Chapter 1 Lecture 1

Sunday, February 18, 2018 9:01 AM

DataBase: a set of logically related files (created using the Same Programming Language / suit or can interact with each other)

example:

Student / Java Prof / C++ Regis / Pro's

Data: Stored Value

Example: Name: Fatimah, ID: 201300111, Grade: 30

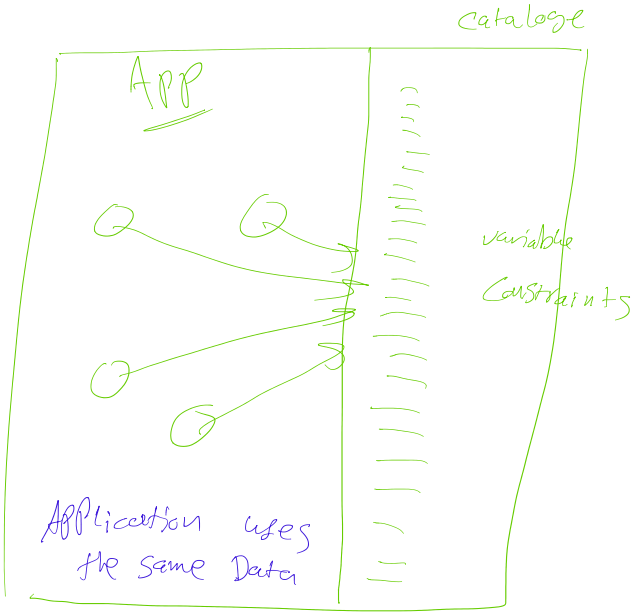
slide 1-3

- Numbers like temperture / texts
- like videos
- like maps
- Data that Shored long time "Historical Data"
- Real-time: Response at time

slide 1-5

-
- Back up
-
- Mulity uses

Slide 1-9



Slide 1-10

Model: a representation of real life situation

Chapter 1 Lecture 2

Tuesday, February 20, 2018 9:02 AM

The model : Description of something

Data Abstraction: High level of Description

Concurrent: Accesses the same Item

unessacry repetion called redundancy

Chapter 2 Lecture 2

Tuesday, February 20, 2018 9:28 AM

Slide 2-4

- not showing models
- Exact models
- Exact specification

Slide 2-9

Schem: Description ex: student Id student name, address, tell Phone

Instances: example such as: (1(1', Fatima, dolu) 555

Slide 2-10

Initial State: when it's Null

valid State: free from errors

Slide 2_

Data Difinition Language: Language that create to Database.

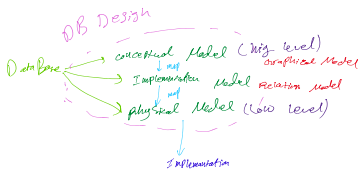
(DDL)

slide 2-23

Centrlized means The Database in one place

Chapter 3 lecture 3

Thursday, February 22, 2018 9:03 AM



Entities

In a hospital, we have patients, Doctors, Departments, ward, nurses, for patients we'd like to store Pna, age, diagnosis, admitdate.

Doctors have: Dna, Did, Ddep, depna, dep tel, dep ward no, Ward, have: id, no of room. Nurses have: eno, ename, sal.

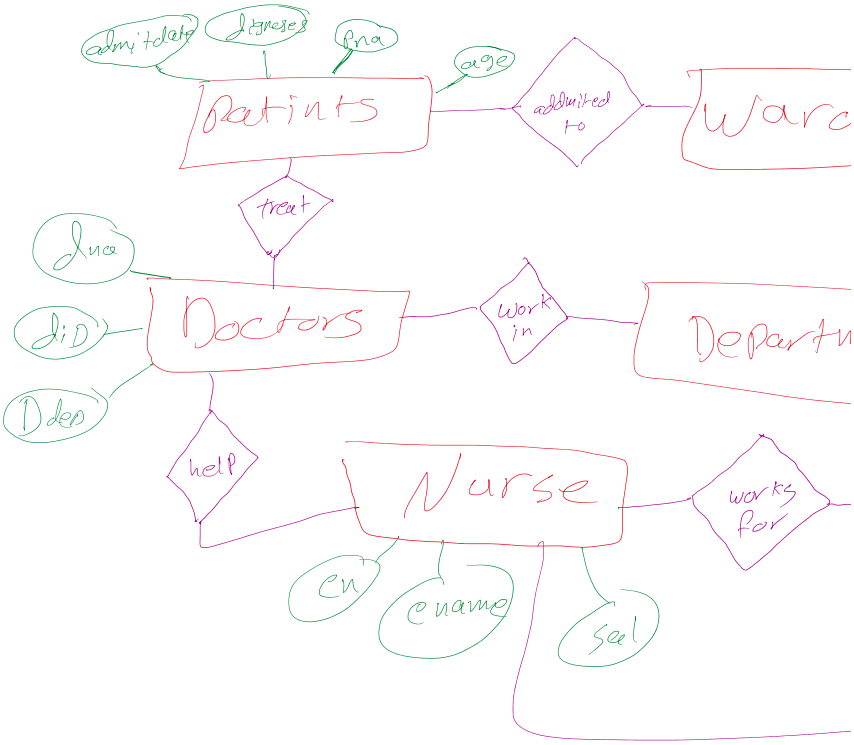
Nurses help doctors. doctors treat patients. Doctors work in department. Nurse work for department. patients admitted to ward. Nurse assigned to ward

Attributes

Relationship

Entity Relationship Diagram:

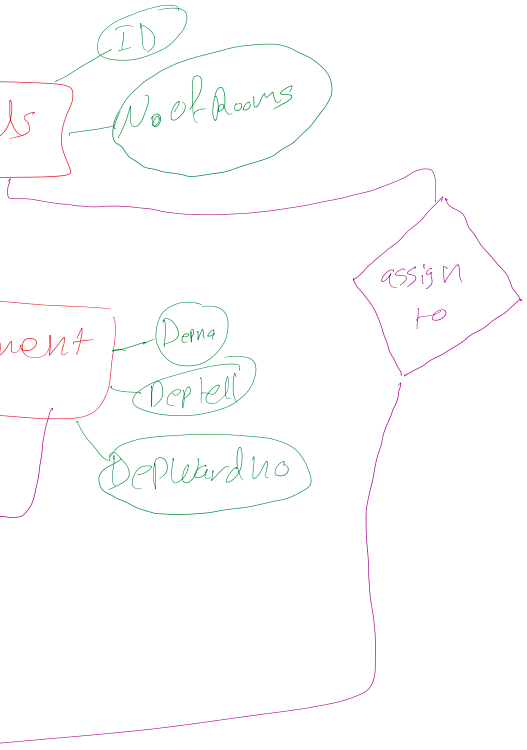
A tool used to made a DB at a conceptal level (high level)



Entity: Something we like to store Information about

Attributes: Characteristics of entities.

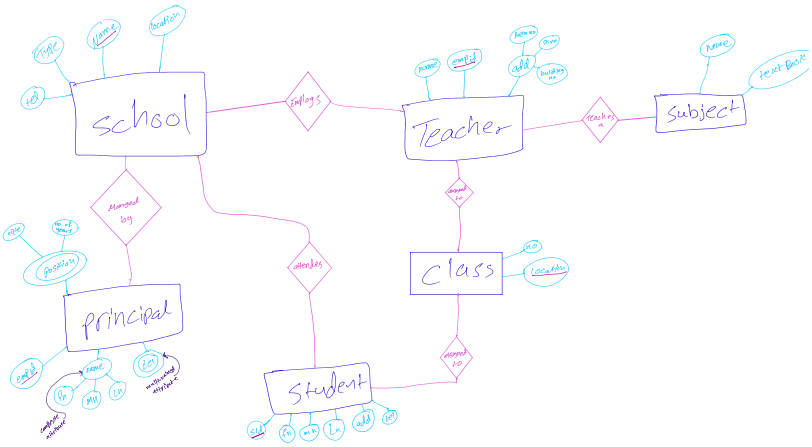
Relationship: Association Between entities


ID
No of Rooms
assign to
Dept Name
Dept Id
Dept Head No

Chapter 3 Lecture 4

Sunday, February 25, 2018 9:01 AM

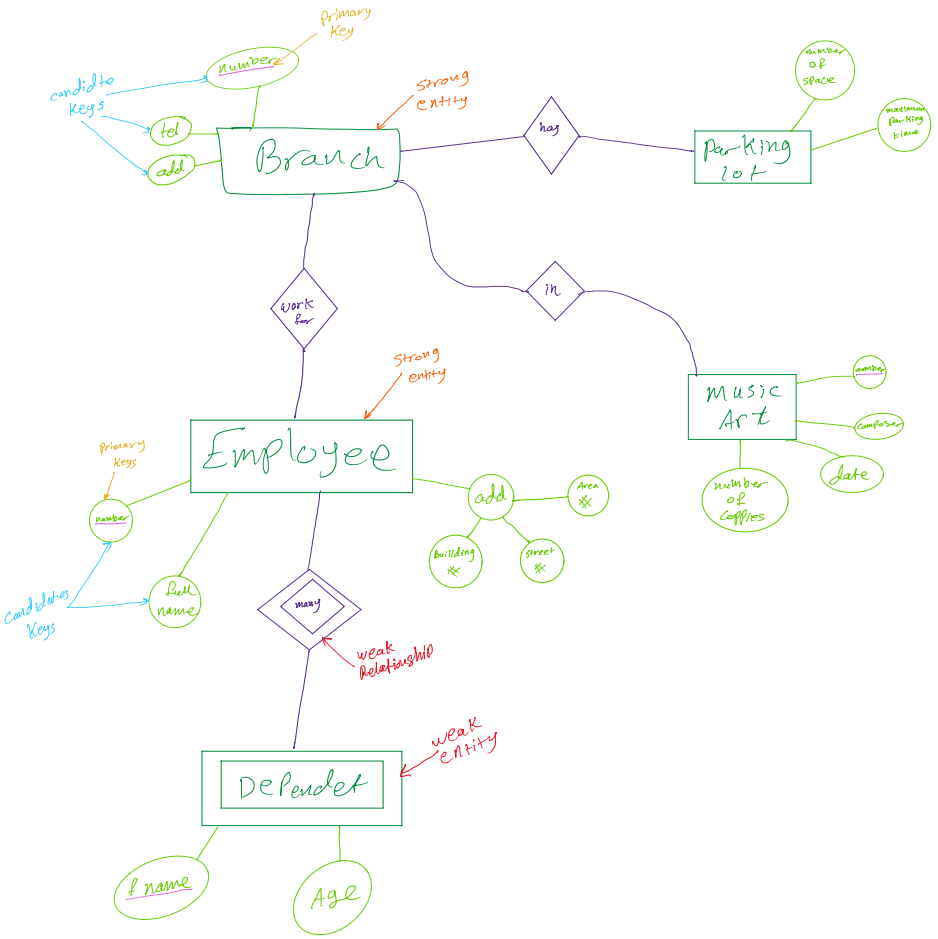
A city has many schools of various stages (Eg. Elementary, Primary, secondary). So, each school has a type, unive name, location, tel. Each school is managed by a principal who has emp id, a name composed of fn, mn, ln, tel. at least two contact numbers. A principal must have at least 10 years experience. Solid working history showing many postings. Each position is composed of title, No. years. Each school employs teachers. Each teacher has a name, empid, add. The address is composed of building No, St no, Area no. Each student attend, the school close to his/her house. Each student has sid, fn, mn, ln, add, tel. students are assigned to classes. Each class has a no, location. Teachers are assigned to classes. Each Teacher teaches at least 3 different classes, & 3-5 subjects. Each subject has name & textbook



Chapter 3 Lecture 5

Tuesday, February 27, 2018 9:04 AM

A large music chain consists of many branches. Each Branch has unique number, add, tel. Each Branch must have a parking lot associated with it for customers convinience. A parking lot has number of spaces, maximum parking time. Each brach has a number of employees working in it. An employee has an employee number, full name, adds composed of building#, street#, Area#. If an employee is married then the company keeps information about his/her dependets such as fname, age. Each music art in a branch has a unize number among all other Product in all branches, but the same in all branches. An Art has also composer, date, number of copies, the date is used to calculate age of work which is also recorded.



Chapter 3 Lecture 6

Thursday, March 1, 2018 9:05 AM

A national bank has 25 Branches

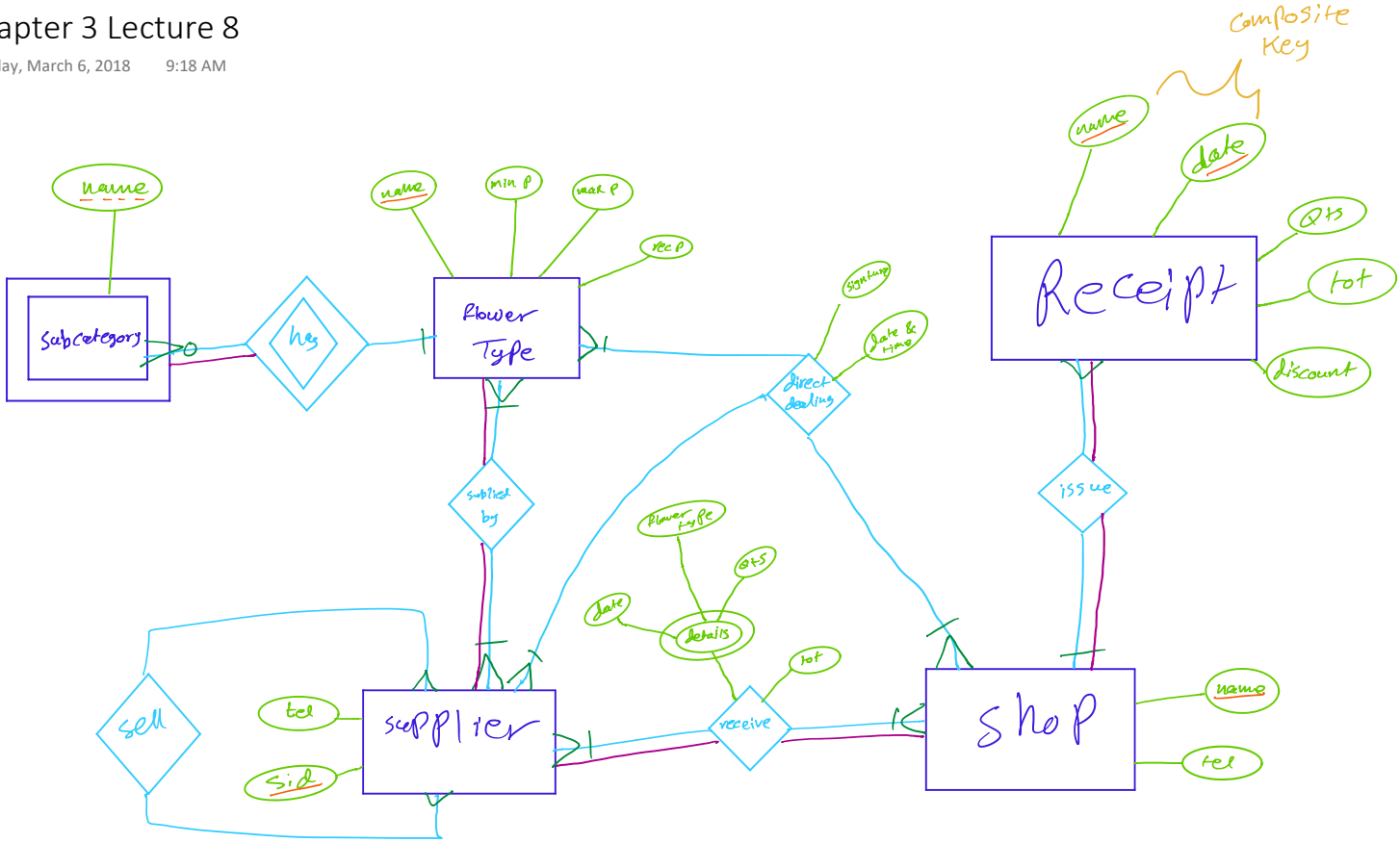
Each branch is identified by bnum,
& has at least 3 telphone numbers.

Chapter 3 Lecture 7

Tuesday, March 6, 2018 9:17 AM

Chapter 3 Lecture 8

Tuesday, March 6, 2018 9:18 AM



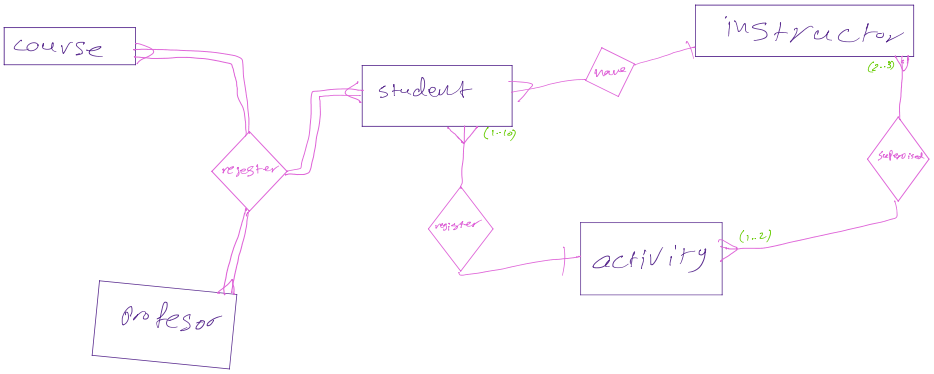
Chapter 3 Lecture 9

Thursday, March 8, 2018 9:07 AM

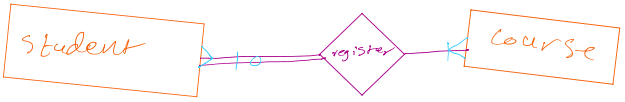
Many more than one

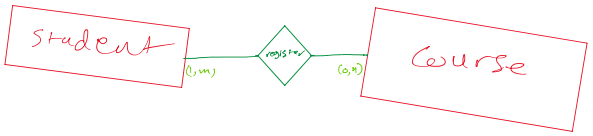
In the following, attributes are omitted for simplicity.

Students register in courses with professor. A student may register in many courses. A course may be given by many professors & have students registered with him / her. Students register in activities supervised by instructors. A student may register in one activity only, hence have one instructor. An activity may have many (upto 3) instructors & many students registered in it. An instructor (max 3) may supervise more than one activity & many students (max 10)



A student must register in one or more courses. A course may have zero or more student registerd in it





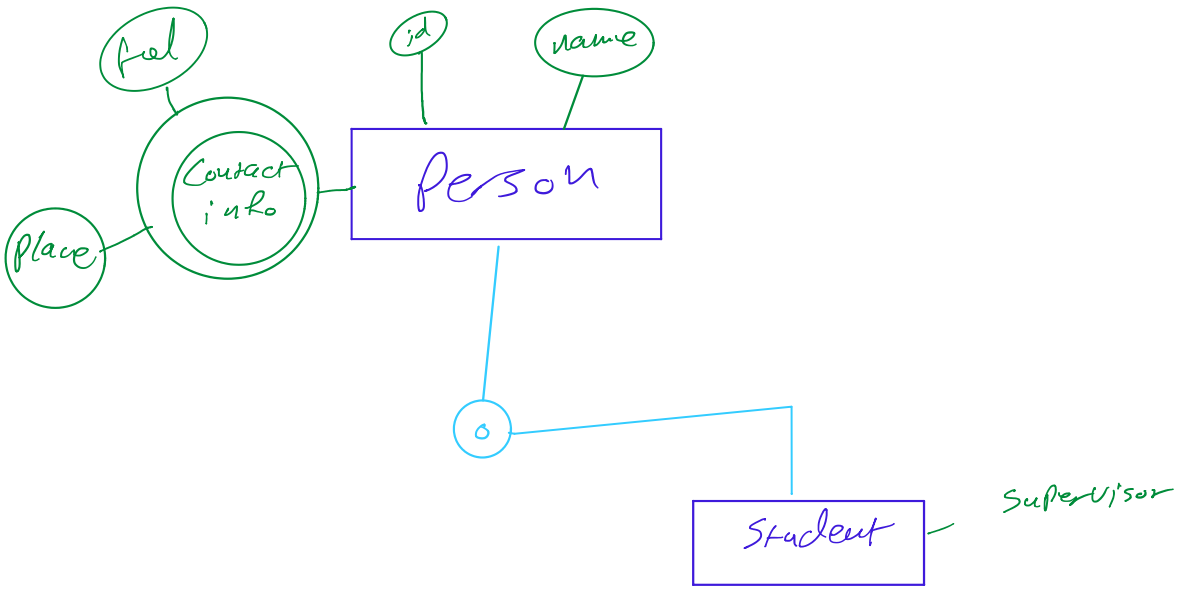
Lecture 10

Sunday, March 11, 2018 9:10 AM

Student at A A university
Can be undergrad or postgrad.
an undergrad student has
Sid, name, tel, Program major.
A Postgrad has sid, name, tel,
speci., Supervisor. A Postgrad enrolls
in Postgrad Course & their
regist

Lecture 11

Thursday, March 15, 2018 9:06 AM



Lecture 12

Tuesday, March 20, 2018 9:24 AM

PR(Pna, Pid, Cno

Co(Cno, Cna, Ch,

ST(Sno, Sna, build, St, name

Act(name, fee,

INS(Iid, ina, name

Sec(Scode, Cno,

Supp. mat. (type, price, S. code

Qualt(Qualit, Pid

teach (Pid, Sno

Assisn (Scode, Cno, Pid, room, time

register (Pid, Sno, Cno, time

Reg (days, pid, cno, sno

Lecture 13

Sunday, March 25, 2018 9:14 AM

A ($\underline{a_1}$, a_2, a_4, a_5)

B (b_1, $\underline{a_1}$)

C (c_1, $\underline{a_1}$)

G (g_1, g_2, b_1, c_1, $\underline{a_1}$, d_1, d_2)

D ($\underline{d_1}$, $\underline{d_2}$, d_3)

E ($\underline{e_1}$, e_2, $\underline{d_1}$, $\underline{d_2}$, f_1, f_5)

F ($\underline{f_1}$, f_2)

R_6 (r_{G1}, r_{G2}, b_1, $\underline{d_1}$, $\underline{d_2}$, d_3, $\underline{a_1 b}$, $\underline{a_1 g}$)

R_4 ($\underline{e_4}$, $\underline{f_1}$, $\underline{a_1}$)

R_1 (a_{11}, a_{12})

Lecture 14

Sunday, April 22, 2018 9:11 AM

AA Tel. company likes to store it's customer's Bills in 4NF Relational model. Show your work ONE STEP AT A TIME.

A sample Bill

Tel: YYYYY
Customer Name: ABC
Bill #: 123
date: xxxxxxx

AA Company
BB City
123 street 111222

Local Calls:

Answer:

Bill(name, Billno, date, email, Tel, {Lnumber, Lduration, Cost, date}*, {Inum, Iduration, Country, rate, Cost, date}*, LTotal, ITotal, Bill Total)

remove whats repeated:

Bill(name, Billno, date, email, Tel, Lnumber, Lduration, LTime, Cost, date, Inum, Iduration, Country, rate, Cost, date, ITime, LTotal, ITotal, Bill Total)

Function Determans:

Tel ⟶ name, email

Bill no, date ⟶ Tel, email, name, Bill Total, I total, L total

L duration ⟶ Cost

I duration, Country ⟶ cost

Country ⟶ rate

Bill is in 1 NF

Decompose Bill:

Bill1(billno, date, name, email, Tel, Ltotal, Itotal, Bill total)

Bill2(Lduration, cost)

Bill3(Country, rate)

Bill4(Iduration, Cost)

Lecture 15

Tuesday, April 24, 2018 9:03 AM

ID:
Report Date: (RD)

Car license plate (CLP)	Traffic Violation Name (TVN)	Date	Time	Amount
C1 xxxx	Speed	23/2/2018	9:00	200
C2 yyyy	parking	25/3/2019	9:00	500

Total to be paid xxx

Ticket (ID, RD, {CLP, TVN, Date, Time, Amount}*, total)

Ticket is not in 1NF relation (repeating rules)

Ticket (ID, RD, CLP, TVN, Date, Time, Amount, total)

ID, RD → Total

TVN → Amount

Date, Time, CLP → TVN, Amount

CLP → ID

CLP, RD → total

Ticket is in 1NF by Difinition Relation

Ticket is not in 2NF
not every non-key attribute depend on the whole key
CLP → ID

T_1(CLP, ID) 1✓ 2✓ 3✓

T_2(CLP, date, time, TVN, Amount) 1✓ 2✓ 3✗ CLP, Date, time → TVN → Amount

T_3(CLP, RD, total) 1✓ 2✓ 3✓

T_2 is not in 3NF
Transitive dependency

T_{21}(CLP, date, time, TVN)

T_{22}(TVN, amount)

The final model is composed of

T_1, T_3, T_{21}, T_{22} "All in 3NF"

Also in BCNF

Also in 4NF

Lecture 16

Thursday, April 26, 2018 9:12 AM

R

A_1	A_2	A_3	A_4	A_5
B_1	C_1	D_1	E_1	F_1
B_1	C_1	D_1	E_2	F_2
B_2	C_1	D_1	E_1	F_1
B_2	C_2	D_1	E_2	F_2
B_3	C_3	D_2	E_1	F_2
B_4	C_4	D_1	E_2	F_1

$A_1 \longrightarrow A_3$

$A_2 \longrightarrow A_3$

$A_1, A_4 \longrightarrow A_2, A_3, A_5$

$A_1, A_5 \longrightarrow A_2, A_3, A_4$

$R(\underline{A_1}, A_2, A_3, A_4, \underline{A_5})$

It's Not in 2NF So

$R_1(\underline{A_1}, A_3)$

$R_2(\underline{A_1}, A_2, A_4, \underline{A_5})$

It's 3NF, BCNF, 4NF

S

A_1	A_2	A_3	A_4	A_5
B_1	C_1	D_1	E_1	F_1

B_1	C_1	D_1	E_1	F_1
B_1	C_1	D_1	E_2	F_2
B_2	C_1	D_1	E_1	F_1
B_2	C_2	D_1	E_2	F_2
B_3	C_3	D_2	E_1	F_2
B_4	C_4	D_1	E_2	F_1
B_1	C_3	D_3	E_6	F_1

$S_1(\underline{A_1}, A_3)$

$S_2(\underline{A_1}, \underline{A_5}, A_4, A_2)$ Decompose → $S_{21}(\underline{A_1}, \underline{A_5}, A_4)$

→ $S_{22}(\underline{A_1}, \underline{A_5}, A_2)$

S#	C#	Tel#
S1	351	7771111
S1	352	7771111
S2	451	6666666
S2	456	4444444
S2	451	4444444
S2	452	6666666

Lecture 17

Sunday, April 29, 2018 9:03 AM

$R(A, B, C, D, E, F)$

FD

$CD \rightarrow E$, $CD \rightarrow F$, $B \rightarrow E$

* select the wrong one "dose not holt"

$A \rightarrow E$ $\quad$ $CD \rightarrow EF$

$AD \rightarrow F$ $\quad$ $B \rightarrow CD$

* select the

$A \rightarrow C$ $\quad$ $B \rightarrow A$

$CD \rightarrow A$ $\quad$ $C \rightarrow D$

RA:

σ_θ select: selcts tuples from a table or relation based on a condition θ

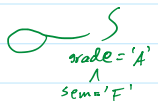
S

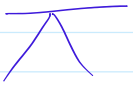
Sno	sem	Cno	grade
1	F	351	A
1	F	351	A
1	S	451	B
2	S	451	B
2	S	111	C
2	F	351	A
3	S	451	B

select form S where grade =A

$\sigma_{grad='A'}(S)$

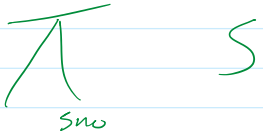
select from S where grade = A and sem : F





Project: select attributes from a relation

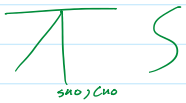
Project sno From S



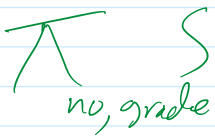
output:

sno
1
2
3

Student with courses they are taking



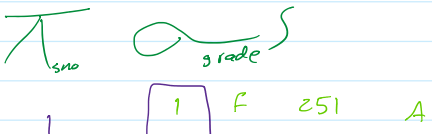
Students with their grade



output:

1	A
2	B
3	B
2	C
2	A
3	B

Sno of students got an A grade:



	sno	grade		
1	1	F	251	A
	1	F	351	A
2	2	F	351	A

X Cartisian Product : Between TWO Relation
contacaterites every tuple of
the first with every
tuple of the seond

A

A_1
A_2

B

B_1
B_2

A x B

$A_1 B_1$
$A_1 B_2$
$A_2 B_1$
$A_2 B_2$

S1

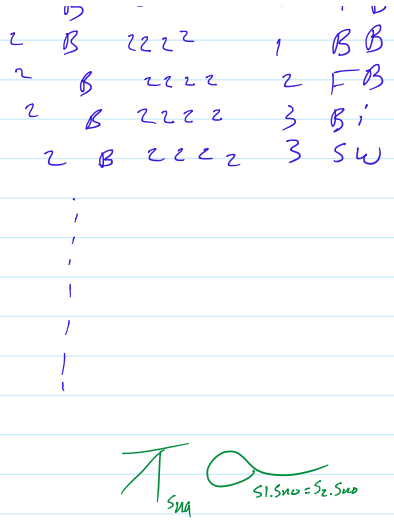
Sno	Sna	Tel
1	A	1111
2	B	2222
3	C	3333
4	D	4444
5	E	5555
6	F	6666

S_2

Sno	Act
1	FB
1	BB
2	FB
3	Bi
3	SW

S_1 X S_2

1	A	1111	1	FB
1	A	1111	1	BB
1	A	1111	2	FB
1	A	1111	3	Bi
1	A	1111	3	SW
2	B	2222	1	FB
2	B	2222	1	BB
2	B	2222	2	FB



SNA
S1.SNO = S2.SNO

Lecture 18

Tuesday, May 1, 2018 9:35 AM

σ_θ R/name

π_{A_1} R/name

M X N

∪ union: 2 relations must be union compatible
- have same number of attributes
- corresponding attributes must have the same domain

ex:

UGS

Sno	Sua
S1	A
S2	B
S3	C

PGS

Pno	Pna
P1	D
P2	E

S1	A
S2	B
S3	C
P1	D
P2	E

— Difference (set difference): Tuples which exist in the first relation but do not exist in the secon relation

ex:

UGS

S1	A
S2	B
S3	C

PGS

P1	D
P2	E

S_2	B
S_3	C
P_1	D

P_2	E

UGS – PGS

S_1	A
S_2	B
S_3	C

PGS – UGS

P_2	E

Exercise:

EMP

eno	ena	supno
e_1	A	e_3
e_2	B	e_3
e_3	C	e_1
e_4	D	e_1
e_5	E	e_1

Π_{eno} EMP

output

eno
A
B
C
D
E

Lecture 19

Sunday, May 6, 2018 8:46 AM

S

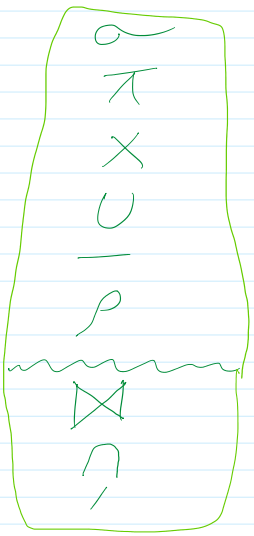
Sno	Sna	Supno
e_1	A	e_3
e_2	B	e_5
e_3	C	e_4
e_4	D	e_1
e_5	E	e_2

```
select b.sna
from S a, S b
where a.supno = b.sno

select sna from S
where sno in
    select supno from S
```

$$\rho_{S1}\, S$$

$$\pi_{S1.sna}\left(S \bowtie_{S1.supno = S.sno} S1\right)$$



Ex:

Pat(Pno, Pna, Padd)

Dr(Did, Dno, Special)

Treatment(Pid, Did, treat, date)

patients who have same name as one of their drs.

$$\pi_{Pat.Pna}\left(\sigma_{P.Pno = Dr.Dno}\left(Dr \bowtie_{Dr.did = Treatment.did}\left(Treatment \bowtie_{Treatment.Pno = Pat.Pno} Pat\right)\right)\right)$$

names of Dr.s who treated patients 'Fatma'

"Two Solutions"

$$\pi_{Dr.dna}\left(\sigma_{Pat.na = \text{"FatmA"}}\left(Dr \bowtie_{Dr.did = Treatment.did}\left(Treatment \bowtie_{Treatment.Pno = Pat.Pno} Pat\right)\right)\right)$$

$$\pi_{Dr.dna}\left(Dr \bowtie_{Dr.did = Treatment.did}\left(Treatment \bowtie_{Treatment.Pno = Pat.Pno} \sigma_{Pat.Pna = \text{'FATMA'}}(Pat)\right)\right)$$

$$\pi_{Dname}\left(Dr \underset{Dr.Drid = Treat.Did}{\bowtie}\left(Treat \underset{Pat.Pno = Treat.Pno}{\bowtie}\left(\sigma_{Pname='FATMA'} Pat\right)\right)\right)$$

* Patient number who have recived more than one treatment on the same day

$$\rho_{T_1} Treat$$

$$\pi_{T_1.Pno}\left(T_1 \bowtie Treat\right)$$

$$T_1.Pno = Treat.Pno$$

$$\wedge$$

$$T1.date = treat.date$$

$$\wedge$$

$$T_1.Treat = treat.treat$$

* Patient number who have recived more than one treatment on the same day with the same dr.

$$\rho_{T_1} Treat$$

$$\pi_{T_1.Pno}\left(T_1 \bowtie Treat\right)$$

$T_1.Pno = Treat.Pno$

$\wedge$

$T1.date = treat.date$

$\wedge$

$T_1.Treat = treat.treat$

$\wedge$

$T.Did = treat.Did$

names of patients who have been treated by Dr X OR Dr Y

$\pi_{Pat.Pname}(Pat \bowtie_{Pat.Pno = Treat.Pno} (Dr \bowtie_{Dr.did = Treat.J.id \wedge (Dr.dna = 'X' \vee Dr.dna = 'Y')} Treat))$

another answer :-

$\pi_{Pat.Pname}(Pat \bowtie_{Pat.no = Treat.Pno} (Dr \bowtie_{drdid = treat.did \wedge dr.na = 'X'} Treat)) \cup \pi_{Pat.Pname}(Pat \bowtie_{Pat.no = Treat.Pno} (Dr \bowtie_{drdid = treat.did \wedge dr.na = 'Y'} Treat))$

names of patients who have been treated by Dr X OR Dr Y

$\pi_{Pat.Pname}(Pat \bowtie_{Pat.no = Treat.Pno} (Dr \bowtie_{drdid = treat.did \wedge dr.na = 'X'} Treat)) \cap \pi_{Pat.Pname}(Pat \bowtie_{Pat.no = Treat.Pno} (Dr \bowtie_{drdid = treat.did \wedge dr.na = 'Y'} Treat))$

Example

S

Sno	Sname	Sadd	Act
1	A	UA	BB
2	B	UA	FB
1	A	UA	SW
1	A	UA	FB
2	B	UA	FB
3	C	UA	SW
4	D	UA	SW

Acts

BB
FB
SW

10
4
12
15
16
20

3/5

Lecture 20

Tuesday, May 15, 2018 9:15 AM

Pat(Pno, Pna, add

Dr(did, dna, add, dep, sal

Treat(Pno, did, treat, date

- id's of drs paid more than the average of their department?

  select did from dr a where a.sal >

  (select avg(sal) from dr b, a where a.dept = b.dept)

- id's of drs paid more than the average of all drs?

  select did from dr where sal >

  (select Avg(sal) From dr)

CH8-1-Functional Dependency

Tuesday, May 8, 2018 2:12 AM

CH8-1-Functional De...

Functional Dependencies

R&G Chapter 19

Science is the knowledge of consequences, and dependence of one fact upon another.

Thomas Hobbes
(1588-1679)

Review: Database Design

- **Requirements Analysis**
  - user needs; what must database do?
- **Conceptual Design**
  - high level descr (often done w/ER model)
- **Logical Design**
  - translate ER into DBMS data model
- **Schema Refinement**
  - consistency,normalization
- **Physical Design - indexes, disk layout**
- **Security Design - who accesses what**

The Evils of Redundancy

- ***Redundancy:*** **root of several problems with relational schemas:**
  - redundant storage, *insert/delete/update anomalies*
- ***Functional dependencies:***
  - a form of *integrity constraint* that can identify schemas with such problems and 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?

Functional Dependencies (FDs)

- **A functional dependency X → Y holds over relation schema R if, for every allowable instance *r* of R:**

  $t1 \in r,\ t2 \in r,\ \pi_X(t1) = \pi_X(t2)$

  implies $\pi_Y(t1) = \pi_Y(t2)$

  (where t1 and t2 are tuples;X and Y are sets of attributes)

- **In other words: X → Y means**
  Given any two tuples in *r*, if the X values are the same, then the Y values must also be the same. (but not vice versa)
- **Read "→" as "determines"**

FD's Continued

- **An FD is a statement about *all* allowable relations.**
  - Must be identified based on semantics of application.
  - Given some instance *r1* of R, we can check if *r1* violates some FD *f*, but we cannot determine if *f* holds over R.
- **Question: How related to keys?**
- **if "K → all attributes of R" then K is a *superkey* for R**
  (does not require K to be *minimal*.)
- **FDs are a generalization of keys.**

Example: Constraints on Entity Set

- **Consider relation obtained from** Hourly_Emps:
  Hourly_Emps (*ssn, name, lot, rating, wage_per_hr, hrs_per_wk*)
- We sometimes denote a relation schema by listing the attributes: e.g., SNLRWH
- This is really the *set* of attributes {S,N,L,R,W,H}.
- Sometimes, we refer to the set of *all attributes* of a relation by using the relation name. e.g., "Hourly_Emps" for SNLRWH

What are some FDs on Hourly_Emps?

ssn is the key: S → SNLRWH
rating determines *wage_per_hr*: R → W
lot determines *lot*: L → L ("trivial" dependnency)

1

Problems Due to R → W

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Hourly_Emps

- ***Update anomaly*: Should we be allowed to modify W in only the 1st tuple of SNLRWH?**
- ***Insertion anomaly*: What if we want to insert an employee and don't know the hourly wage for his or her rating? (or we get it wrong?)**
- ***Deletion anomaly*: If we delete all employees with rating 5, we lose the information about the wage for rating 5!**

Detecting Reduncancy

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Hourly_Emps

Q: Why was R → W problematic, but S →W not?

Decomposing a Relation

- **Redundancy can be removed by "chopping" the relation into pieces (vertically!)**
- **FD's are used to drive this process.**
  R → W is causing the problems, so decompose SNLRWH into what relations?

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Hourly_Emps2

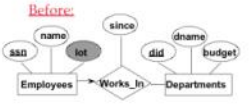
R	W
8	10
5	7

Wages

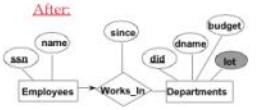
Refining an ER Diagram

- **1st diagram becomes:**
  Workers(S,N,L,D,Si)
  Departments(D,M,B)
  - Lots associated with workers.
- **Suppose all workers in a dept are assigned the same lot: D → L**
- **Redundancy; fixed by:**
  Workers2(S,N,D,Si)
  Dept_Lots(D,L)
  Departments(D,M,B)
- **Can fine-tune this:**
  Workers2(S,N,D,Si)
  Departments(D,M,B,L)

Before:



After:



Reasoning About FDs

- **Given some FDs, we can usually infer additional FDs:**
  title → studio, star implies *title → studio* and *title → star*
  title → studio and *title → star* implies *title → studio, star*
  title → studio, studio → star implies *title → star*

  But,
  title, star → studio does NOT necessarily imply that *title → studio* or that *star → studio*
- **An FD *f* is *implied by* a set of FDs *F* if *f* holds whenever all FDs in *F* hold.**
- **F^+ = *closure of F* is the set of all FDs that are implied by *F*. (includes "trivial dependencies")**

Rules of Inference

- **Armstrong's Axioms (X, Y, Z are sets of attributes):**
  - *Reflexivity*: If $X \supseteq Y$, then $X \rightarrow Y$
  - *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- **These are *sound* and *complete* inference rules for FDs!**
  - i.e., using AA you can compute all the FDs in F+ and only these FDs.
- **Some additional rules (that follow from AA):**
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Example

- **Contracts(*cid,sid,jid,did,pid,qty,value*), and:**
  - C is the key: C → CSJDPQV
  - Proj purchases each part using single contract: JP → C
  - Dept purchases at most 1 part from a supplier: SD → P
- **Problem: Prove that SDJ is a key for Contracts**
- **JP → C, C → CSJDPQV imply JP → CSJDPQV** (by transitivity) (shows that JP is a key)
- **SD → P implies SDJ → JP** (by augmentation)
- **SDJ → JP, JP → CSJDPQV imply SDJ → CSJDPQV** (by transitivity) thus SDJ is a key.

Q: can you now infer that SD → CSDPQV (i.e., drop J on both sides)?

No! FD inference is not like arithmetic multiplication.

Attribute Closure

- **Computing the closure of a set of FDs can be expensive. (Size of closure is exponential in # attrs!)**
- **Typically, we just want to check if a given FD *X → Y* is in the closure of a set of FDs *F*. An efficient check:**
  - Compute *attribute closure* of X (denoted X^+) wrt *F*.
    X^+ = Set of all attributes A such that $X \rightarrow A$ is in F^+
    - $X^+ := X$
    - Repeat until no change: if there is an fd $U \rightarrow V$ in *F* such that U is in X^+, then add V to X^+
  - Check if Y is in X^+
  - Approach can also be used to find the keys of a relation.
    - If all attributes of R are in the closure of X then X is a superkey for R.
    - Q: How to check if X is a "candidate key"?

Attribute Closure (example)

- **R = {A, B, C, D, E}**
- **F = { B →CD, D → E, B → A, E → C, AD →B }**
- **Is B → E in F^+ ?**
  - B^+ = B
  - B^+ = BCD
  - B^+ = BCDA
  - B^+ = BCDAE ... Yes!
    and B is a key for R too!
- **Is D a key for R?**
  - D^+ = D
  - D^+ = DE
  - D^+ = DEC
    ... Nope!
- **Is AD a key for R?**
  AD^+ = AD
  AD^+ = ABD and B is a key, so Yes!
- **Is AD a *candidate* key for R?**
  A^+ = A, D+ = DEC
  ... A,D not keys, so Yes!
- **Is ADE a *candidate* key for R?**
  ... No! AD is a key, so ADE is a superkey, but not a cand. key

Next Class...

- **Normal forms and normalization**
- **Table decompositions**

CH8-2-Functional Dependency

Tuesday, May 8, 2018 2:19 AM

CH8-2-Functional De...

Examples: State FD and Keys

http://jcsites.juniata.edu/faculty/rhodes/dbms/funcdep.htm

Ex. All addresses in the same town have the same zip code

SSN	Name	Town	Zip
1234	Joe	Huntingdon	16652
2345	Mary	Huntingdon	16652
3456	Tom	Huntingdon	16652
5948	Harry	Alexandria	16603

SSN	Name	Address	Hobbies
1111	Joe	123 Main	hiking
1111	Joe	123 Main	biking
2222	Mary	321 Elm	lacross

- ***{stuId}→{lastName}***, but not the reverse
- ***{stuId} →{lastName, major, credits, status, socSecNo, stuId}***
- ***{socSecNo} →{stuId, lastName, major, credits, status, socSecNo}***

- ***{credits}→{status}***, but not ***{status}→{credits}***

Trivial Functional Dependency

The FD X→Y is ***trivial*** if set {Y} is a subset of set {X}

Examples: If A and B are attributes of R,

- {A}→{A}
- {A,B} →{A}
- {A,B} →{B}
- {A,B} →{A,B}

are all trivial FDs and will not contribute to the evaluation of normalization.

http://www.rlvision.com/blog/method-for-determining-candidate-keys-and-highest-normal-form-of-a-relation-based-on-functional-dependencies/

```
R(A,B,C)
A → B
B → {A,C}
```

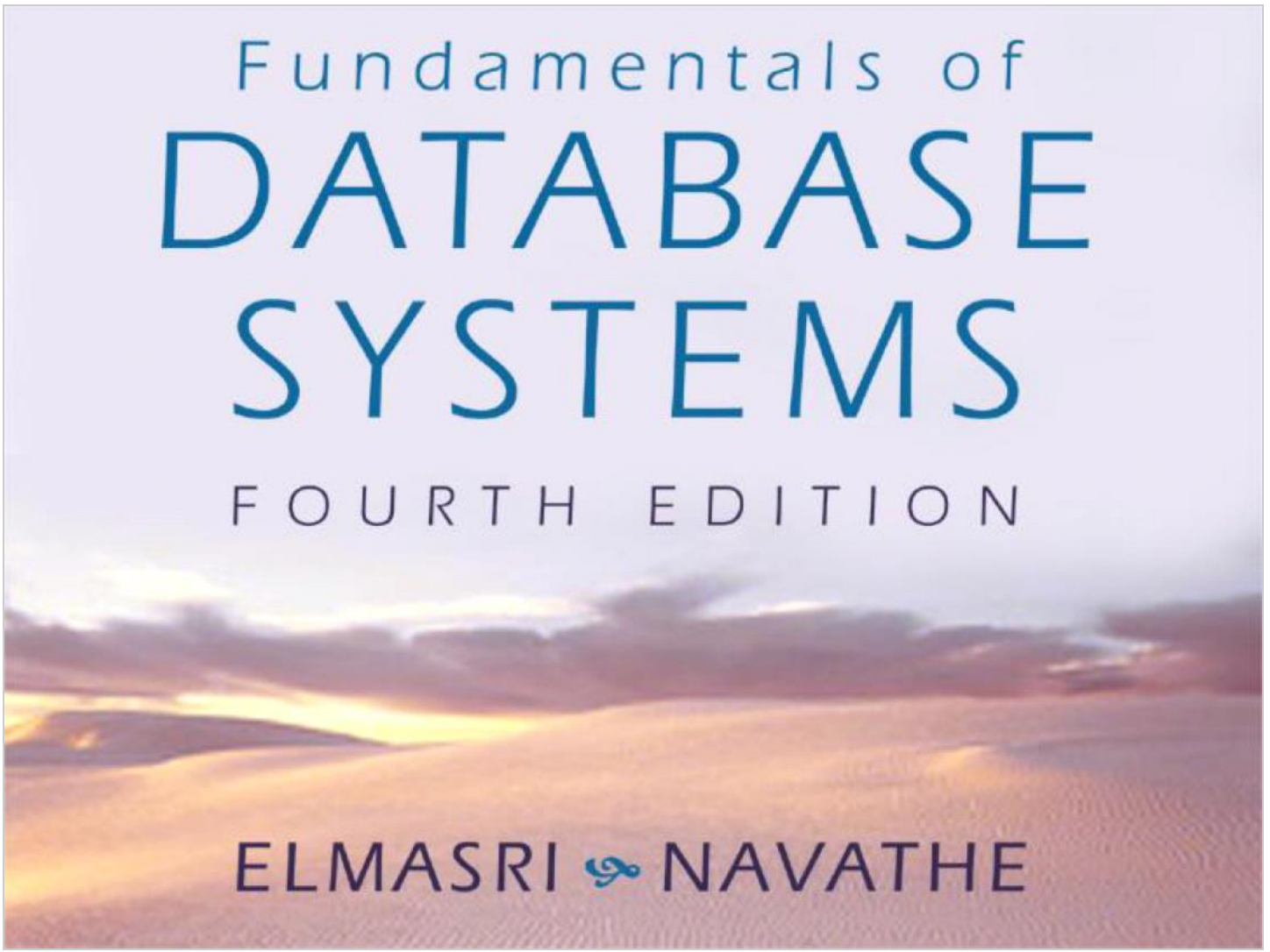
```
R(A,B,C,D,E,F)
A → B
B → A
{B,C} → D
C → E
```

```
R(A,B,C,D,E)
A → {B,C}
{B,C} → A,D
D → E
```

CH8-3-Normalization

Tuesday, May 8, 2018 2:19 AM

CH8-3-Normalization



Chapter 10

Functional Dependencies and Normalization for Relational Databases

Chapter Outline

1 Informal Design Guidelines for Relational Databases
- 1.1Semantics of the Relation Attributes
- 1.2 Redundant Information in Tuples and Update Anomalies
- 1.3 Null Values in Tuples
- 1.4 Spurious Tuples

2 Functional Dependencies (FDs)
- 2.1 Definition of FD
- 2.2 Inference Rules for FDs
- 2.3 Equivalence of Sets of FDs
- 2.4 Minimal Sets of FDs

Chapter Outline(contd.)

3 Normal Forms Based on Primary Keys

- 3.1 Normalization of Relations
- 3.2 Practical Use of Normal Forms
- 3.3 Definitions of Keys and Attributes Participating in Keys
- 3.4 First Normal Form
- 3.5 Second Normal Form
- 3.6 Third Normal Form

4 General Normal Form Definitions (For Multiple Keys)

5 BCNF (Boyce-Codd Normal Form)

1 Informal Design Guidelines for Relational Databases (1)

- What is relational database design?

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

Informal Design Guidelines for Relational Databases (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Normal Form)
  - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 11

1.1 Semantics of the Relation Attributes

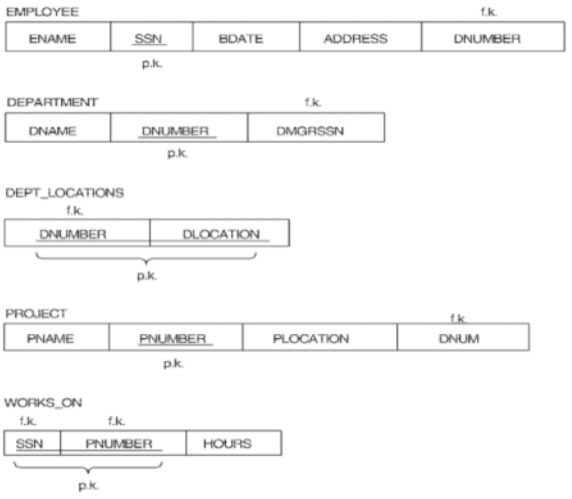
GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- 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.

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

Figure 10.1 A simplified COMPANY relational database schema

Figure 14.1 Simplified version of the COMPANY relational database schema.



Note: The above figure is now called Figure 10.1 in Edition 4

1.2 Redundant Information in Tuples and Update Anomalies

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

EXAMPLE OF AN UPDATE ANOMALY (1)

Consider the relation:

EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from "Billing" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project P1.

EXAMPLE OF AN UPDATE ANOMALY (2)

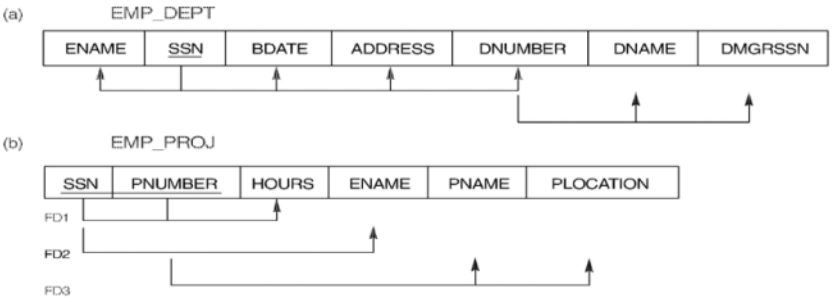
- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .

  Inversely - Cannot insert an employee unless an he/she is assigned to a project.

- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Figure 10.3 Two relation schemas suffering from update anomalies

Figure 14.3 Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.



Note: The above figure is now called Figure 10.3 in Edition 4

Figure 10.4 Example States for EMP_DEPT and EMP_PROJ

Figure 14.4 Example relations for the schemas in Figure 14.3 that result from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
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

Note: The above figure is now called Figure 10.4 in Edition 4

Guideline to 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 present, then note them so that applications can be made to take them into account

1.3 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

1.4 Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations

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.

Spurious Tuples (2)

There are two important properties of decompositions:

(a) non-additive or losslessness of the corresponding join

(b) preservation of the functional dependencies.

Note that property (a) is extremely important and *cannot* be sacrificed. Property (b) is less stringent and may be sacrificed. (See Chapter 11).

2.1 Functional Dependencies (1)

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

Functional Dependencies (2)

- X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
- For any two tuples t1 and t2 in any relation instance r(R): *If* t1[X]=t2[X], *then* t1[Y]=t2[Y]
- X -> Y in R specifies a *constraint* on all relation instances r(R)
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints (1)

- social security number determines employee name

  SSN -> ENAME
- project number determines project name and location

  PNUMBER -> {PNAME, PLOCATION}
- employee ssn and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} -> HOURS

Examples of FD constraints (2)

- 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 t1[K]=t2[K])

2.2 Inference Rules for FDs (1)

- Given a set of FDs F, we can *infer* additional FDs that hold whenever the FDs in F hold

Armstrong's inference rules:

IR1. (**Reflexive**) If Y *subset-of* X, then X -> Y

IR2. (**Augmentation**) If X -> Y, then XZ -> YZ
(Notation: XZ stands for X ∪ Z)

IR3. (**Transitive**) If X -> Y and Y -> Z, then X -> Z

- IR1, IR2, IR3 form a *sound* and *complete* set of inference rules

Inference Rules for FDs (2)

Some **additional inference rules** that are useful:

(**Decomposition**) If X -> YZ, then X -> Y and X -> Z

(**Union**) If X -> Y and X -> Z, then X -> YZ

(**Psuedotransitivity**) If X -> Y and WY -> Z, then WX -> Z

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Inference Rules for FDs (3)

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

2.3 Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if:
  - every FD in F can be inferred from G, *and*
  - every FD in G can be inferred from F
- Hence, F and G are equivalent if $F^+ = G^+$

Definition: F **covers** G if every FD in G can be inferred from F (i.e., if G^+ *subset-of* F^+)

- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

2.4 Minimal Sets of FDs (1)

- A set of FDs is **minimal** if it satisfies the following conditions:

(1) Every dependency in F has a single attribute for its RHS.

(2) We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.

(3) We cannot replace any dependency X -> A in F with a dependency Y -> A, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.

Minimal Sets of FDs (2)

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set (e.g., see algorithms 11.2 and 11.4)

3 Normal Forms Based on Primary Keys

3.1 Normalization of Relations

3.2 Practical Use of Normal Forms

3.3 Definitions of Keys and Attributes Participating in Keys

3.4 First Normal Form

3.5 Second Normal Form

3.6 Third Normal Form

3.1 Normalization of Relations (1)

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

Normalization of Relations (2)

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies : MVDs; 5NF based on keys, join dependencies : JDs (Chapter 11)
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; Chapter 11)

3.2 Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to **detect**
- The database designers ***need not*** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)
- **Denormalization:** the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

3.3 Definitions of Keys and Attributes Participating in Keys (1)

- A **superkey** of a relation schema $R = \{A_1, A_2, ..., A_n\}$ 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 (2)

- 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.

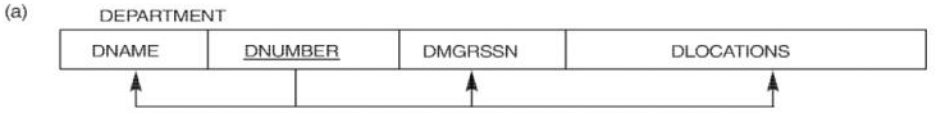
3.2 First Normal Form

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

- Considered to be part of the definition of relation

Figure 10.8 Normalization into 1NF

Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.



(b)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

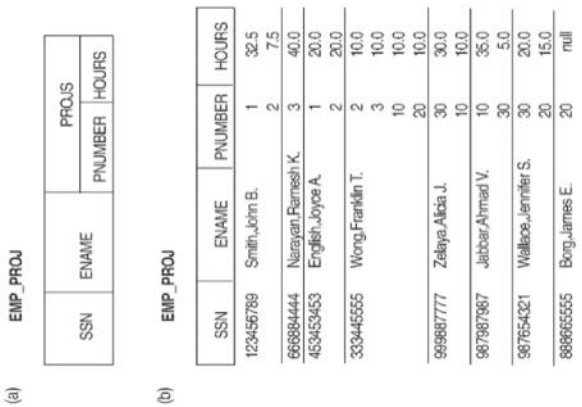
DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Note: The above figure is now called Figure 10.8 in Edition 4

Figure 10.9 Normalization nested relations into 1NF

Figure 14.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.



Note: The above figure is now called Figure 10.9 in Edition 4

3.3 Second Normal Form (1)

- Uses the concepts of **FD**s, **primary key**

Definitions:

- **Prime attribute** - attribute that is member of the primary key K
- **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Examples: - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

- {SSN, PNUMBER} -> ENAME is *not* a full FD (it is called a *partial dependency*) since SSN -> ENAME also holds

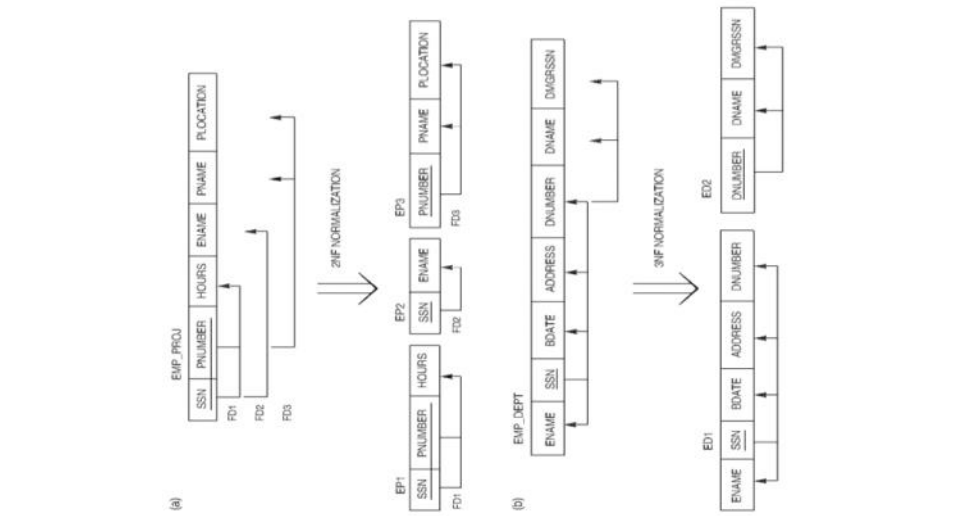
Second Normal Form (2)

- 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

Figure 10.10 Normalizing into 2NF and 3NF

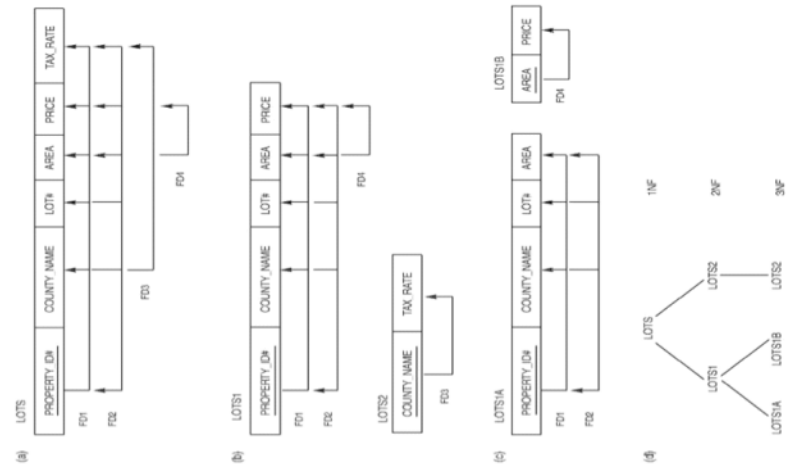
Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



Note: The above figure is now called Figure 10.10 in Edition 4

Figure 10.11 Normalization into 2NF and 3NF

Figure 14.11 Normalization to 2NF and 3NF. (a) The lots relation schema and its functional dependencies fd1 through FD4. (b) Decomposing lots into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of lots.



Note: The above figure is now called Figure 10.11 in Edition 4

3.4 Third Normal Form (1)

Definition:

- **Transitive functional dependency** - a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

  Examples:

  - SSN -> DMGRSSN is a *transitive* FD since

  SSN -> DNUMBER and DNUMBER -> DMGRSSN hold

  - SSN -> ENAME is *non-transitive* since there is no set of attributes X where SSN -> X and X -> ENAME

Third Normal Form (2)

- 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 -> Y and Y -> 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 -> Emp# -> Salary and Emp# is a candidate key.

4 General Normal Form Definitions (For Multiple Keys) (1)

- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R

General Normal Form Definitions (2)

Definition:

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form (3NF)** if whenever a FD X -> A holds in R, then either:

  (a) X is a superkey of R, or

  (b) A is a prime attribute of R

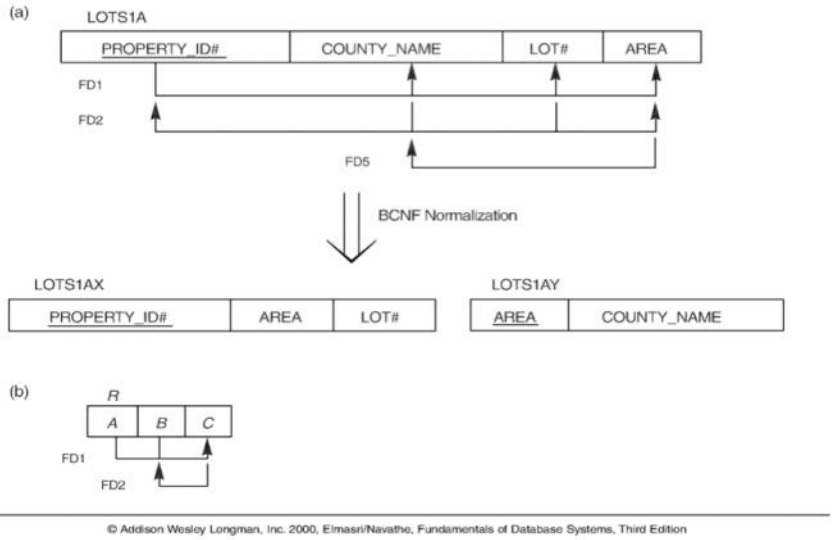
NOTE: Boyce-Codd normal form disallows condition (b) above

5 BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Figure 10.12 Boyce-Codd normal form

Figure 14.12 Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being "lost" in the decomposition. (b) A relation *R* in 3NF but not in BCNF.



Note: The above figure is now called Figure 10.12 in Edition 4

Figure 10.13 a relation TEACH that is in 3NF but not in BCNF

Figure 14.13 A relation TEACH that is in 3NF but not in BCNF.

TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Note: The above figure is now called Figure 10.13 in Edition 4

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:

  fd1: { student, course} -> instructor

  fd2: instructor -> course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 10.12 (b). So this relation is in 3NF but not in BCNF
- A relation **NOT** in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations. (See Algorithm 11.3)

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  1. {student, instructor} and {student, course}
  2. {course, instructor } and {course, student}
  3. {instructor, course } and {instructor, student}
- All three decompositions will lose fd1. We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity property after decomposition.
- Out of the above three, only the 3rd decomposition will not generate spurious tuples after join.(and hence has the non-additivity property).
- A test to determine whether a binary decomposition (decomposition into two relations) is nonadditive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.