

CS354: Database

Universal Relation

- Put every attribute that you need to store into one single (huge) relation
- Example: Company database model
COMPANY(SSN, name, ..., dno, dname, ..., pno, pname, ..., dept_name, ... , dependent_name, ...)
- What is so bad about this relation?

Are these Bad Designs?

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland

When is a Relation “Good” or “Bad”?

- Database researches have found a number of bad properties called anomalies
- When a relation exhibits one or more of these anomalies, the relation is deemed bad
- CAVEAT:
 - “Good” relations can be inefficient
 - DB designers may decide to use “bad” relations for performance reasons, but need to take precaution to make sure “bad” things do not happen

Database Anomalies: Insert Anomaly

- Normal behavior of inserting ONE item of information
 - One tuple is introduced in one or more tables
 - No NULL values are introduced
- Insert anomaly occurs when inserting ONE item of information
 - Multiple tuples into some relation
 - Needs to use NULL values

Example: Insert Anomaly

Relation to represent information about employees and departments

<u>SSN</u>	FName	LName	DNo	DName	MgrSSN
111-11-1111	John	Smith	5	Research	123-45-6789
222-22-2222	Jane	Doe	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	1	Payroll	777-77-7777

What if a new department is introduced (dno = 6, dname = "Administration") that does not have any employees yet?

<u>SSN</u>	FName	LName	DNo	DName	MgrSSN
NULL	NULL	NULL	6	Administration	NULL

Database Anomalies: Delete Anomaly

- Normal behavior of deleting ONE item of information
 - One tuple is removed in one or more tables
 - Only intended information is deleted and does not cause loss of additional information
- Delete anomaly occurs when deleting ONE item of information
 - Deletes multiple tuples into some relation
 - Causes additional (unintended) information

Example: Delete Anomaly

<u>SSN</u>	FName	LName	DNo	DName	MgrSSN
111-11-1111	John	Smith	5	Research	123-45-6789
222-22-2222	Jane	Doe	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	1	Payroll	777-77-7777

What if Jack Rabbit leaves the company?

DELETE employee WHERE fname = 'Jack' AND lname = 'Rabbit';

<u>SSN</u>	FName	LName	DNo	DName	MgrSSN
111-11-1111	John	Smith	5	Research	123-45-6789
222-22-2222	Jane	Doe	5	Research	123-45-6789

Payroll department is also deleted!

Database Anomalies: Update Anomaly

- Normal behavior of updating ONE item of information
 - One tuple in one or more tables is updated
- Update anomaly occurs when updating ONE item of information
 - Updates multiple tuples from some relation

Example: Update Anomaly

<u>SSN</u>	FName	LName	DNo	DName	MgrSSN
111-11-1111	John	Smith	5	Research	123-45-6789
222-22-2222	Jane	Doe	5	Research	123-45-6789
333-33-3333	Jack	Rabbit	1	Payroll	777-77-7777

What if manager of research department changes?

UPDATE employee SET MgrSSN = '888-88-8888'
WHERE DName = 'Research';

<u>SSN</u>	FName	LName	DNo	DName	MgrSSN
111-11-1111	John	Smith	5	Research	888-88-8888
222-22-2222	Jane	Doe	5	Research	888-88-8888
333-33-3333	Jack	Rabbit	1	Payroll	777-77-7777

Operation has modified multiple tuples in single relation!

Generation of Spurious Tuples

- Natural join results in more tuples than “expected”
- Represents spurious information that is not valid
- Example: What happens during a natural join?

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford

Example: Generation of Spurious Tuples

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.

Asterisk denotes the tuples that don't make sense

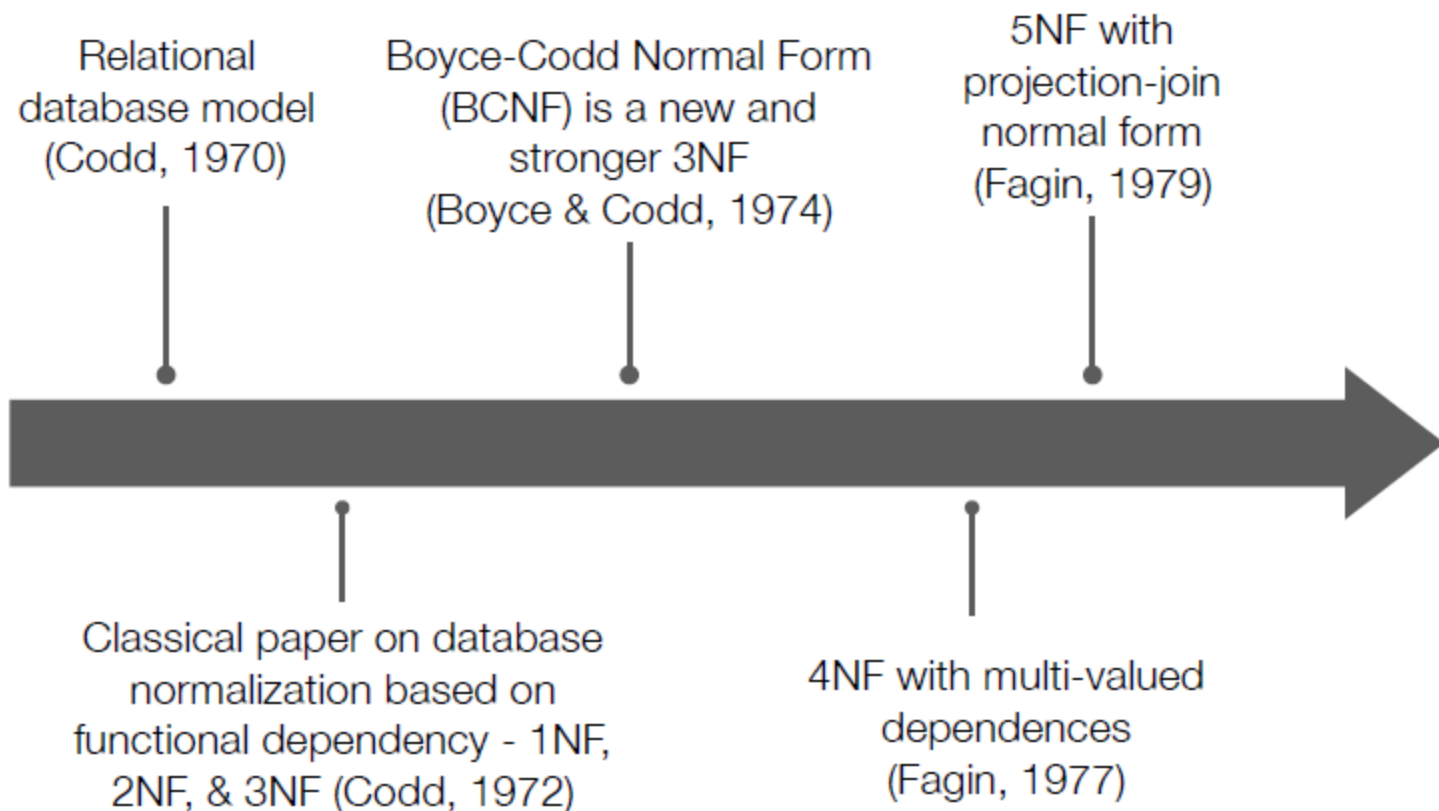
Informal Design Guidelines

- Design relations where meaning of a relation's attributes can be easily explained — avoid combining multiple entity types and relationship types into a single relation
- Avoid insertion, deletion, and update anomalies — minimize redundant information
- Reduce NULL values in tuples — use space efficiently and avoid joins with NULL values
- Design relation schemas to guarantee no spurious tuples — avoid relations that contain matching attributes that are not (foreign key, primary key) combinations

Formal Database Design Theory

- Normal forms
 - Set of properties that relations must satisfy
 - Successively higher degrees of stringency
- Database normalization
 - Certify whether a database design satisfies a certain normal form
 - Correct designs to achieve certain normal form

History of Database Design



Relationship amongst Normal Forms

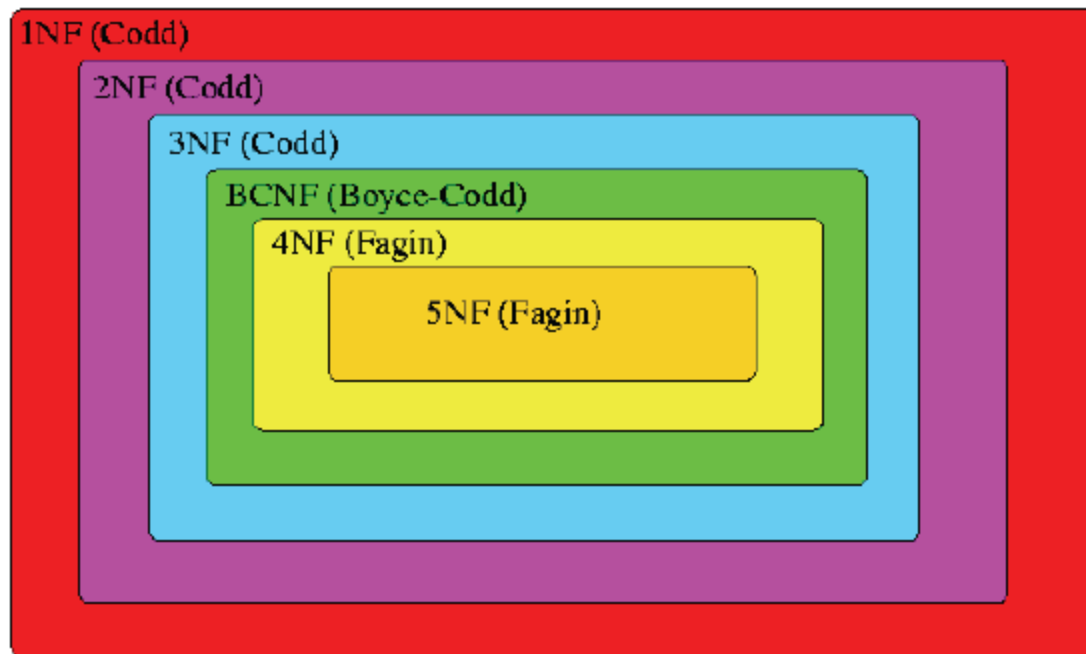


Image courtesy of Prof Cheung's notes

Each rectangle represents all possible relations

Normalization: General Idea

- Designers should aim for the “ultimate” 5NF
 - However, designers typically stop at 3NF or BCNF
- Designing a good database is a complex task
 - Normalization is useful aid but should not be panacea
- Normal forms can be violated deliberately to achieve better performance (less join operations)

First Normal Form (1NF)

- Simplest one that does not depend on “functional dependency”
- Basic relational model where every attribute has atomic (single, not multi) values
- Techniques to achieve 1NF (if not already done)
 - Remove attribute violating 1NF and place in separate relation
 - Expand the key

Example: 1NF Conversion

DEPARTMENT

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



DEPARTMENT

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

Adapted from Figure 14.9 (Book)

Functional Dependencies (FD)

- Constraint between two sets of attributes
- Generalize the concept of keys
- Definition:
 - Let X and Y be 2 sets of attributes of R
 - A functional dependency ($X \rightarrow Y$) occurs if for any two tuples t_1 and t_2 of the relation R , if $t_1[X] = t_2[X]$ (i.e., the attribute values for X is the same in both tuples) then $t_1[Y] = t_2[Y]$

$X \rightarrow Y$ means that whenever two tuples agree on X , then they agree on Y

FD Pictorially

		X			Y			
		A	B	C	D	E	F	G
t1
	b7	c4	...	e1	f3	g4
t2
	b7	c4	...	e1	f3	g4

If t1 and t2 agree here...

they also agree here!

Example: Company Database

- Relation that represent information about employees and the projects they work on

<u>SSN</u>	FName	LName	<u>PNo</u>	PName	Hours
111-11-1111	John	Smith	pj1	ProjectX	20
111-11-1111	John	Smith	pj2	ProjectY	10
333-33-3333	Jack	Rabbit	pj1	ProjectX	5

- FDs in the relation
 - $SSN \rightarrow \text{fname, lname}$
 - $PNo \rightarrow PName$
 - $SSN, PNo \rightarrow \text{Hours}$

Example: Company Database (2)

<u>SSN</u>	FName	LName	<u>PNo</u>	PName	Hours
111-11-1111	John	Smith	pj1	ProjectX	20
111-11-1111	John	Smith	pj2	ProjectY	10
333-33-3333	Jack	Rabbit	pj1	ProjectX	5

- FDs can cause anomalies due to dependency between attributes
- Insert anomaly - new project (pj3) with no employees
- Delete anomaly - deleting John Smith from pj2 deletes information about pj2

Example: Course Database

- Relation with courses, students, and instructors

<u>studentID</u>	semester	courseNo	section	instructor
123455	Spring16	CS377	0	Ho
234097	Spring16	CS377	0	Ho
234107	Fall15	CS377	0	Cheung
140701	Fall15	CS377	0	Cheung

- FDs in the relation
 - courseNo, semester \rightarrow instructor
 - studentID \rightarrow courseNo, semester

Inferring FDs

- An FD is
 - Inherent property of an application
 - Defined based on the semantics of the attributes
 - Not something we can infer from a set of tuples
- Given a table with a set of tuples
 - Can confirm that a FD **seems** to be valid
 - Infer a FD is **definitely invalid**
 - Can **never prove** that FD is valid

Refresher: Keys

- Set of attributes S is a super key of a relation R if S functionally determines all attributes in R

$$\forall t_1, t_2 \in R : t_1[SK] \neq t_2[SK]$$

- Set of attributes K is a key of a relation if and only if
 - K functionally determines all attributes in R
 - K is minimal superkey
 - None of its subsets functionally determines all attributes in R

“Good” vs “Bad” FDs

- A key of a relation functionally determines all attributes in that relation
 - This is called natural or trivial
- “Good” functional dependency is a natural or trivial functional dependency
- Functional dependencies other than natural dependencies will cause anomalies

Example: Company DB Revisited


<u>SSN</u>	FName	LName	<u>PNo</u>	PName	Hours
------------	-------	-------	------------	-------	-------

- SSN, PNo \rightarrow Hours is a “good” functional dependency
 - (SSN, PNo, Hours) should be in the same relation
- SSN \rightarrow fname, lname is a “bad” functional dependency and should be taken out and put together in another relation on their own
- PNo \rightarrow PName is a “bad” functional dependency and should be taken out and put in another relation on their own

“Bad” FDs Cause Anomalies

- Since the LHS of a functional dependency is not a key, you can have multiple tuples in the database
 - Leads to update anomalies as well as insert and delete anomalies
 - Duplication of information is guaranteed!
- Solution: break up the relation into multiple tuples

Relation Decomposition

- A decomposition of relation R is a collection of relations R_1, R_2, \dots, R_n such that every attribute of R appears in R_1, R_2, \dots, R_n at least once
- Some decompositions are useful and some aren't
- Example:
Employee(SSN, Fname, LName, PNo, PName, Hours) —
R1(SSN, PName, Hours)
R2(PNumber, Fname, LName) 
What does this mean?
- Decompose with a goal!

What is a Good Decomposition?

- Normal forms will be guiding criteria for better relations
- When a relation R violates the guiding criteria of normal form, we decompose the relation to comply with the guiding criteria of the normal form
- Use functional dependencies to determine if dependency is “good” or “bad”
 - Find all keys of the relation R via inference rules

Armstrong's Axioms

- Most basic inference rules
 - Given a set of functional dependencies, we can derive additional functional dependencies using inference rules
- Sound — any FD inferred using Armstrong's axioms will hold in R
- Complete — Every valid FD on R can be found by applying only Armstrong's axioms

Armstrong's Axiom 1: Reflexivity

- For attribute sets X, Y : If Y is subset of X , then $X \rightarrow Y$
- Examples:
 - $A, B \rightarrow B$
 - $A, B, C \rightarrow A, B$
 - $A, B, C \rightarrow A, B, C$

Armstrong's Axiom 2: Augmentation

- For attribute sets X, Y, Z : If $X \rightarrow Y$, then $X, Z \rightarrow Y, Z$
- Examples:
 - $A \rightarrow B$ implies $A, C \rightarrow B, C$
 - $A, B \rightarrow C$ implies $A, B, C \rightarrow C$

Armstrong's Axiom 3: Transitivity

- For attribute sets X, Y, Z : If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- Examples:
 - $A \rightarrow B$ and $B \rightarrow C$ implies $A \rightarrow C$
 - $A \rightarrow C, D$ and $C, D \rightarrow E$ implies $A \rightarrow E$

Example: Armstrong's Axioms

- **Product**(name, category, color, department, price)
- Given initial set of FDs:
 - name \rightarrow color
 - category \rightarrow department
 - color, category \rightarrow price
- Inferred FDs:
 - name, category \rightarrow price: augmentation & transitivity
 - name, category \rightarrow color: reflexivity & transitivity

Other Useful Inference Rules

- Derived from Armstrong's Axioms
- Decomposition rule: If $X \twoheadrightarrow Y, Z$ then $X \twoheadrightarrow Y, X \twoheadrightarrow Z$
- Union rule: If $X \twoheadrightarrow Y$ and $X \twoheadrightarrow Z$, then $X \twoheadrightarrow Y, Z$
- Pseudo transitivity rule: If $X \twoheadrightarrow Y$ and $Y, W \twoheadrightarrow Z$ then $X, W \twoheadrightarrow Z$

Database Design: Recap

- Informal guidelines for good design
- 1NF
- Functional dependency
- Inference rules for FD

