CMPT 321
FALL 2017

Introduction to normalization

Lecture 06.01

By Marina Barsky

Relational Schema Design

- Goal of relational schema design is to avoid anomalies and redundancy.
  - *Update anomaly* : one occurrence of a fact is changed, but not all occurrences.
  - *Deletion anomaly*: valid fact is lost when a tuple is deleted.

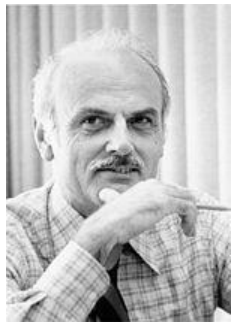
- Now we will tackle the problem of refining relational schemas.

- The main method: *decomposition* of relations into smaller relations

Normalization

- *Database normalization* is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency
- Normalization involves dividing large tables into smaller (and less redundant) tables and defining relationships between them.
- The objective: isolate each fact so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships

The goals of normalization defined by Codd:



1. To free the collection of relations from undesirable insertion, update and deletion dependencies

2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs

3. To make the relational model more informative to users

4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by

— E.F. Codd,

"Further Normalization of the Data Base Relational Model"

Functional dependencies

- The primary key identifies an entity
- All non-key attributes are ***functionally*** *dependent* on the key
- It is a special type of a **function** - a database function: knowing an argument (value of attribute(s) A), we can find all the other attributes (not from a formula but by lookup)

- Notation: *A* ➔ *B*
  - *A* uniquely identifies *B*
  - *B* is functionally dependent on *A*

Example of functional dependencies

Grades		
Name	Course	Grade
Bob	Databases	In pr
Maria	HCI	A
John	Python	B
Tom	HCI	A
Maria	Algorithms	A
Bob	HCI	B
Maria	Python	A

Name, Course → Grade

~~Name → Grade~~

~~Grade → Course~~

~~Course → Grade~~

- **Functional dependencies** reflect real-life facts and are nothing else but **constraints on data**

FD example

- Constraint: “no two courses can meet in the same room at the same time”

{hour, room} → course

Example: database of student records

Students (Name, Courses (with grades))

Students	
Name	**Courses**
Bob	Databases (in progr.), HCI (A)
John	Python (B)
Tom	HCI (A)
Maria	Algorithms (A), Python(A), HCI(A)

Constraints:
Name is unique
One student may take multiple courses

Usage:
Report one student's courses and grades

The first thing– identify key

- The key of the relation is a set of one or more attributes that identify each tuple (row)
  - The value of a key is **unique** for each row
  - Each relation has **only one** key
  - Key is **immutable**: once the value of the key is assigned to a tuple, it cannot be updated through the lifetime of the database, but only deleted with the entire tuple
- All non-key attributes are functionally dependent on key

Database of student records: key?

Students (Name, Courses (with grades))

Students	
Name	**Courses**
Bob	Databases (in progr.), HCI (A)
John	Python (B)
Tom	HCI (A)
Maria	Algorithms (A), Python(A), HCI(A)

Student records: key defined

Students (ID, Name, Courses (with grades))

Students		
ID	**Name**	**Courses**
1	Bob	Databases (in progr.), HCI (A)
2	John	Python (B)
3	Tom	HCI (A)
4	Maria	Algorithms (A), Python(A), HCI(A)

Updated constraints:
ID is unique
ID determines Name
ID determines courses

Student records: functional dependencies

Students (ID, Name, Courses (with grades))

Students		
ID	**Name**	**Courses**
1	Bob	Databases (in progr.), HCI (A)
2	John	Python (B)
3	Tom	HCI (A)
4	Maria	Algorithms (A), Python(A), HCI(A)

ID →Name
ID → Courses
ID, Name → Courses

Student record database is anomalous

Students (ID, Name, Courses (with grades))

Students		
ID	**Name**	**Courses**
1	Bob	Databases (in progr.), HCI (A)
2	John	Python (B)
3	Tom	HCI (A)
4	Maria	Algorithms (A), Python(A), HCI(A)

- There is no redundancy (good!)
- Insertion anomaly
- Deletion anomaly
- General-purpose queries are inefficient

Normal forms –testing for normalization

- The database is normalized when all its relations are normalized
- There are rules to test each relation – normal forms:
  - 1NF
  - 2NF
  - 3NF
  - BCNF
  - 4NF
  - 5NF

- In most cases, the relation is normalized if it is in 3NF

First normal form 1NF

- The **primary key** is defined
- The domain of each attribute is represented as a single column (**no duplicative attributes** with the same meaning)
- Every row-and-column intersection contains exactly one value from the applicable domain (**atomicity**)

Students: in 1NF?

Students relation		
ID	**Name**	**Courses**
1	Bob	Databases (in progr.), HCI (A)
2	John	Python (B)
3	Tom	HCI (A)
4	Maria	Algorithms (A), Python(A), HCI(A)

Students: in 1NF?

Students relation		
ID	**Name**	**Courses**
1	Bob	Databases (in progr.), HCI (A)
2	John	Python (B)
3	Tom	HCI (A)
4	Maria	Algorithms (A), Python(A), HCI(A)

- The *courses* are not atomic: each cell in this column contains multiple values

Students: in 1NF?

Students relation							
ID	**Name**	**Course1**	**Grade1**	**Course2**	**Grade2**	**Course3**	**Grade3**
1	Bob	Databases	In progr.	HCI	A		
2	John	Python	B				
3	Tom	HCI	A				
4	Maria	Algorithms	A	Phyton	A	HCI	A

Students: in 1NF?

Students relation							
ID	**Name**	**Course1**	**Grade1**	**Course2**	**Grade2**	**Course3**	**Grade3**
1	Bob	Databases	In progr.	HCI	A		
2	John	Python	B				
3	Tom	HCI	A				
4	Maria	Algorithms	A	Phyton	A	HCI	A

- The attributes are duplicative (multiple columns for the same domain)
- Still general queries are difficult
- Waste of space
- Limited number of courses per student

Students: in 1NF?

Students (ID, Name, Course, Grade)

Students			
ID	**Name**	**Course**	**Grade**
1	Bob	Databases	In pr
2	Maria	HCI	A
3	John	Python	B
4	Tom	HCI	A
2	Maria	Algorithms	A
1	Bob	HCI	B
2	Maria	Python	A

Primary key?

Students: in 1NF!

Students (ID, Name, Course, Grade)

Students			
ID	**Course**	**Name**	**Grade**
1	**Databases**	Bob	In pr
2	**HCI**	Maria	A
3	**Python**	John	B
4	**HCI**	Tom	A
2	**Algorithms**	Maria	A
1	**HCI**	Bob	B
2	**Python**	Maria	A

New data: Students extended

Students (ID, Course, Name, Phone, Major, Professor, Grade)

Students						
ID	**Course**	**Name**	**Phone**	**Major**	**Prof**	**Grade**
1	**Databases**	Bob	211-2112	CSCI	Dr. Monk	In pr
2	**HCI**	Maria	344-3344	BIOL	Dr. Pooh	A
3	**Python**	John	500-5005	MATH	Dr. Patel	B
4	**HCI**	Tom	601-6778	PHYS	Dr. Pooh	A
2	**Algorithms**	Maria	344-3344	BIOL	Dr. Monk	A
1	**HCI**	Bob	211-2112	CSCI	Dr. Pooh	B
2	**Python**	Maria	344-3344	BIOL	Dr. Patel	A

Students extended

Students (ID, Course, Name, Phone, Major, Professor, Grade)

Students						
ID	**Course**	**Name**	**Phone**	**Major**	**Prof**	**Grade**
1	Databases	Bob	211-2112	CSCI	Dr. Monk	In pr
2	HCI	Maria	344-3344	BIOL	Dr. Pooh	A
3	Python	John	500-5005	MATH	Dr. Patel	B
4	HCI	Tom	601-6778	PHYS	Dr. Pooh	A
2	Algorithms	Maria	344-3344	BIOL	Dr. Monk	A
1	HCI	Bob	211-2112	CSCI	Dr. Pooh	B
2	Python	Maria	344-3344	BIOL	Dr. Patel	A

Constraints:

- ID is unique
- Student can take many courses
- Student majors in one subject
- Student has only one phone
- Course is taught by one professor

Students extended

Students (ID, Course, Name, Phone, Major, Professor, Grade)

Students						
ID	**Course**	**Name**	**Phone**	**Major**	**Prof**	**Grade**
1	Databases	Bob	211-2112	CSCI	Dr. Monk	In pr
2	HCI	Maria	344-3344	BIOL	Dr. Pooh	A
3	Python	John	500-5005	MATH	Dr. Patel	B
4	HCI	Tom	601-6778	PHYS	Dr. Pooh	A
2	Algorithms	Maria	344-3344	BIOL	Dr. Monk	A
1	HCI	Bob	211-2112	CSCI	Dr. Pooh	B
2	Python	Maria	344-3344	BIOL	Dr. Patel	A

Constraints (functional dependencies)

- ID → Name
- ID → Phone
- ID → Major
- Course → Professor
- ID, Course → Grade

Students extended: problems

Students (ID, Course, Name, Phone, Major, Professor, Grade)

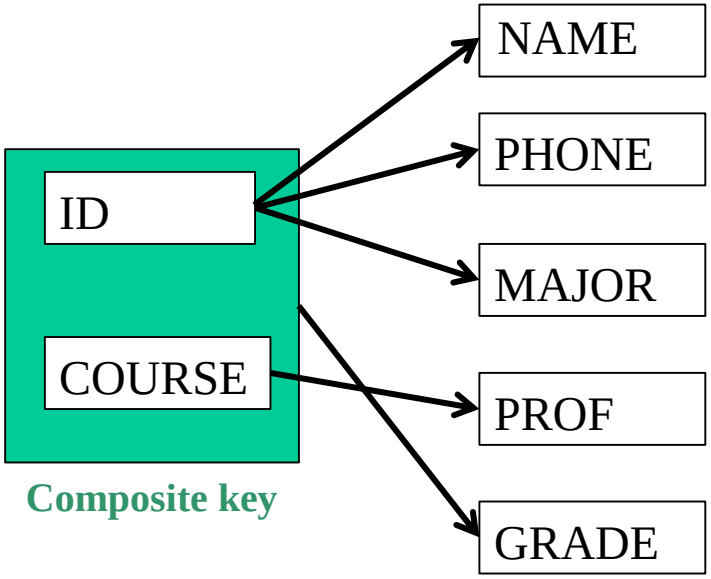
Students						
ID	**Course**	**Name**	**Phone**	**Major**	**Prof**	**Grade**
1	**Databases**	Bob	211-2112	CSCI	Dr. Monk	In pr
2	**HCI**	Maria	344-3344	BIOL	Dr. Pooh	A
3	**Python**	John	500-5005	MATH	Dr. Patel	B
4	**HCI**	Tom	601-6778	PHYS	Dr. Pooh	A
2	**Algorithms**	Maria	344-3344	BIOL	Dr. Monk	A
1	**HCI**	Bob	211-2112	CSCI	Dr. Pooh	B
2	**Python**	Maria	344-3344	BIOL	Dr. Patel	A

- Redundancy
- Insertion anomaly
- Deletion anomaly
- Update anomaly

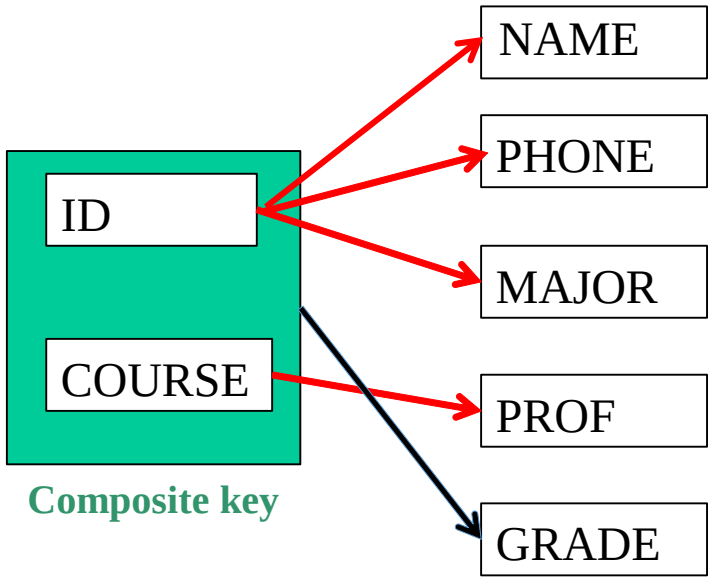
Second normal form: 2NF

- The relation is in 1NF
- All non-key attributes are **fully dependent on the key** (no attributes depend on a part of a key)

Functional dependency diagram

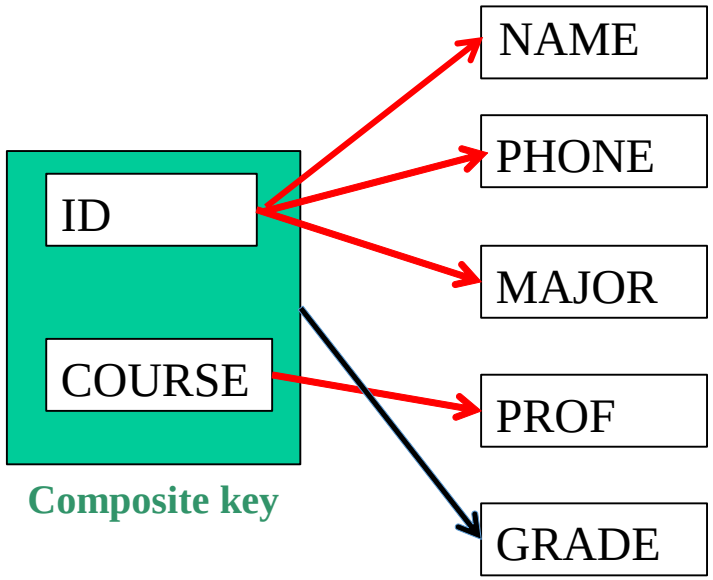


Extended Students: in 2NF?



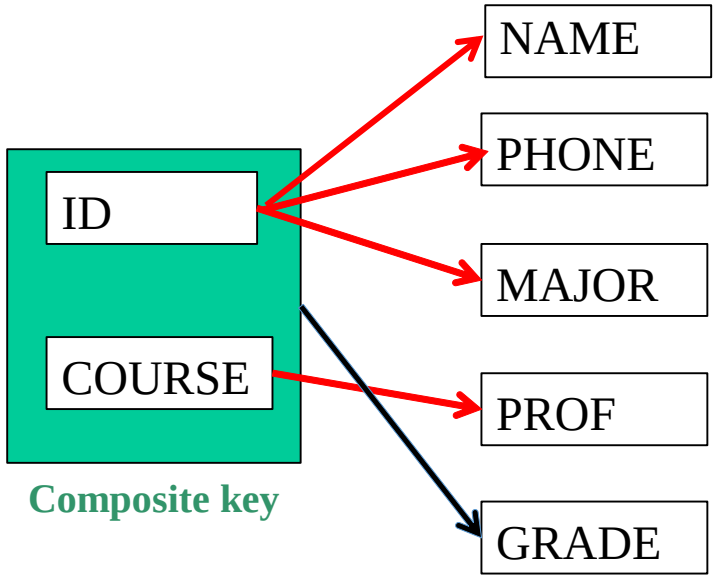
Partial dependencies

Extended Students: in 2NF?



- Solution: decompose all partial dependencies into separate relations

Students in 2NF



Students

ID	Name	Phone	Major
1	Bob	211-2112	CSCI
2	Maria	344-3344	BIOL
3	John	500-5005	MATH
4	Tom	601-6778	PHYS

Grades

ID	Course	Grade
1	Databases	In pr
2	HCI	A
3	Python	B
4	HCI	A
2	Algorithms	A
1	HCI	B
2	Python	A

Courses

Course	Prof
Databases	Dr. Monk
HCI	Dr. Pooh
Python	Dr. Patel
Algorithms	Dr. Monk

Students (<u>ID</u>, Name, Phone, Major)

Courses (<u>Course</u>, Prof)

Grades (<u>ID</u>, <u>Course</u>, Grade)

Students relation: new information

Students (ID, Name, Phone, Major, Department)

Students				
ID	**Name**	**Phone**	**Major**	**Department**
1	Bob	211-2112	CSCI	Computer Science
2	Maria	344-3344	BIOL	Life Sciences
3	John	500-5005	MATH	Mathematics and Statistics
4	Tom	601-6778	PHYS	Physics
5	Andrew	222-2341	CSCI	Computer Science
6	Ann	544-6778	STAT	Mathematics and Statistics

New constraint:

Major → Department

Students relation: new information

Students (ID, Name, Phone, Major, Department)

Students				
ID	**Name**	**Phone**	**Major**	**Department**
1	Bob	211-2112	CSCI	Computer Science
2	Maria	344-3344	BIOL	Life Sciences
3	John	500-5005	MATH	Mathematics and Statistics
4	Tom	601-6778	PHYS	Physics
5	Andrew	222-2341	CSCI	Computer Science
6	Ann	544-6778	STAT	Mathematics and Statistics

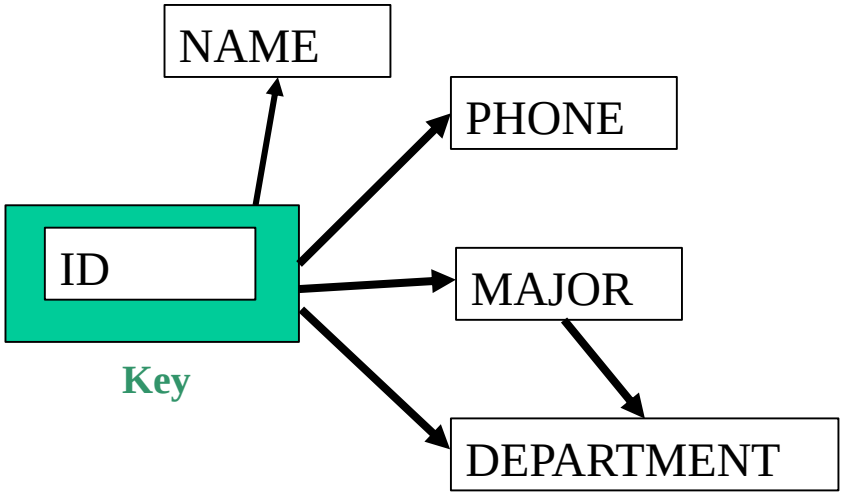
- Redundancy
- Update anomalies

Major → Department

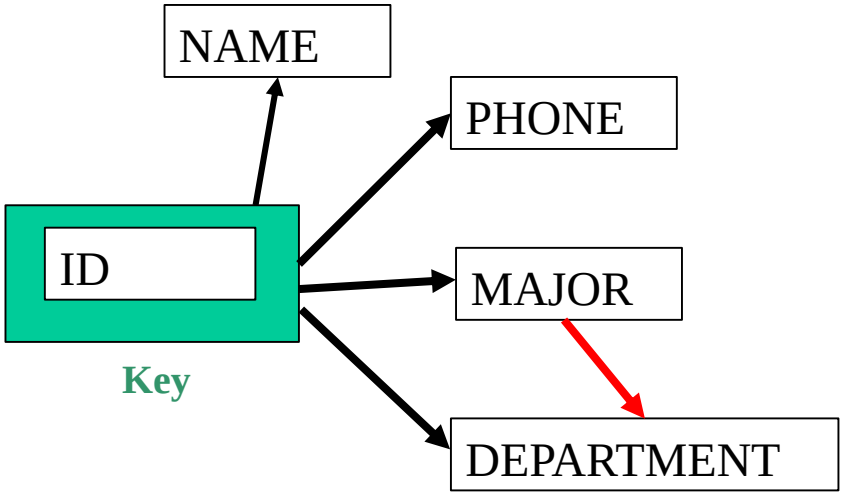
Third normal form 3NF

- The relation is in 2NF
- All attributes are functionally dependent only on the key: **no dependency on** another **non-key attribute**

Functional dependency diagram

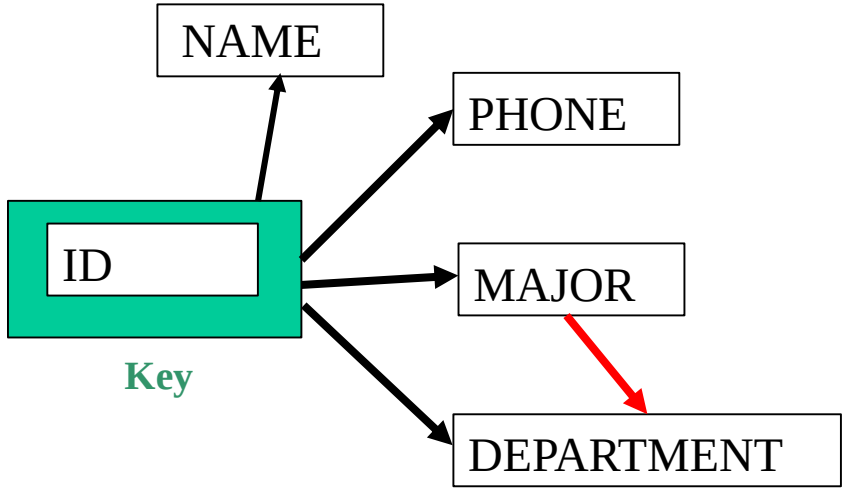


Functional dependency diagram



Transitive dependency

Students in 3NF



Students			
ID	**Name**	**Phone**	**Major**
1	Bob	211-2112	CSCI
2	Maria	344-3344	BIOL
3	John	500-5005	MATH
4	Tom	601-6778	PHYS
5	Andrew	222-2341	CSCI
6	Ann	544-6778	STAT

MajorsOffered	
Major	Department
CSCI	Computer Science
BIOL	Life Sciences
PHYS	Physics
MATH	Mathematics and Statistics
STAT	Mathematics and Statistics

Students (ID, Name, Phone, Major)

MajorsOffered (Major, Department)

Boyce-Codd normal form - BCNF

- Relation is in 3NF
- All attributes **depend on the key, full key and nothing but the key**

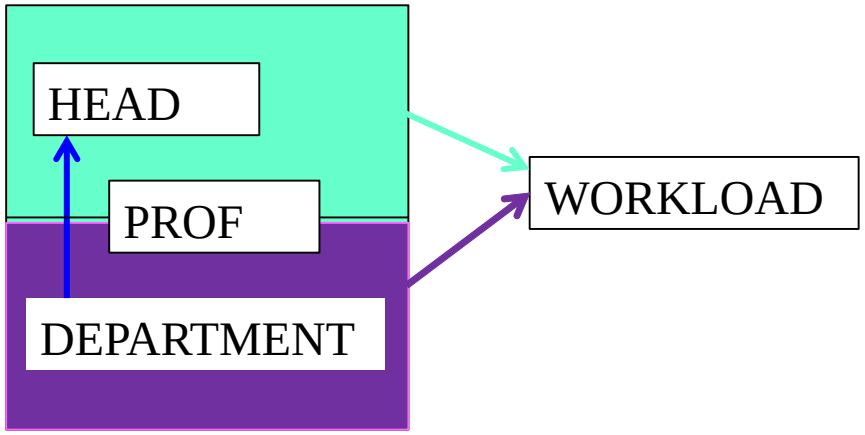
Professor workload: in BCNF?

	Professors		
Prof	**Department**	**Head**	**WorkLoad**
Dr. Monk	CSCI	Prof. Ming	30%
Dr. Pooh	MATH	Prof. Doe	70%
Dr. Patel	PHYS	Prof. Bond	100%
Dr. Pooh	CSCI	Prof. Ming	30%
Dr. Monk	BIOL	Prof. Bond	30%
Dr. Monk	MATH	Prof. Doe	40%

Department → Head

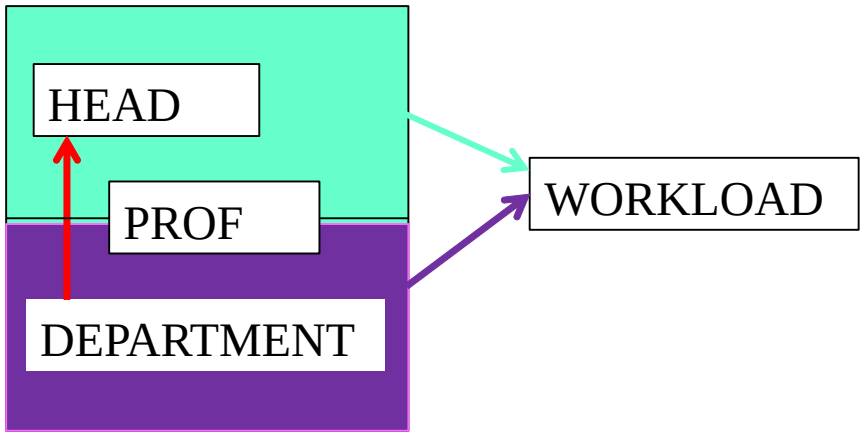
Prof, Department → Workload

Functional dependency diagram



Two overlapping composite candidate keys

Functional dependency diagram



Two overlapping composite candidate keys

- BCNF violation: part of two candidate keys depends on another part

Professors in BCNF

Professors		
Prof	**Department**	**WorkLoad**
Dr. Monk	**CSCI**	30%
Dr. Pooh	**MATH**	70%
Dr. Patel	**PHYS**	100%
Dr. Pooh	**CSCI**	30%
Dr. Monk	**BIOL**	30%
Dr. Monk	**MATH**	40%

Department	
Department	**Head**
CSCI	Prof. Ming
MATH	Prof. Doe
PHYS	Prof. Bond
BIOL	Prof. Bond

Professors (Prof, Department, Workload)

Department (Department, Head)

Apply

Orders								
Order ID	**Customer**	**Phone**	**Item1**	**Qty1**	**Price1**	**Item2**	**Qty2**	**Price2**
1	Bob	211-2112	Pen	2	5.49	Eraser	1	2.00
2	John	344-3344	Pen	5	5.49			
3	Bob	211-2112	Pen	1	5.49			
4	Maria	500-5005	Eraser	3	2.00			

Relationship between normal forms