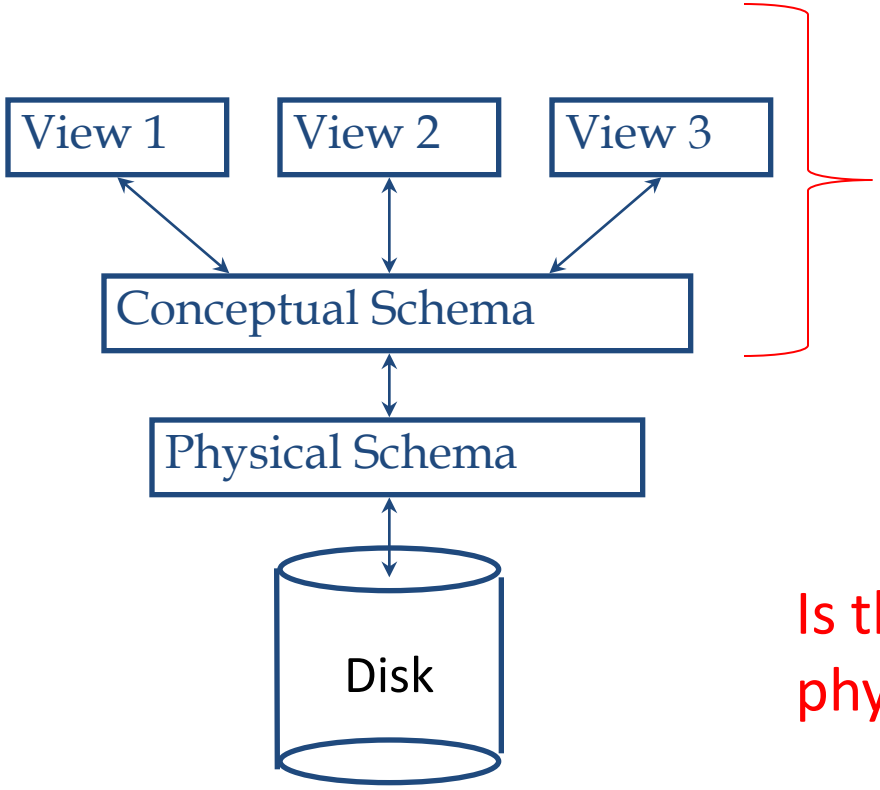
NYU
New York University

CSCI-GA.2433-001

Database Systems

Lecture 7: Schema Refinement and Normalization

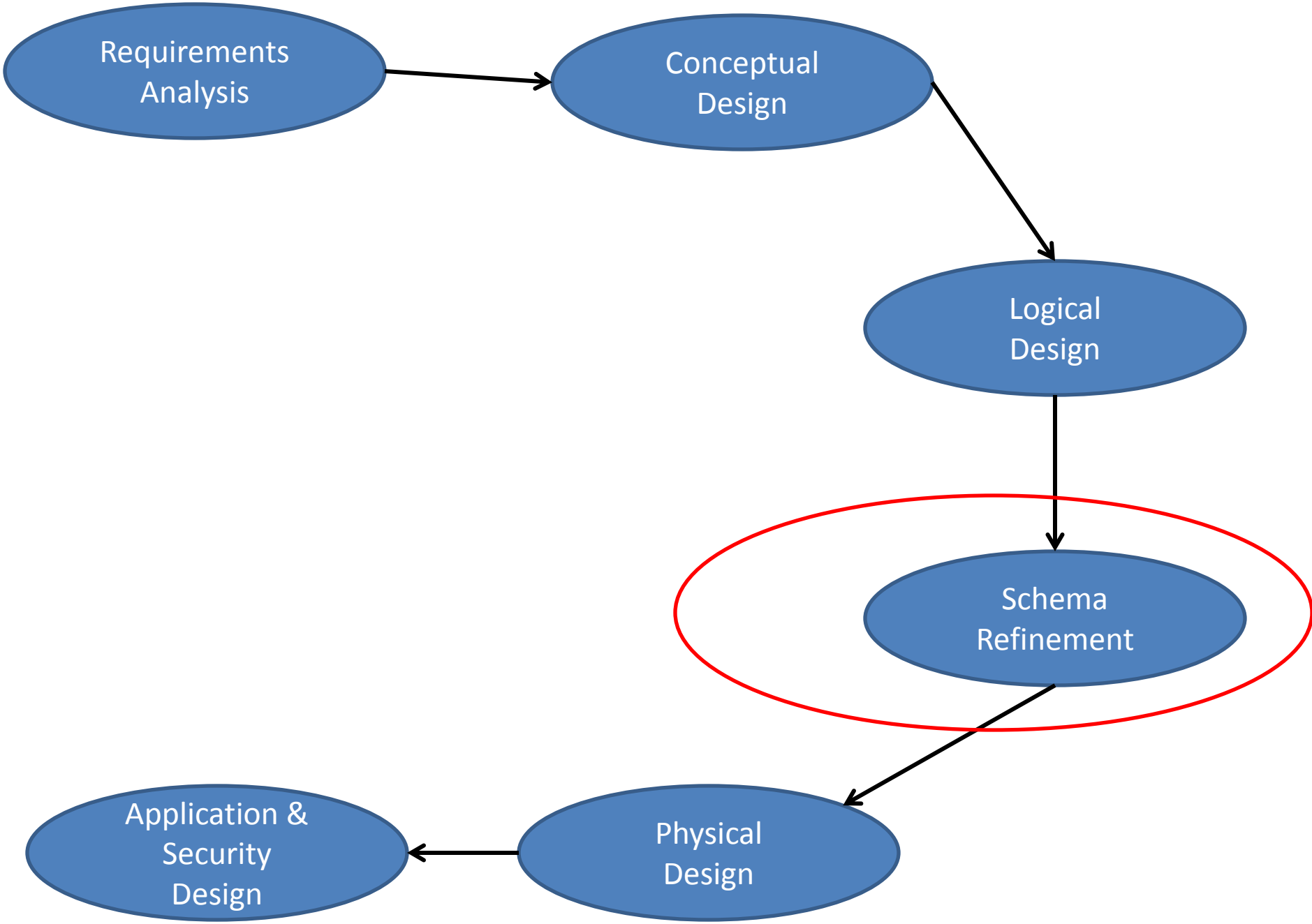
Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

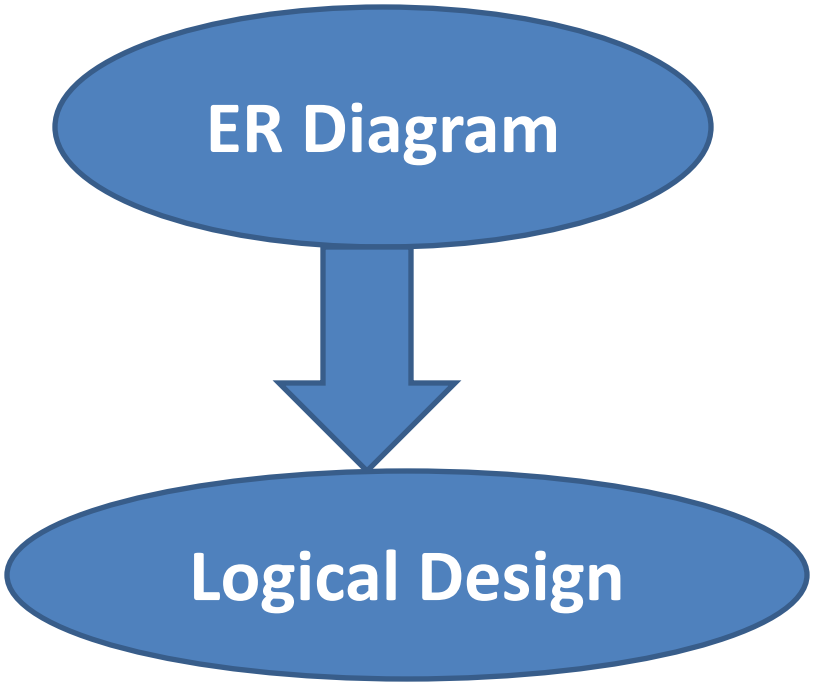


At that point we have:

- ER model
- Set of constraints
- Schema

Is that all? Can we move now to the physical part?


Requirements Analysis
Conceptual Design
Logical Design
Schema Refinement
Physical Design
Application & Security Design



1

To

Many

How to choose?

What Do We Want to Do?

- We are given a set of tables specifying the database
- They come from some ER diagram
- We will need to examine whether the specific choice of tables is good for
  - Storing the information needed
  - Enforcing constraints
  - Avoiding anomalies, such as redundancies
- If there are issues to address, we may want to restructure the database, of course not losing any information

The Evils of Redundancy

- *Redundancy* is at the root of several problems associated with relational schemas:
  - redundant storage, insert/delete/update anomalies
- Integrity constraints, in particular *functional dependencies*, can be used to identify schemas with such problems and to suggest refinements.
- Main refinement technique: *decomposition* (replacing ABCD with, say, AB and BCD, or ACD and ABD).
- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - What problems (if any) does the decomposition cause?

Problems with Decompositions

- There are three potential problems to consider:
  - Some queries become more expensive.
  - Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
  - Checking some dependencies may require joining the instances of the decomposed relations.

Must consider these issues vs. redundancy.

Are there any inefficiencies here?

Name	SSN	DOB	Grade	Salary
A	121	2367	2	80
A	132	3678	3	70
B	101	3498	4	70
C	106	2987	2	80

Assume grade determines salary.

Redundancy

Additional Problems:

- We are unable to store the salary structure for a Grade that does not currently exist for any employee.
  For example, we cannot store that Grade = 1 implies Salary = 90
- For example, if employee with SSN = 132 leaves, we forget which Salary should be paid to employee with Grade = 3
- We could perhaps invent a fake employee with such a Grade and such a Salary, but this brings up additional problems, e.g., What is the SSN of such a fake employee? It cannot be NULL as SSN is the primary key

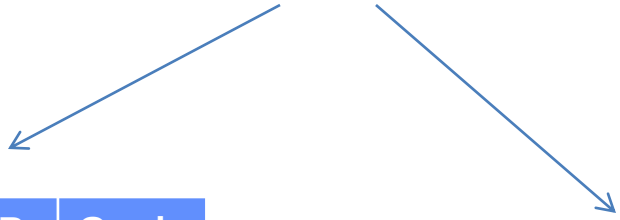
Better Presentation

```
SELECT INTO S
Name, SSN, DOB, Grade
FROM R;
```

```
SELECT INTO T
Grade, Salary
FROM R;
```

R	Name	SSN	DOB	Grade	Salary
	A	121	2367	2	80
	A	132	3678	3	70
	B	101	3498	4	70
	C	106	2987	2	80

Use two tables!



S	Name	SSN	DOB	Grade
	A	121	2367	2
	A	132	3678	3
	B	101	3498	4
	C	106	2987	2

T	Grade	Salary
	2	80
	3	70
	4	70

Did We Loose Any Information?

S	Name	SSN	DOB	Grade
	A	121	2367	2
	A	132	3678	3
	B	101	3498	4
	C	106	2987	2

T	Grade	Salary
	2	80
	3	70
	4	70

Natural Join

R	Name	SSN	DOB	Grade	Salary
	A	121	2367	2	80
	A	132	3678	3	70
	B	101	3498	4	70
	C	106	2987	2	80

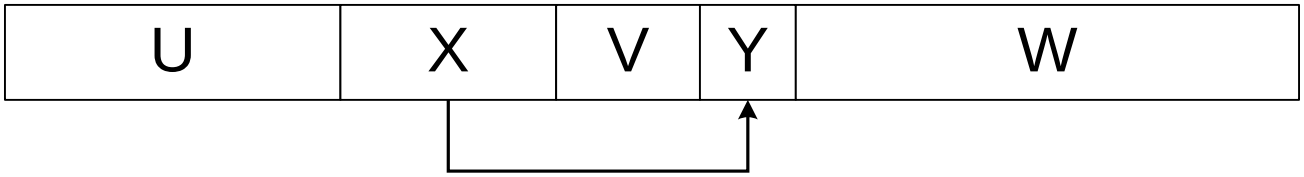
```
SELECT INTO R
Name, SSN, DOB, S.Grade AS Grade,
Salary
FROM T, S
WHERE T.Grade = S.Grade;
```

Can We Generalize The Solution?

We have a problem whenever we have two sets of columns X and Y such that

1. **X does not contain a key either primary or unique** (thus there could be several/many **non-identical** rows with the same value of X)
2. **Whenever two rows are equal on X, they must be equal on Y**

We had one table



We replaced that table with two tables

What Will We Do?

- Formally, we will study normalization (decompositions as in the above example) and normal forms (forms for relation specifying some "niceness" conditions)
- There will be three very important issues of interest:
  - Removal of redundancies
  - Lossless-join decompositions
  - Preservation of dependencies

Normal Forms

- A normal form applies to a table/relation, not to the database
- So the question is individually asked about a table: is it of some specific desirable normal form?
- The ones you need to know about in increasing order of "quality" and complexity:
  - First Normal Form (**1NF**); it essentially states that we have a table/relation
  - Second Normal Form (**2NF**); intermediate form
  - Third Normal Form (**3NF**); very important; a final form
  - Boyce-Codd Normal Form (**BCNF**); very important; a final form

Normalization deals with "reorganizing" a relational database by, generally, breaking up tables (relations) to remove various anomalies

Our Running Example

	S	B	C	T	F	C	T	F
	Fang	1990	DB	Zvi	1	OS	Allan	2
	John	1980	OS	Allan	2	PL	Marsha	4
	Mary	1990	PL	Vijay	1			

- S, which is a Student
- B, which is the Birth Year of the Student
- C, which is a Course that the student took
- T, which is the Teacher who taught the Course the Student took
- F, which is the Fee that the Student paid the Teacher for taking the course

Assumption:

- **A student can take any number of courses**

The number of columns is not fixed!!!
This means the above table is NOT a relation!

Our Running Example

- Constraints:
  - A student can have only one birth year
  - A teacher has to charge the same fee from every student he/she teaches.
  - A teacher can teach only one course
  - A student can take any specific course from one teacher only (or not at all)

First Normal Form (1NF): A Table With Fixed Number Of Column

- This ***was not*** a relation, because we are told that each Student may have taken any number of Courses
- Therefore, the number of columns is not fixed/bounded
- It is easy to make this a relation, getting

R	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4

Formally, we have a relation in ***First Normal Form (1NF)***, this means that there are no repeating groups and the number of columns is fixed

Functional Dependencies (FD)

- Let P and Q be sets of columns, then:
  *P **functionally determines*** Q, written P → Q
  if and only if
  any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q
- In simpler terms, less formally, but really the same, it means that:

If a value of P is specified, it "forces" some (specific) value of Q; in other words: Q is a function of P

Example

R	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4

- Dependencies:
  1. A student can have only one birth year: $S \rightarrow B$
  2. A teacher has to charge the same fee from every student he/she teaches : $T \rightarrow F$
  3. A teacher can teach only one course (perhaps at different times, different offerings, etc, but never another course) : $T \rightarrow C$
  4. A student can take a course from one teacher only: $SC \rightarrow T$

Possible Primary Key

- Our rules: $S \rightarrow B$, $T \rightarrow F$, $T \rightarrow C$, $SC \rightarrow T$
- ST possible primary key, because given ST
  1. S determines B
  2. T determines F
  3. T determines C
- A part of ST is not sufficient
  1. From S, we cannot get T, C, or F
  2. From T, we cannot get S or B

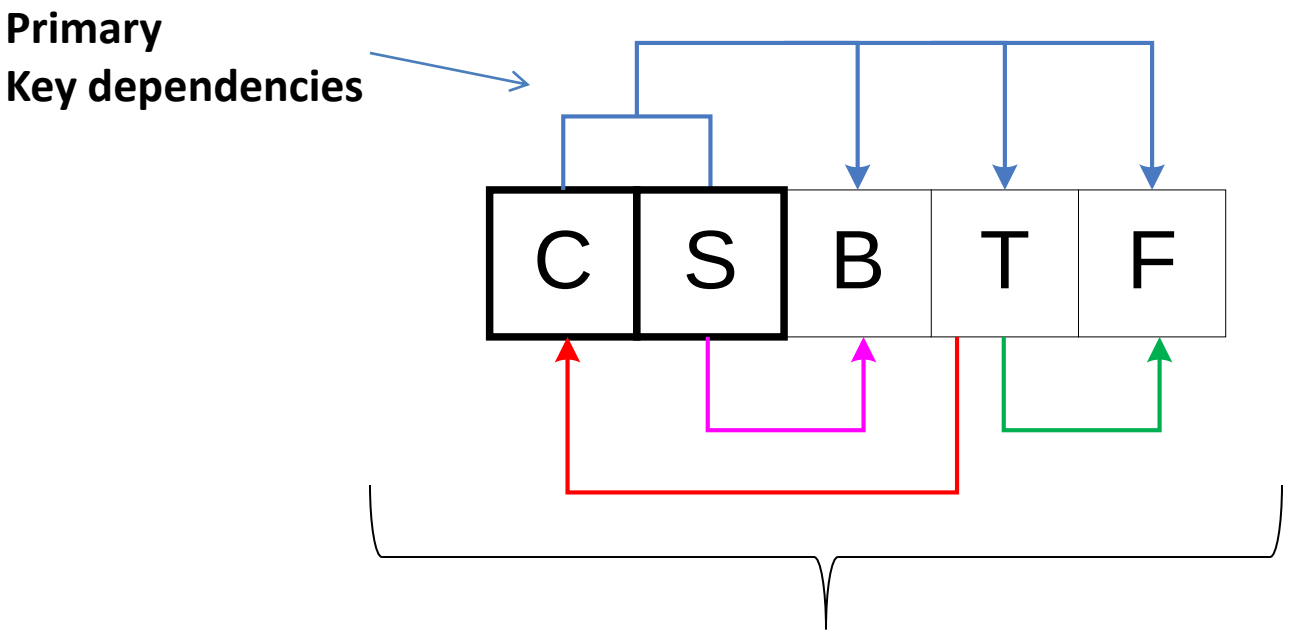
R	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4

Possible Primary Key

- Our rules: $S \rightarrow B$, $T \rightarrow F$, $T \rightarrow C$, $SC \rightarrow T$
- SC possible primary key, because given SC
  1. S determines B
  2. SC determines T
  3. T determines F (we can now use T to determine F because of 2)
- A part of SC is not sufficient
  1. From S, we cannot get T, C, or F
  2. From C, we cannot get S, B, T, or F

R	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4

Drawing FD



Partial dependency: From a part of the primary key to outside the key

Transitive dependency: From outside the key to outside the key

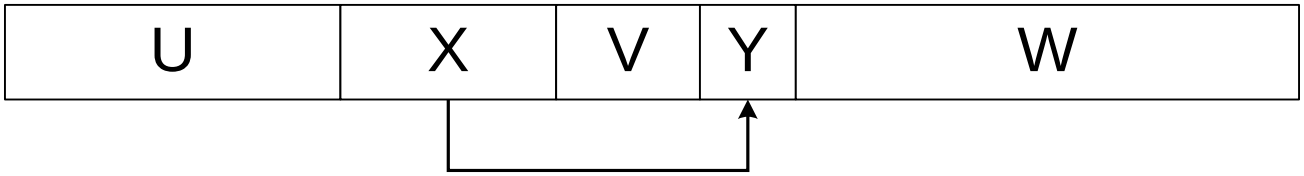
Into key dependency: From outside the key into (all or part of) the key

Those dependencies not from the full primary key are the reason for **anomalies**:

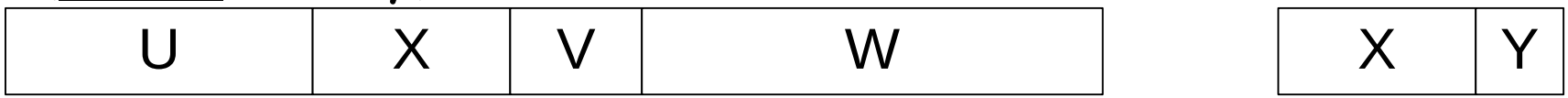
- Inability to store important information
- Redundancies

A Procedure For Removing Anomalies

- In general, we will have sets of attributes: U, X, V, Y, W
- We replaced R(Name,SSN,DOB,Grade,Salary), where Grade → Salary; in the drawing "X" stands for "Grade" and "Y" stands for "Salary"



by two tables S(Name,SSN,DOB,Grade) and T(Grade,Salary)



- We will do the same thing, dealing with one anomaly at a time

We do this ONLY if Y is not part of the primary key.

What do you think about the following decomposition?

R	Name	SSN	DOB	Grade	Salary
	A	121	2367	2	80
	A	132	3678	3	70
	B	101	3498	4	70
	C	106	2987	2	80

S	Name	DOB	Grade	Salary
	A	2367	2	80
	A	3678	3	70
	B	3498	4	70
	C	2987	2	80

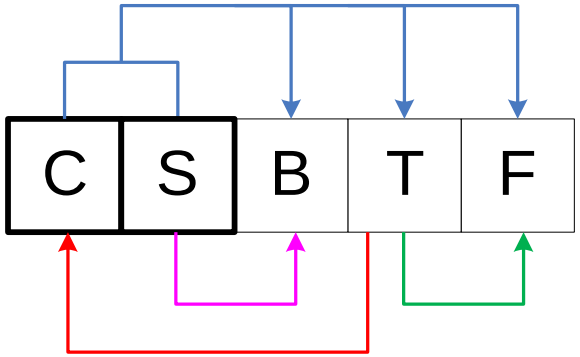
T	SSN	Salary
	121	80
	132	70
	101	70
	106	80

what is the Name for SSN = 121

We lost some information!!

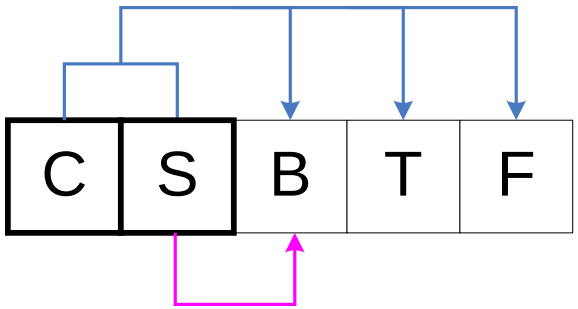
Our Example Again

	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4



Partial Dependency: S → B

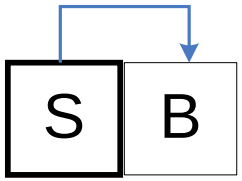
	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4



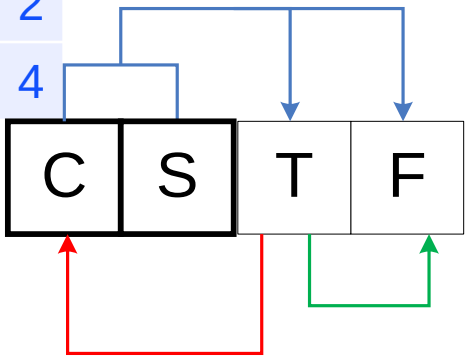
Decomposition

	S	B	C	T	F
	Fang	1990	DB	Zvi	1
	John	1980	OS	Allan	2
	Mary	1990	PL	Vijay	1
	Fang	1990	OS	Allan	2
	John	1980	PL	Marsha	4

No Anomalies



Some anomalies



	S	B
	Fang	1990
	John	1980
	Mary	1990
	Fang	1990
	John	1980

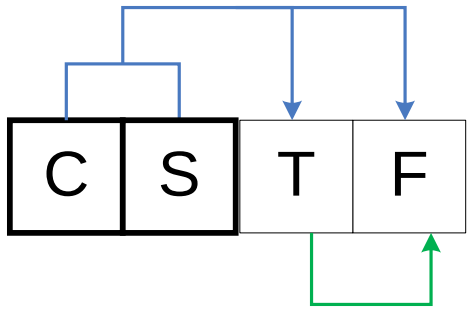
	S	C	T	F
	Fang	DB	Zvi	1
	John	OS	Allan	2
	Mary	PL	Vijay	1
	Fang	OS	Allan	2
	John	PL	Marsha	4

Second Normal Form (2NF): 1NF And No Partial Dependencies

- Second Normal Form means:
  - First Normal Form
  - No Partial dependencies
- The above is checked individually for each table
- Decomposition is a lossless join decomposition
- This means that by "combining" all the tables we get exactly the original table back

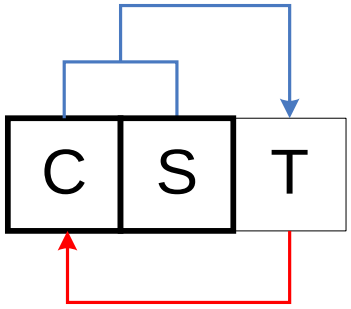
T → F

	S	C	T	F
	Fang	DB	Zvi	1
	John	OS	Allan	2
	Mary	PL	Vijay	1
	Fang	OS	Allan	2
	John	PL	Marsha	4



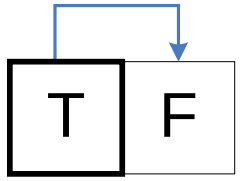
Decomposition

Anomalies



	S	C	T	F
	Fang	DB	Zvi	1
	John	OS	Allan	2
	Mary	PL	Vijay	1
	Fang	OS	Allan	2
	John	PL	Marsha	4

No Anomalies

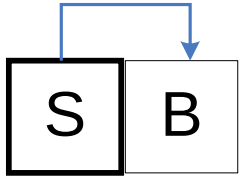


	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha

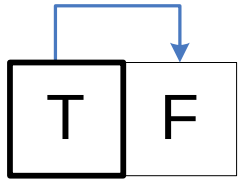
	T	F
	Zvi	1
	Allan	2
	Vijay	1
	Allan	2
	Marsha	4

Decomposition So Far

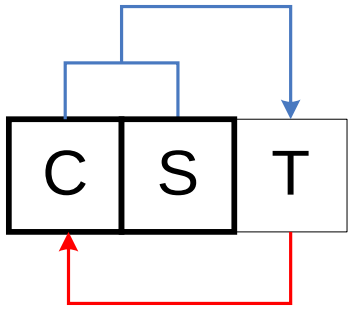
	S	B
	Fang	1990
	John	1980
	Mary	1990



	T	F
	Zvi	1
	Allan	2
	Vijay	1
	Marsha	4



	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha

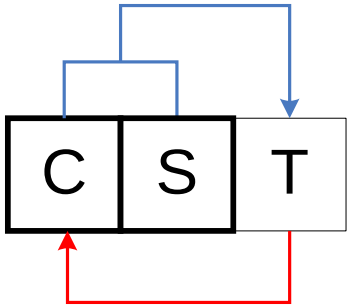


Third Normal Form (3NF): 2NF And No Transitive Dependencies

- Third Normal Form means:
  - Second Normal Form (therefore in 1NF and no partial dependencies)
  - No transitive dependencies
- The above is checked individually for each table
- A lossless join decomposition
- This means that by "combining" all the tables we get exactly the original table back

Anomaly

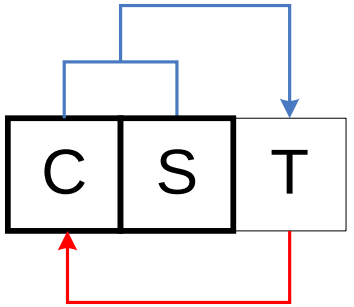
	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha



- We are worried about decomposing by "pulling out" C and getting CS and TC, as we are pulling out a part of the key
- What Can We do?

Anomaly

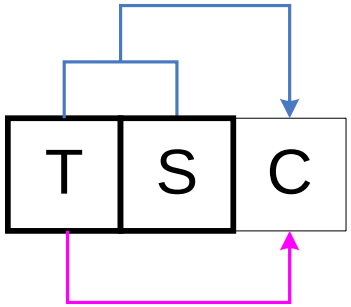
	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha



- Can we pick an alternative primary key for this table?
- How about TS?

Anomaly

	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha



- Now are anomaly is a partial dependency, which we know how to handle

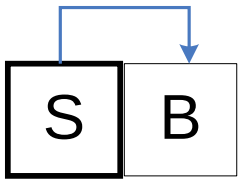
Decomposition

	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha

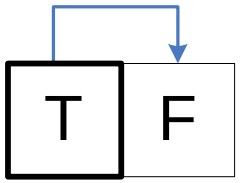
	S	T
	Fang	Zvi
	John	Allan
	Mary	Vijay
	Fang	Allan
	John	Marsha

	C	T
	DB	Zvi
	OS	Allan
	PL	Vijay
	OS	Allan
	PL	Marsha

Our Decomposition

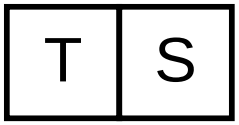


	S	B
	Fang	1990
	John	1980
	Mary	1990

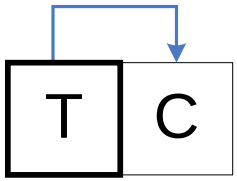


	T	F
	Zvi	1
	Allan	2
	Vijay	1
	Marsha	4

Can we combine those while keeping the good properties?



	S	T
	Fang	Zvi
	John	Allan
	Mary	Vijay
	Fang	Allan
	John	Marsha

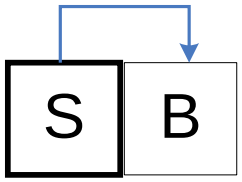


	C	T
	DB	Zvi
	OS	Allan
	PL	Vijay
	PL	Marsha

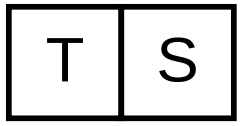
No Anomalies

Our Decomposition

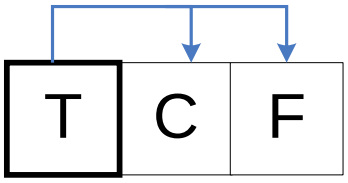
Combine tables if they have the same key and we can still maintain good properties



S	B
Fang	1990
John	1980
Mary	1990



S	T
Fang	Zvi
John	Allan
Mary	Vijay
Fang	Allan
John	Marsha



T	F	C
Zvi	1	DB
Allan	2	OS
Vijay	1	PL
Marsha	4	PL

Boyce-Codd Normal (BCNF): 1NF And All Dependencies From Full Key

- Boyce-Codd Normal Form (BCNF) means:
  - 1NF
  - Every functional dependency is from a full key

    This definition is "loose." Later, a complete, formal definition
- A table is BCNF is automatically in 3NF
- The above is checked individually for each table
- A lossless join decomposition
- This means that by "combining" all the tables we get exactly the original table back

Do You Remember this Decomposition?

Constraints:

T → C

SC → T

	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha

	S	T
	Fang	Zvi
	John	Allan
	Mary	Vijay
	Fang	Allan
	John	Marsha

	C	T
	DB	Zvi
	OS	Allan
	PL	Vijay
	OS	Allan
	PL	Marsha

An Insert Attempt

- A user wants to specify that now John is going to take PL from Vijay
- If we look at the database, we realize this update should not be permitted because
  - John can take PL from at most one teacher
  - John already took PL
- But can the system figure this out just by checking whether FDs continue being satisified
- Let us find out what will happen in each of the two scenarios (before and after decomposition)

Scenario 1

- We maintain SCT, knowing that its keys are SC and ST
- Before the INSERT, constraints are satisfied; keys are OK
- After the INSERT, constraints are not satisfied; SC is no longer a key
- INSERT rejected after the constraint is checked

	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha

	S	C	T
	Fang	DB	Zvi
	John	OS	Allan
	Mary	PL	Vijay
	Fang	OS	Allan
	John	PL	Marsha
	John	PL	Vijay

Scenario 2

Dependencies are NOT preserved!!

- We maintain ST, knowing that its key ST
- We maintain CT, knowing that its key is T

	S	T
	Fang	Zvi
	John	Allan
	Mary	Vijay
	Fang	Allan
	John	Marsha

	C	T
	DB	Zvi
	OS	Allan
	PL	Vijay
	OS	Allan
	PL	Marsha

- Before the INSERT, constraints are satisfied; keys are OK
- After the INSERT constraints are still satisfied; keys remain keys
- But the INSERT ***must*** still be rejected

	S	T
	Fang	Zvi
	John	Allan
	Mary	Vijay
	Fang	Allan
	John	Marsha
	John	Vijay

	C	T
	DB	Zvi
	OS	Allan
	PL	Vijay
	OS	Allan
	PL	Marsha
	PL	Vijay

VERY Important

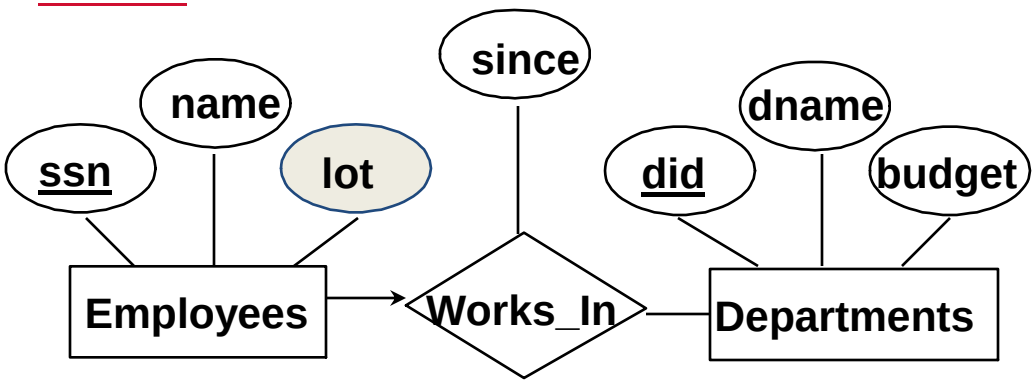
Generally, normalize up to 3NF and not up to BCNF

- So the database is not fully normalized

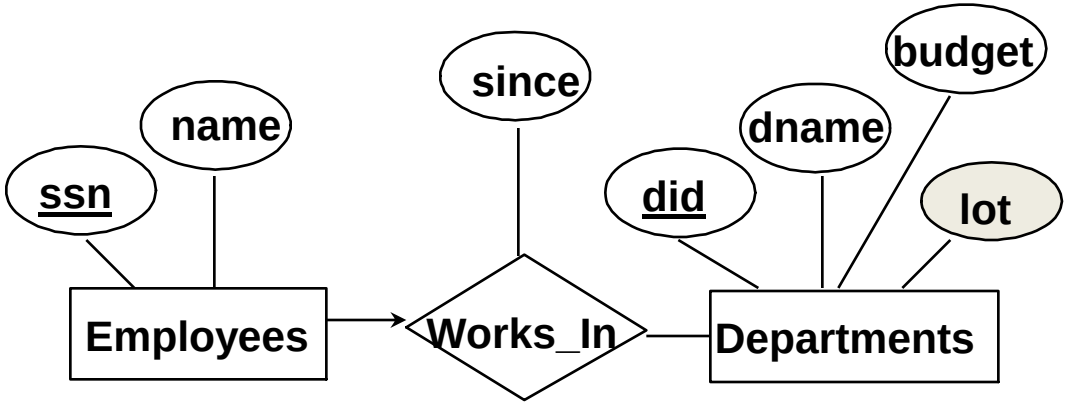
Refining an ER Diagram

- 1st diagram translated:
  - Lots associated with workers.
- Suppose all workers in a dept are assigned the same lot: D → L

Before:



After:



Conclusions

- Avoiding redundancy makes the design more effective and efficient
- You have to weight pros and cons of getting rid of redundancy
- You can refine the ER diagram and also refine at the logical level too
- Generally, normalize up to 3NF and not up to BCNF
  - So the database is not fully normalized