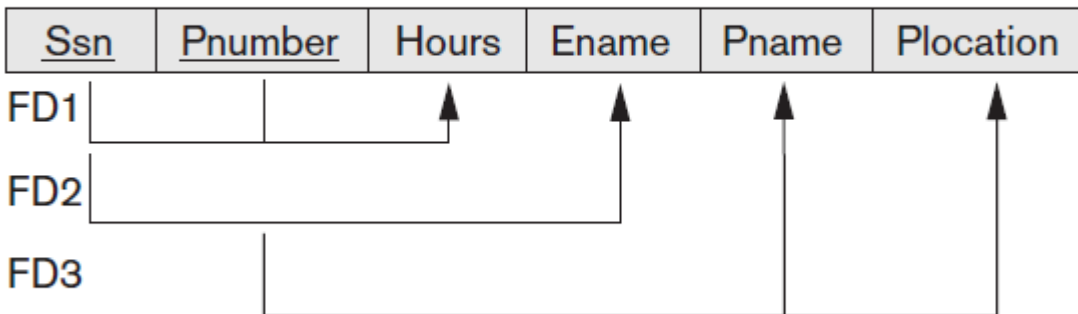
P.K.

EMP\_PROJ1

<u>Ssn</u>	<u>Pnumber</u>	Hours	Pname	Plocation
------------	----------------	-------	-------	-----------

P.K.

EMP\_PROJ



(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.
666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
453453453	1	20.0	ProductX	Bellaire	Smith, John B.
453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
453453453	2	20.0	ProductY	Sugarland	Smith, John B.
453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
333445555	2	10.0	ProductY	Sugarland	Smith, John B.
333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.



# Summary

- Anomalies that cause redundant work to be done during insertion into and modification of a relation, and that may cause accidental loss of information during a deletion from a relation
- Waste of storage space due to NULLs and the difficulty of performing selections, aggregation operations, and joins due to NULL values
- Generation of invalid and spurious data during joins on base relations with matched attributes that may not represent a proper (foreign key, primary key) relationship



## Functional Dependency

## 2. Functional Dependencies

- Functional dependencies (FDs)
  - Are used to specify *formal measures* of the "goodness" of relational designs
  - And keys are used to define **normal forms** for relations
  - 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  $Y$
- Relation extension  $r(R)$  that satisfies the functional dependency constraint are called **legal relation states**.

# Functional Dependencies

## Functional Dependencies

We say an attribute, B, has a *functional dependency* on another attribute, A, if for any two records, which have the same value for A, then the values for B in these two records must be the same. We illustrate this as:

$$A \rightarrow B$$

**Example:** Suppose we keep track of employee email addresses, and we only track one email address for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of email address on employee number:

$$\text{employee number} \rightarrow \text{email address}$$

**Example:** We only track one name for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of name on employee number:

$$\text{employee number} \rightarrow \text{employee name}$$

# Functional Dependencies

<u>EmpNum</u>	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

If EmpNum is the PK then the FDs:

EmpNum  $\rightarrow$  EmpEmail

EmpNum  $\rightarrow$  EmpFname

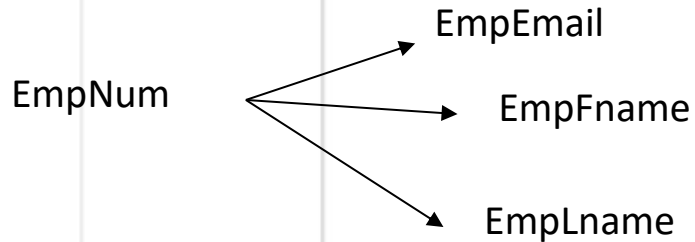
EmpNum  $\rightarrow$  EmpLname

must exist.

# Functional Dependencies

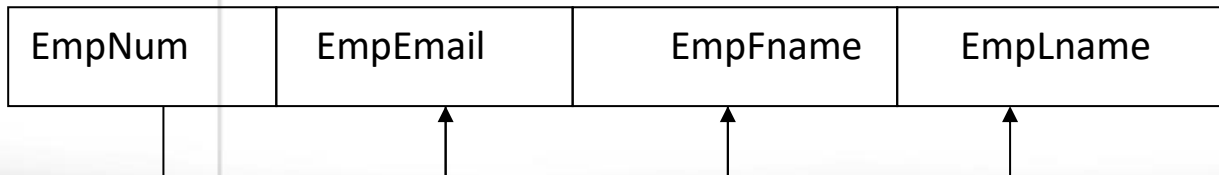
EmpNum  $\rightarrow$  EmpEmail  
EmpNum  $\rightarrow$  EmpFname  
EmpNum  $\rightarrow$  EmpLname

*3 different ways  
you might see FDs  
depicted*



Attribute on the LHS is known as the **determinant**

- EmpNum is a determinant of EmpEmail



ID	Name	DName	DBuilding
1	Javeria	CS	1
2	Arwa	AI	2
3	Mustafa	CS	1
4	Mustafa	AI	2
5	Rubab	EE	3
6	Waqas	SS	3

# Examples of FD constraints

1. Identify if F.D exists based on ID.

ID-→{Name, Dname,Dbuilding}

3-→{Mustafa, CS,1}

Valid F.D

Determinant	Dependant
1	a
2	b
Valid F.D	



# Examples of FD constraints

2. Identify the department building based on department name

Dname->Dbuilding

CS->{1}

Valid F.D

Determinant	Dependant
1	a
1	a
Valid F.D	

# Examples of FD constraints

## 3. Identify department building using department name

Dname->Dbuilding

EE->{3}  
SS->{3}

Valid F.D

Determinant	Dependant
1	a
2	a
Valid F.D	

# Examples of FD constraints

## 4. Student name identify department name

name  $\rightarrow$  Dname

Mustafa  $\rightarrow$  CS  
Mustafa  $\rightarrow$  AI

Invalid F.D

Determinant	Dependant
1	a
1	b
In Valid F.D	

# Examples of FD constraints

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

# Examples of FD constraints

- An FD is a property of the attributes in the schema  $R$
- The constraint must hold on *every* relation instance  $r(R)$
- If  $K$  is a key of  $R$ , then  $K$  functionally determines all attributes in  $R$ 
  - (since we never have two distinct tuples with  $t_1[K]=t_2[K]$ )

# 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.

# Ruling Out FDs

Note that given the state of the TEACH relation, we can say that the FD: Text  $\rightarrow$  Course may exist. However, the FDs Teacher  $\rightarrow$  Course, Teacher  $\rightarrow$  Text and Course  $\rightarrow$  Text are ruled out.

TEACH

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

# Functional Dependencies

- **Partial Dependency** – when an **non-key attribute** is determined by a part, but not the whole, of a **COMPOSITE** primary key (The Primary Key must be a Composite Key).
- A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.

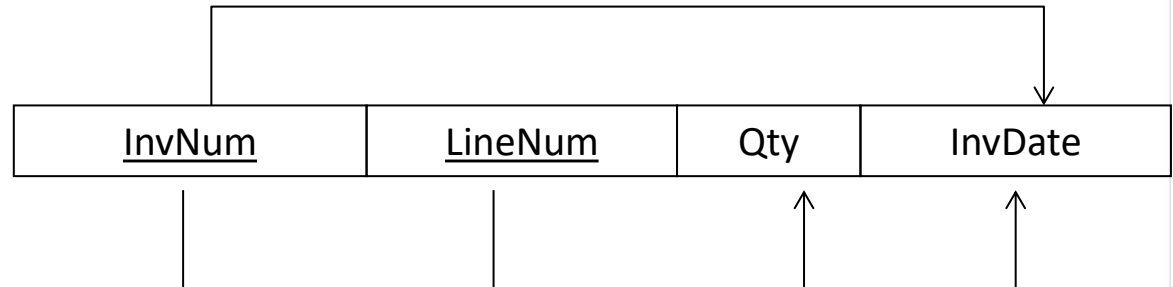


## CUSTOMER

Partial  
Dependency

<u>Cust_ID</u>	Name	<u>Order_ID</u>
101	AT&T	1234
101	AT&T	156
125	Cisco	1250

Cust\_ID → Name



Candidate keys: {InvNum, LineNum} InvDate is *partially dependent* on {InvNum, LineNum} as **InvNum is a determinant of InvDate and InvNum is part of a candidate key**

# Transitive dependency

- **Transitive Dependency** – when a non-key attribute determines another non-key attribute.
- The primary key is a determinant for another attribute which is determinant for a third attribute.
- Consider attributes A, B, and C, and where:

$$A \rightarrow B \text{ and } B \rightarrow C.$$

- Functional dependencies are transitive, which means that we also have the functional dependency

$$A \rightarrow C$$

- We say that C is transitively dependent on A through B.

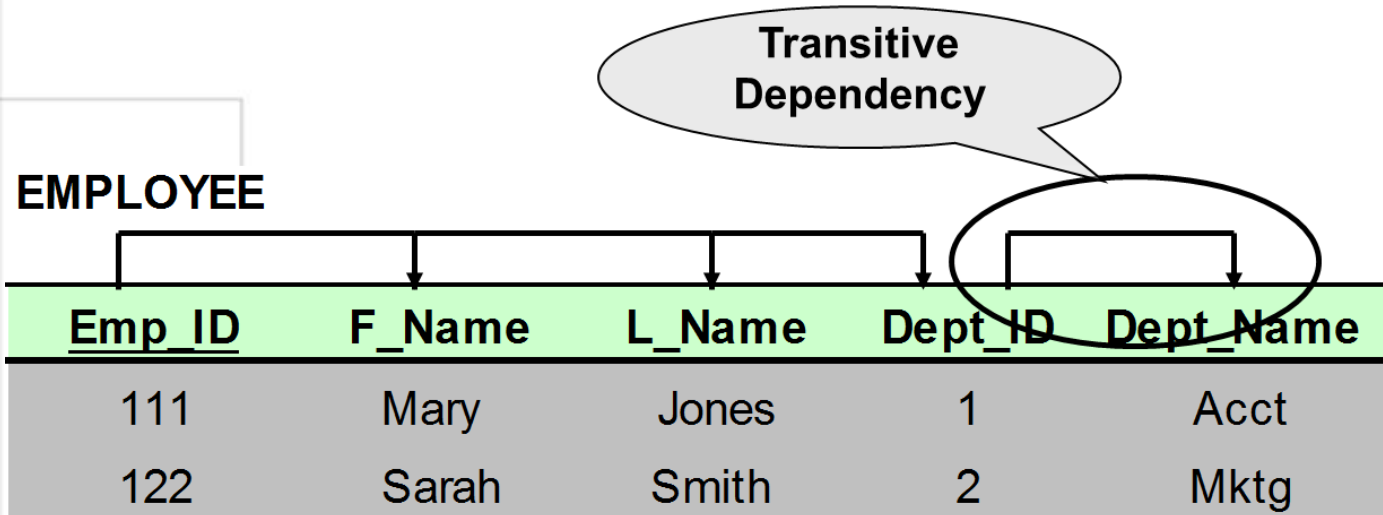
# Transitive Functional Dependency Example

- Consider the functional dependencies:
- Orderid->order\_date, customer\_id,name,address
- Order\_id-> quantity
- Product\_id-> description, finish, price
- customer\_id-> name,address

A->B->C

Order\_id->customer\_id->name,address

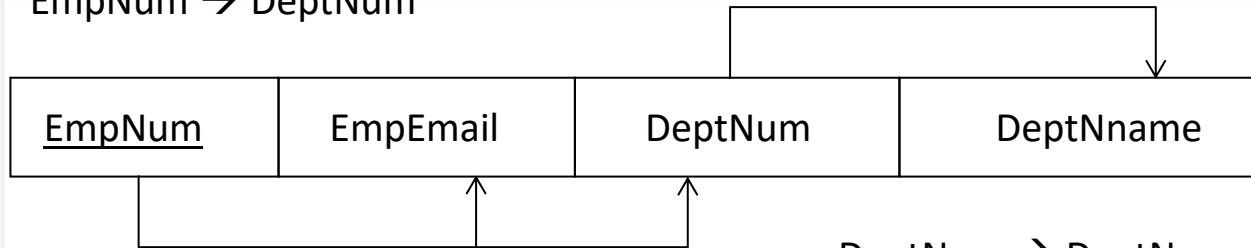
# Functional Dependencies



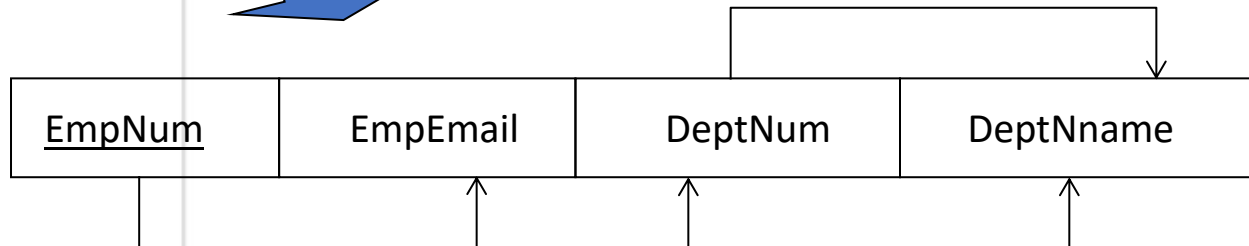
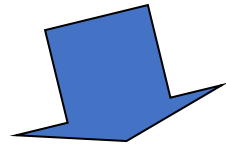
Dept\_ID → Dept\_Name

# Transitive dependency

$\text{EmpNum} \rightarrow \text{DeptNum}$



$\text{DeptNum} \rightarrow \text{DeptName}$



DeptName is *transitively dependent* on EmpNum via DeptNum

$\text{EmpNum} \rightarrow \text{DeptName}$

# Example

- Identify any transitive functional dependencies and partial functional dependencies in the following functional dependencies.
- Shipment\_id  $\rightarrow$  shipment\_date, origin, destination, ship\_id, ship\_name, captain\_id, captain\_name.
- Item\_id  $\rightarrow$  description, weight
- Ship\_id  $\rightarrow$  ship\_name, captain\_id, captain\_name
- Captain\_id  $\rightarrow$  captain\_name
- Shipment\_id, item\_id  $\rightarrow$  quantity.

# Full Functional dependency

- Indicates that if A and B are attributes of a relation, B is fully functionally dependent on A if B is functionally dependent on A, but not on any proper subset of A.
- $\{\text{StaffNo}, \text{StaffName}\} \rightarrow \text{BranchNo}$
- $\text{StaffNo} \rightarrow \text{BranchNo}$



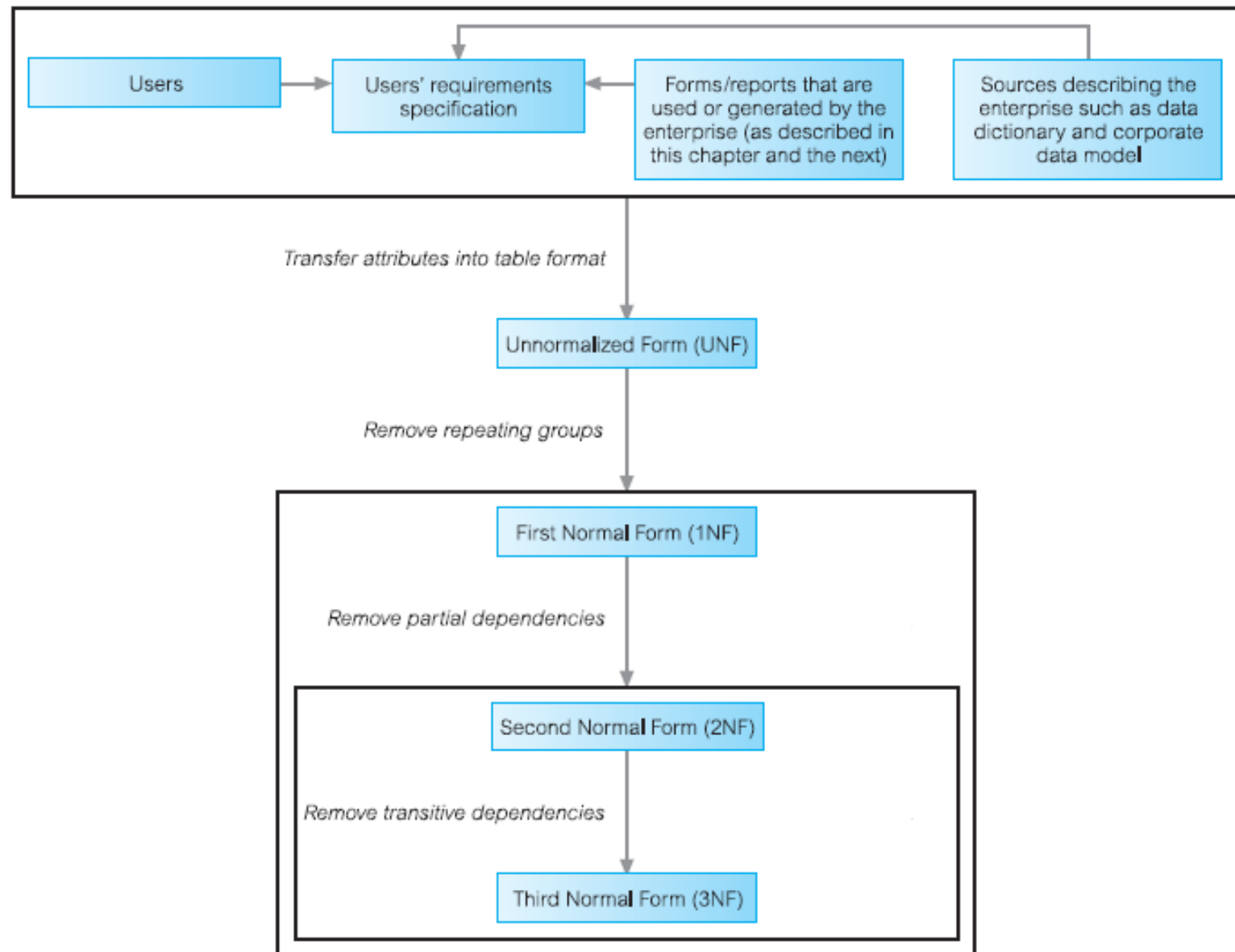
# Normalization



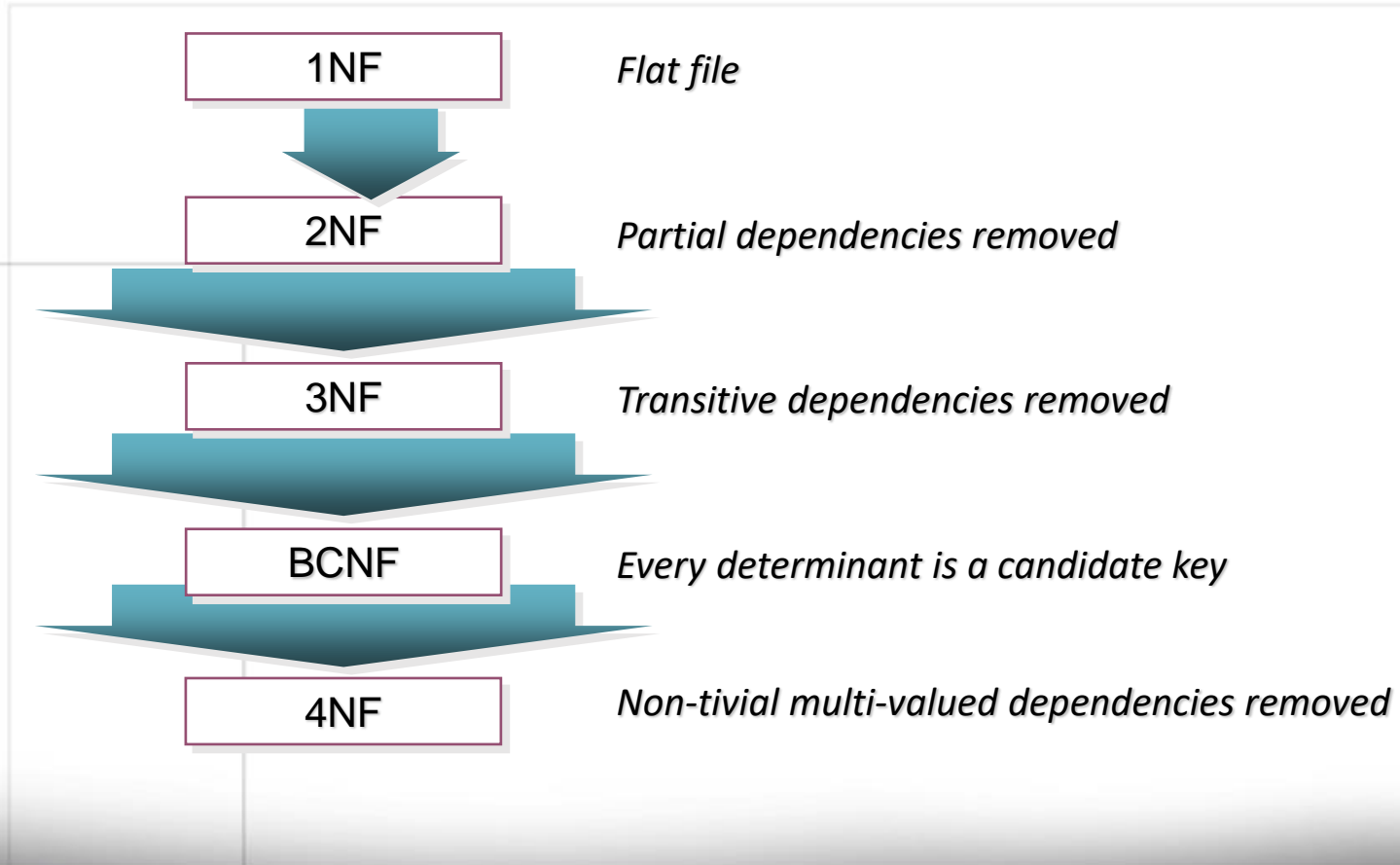
# Database Tables and Normalization

- **Normalization** is a process for assigning attributes to entities. It reduces data redundancies and helps eliminate the data anomalies.
- The normalization procedure provides database designer with the followings:
  - A formal framework for analyzing relations schema based on their keys and on the functional dependencies among their attribute.
  - A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalized to any desired degree.
- Normalization works through a series of stages called normal forms:
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
  - Fourth normal form (4NF)
  - Fifth normal form (5NF)
- The highest level of normalization is not always desirable.

## Data sources



# Normalization



# Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema  $R = \{A_1, A_2, \dots, A_n\}$  is a set of attributes  $S$  *subset-of*  $R$  with the property that no two tuples  $t_1$  and  $t_2$  in any legal relation state  $r$  of  $R$  will have  $t_1[S] = t_2[S]$
- 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.

# Definitions of Keys and Attributes Participating in Keys

- 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.

# First Normal Form

- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic

title	author_list	date	keyword_list
		day month year	
salesplan	{Smith, Jones}	1 April 89	{profit, strategy}
stat. report	{Jones, Frick}	17 July 94	{profit, personnel}

- 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

```
graph TD; A[Table/Relation] --> B[1st normal form]; A --> C[Not in first normal form];
```

Table/Relation

1<sup>st</sup> normal form

Must not contain composite  
attribute

Must not contain multi-value  
attribute

Not in first normal  
form

# Scenario

**A few employees works for one project.**

**Employee Num :**  
**101, 102, 103,**  
**105**

**Project Num : 15**

**Project Name :**  
**Evergreen**





# Sample Form

**Project Num : 15**

**Project Name : Evergreen**



Emp Num	Emp Name	Job Class	Chr Hours	Hrs Billed	Total
101					
102					
103					
105					

# Sample Report Layout

PROJ. NUM.	PROJECT NAME	EMPLOYEE NUMBER	EMPLOYEE NAME	JOB CLASS.	CHG/ HOUR	HOURS BILLED	TOTAL CHARGE
15	Evergreen	103	June E.Arbough	Elec. Engineer	\$84.50	23.8	\$2,011.10
		101	John G. News	Database Designer	\$105.00	19.4	\$2,037.00
		105	Alice K. Johnson *	Database Designer	\$105.00	35.7	\$3,748.50
		106	William Smithfield	Programmer	\$35.75	12.6	\$450.45
		102	David H. Senior	Systems Analyst	\$96.75	23.8	\$2,302.65
Subtotal							\$10,549.70
18	Amber Wave	114	Annelise Jones	Applications Designer	\$48.10	24.6	\$1,183.26
		118	James J. Frommer	General Support	\$18.36	45.3	\$831.71
		104	Anne K. Ramoras *	Systems Analyst	\$96.75	32.4	\$3,134.70
		112	Darlene M. Smithson	DSS Analyst	\$45.95	44.0	\$2,021.80
Subtotal							\$7,171.47
22		105	Alice K. Johnson	Database Designer	\$105.00	64.7	\$6,793.50
		104	Anne K. Ramoras	Systems Analyst	\$96.75	48.4	\$4,682.70
		113	Delbert K. Joenbrood*	Applications Designer	\$48.10	23.6	\$1,135.16
		111	Geoff B.Wabash	Clerical Support	\$26.87	22.0	\$591.14
		106	William Smithfield	Programmer	\$35.75	12.8	\$457.60
Subtotal							\$13,660.10
25		107	Maria D.Alonzo	Programmer	\$35.75	24.6	\$879.45
		115	Travis B. Bawangi	Systems Analyst	\$96.75	45.8	\$4,431.15
		101	John G. News *	Database Designer	\$105.00	56.3	\$5,911.50
		114	Annelise Jones	Applications Designer	\$48.10	33.1	\$1,592.11
		108	Ralph B.Washington	Systems Analyst	\$96.75	23.6	\$2,283.30
		118	James J. Frommer	General Support	\$18.36	30.5	\$559.98
		112	Darlene M. Smithson	DSS Analyst	\$45.95	41.4	\$1,902.33
Subtotal							\$17,559.82
Note: * indicates project leader				Total	48,941.09		

Note: \* indicates project leader

# Table Structure Matches the Report Format

	PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
► 15		Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8
			101	John G. News	Database Designer	\$105.00	19.4
			105	Alice K. Johnson *	Database Designer	\$105.00	35.7
			106	William Smithfield	Programmer	\$35.75	12.6
			102	David H. Senior	Systems Analyst	\$96.75	23.8
18		Amber Wave	114	Annelise Jones	Applications Designer	\$48.10	24.6
			118	James J. Frommer	General Support	\$18.36	45.3
			104	Anne K. Ramoras *	Systems Analyst	\$96.75	32.4
			112	Darlene M. Smithson	DSS Analyst	\$45.95	44.0
22		Rolling Tide	105	Alice K. Johnson	Database Designer	\$105.00	64.7
			104	Anne K. Ramoras	Systems Analyst	\$96.75	48.4
			113	Delbert K. Joenbrood *	Applications Designer	\$48.10	23.6
			111	Geoff B. Wabash	Clerical Support	\$26.87	22.0
			106	William Smithfield	Programmer	\$35.75	12.8
25		Starflight	107	Maria D. Alonzo	Programmer	\$35.75	24.6
			115	Travis B. Bawangi	Systems Analyst	\$96.75	45.8
			101	John G. News *	Database Designer	\$105.00	56.3
			114	Annelise Jones	Applications Designer	\$48.10	33.1
			108	Ralph B. Washington	Systems Analyst	\$96.75	23.6
			118	James J. Frommer	General Support	\$18.36	30.5
			112	Darlene M. Smithson	DSS Analyst	\$45.95	41.4

# Database Tables and Normalization

- Problems with the Figure at previous slide
- The project number is intended to be a primary key, but it contains nulls.
  - The table displays data redundancies.
  - The table entries invite data inconsistencies.
  - The data redundancies yield the following anomalies:
    - Update anomalies.
    - Addition anomalies.
    - Deletion anomalies.

# Database Tables and Normalization

- Conversion to First Normal Form
  - A relational table must not contain repeating groups.
  - Repeating groups can be eliminated by adding the appropriate entry in at least the primary key column(s).

	PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
▶	15	Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8
			101	John G. News	Database Designer	\$105.00	19.4
			105	Alice K. Johnson *	Database Designer	\$105.00	35.7
			106	William Smithfield	Programmer	\$35.75	12.6
			102	David H. Senior	Systems Analyst	\$96.75	23.8



# Data Organization: First Normal Form

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	03	June E. Arbough	Elect. Engineer	\$84.50	23.8
		01	John G. News	Database Designer	\$105.00	19.4
		05	Alice K. Johnson *	Database Designer	\$105.00	35.7
		06	William Smithfield	Programmer	\$35.75	12.6
		02	David H. Senior	Systems Analyst	\$96.75	23.9
18	Amber Wave	14	Annelise Jones	Applications Designer	\$48.10	24.6
		18	James J. Frommer	General Support	\$18.36	45.3
		04	Anne K. Ramoras *	Systems Analyst	\$96.75	32.4
		12	Darlene M. Smithson	DSS Analyst	\$45.95	44.0
22	Rolling Tide	05	Alice K. Johnson	Database Designer	\$105.00	64.7
		04	Anne K. Ramoras	Systems Analyst	\$96.75	48.4
		13	Delbert K. Joenbrood *	Applications Designer	\$48.10	23.6
		11	Geoff B. Wabash	Clerical Support	\$26.87	22.0
		06	William Smithfield	Programmer	\$35.75	12.8
25	Starflight	07	Maria D. Alonzo	Programmer	\$35.75	24.6
		15	Travis B. Bawangi	Systems Analyst	\$96.75	45.8
		01	John G. News *	Database Designer	\$105.00	56.3
		14	Annelise Jones	Applications Designer	\$48.10	33.1
		08	Ralph B. Washington	Systems Analyst	\$96.75	23.9
		18	James J. Frommer	General Support	\$18.36	30.5
		12	Darlene M. Smithson	DSS Analyst	\$45.95	41.4

Before

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8
15	Evergreen	101	John G. News	Database Designer	\$105.00	19.4
15	Evergreen	105	Alice K. Johnson *	Database Designer	\$105.00	35.7
15	Evergreen	106	William Smithfield	Programmer	\$35.75	12.5
15	Evergreen	102	David H. Senior	Systems Analyst	\$96.75	23.9
18	Amber Wave	114	Annelise Jones	Applications Designer	\$48.10	24.6
18	Amber Wave	118	James J. Frommer	General Support	\$18.36	45.3
18	Amber Wave	104	Anne K. Ramoras *	Systems Analyst	\$96.75	32.1
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	\$45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	\$105.00	64.7
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	\$96.75	48.9
22	Rolling Tide	113	Delbert K. Joenbrood *	Applications Designer	\$48.10	23.6
22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	\$26.87	22.5
22	Rolling Tide	106	William Smithfield	Programmer	\$35.75	12.1
25	Starflight	107	Maria D. Alonzo	Programmer	\$35.75	24.7
25	Starflight	115	Travis B. Bawangi	Systems Analyst	\$96.75	45.8
25	Starflight	101	John G. News *	Database Designer	\$105.00	56.3
25	Starflight	114	Annelise Jones	Applications Designer	\$48.10	33.1
25	Starflight	108	Ralph B. Washington	Systems Analyst	\$96.75	23.9
25	Starflight	118	James J. Frommer	General Support	\$18.36	30.2
25	Starflight	112	Darlene M. Smithson	DSS Analyst	\$45.95	41.4


After

# Normalization into 1NF

(a)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations



(b)

DEPARTMENT

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

(c)

DEPARTMENT

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

# Solution with respect to the above example

1. Remove the attribute Dlocation that violates 1NF and place it in a separate relation DEPT\_LOCATIONS along with the primary key Dnumber of DEPARTMENT. The primary key of this newly formed relation is the combination {Dnumber, Dlocation}. A distinct tuple in DEPT\_LOCATIONS exists for *each location* of a department. This decomposes the non-1NF relation into two 1NF relations.

## DEPT\_LOCATIONS

F.K.

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

P.K.



2. Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT. In this case, the primary key becomes the combination {Dnumber, Dlocation}. This solution has the disadvantage of introducing redundancy in the relation and hence is rarely adopted.

#### DEPARTMENT

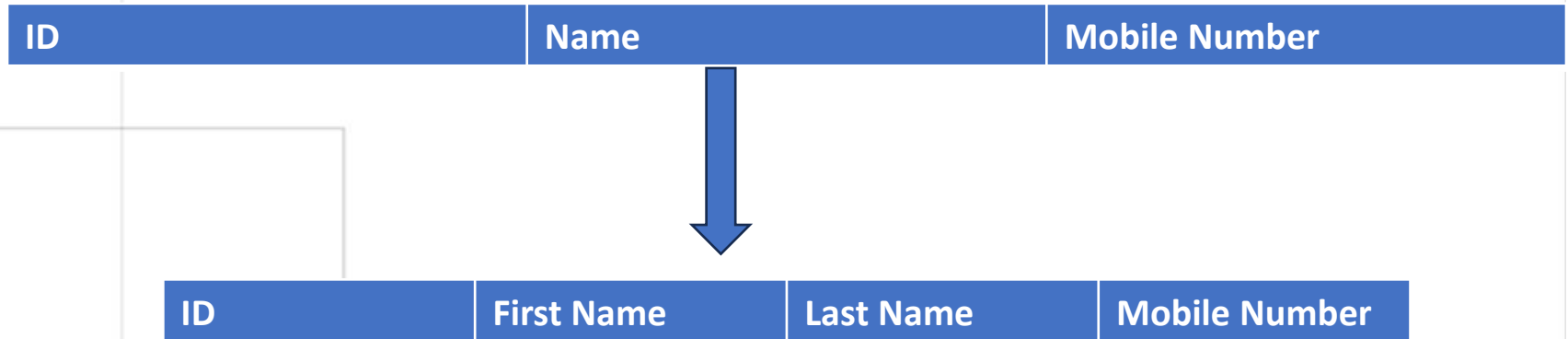
Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

3. If a maximum number of values is known for the attribute—for example, if it is known that at most three locations can exist for a department—replace the Dlocations attribute by three atomic attributes: Dlocation1, Dlocation2, and Dlocation3. This solution has the disadvantage of introducing NULL values if most departments have fewer than three locations. It further introduces spurious semantics about the ordering among the location values; that ordering is not originally intended. Querying on this attribute becomes more difficult; for example, consider how you would write the query: List the departments that have ‘Bellaire’ as one of their locations in this design. For all these reasons, it is best to avoid this alternative

ID	Name	Mobile Number
1	Javeria Farooq	123456789, 574898696
2	Mustafa Tauseef	11235667
3	Arwa Tauseef	37475859, 73475485

# What to do if table is not in First Normal Form

- Composite Attribute



# What to do if table is not in First Normal Form

- Multivalued Attribute

ID	Name	Mobile Number
1	Javeria Farooq	123456789, 574898696
2	Mustafa Tauseef	11235667
3	Arwa Tauseef	37475859, 73475485

ID	First Name	Last Name	Mobile Number	AlternateMobile
1	Javeria	Farooq	123456789	574898696
2	Mustafa	Tauseef	11235667	NULL
3	Arwa	Tauseef	37475859	73475485

ID	Name	Mobile Number
1	Javeria Farooq	123456789, 574898696
2	Mustafa Tauseef	11235667
3	Arwa Tauseef	37475859, 73475485
4	Tauseef	34567778

ID	Mobile Number	Employee ID
1	123456789	1
2	11235667	2
3	37475859	3
4	574898696	1
5	73475485	2

ID	Name
1	Javeria Farooq
2	Mustafa Tauseef
3	Arwa Tauseef
4	Tauseef

# Normalizing nested relations into 1NF

(a)

EMP\_PROJ

Ssn	Ename	Projs	
		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
-----	---------	-------

# First Normal Form (1 NF)

- **1NF Definition**

- The term first normal form (**1NF**) describes the tabular format in which:
  - All the key attributes are defined.
  - There are no repeating groups in the table.
  - All attributes are dependent on the primary key.



# Second Normal Form

- 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  $Y \rightarrow Z$  where removal of any attribute from Y means the FD does not hold any more
- Examples:
  - $\{SSN, PNUMBER\} \rightarrow HOURS$  is a full FD since neither  $SSN \rightarrow HOURS$  nor  $PNUMBER \rightarrow HOURS$  hold
  - $\{SSN, PNUMBER\} \rightarrow ENAME$  is not a full FD (it is called a partial dependency ) since  $SSN \rightarrow ENAME$  also holds

# Second Normal Form

- 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”

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Student\_ID-> {student\_ID, course\_ID, Course\_Fee}

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

No

Student\_ID-> {student\_ID, course\_ID, Course\_Fee}

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Student\_ID-> {student\_ID, course\_ID, Course\_Fee}

Course\_ID-> {student\_ID, course\_ID, Course\_Fee}

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Student\_ID-> {student\_ID, course\_ID, Course\_Fee}

Course\_ID-> {student\_ID, course\_ID, Course\_Fee}

No

No

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Student\_ID-> {student\_ID, course\_ID, Course\_Fee}

Course\_ID-> {student\_ID, course\_ID, Course\_Fee}

Course\_Fee-> {student\_ID, course\_ID, Course\_Fee}

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Student\_ID-> {student\_ID, course\_ID, Course\_Fee}

Course\_ID-> {student\_ID, course\_ID, Course\_Fee}

Course\_Fee-> {student\_ID, course\_ID, Course\_Fee}

No

No

No



Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

(Student\_ID, Course\_ID -> {student\_ID, course\_ID, Course\_Fee})

Student\_ID and course\_ID  
together form a candidate key

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Prime Attribute	Non-Prime Attribute
Student_ID	Course_Fee
Course_ID	

(Student\_ID, Course\_ID → {Course\_Fee})

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Prime Attribute	Non-Prime Attribute
Student_ID	Course_Fee
Course_ID	

$(\text{Student\_ID}, \text{Course\_ID} \rightarrow \{\text{Course\_Fee}\})$

$\text{Course\_ID} \rightarrow \{\text{Course\_Fee}\}$

Student_ID	Course_ID	Course_Fee
1	1	7500
2	1	7500
1	2	8500
2	2	8500

Courses Fees	
Course_ID	Course_Fee
1	7500
2	8500

Student Courses	
Student_ID	Course_ID
1	1
2	1
1	2
2	2

# Second Normal Form (2 NF)

A table is in 2NF if:

- It is in 1NF and
- It includes no partial dependencies; that is, no attribute is dependent on only a portion of the primary key.

(It is still possible for a table in 2NF to exhibit **transitive dependency**; that is, one or more attributes may be functionally dependent on nonkey attributes.)

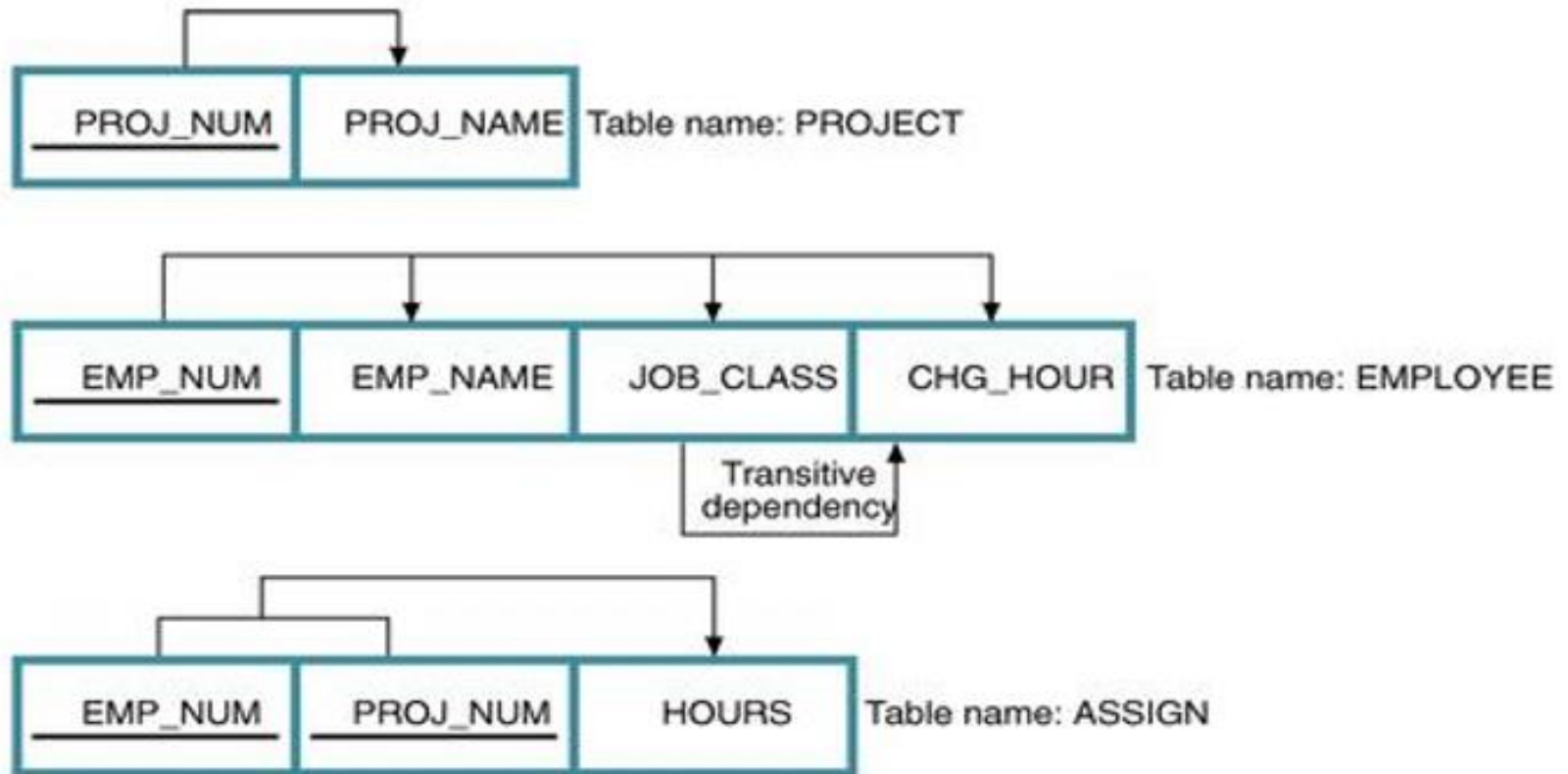
## 3.6 Third Normal Form

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

# Third Normal Form

- 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  $X \rightarrow Y$  and  $Y \rightarrow Z$ , 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,  $SSN \rightarrow Emp\# \rightarrow Salary$  and Emp# is a candidate key.

# Dependency Diagram





# Third Normal Form (3 NF)

- **Conversion to Third Normal Form**
  - Create a separate table with attributes in a transitive functional dependence relationship.

PROJECT (PROJ\_NUM, PROJ\_NAME)

ASSIGN (PROJ\_NUM, EMP\_NUM, HOURS)

EMPLOYEE (EMP\_NUM, EMP\_NAME, JOB\_CLASS)

JOB (JOB\_CLASS, CHG\_HOUR)

### Tournament winners

<u>Tournament</u>	<u>Year</u>	<u>Winner</u>	<u>Winner's date of birth</u>
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

(Tournament,Year)->{winner}  
Winner->{Winner's DateofBirth}

# Solution

**Tournament winners**

<u>Tournament</u>	<u>Year</u>	<u>Winner</u>
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

**Winner's dates of birth**

<u>Winner</u>	<u>Date of birth</u>
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

# Third Normal Form (3 NF)

- **3NF Definition**

- A table is in 3NF if:
  - It is in 2NF and
  - It contains no transitive dependencies.

# Normalization Stages

- Process involves applying a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form. When a test fails, the relation is decomposed into simpler relations that individually meet the normalization tests.
- The higher the normal form the less vulnerable to update anomalies the relations become.
- Three Normal forms: 1NF, 2NF and 3NF were initially proposed by Codd.
- All these normal forms are based on the functional dependencies among the attributes of a relation.
- Normalization follows a staged process that obeys a set of rules. The steps of normalization are:
  - **Step 1:** Select the data source and convert into an unnormalized table (UNF)
  - **Step 2:** Transform the unnormalized data into first normal form (1NF)
  - **Step 3:** Transform data in first normal form (1NF) into second normal form (2NF)
  - **Step 4:** Transform data in second normal form (2NF) into third normal form (3NF)
- Occasionally, the data may still be subject to anomalies in third normal form. In this case, we may have to perform further transformations.
- Transform third normal form to Boyce-Codd normal form (BCNF)
- Transform Boyce-Codd normal form to fourth normal form (4NF)
- Transform fourth normal form to fifth normal form (5NF)

# Normalization Example

- We will demonstrate the process of normalization (to 3NF) by use of an example. Normalization is a bottom-up technique for database design, normally based on an existing system (which may be paper-based). We start by analyzing the documentation, eg reports, screen layouts from that system. We will begin with the Project Management Report, which describes projects being worked upon by employees. This report is to be 'normalised'. Each of the first four normalization steps is explained.

Project Management Report				
Project Code:		PC010		
Project Title:		Pensions System		
Project Manager:		M Phillips		Project Budget:
£24,500				
Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
=====				
==				
S10001	A Smith	L004	IT	£22.00
S10030	L Jones	L023	Pensions	£18.50
S21010	P Lewis	L004	IT	£21.00
S00232	R Smith	L003	Programming	£26.00
=====				
Total Staff on Project: 4		Average Hourly Rate:		£21.88

Calculated Fields

# Step 1

- Select the data source (ie the report from the previous page) and convert into an unnormalized table (UNF). The process is as follows:
- Create column headings for the table for each data item on the report (**ignoring any calculated fields**). A calculated field is one that can be derived from other information on the form. In this case **total staff** and **average hourly rate**.
- Enter sample data into table. (This data is not simply the data on the report but a representative sample. In this example it shows several employees working on several projects. In this company the same employee can work on different projects and at a different hourly rate.)
- Identify a **key** for the table (and underline it).
- Remove duplicate data. (In this example, for the chosen key of Project Code, the values for Project Code, Project Title, Project Manager and Project Budget are duplicated if there are two or more employees working on the same project. **Project Code** chosen for the key and duplicate data, associated with each project code, is removed. Do not confuse duplicate data with repeating attributes which is described in the next step.

<b><u>Project Code</u></b>	<b><u>Project Title</u></b>	<b><u>Project Manager</u></b>	<b><u>Project Budget</u></b>	<b><u>Employee No.</u></b>	<b><u>Employee Name</u></b>	<b><u>Department No.</u></b>	<b><u>Department Name</u></b>	<b><u>Hourly Rate</u></b>
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50
PC045	Salaries System	H Martin	17400	S13210	W Richards	L008	Salary	17.00
PC064	HR System	K Lewis	12250	S31002	T Gilbert	L028	Database	23.25
PC064	HR System	K Lewis	12250	S21010	P Lewis	L004	IT	17.50
PC064	HR System	K Lewis	12250	S10034	B James	L009	HR	16.50



## Step 2

- Transform a table of unnormalized data into first normal form (1NF). any repeating attributes to a new table. A repeating attribute is a data field within the UNF relation that may occur with multiple values for a single value of the key. The process is as follows: Identify repeating attributes.
- Remove these repeating attributes to a new table together with a **copy** of the key from the UNF table.
- Assign a key to the new table (and underline it). The key from the original unnormalized table **always** becomes **part** of the key to the new table. A **compound key** is created. The value for this key must be unique for each entity occurrence.
- **Notes:**
- After removing the duplicate data, the repeating attributes are easily identified.
- In the previous table the Employee No, Employee Name, Department No, Department Name and Hourly Rate attributes are repeating. That is, there is potential for more than one occurrence of these attributes for each project code. These are the repeating attributes and have been to a new table together with a copy of the original key (ie: Project Code).
- A key of Project Code and Employee No has been defined for this new table. This combination is unique for each row in the table.

<b><u>Project Code</u></b>	<b><u>Project Title</u></b>	<b><u>Project Manager</u></b>	<b><u>Project Budget</u></b>
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

<b><u>Project Code</u></b>	<b><u>Employee No.</u></b>	<b><u>Employee Name</u></b>	<b><u>Department No.</u></b>	<b><u>Department Name</u></b>	<b><u>Hourly Rate</u></b>
PC010	S10001	A Smith	L004	IT	22.00
PC010	S10030	L Jones	L023	Pensions	18.50
PC010	S21010	P Lewis	L004	IT	21.00
PC045	S10010	B Jones	L004	IT	21.75
PC045	S10001	A Smith	L004	IT	18.00
PC045	S31002	T Gilbert	L028	Database	25.50
PC045	S13210	W Richards	L008	Salary	17.00
PC064	S31002	T Gilbert	L028	Database	23.25
PC064	S21010	P Lewis	L004	IT	17.50
PC064	S10034	B James	L009	HR	16.50

# Step 3

- Transform 1NF data into second normal form (2NF). Remove any -key attributes (partial Dependencies) that only depend on part of the table key to a new table. What must be determined "is field A dependent upon field B or vice versa?" This means: "Given a value for A, do we then have only one possible value for B, and vice versa?" If the answer is yes, A and B should be put into a new relation with A becoming the primary key. A should be left in the original relation and marked as a foreign key.
- Ignore tables with (a) a simple key or (b) with no non-key attributes (these go straight to 2NF with no conversion).
- The process is as follows:
  - Take each non-key attribute in turn and ask the question: is this attribute dependent on **one part** of the key?
  - If yes, remove the attribute to a new table with a **copy** of the **part** of the key it is dependent upon. The key it is dependent upon becomes the key in the new table. Underline the key in this new table.
  - If no, check against other part of the key and repeat above process
  - If still no, i.e.: not dependent on either part of the key, keep attribute in current table.
- **Notes:**
  - The first table went straight to 2NF as it has a simple key (Project Code).
  - Employee name, Department No and Department Name are dependent upon Employee No only. Therefore, they were moved to a new table with Employee No being the key.
  - However, Hourly Rate is dependent upon both Project Code and Employee No as an employee may have a different hourly rate depending upon which project they are working on. Therefore, it remained in the original table.

<u>Project Code</u>	Project Title	Project Manager	Project Budget
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

<u>Project Code</u>	Employee No.	Hourly Rate		<u>Employee No.</u>	Employee Name	Department No.	Department Name
PC010	S10001	22.00		S10001	A Smith	L004	IT
PC010	S10030	18.50		S10030	L Jones	L023	Pensions
PC010	S21010	21.00		S21010	P Lewis	L004	IT
PC045	S10010	21.75		S10010	B Jones	L004	IT
PC045	S10001	18.00		S31002	T Gilbert	L028	Database
PC045	S31002	25.50		S13210	W Richards	L008	Salary
PC045	S13210	17.00		S10034	B James	L009	HR
PC064	S31002	23.25					
PC064	S21010	17.50					
PC064	S10034	16.50					

# Step 4

- Data in second normal form (2NF) into third normal form (3NF). Remove to a new table any non-key attributes that are more dependent on other non-key attributes than the table key.
- What must be determined is "is field A dependent upon field B or vice versa?" This means: "Given a value for A, do we then have only one possible value for B, and vice versa?" If the answer is yes, then A and B should be put into a new relation, with A becoming the primary key. A should be left in the original relation and marked as a foreign key.
- Ignore tables with zero or only one non-key attribute (these go straight to 3NF with no conversion).
- The process is as follows: If a non-key attribute is more dependent on another non-key attribute than the table key:
- Move the **dependent** attribute, together with a **copy** of the non-key attribute upon which it is dependent, to a new table.
- Make the non-key attribute, upon which it is dependent, the key in the new table. Underline the key in this new table.
- **Leave** the non-key attribute, upon which it is dependent, in the original table and mark it a **foreign key (\*)**.
- **Notes:**
- The project team table went straight from 2NF to 3NF as it only has one non-key attribute.
- Department Name is more dependent upon Department No than Employee No and therefore was moved to a new table. Department No is the key in this new table and a foreign key in the Employee table.

<u>Project Code</u>	<u>Project Title</u>	<u>Project Manager</u>	<u>Project Budget</u>
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

<u>Department No.</u>	<u>Department Name</u>
L004	IT
L023	Pensions
L004	IT
L004	IT
L028	Database
L008	Salary
L009	HR

<u>Project Code</u>	<u>Employee No.</u>	<u>Hourly Rate</u>
PC010	S10001	22.00
PC010	S10030	18.50
PC010	S21010	21.00
PC045	S10010	21.75
PC045	S10001	18.00
PC045	S31002	25.50
PC045	S13210	17.00
064	S31002	23.25
PC064	S21010	17.50
PC064	S10034	16.50

<u>Employee No.</u>	<u>Employee Name</u>	<u>Department No.</u>
S10001	A Smith	L004
S10030	L Jones	L023
S21010	P Lewis	L004
S10010	B Jones	L004
S31002	T Gilbert	L028
S13210	W Richards	L008
S10034	B James	L009

3NF Tables: Non-Key Dependencies Removed

# Summary of Normalization Rules

- That is the complete process. Having started off with an unnormalized table we finished with four normalized tables in 3NF. You will notice that duplication has been removed (apart from the keys needed to establish the links between those tables).
- The process may look complicated. However, if you follow the rules **completely**, and **do not** miss out any steps, then you should arrive at the correct solution. If you omit a rule there is a high probability that you will end up with too few tables or incorrect keys.
- The following normal forms were discussed in this section:
- **First normal form:** A table is in the first normal form if it contains no repeating columns.
- **Second normal form:** A table is in the second normal form if it is in the first normal form and contains only columns that are dependent overall (primary) key.
- **Third normal form:** A table is in the third normal form if it is in the second normal form and all the non-key columns are dependent only on the primary key. If the value of a non-key column is dependent on the value of another non-key column we have a situation known as transitive dependency. This can be resolved by removing the columns dependent on non-key items to another table.

# Boyce-Codd Normal Form (BCNF)

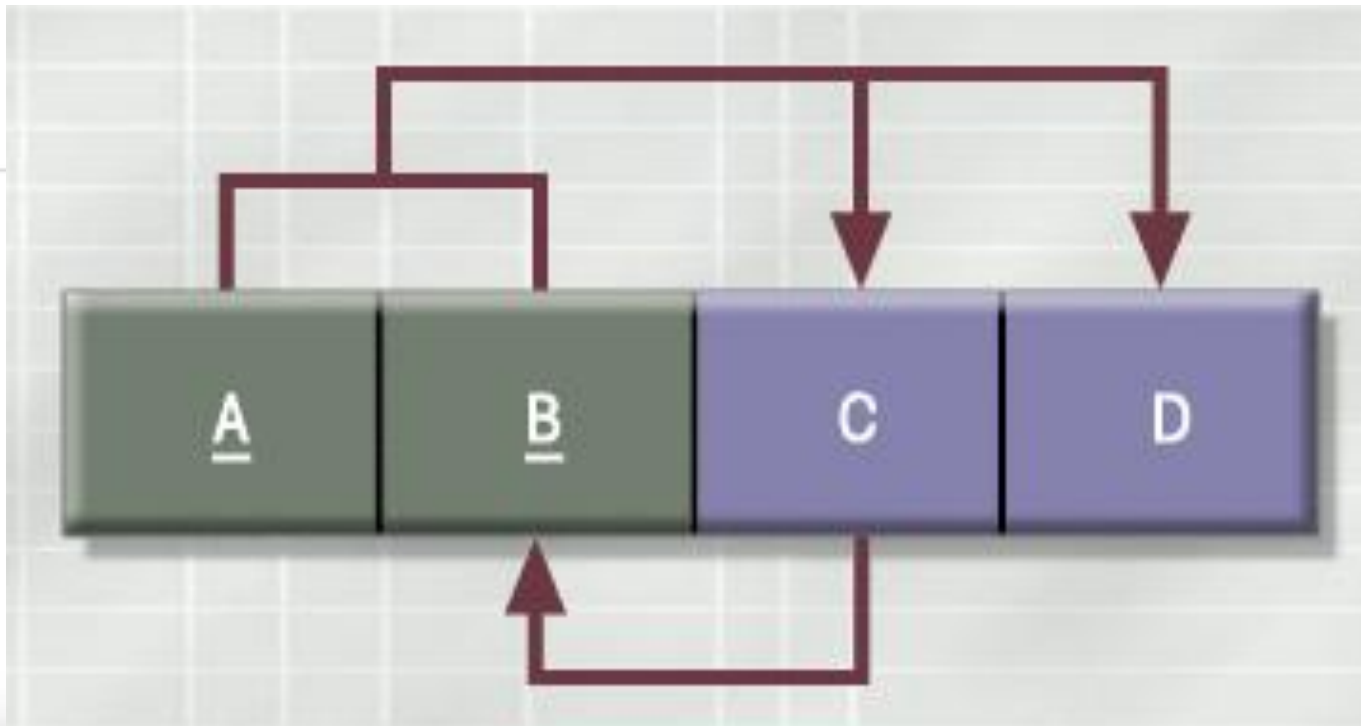
- A table is in **Boyce-Codd normal form (BCNF)** if every determinant in the table is a candidate key.

(A determinant is any attribute whose value determines other values with a row.)

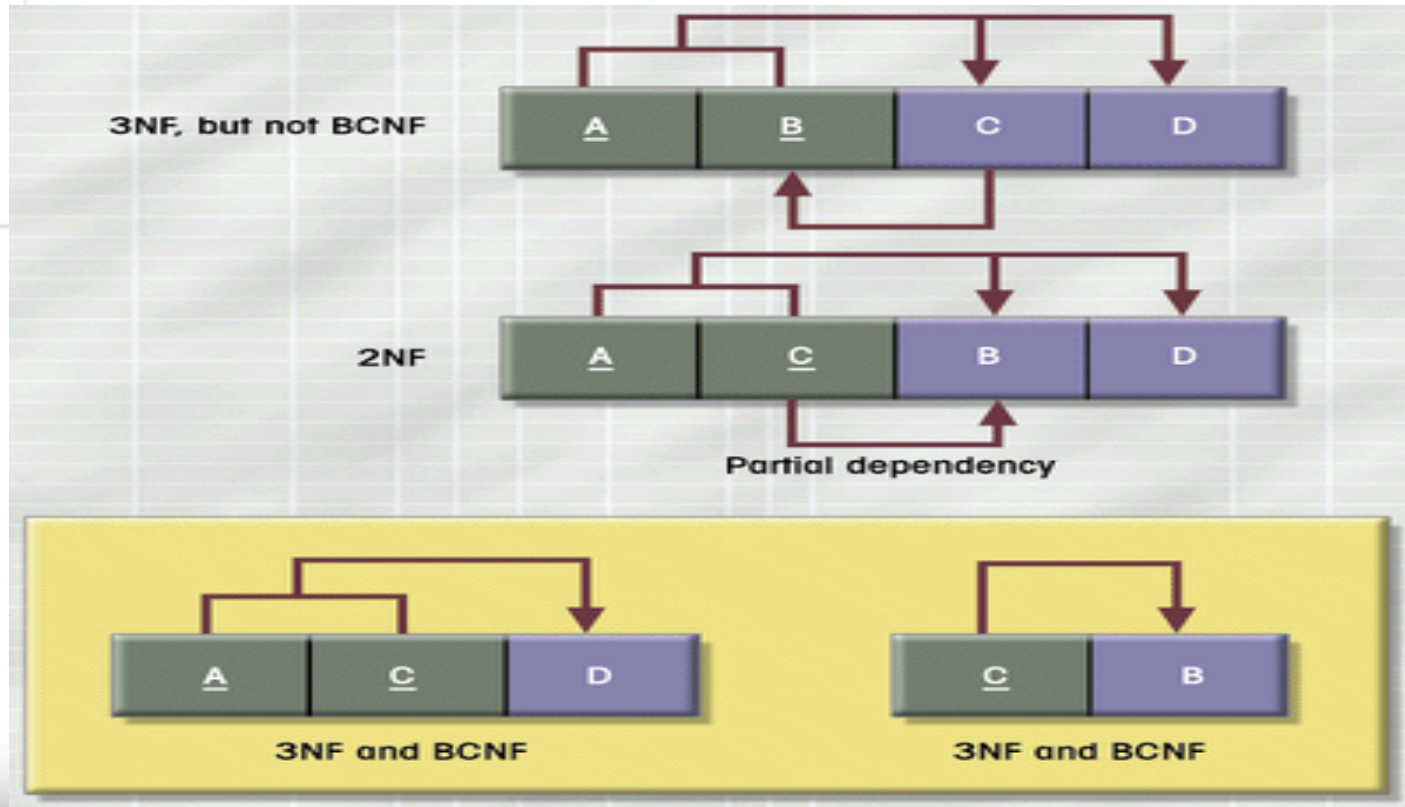
- If a table contains only one candidate key, the 3NF and the BCNF are equivalent.
- BCNF is a special case of 3NF.



# A Table That Is In 3NF But Not In BCNF



# The Decomposition of a Table Structure to Meet BCNF Requirements

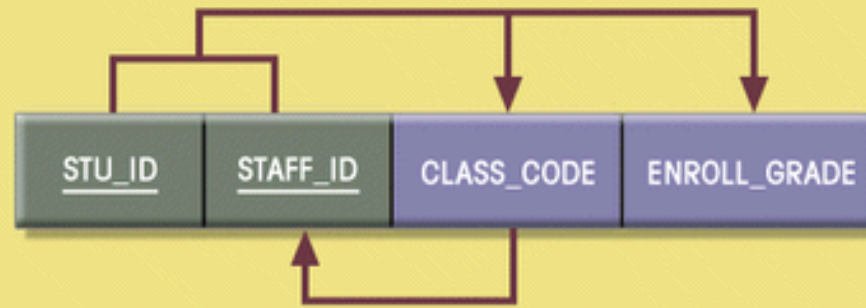


# Sample Data for a BCNF Conversion

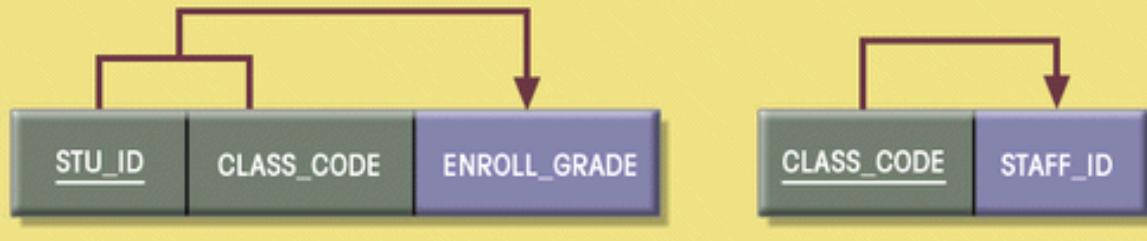
STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

# Decomposition into BCNF

Panel A: 3NF, but not BCNF



Panel B: 3NF and BCNF



Student	Subject	Teacher
Shahyan	Database	Javeria
Shahyan	SDA	Rubab
Mustafa	Database	Javeria
Mustafa	SDA	Abdul Rahman

Functional Dependencies:  
Student,Subject-> Teacher  
Teacher->Subject

Checking transitive dependencies  
Student,Subject-> Teacher  
Teacher->Subject

No Transitive dependency

No Transitive dependency

Every FD in the table is of form X determines Y where X or LHS is strictly A candidate key or a Super Key

Checking transitive dependencies

X  $\rightarrow$  Y  
Candidate or super key

Student,Subject  $\rightarrow$  Teacher

Verifies the condition of BCNF

Teacher  $\rightarrow$  Subject

Doesn't verify the condition of BCNF

Student	Subject	Teacher
Shahyan	Database	Javeria
Shahyan	SDA	Rubab
Mustafa	Database	Javeria
Mustafa	SDA	Abdul Rahman

Candidate key: Student,Subject

Functional Dependencies:  
Student,Subject-> Teacher  
Teacher->Subject

Student (PK)	Teacher (FK)
Shahyan	Javeria
Shahyan	Rubab
Mustafa	Javeria
Mustafa	Abdul Rahman

Teacher (PK)	Subject
Javeria	Database
Rubab	SDA
Javeria	Database
Abdul Rahman	SDA

# BCNF (Activity)

- Convert the following table into BCNF

<u>S_Num</u>	<u>T_Code</u>	Offering#	Review Date
123599	FIT104	01764	2nd March
123599	PIT305	01765	12th April
123599	PIT107	01789	2nd May
346700	FIT104	01764	3rd March
346700	PIT305	01765	7th May



## StudentReview

<u>S_Num</u>	<u>Offering#</u>	<u>Review Date</u>
123599	01764	2nd March
123599	01765	12th April
123599	01789	2nd May
346700	01764	3rd March
346700	01765	7th May

## OfferingTeacher

<u>Offering#</u>	<u>T Code</u>
01764	FIT104
01765	PIT305
01789	PIT107

# BCNF Definition

- **BCNF Definition**
  - A table is in BCNF if every determinant in that table is a candidate key. If a table contains only one candidate key, 3NF and BCNF are equivalent.

Take the following table.

StudentID is the primary key.

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	<u>10 Charles Street</u>	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+

Is it 1NF?

No. There are repeating groups (subject, subjectcost, grade)

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	<u>10 Charles Street</u>	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+

How can you make it 1NF?

Create new rows so each cell contains only one value

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+



StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

But now look – is the *studentID* primary key still valid?

No – the studentID no longer uniquely identifies each row

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

You now need to declare *studentID* **and** *subject* **together** to uniquely identify each row.

So the new **key** is StudentID *and* Subject.

So. We now have 1NF.

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

Is it 2NF?

**Studentname** and **address** are dependent on **studentID** (which is part of the key)  
This is good.

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

But they are **not** dependent on  
*Subject* (the *other* part of the  
key)



# And 2NF requires...

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

All non-key fields are  
dependent on the ENTIRE  
key (studentID + subject)

# So it's not 2NF

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

## How can we fix it?

# Make new tables

- Make a new table for each primary key field
- Give each new table its own primary key
- Move columns from the original table to the new table that matches their primary key...

# Step 1

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

## Step 2

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

# Step 3

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)

# Step 3

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	\$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	\$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

# Step 4 - relationships

STUDENT TABLE (key = StudentID)

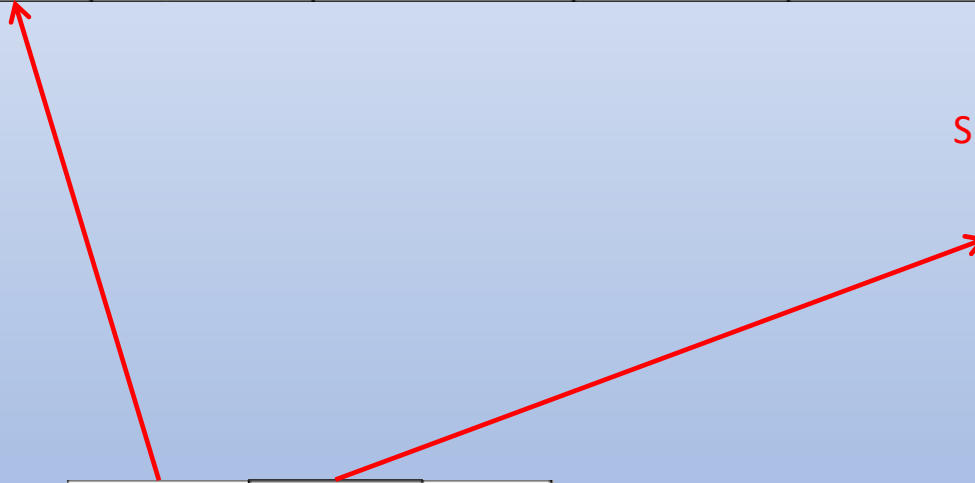
StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)





# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

Each student can only appear  
ONCE in the student table

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

Each subject can only appear  
ONCE in the subjects table

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

A subject can be listed MANY times in the results table (for different students)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

1

1

∞

# Step 4 - cardinality

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

A student can be listed  
MANY times in the results  
table (for different subjects)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

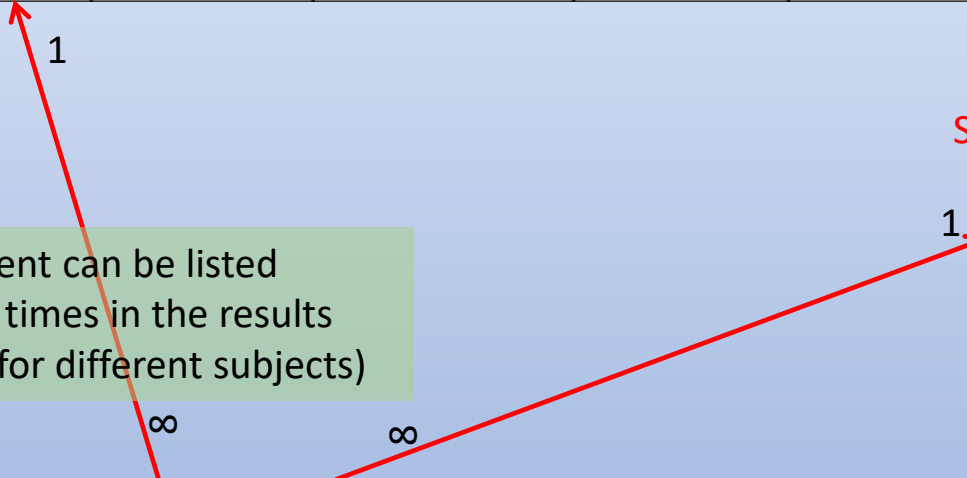
RESULTS TABLE (key = StudentID+Subject)

1

1

∞

∞



# A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

**SubjectCost** is only dependent on the primary key, *Subject*



# A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

**Grade** is only dependent  
on the primary key  
(*studentID* + *subject*)

RESULTS TABLE (key = StudentID+Subject)



# A 2NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

1

**Name, Address** are only dependent on the primary key (*StudentID*)



SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

1

∞

∞

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

**So it is  
2NF!**

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

But is it 3NF?



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Oh oh...  
What?

1

∞

∞

1

# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

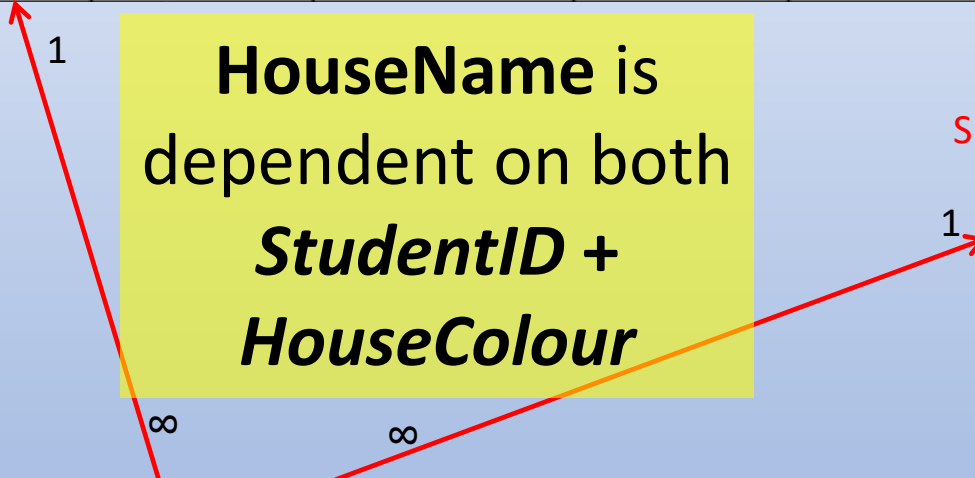
**HouseName** is  
dependent on both  
***StudentID +  
HouseColour***

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

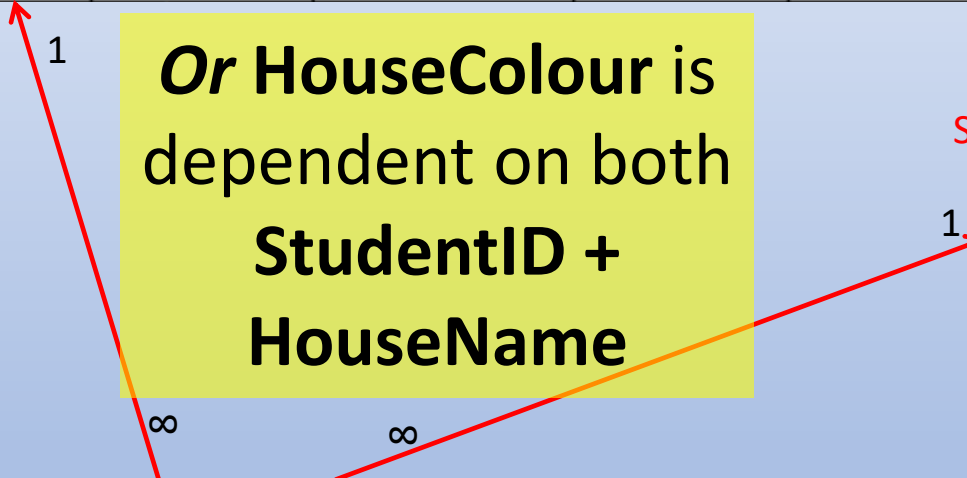
**Or HouseColour is  
dependent on both  
StudentID +  
HouseName**

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

*But either way,  
non-key fields are  
dependent on MORE  
THAN THE PRIMARY  
KEY (studentID)*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

*And 3NF says that  
non-key fields must  
depend on **nothing**  
but the key*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)



# A 3NF check

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

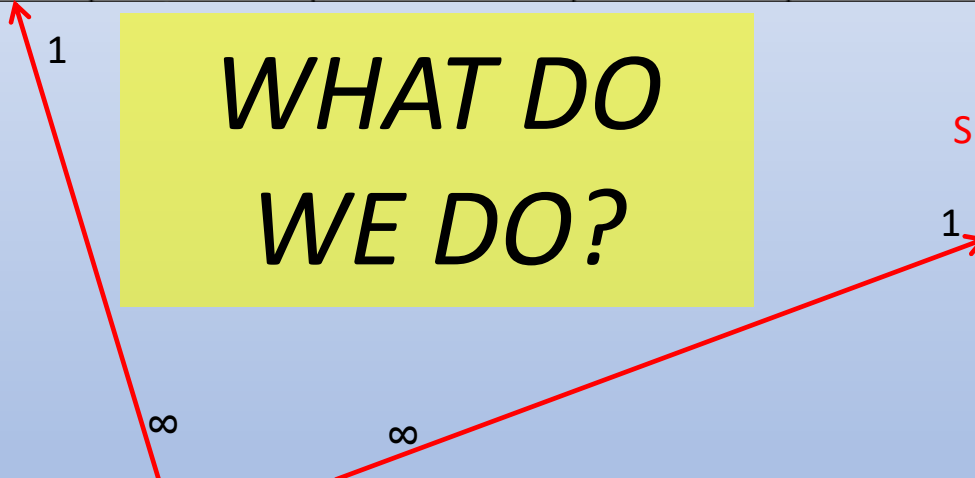
*WHAT DO  
WE DO?*

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

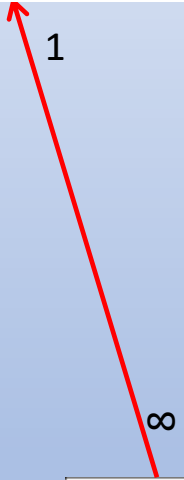


Again, carve off the offending fields

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID



RESULTS TABLE (key = StudentID+Subject)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100



# A 3NF fix

StudentTable			HouseTable	
StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red
Primary key: StudentID			Primary key: HouseName	

1

∞

∞

1

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

RESULTS TABLE (key = StudentID+Subject)



# A 3NF fix

StudentTable			
StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob
Primary key: StudentID			

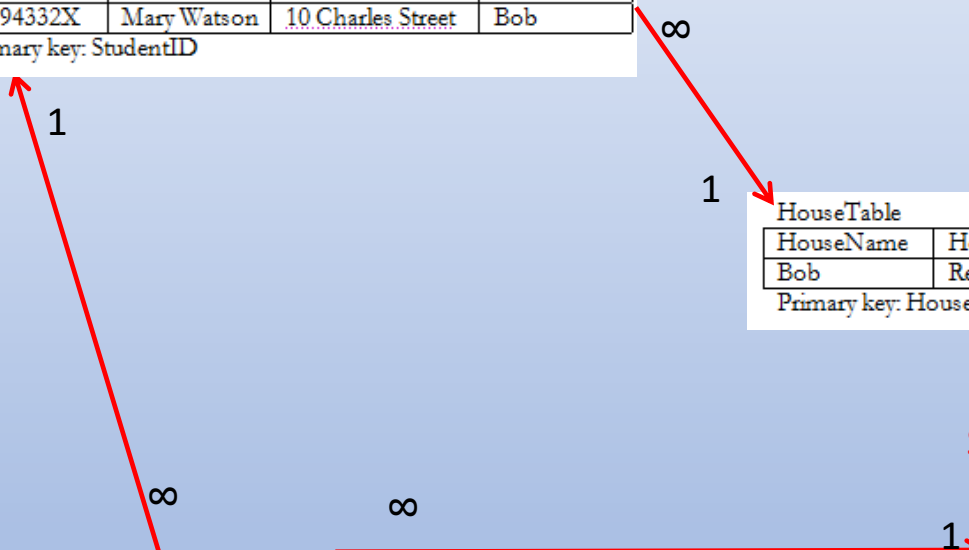
HouseTable	
HouseName	HouseColor
Bob	Red
Primary key: HouseName	

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

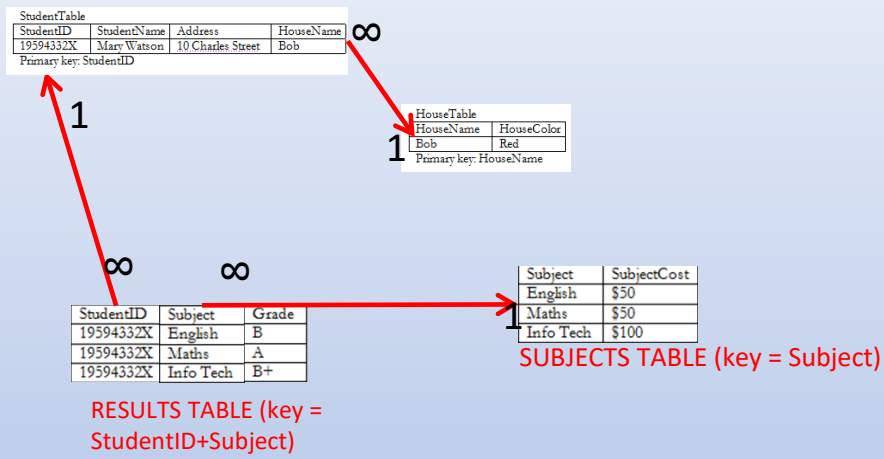
SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

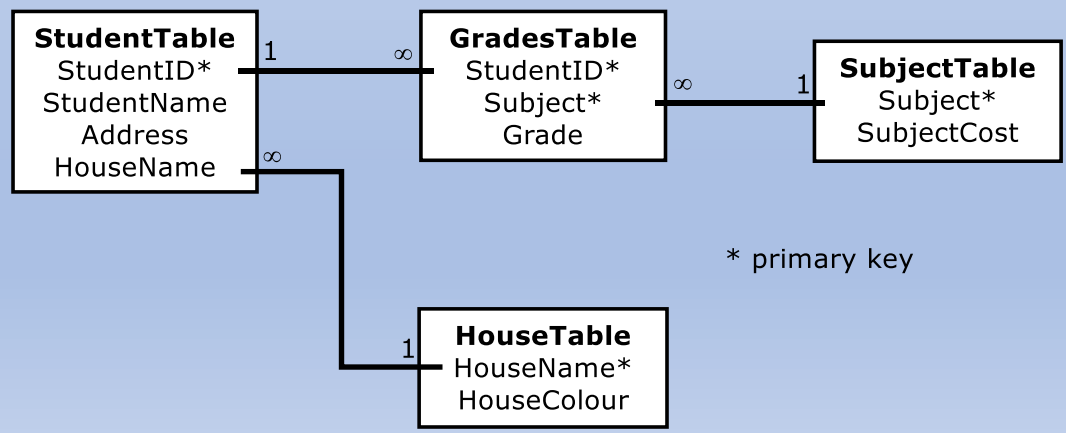
RESULTS TABLE (key = StudentID+Subject)



# A 3NF win!



Or...



# The Reveal

Before...

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	\$50	B
					Maths	\$50	A
					Info Tech	\$100	B+

After...

StudentTable			
StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob
Primary key: StudentID			

HouseTable	
HouseName	HouseColor
Bob	Red
Primary key: HouseName	

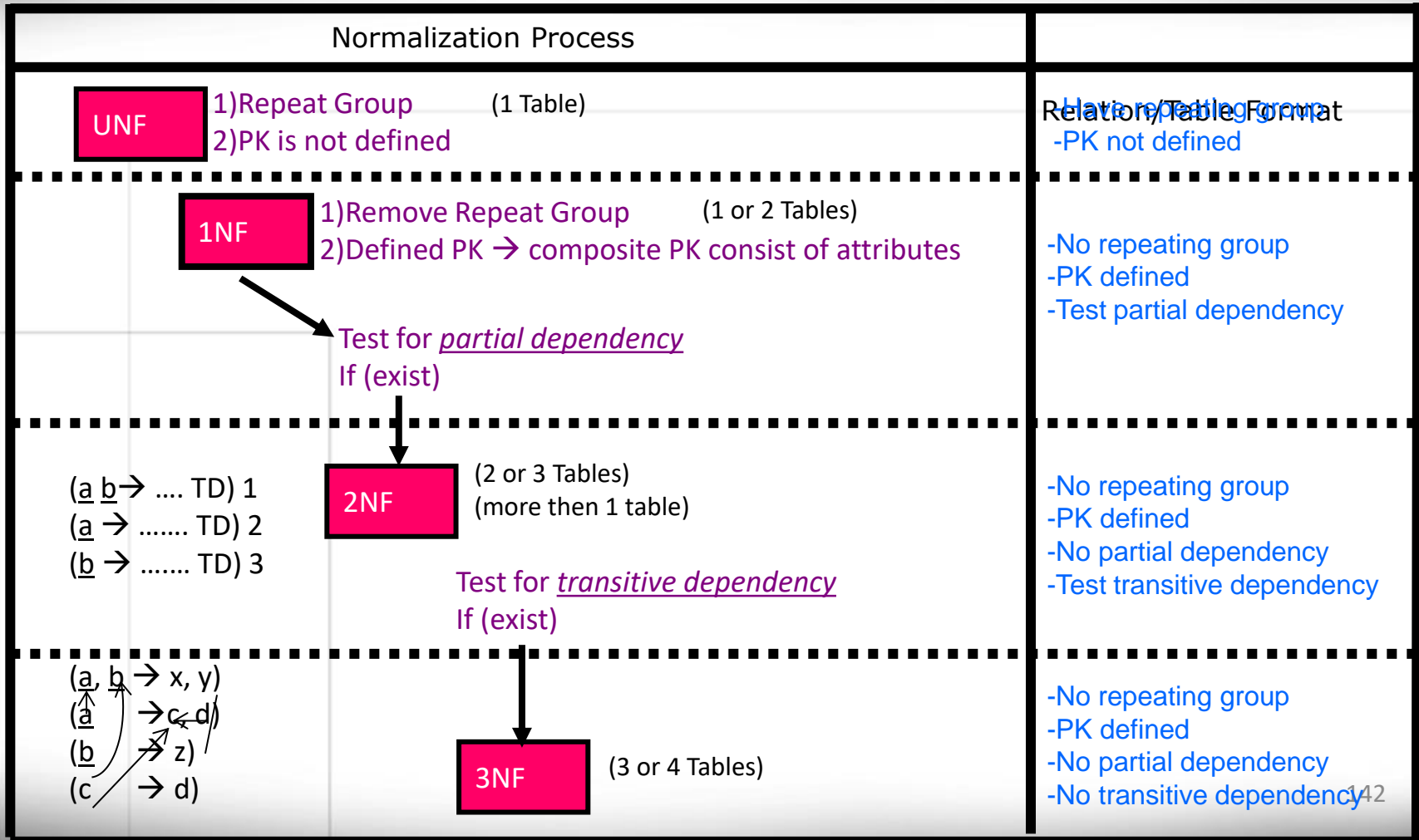
StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

Subject	SubjectCost
English	\$50
Maths	\$50
Info Tech	\$100

SUBJECTS TABLE (key = Subject)

RESULTS TABLE (key =  
StudentID+Subject)

# Normalization Process



# Normalization Process

Normalization Process						Relation/Table Format																																										
<table border="1"> <thead> <tr> <th>PROJ_NUM</th><th>PROJ_NAME</th><th>EMP_NUM</th><th>EMP_NAME</th><th>JOB_CLASS</th><th>CHG_HOUR</th><th>HOURS</th></tr> </thead> <tbody> <tr> <td>15</td><td>Evergreen</td><td>103</td><td>June E. Arbough</td><td>Elect. Engineer</td><td>\$84.50</td><td>23.8</td></tr> <tr> <td></td><td></td><td>101</td><td>John G. News</td><td>Database Designer</td><td>\$105.00</td><td>19.4</td></tr> <tr> <td></td><td></td><td>105</td><td>Alice K. Johnson *</td><td>Database Designer</td><td>\$105.00</td><td>35.7</td></tr> <tr> <td></td><td></td><td>106</td><td>William Smithfield</td><td>Programmer</td><td>\$35.75</td><td>12.5</td></tr> <tr> <td></td><td></td><td>102</td><td>David H. Senior</td><td>Systems Analyst</td><td>\$98.75</td><td>23.9</td></tr> </tbody> </table>						PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS	15	Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8			101	John G. News	Database Designer	\$105.00	19.4			105	Alice K. Johnson *	Database Designer	\$105.00	35.7			106	William Smithfield	Programmer	\$35.75	12.5			102	David H. Senior	Systems Analyst	\$98.75	23.9	<p><b>UNF</b> 1)Repeat Group (1 Table) 2)PK is not defined</p> <p>-Have repeating group -PK not defined</p>
PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS																																										
15	Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8																																										
		101	John G. News	Database Designer	\$105.00	19.4																																										
		105	Alice K. Johnson *	Database Designer	\$105.00	35.7																																										
		106	William Smithfield	Programmer	\$35.75	12.5																																										
		102	David H. Senior	Systems Analyst	\$98.75	23.9																																										
<table border="1"> <thead> <tr> <th>PROJ_NUM</th><th>PROJ_NAME</th><th>EMP_NUM</th><th>EMP_NAME</th><th>JOB_CLASS</th><th>CHG_HOUR</th><th>HOURS</th></tr> </thead> <tbody> <tr> <td>15</td><td>Evergreen</td><td>103</td><td>June E. Arbough</td><td>Elect. Engineer</td><td>\$84.50</td><td>23.8</td></tr> <tr> <td>15</td><td>Evergreen</td><td>101</td><td>John G. News</td><td>Database Designer</td><td>\$105.00</td><td>19.4</td></tr> <tr> <td>15</td><td>Evergreen</td><td>105</td><td>Alice K. Johnson *</td><td>Database Designer</td><td>\$105.00</td><td>35.7</td></tr> <tr> <td>15</td><td>Evergreen</td><td>106</td><td>William Smithfield</td><td>Programmer</td><td>\$35.75</td><td>12.5</td></tr> <tr> <td>15</td><td>Evergreen</td><td>102</td><td>David H. Senior</td><td>Systems Analyst</td><td>\$98.75</td><td>23.9</td></tr> </tbody> </table>						PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS	15	Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8	15	Evergreen	101	John G. News	Database Designer	\$105.00	19.4	15	Evergreen	105	Alice K. Johnson *	Database Designer	\$105.00	35.7	15	Evergreen	106	William Smithfield	Programmer	\$35.75	12.5	15	Evergreen	102	David H. Senior	Systems Analyst	\$98.75	23.9	<p><b>1NF</b> 1)Remove Repeat Group 2)Defined PK → composite PK consist of attributes</p> <p>Test for <u>partial dependency</u> If (exist)</p> <p>-No repeating group -PK defined -Test partial dependency</p>
PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS																																										
15	Evergreen	103	June E. Arbough	Elect. Engineer	\$84.50	23.8																																										
15	Evergreen	101	John G. News	Database Designer	\$105.00	19.4																																										
15	Evergreen	105	Alice K. Johnson *	Database Designer	\$105.00	35.7																																										
15	Evergreen	106	William Smithfield	Programmer	\$35.75	12.5																																										
15	Evergreen	102	David H. Senior	Systems Analyst	\$98.75	23.9																																										
<p>(a → ..... TD) 1 (b → ..... TD) 2 (a b → .... TD) 3</p>						<p><b>2NF</b> (2 or 3 Tables) (more then 1 table)</p> <p>Test for <u>transitive dependency</u> If (exist)</p> <p>-No repeating group -PK defined -No partial dependency -Test transitive dependency</p>																																										

# Normalization Process

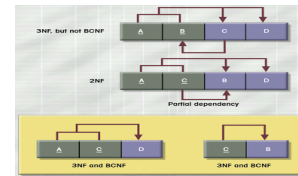
## Normalization Process

$(\underline{a}, \underline{b} \rightarrow x, y)$   
 $\uparrow$   
 $(\underline{a} \rightarrow c, \leftarrow d) \text{ [TD]}$   
 $(\underline{b} \rightarrow z)$   
 $(c \rightarrow d)$

3NF

(3 or 4 Tables)

BCNF



4NF

COURSE	CRS_TEXT	CRS_INSTRUCTOR
S511	DB design	Jones
S511	DB design	Smith
S511	Inside Access 2007	Jones
S511	Inside Access 2007	Smith

COURSE	CRS_TEXT
S511	DB design
S511	Inside Access 2007

COURSE	CRS_INSTRUCTOR
S511	Jones
S511	Smith

## Relation/Table Format

- No repeating group
- PK defined
- No partial dependency
- No transitive dependency

- Identify multiple multi-valued attributes
- Create separate tables containing each of multi-valued attributes

# Exercise 1

- A college maintains details of its lecturers' subject area skills. These details comprise:
- Lecturer Number
- Lecturer Name
- Lecturer Grade
- Department Code
- Department Name
- Subject Code
- Subject Name
- Subject Level
- Assume that each lecturer may teach many subjects but may not belong to more than one department.
- Subject Code, Subject Name and Subject Level are repeating fields.
- Normalise this data to Third Normal Form.

# Exercise 2

- A software contract and consultancy firm maintains details of all the various projects in which its employees are currently involved. These details comprise:
  - Employee Number
  - Employee Name
  - Date of Birth
  - Department Code
  - Department Name
  - Project Code
  - Project Description
  - Project Supervisor
- **Assume the following:**
  - Each employee number is unique.
  - Each department has a single department code.
  - Each project has a single code and supervisor.
  - Each employee may work on one or more projects.
  - Employee names need not necessarily be unique.
  - Project Code, Project Description and Project Supervisor are repeating fields.
  - Normalise this data to Third Normal Form.



# Example

Marketing Promotions Report								
Marketing Promotion	Promotion Date	Subject of Promotion	Branch No	Branch Name	Feature Category Id	Feature No	Feature Desc	Cost of promotion \$
1000	21/01/2007	Outdoor Activities	103	Takapuna	1	100	XXX	3000
			101	Orewa	1	100	XXX	2500
			105	Maritai	2	101	YY	2800
			106	Hastings	2	101	YY	3500
			102	Napier	2	102	Z	2300
Sub Total								14100
1002	21/01/2007	Sports Activities	105	Maritai	1	100	XXX	2500
			103	Takapuna	1	100	XXX	3000
			106	Hastings	2	101	YY	3500
			111	Christ Church	2	101	YY	2300
			108	Tauranga	2	103	ZX	3500
Sub Total								14800
Grand Total								28900

- Derive a set of relation in 3NF. Show each step of normalisation clearly. All PKs should have a solid underline and FKs a dashed line. Draw an ERD for the derived relations

# Example

Customer No	Customer Name	Customer Address	Customer Category	Category Discount	Painting No	Painting Title	Date Of Hire	Date Due Back	Return Flag
10	Gilbert	1, Hope St	Gold	10%	100	The Deer	1/07/2014	15/07/2014	Y
					110	Loch Ness	1/07/2014	15/07/2014	Y
					120	Edinburgh at Night	5/07/2014	20/07/2014	Y
					100	The Deer	16/07/2014	23/07/2014	N
					130	Autumn Leaves	16/07/2014	23/07/2014	N
20	Jones	23, Queen St	Silver	5%	140	Cold Days	2/07/2014	16/07/2014	Y
					110	Loch Ness	16/07/2014	23/07/2014	N
					160	Daffodils	16/07/2014	23/07/2014	N
30	Smith	8, George St	Gold	10%	170	Highland Cattle	3/07/2014	17/07/2014	Y
					170	Highland Cattle	18/07/2014	27/07/2014	N
					120	Edinburgh at Night	23/07/2014	1/08/2014	N