

Normal Forms

Ramakrishnan & Gehrke, Chapter 17 & 18

The Evils of Redundancy

Dept_id	budget	Emp_id	Emp_name	salary
1	100	1	John Williams	60
1	100	2	Phil Coulter	50
2	200	3	Norah Jones	45
3	300	4	Anastacia	40

- **Redundancy** at the root of several relational schema problems
 - redundant storage, insert/delete/update anomalies
- **Integrity constraints** identify problems and suggest refinements
 - in particular: functional dependencies

Functional Dependencies

- Let R be relation, X and Y sets of attributes of R
- Functional dependency (FD)** $X \rightarrow Y$ holds over relation R

if, for every allowable instance r of R :

- $t1 \in r, t2 \in r$:
 $\pi_X(t1) = \pi_X(t2) \Rightarrow \pi_Y(t1) = \pi_Y(t2)$
- FDs in example?

Dept_id	budget	Emp_id	Emp_name	salary
1	100	1	John Williams	60
1	100	2	Phil Coulter	50
2	200	3	Norah Jones	45
3	300	4	Anastacia	40

- K is a **candidate key** for R means that $K \rightarrow R$
 - $K \rightarrow R$ does not require K to be minimal!
- FD is a statement about **all** allowable relation instances
 - Must be identified **based on semantics** of application
 - Given some allowable instance $r1$ of R ,
 we **can check if it violates** some FD f , but we **cannot tell if f holds** over R !

Example: Constraints on Entity Set

- Consider relation obtained from Hourly_Emps:
 - Hourly_Emps (ssn, name, lot, rating, hrly_wages, hrs_worked)
- **Notation:** relation schema by listing the attributes: SNLRWH
 - **set** of attributes {S,N,L,R,W,H}
 - Using equivalently to relation name (e.g., Hourly_Emps for SNLRWH)
- Some FDs on Hourly_Emps:
 - ssn is key: $S \rightarrow \text{SNLRWH}$
 - rating determines hrly_wages: $R \rightarrow W$

Example (Contd.)

- Problems due to $R \rightarrow W$:
 - **Update anomaly:**
change W in just the 1st tuple of SNLRWH?
 - **Insertion anomaly:**
insert employee and don't know the hourly wage for his rating?
 - **Deletion anomaly:**
delete all employees with rating 5
⇒ lose information about the wage for rating 5!

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

Wages

Hourly_Emps2

R	W
8	10
5	7

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

Will 2 smaller tables be better?

Normal Forms & Functional Dependencies

- normal forms avoid / minimize certain kinds of problems
 - helps to decide on decomposing relation
- Role of FDs in detecting redundancy
 - No FDs hold: no redundancy
 - Given relation R with 3 attributes ABC and FD $A \rightarrow B$:
Several tuples might have the same A value; if so, they **all have the same B value**

It's all about hidden repeating information across tuples

First Normal Form

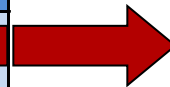
■ First Normal Form (1NF)

- eliminates attributes containing sets = **repeating groups**
- ...by flattening: introduce separate tuples with atomic values

■ Ex:

id	name	skillsList
1	Jane	{C,C++,SQL}
2	John	{Java,python,SQL}

- Skills not f.d. on id, nor name!



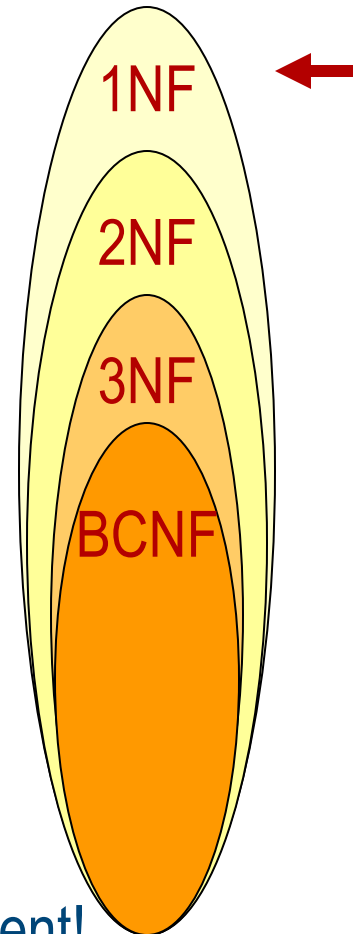
id	name	skill
1	Jane	C
1	Jane	C++
1	Jane	SQL
2	John	Java
2	John	Python
2	John	SQL

■ Oops: lost primary key property.

- Will fix that later.

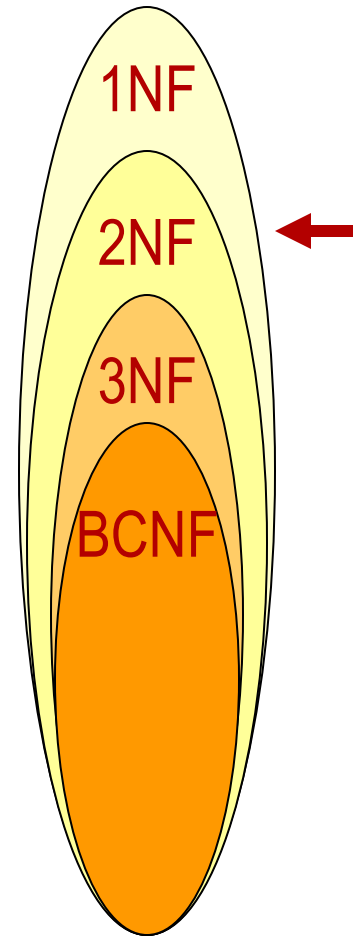
■ Why good? Repeating groups complicate storage management!

- Experimental DBMSs exist for non-1NF (NFNF, NF²) tables



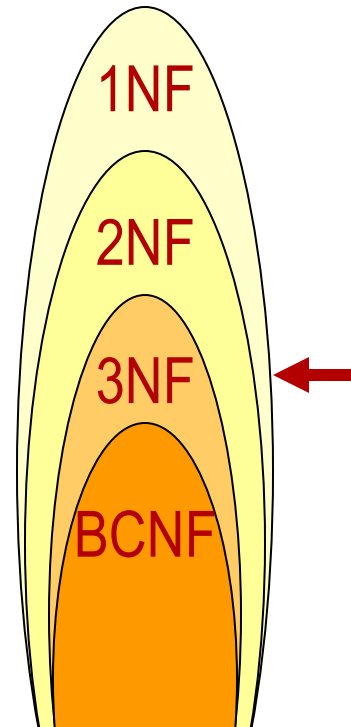
Second Normal Form

- Second Normal Form (2NF):
 - eliminates functional dependencies on a partial key
 - by putting the fields in a separate table from those that are dependent on the whole key
- Ex: ABCD with $B \rightarrow C$
becomes: ABD, BC



Third Normal Form (3NF)

- Relation R with FD set F is in **3NF** if, for all $X \rightarrow A$ in F^+ ,
 - Either $A \in X$ (called a *trivial* FD)
 - Or X contains a key for R
 - Or A is part of some key for R
- In plain words:
 - 3NF eliminates functional dependencies on non-key fields by putting them in a separate table
 - = in 3NF, all non-key fields are dependent on *the key, the whole key, and nothing but the key*



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

• Ex:

Why Is 3NF Good?

- If 3NF **violated** by $X \rightarrow A$, one of the following holds:
- X subset of some key K
 - We store (X, A) pairs **redundantly**
- X not a proper subset of any key
 - Which means: for some key K , there is a chain of FDs $K \rightarrow X \rightarrow A$
 - Which means: we once introduced keys to capture dependencies, but now we have **attributes dependent on a non-key attribute!**
- ...so **non-3NF** means **dangerous updates!**

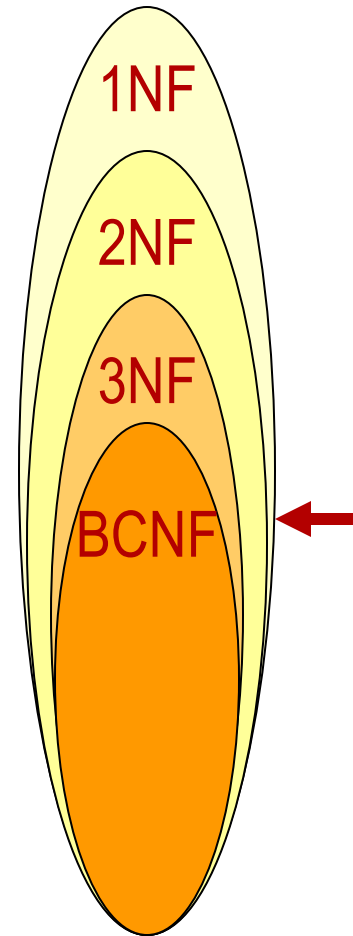
What Does 3NF **NOT** Achieve?

- Some redundancy possible with 3NF
- Ex: Reserves SBDC, $S \rightarrow C$, $C \rightarrow S$
 - is in 3NF
 - but $S \leftrightarrow C$ means:
for each reservation of sailor S , same (S, C) pair is stored
- ...so we still need to capture "nests" **inside** the keys

Boyce-Codd Normal Form (BCNF)

- Relation R with FDs F is in **BCNF** if, for all $X \rightarrow A$ in F^+ ,
 - Either $A \in X$ (called a *trivial* FD)
 - Or X contains a key for R
 - ~~• Or A is part of some key for R~~

- In other words:
 R in BCNF $\Leftrightarrow$ only key-to-nonkey constraints FDs left
 - ✓ = No redundancy in R that can be detected using FDs alone
 - ✓ = No FD constraints "hidden in data"



Discussion: 3NF vs. BCNF

- Always possible?
 - 3NF **always possible**, is “nice” (lossless-join, dependency-preserving)
 - BCNF **not always possible**
- 3NF compromise used when BCNF not achievable
 - Ex: performance considerations
 - Ex: cannot find “good” decomp (see next)

Decomposition of a Relation Scheme

- Given relation R with attributes $A_1 \dots A_n$
- **decomposition** of R = replacing R by two or more relations such that:
 - Each new relation scheme contains a subset of the attributes of R (and no additional attributes), and
 - Every attribute of R appears as an attribute of one of the new relations
- E.g., decompose SNLRWH into SNLRH and RW

Example Decomposition

- SNLRWH has FDs
 $S \rightarrow \text{SNLRWH}$, $R \leftrightarrow W$, $N \rightarrow \text{SN}$
- 2nd FD causes **3NF violation**:
W values repeatedly associated with R values (and vice versa)!
- Easiest fix: create relation RW to store assoc's w/o dups, remove W from main schema
= **decompose** SNLRWH into SNLRH and RW

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

Wages

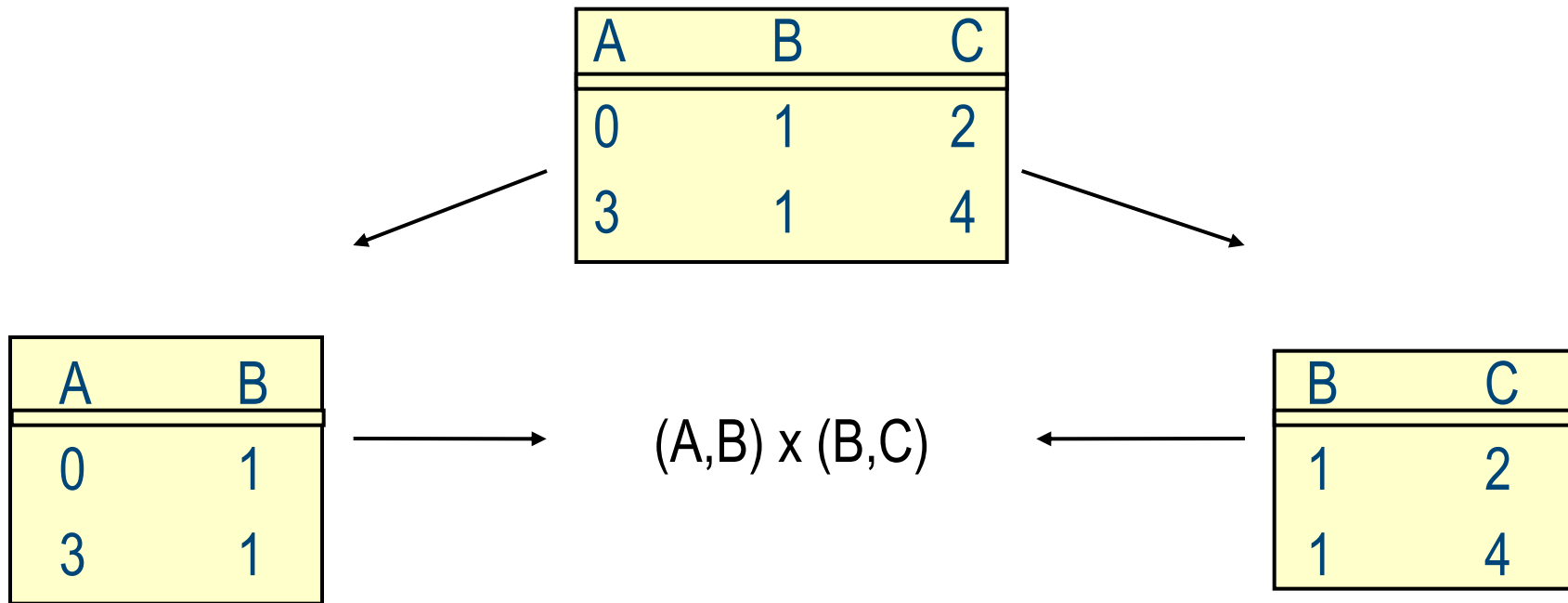
Hourly_Emps2

R	W
8	10
5	7

If we just store projections of SNLRWH tuples onto SNLRH and RW, are there any potential problems?

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

Lossless Join: A Counter Example



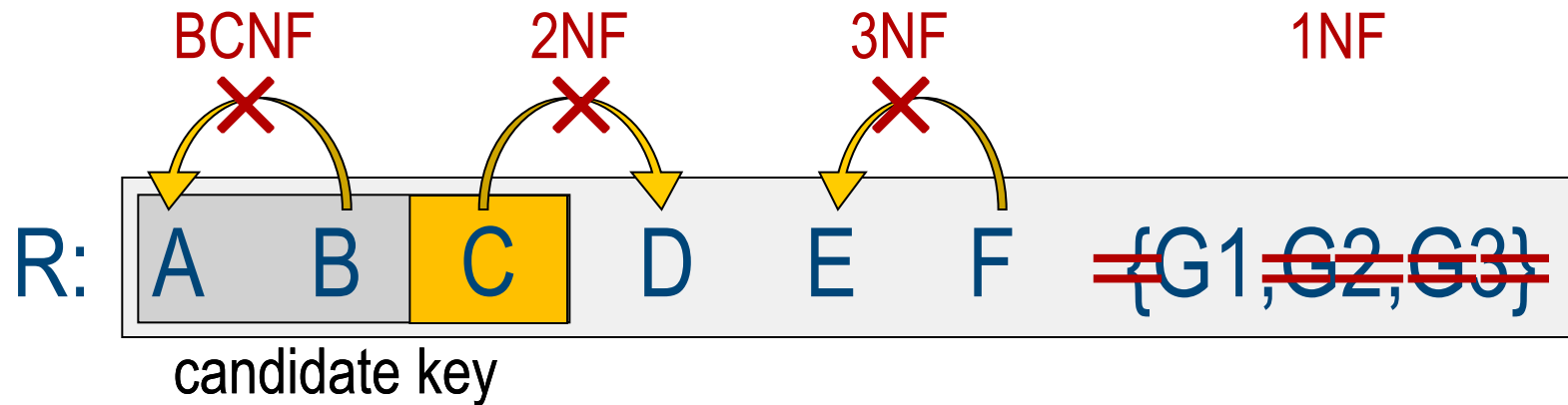
What's wrong?

Summary of Schema Refinement

- BCNF = free of redundancies that can be detected using FDs
 - BCNF good heuristic (consider typical queries!)
 - *Check FDs !*
 - Next best: 3NF
- When not BCNF?
 - not always possible
 - unsuitable, given typical queries - performance requirements
- Use decompositions only when needed!

👉 NF pocket guide

Pocket Guide to NFs



- 1NF = no repeating groups
- 2NF = 1NF + no partial key $\rightarrow$ non-key
- 3NF = 2NF + no non-key $\rightarrow$ anything
- BCNF = 3NF + no key $\rightarrow$ key

Normalization of table R with FD set :

- For all FDs $F = „X \rightarrow Y“$:
 - Create additional table $R_F(X, Y)$
 - Remove Y from R, but keep X
- Drop duplicate tables arising from „ $X \rightarrow Y, Y \rightarrow X$ “ cycles
- Crosscheck all new tables created against all FDs for decomposition need