Lecture:
Design Theory

Example Enrollment table - “v0”

~375 cs145 students

SID	Class	Room	Time	Lat	Lng
4749732	cs 145	Nvidia Aud	T/R 4:30-6	37.4277° N	122.1742° W
2720942	cs 145	Nvidia Aud	T/R 4:30-6	37.4277° N	122.1742° W
4823984	cs 145	Nvidia Aud	T/R 4:30-6	37.4277° N	122.1742° W
4287594	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2984994	cs 145	Nvidia Aud	T/R 4:30-6	...	...
8472374	cs 145	Nvidia Aud	T/R 4:30-6	...	...
4723663	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2478239	cs 145	Nvidia Aud	T/R 4:30-6	...	...
4763268	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2364532	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2364573	cs 145	Nvidia Aud	T/R 4:30-6	...	...
3476382	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2347623	cs 145	Nvidia Aud	T/R 4:30-6	...	...
...	...	...	...	...	...
2364579	cs 245	Nvidia Aud	T/R 3-4:30	37.4277° N	122.1742° W
3476343	cs 245	Nvidia Aud	T/R 3-4:30	37.4277° N	122.1742° W
2322232	cs 245	Nvidia Aud	T/R 3-4:30	37.4277° N	122.1742° W

~300 cs245 students

<u>Problems</u>
Repeats?
Room/time change?
Deletes?

<u>Properties</u>
Class -> Room/time
Room -> Lat, Lng

(more compact)

Example Enrollment table - “v1”

375 cs145 students

300 cs245 students

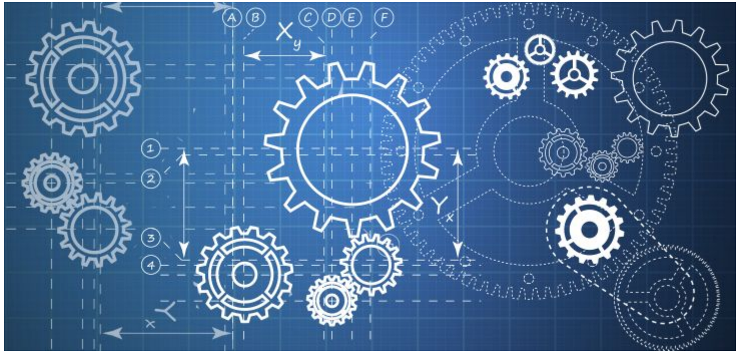
SID	Class
4749732	cs 145
2720942	cs 145
4823984	cs 145
4287594	cs 145
2984994	cs 145
8472374	cs 145
4723663	cs 145
2478239	cs 145
4763268	cs 145
2364532	cs 145
2364573	cs 145
3476382	cs 145
2347623	cs 145
...	...
2364579	cs 245
3476343	cs 245
2322232	cs 245

Class	Room	Time
cs 145	Nvidia Aud	T/R 4:30-6
cs 245	Nvidia Aud	T/R 3-4:30
cs 246	Nvidia Aud	M/W 3-4:30

Room	Lat	Lng
Nvidia Aud	37.4277° N	122.1742° W

Design Theory

- Design theory is about how to represent your data to avoid ***anomalies***.
- Simple algorithms for “best practices”



Data Anomalies & Constraints

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

If every course is in only one room, contains ***redundant*** information!

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	$\boxed{\text{C12}}$
Sam	CS145	B01
..	..	..

If we update the room number for one tuple, we get inconsistent data = an ***update anomaly***

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

Student	Course	Room
..	..	..

If everyone drops the class, we lose what room the class is in! = a ***delete* anomaly**

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

…	CS229	C12

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

Similarly, we can't reserve a room without students = an ***insert* anomaly**

Constraints Prevent (some) Anomalies in the Data

Student	Course
Mary	CS145
Joe	CS145
Sam	CS145
..	..

Course	Room
CS145	B01
CS229	C12

Is this form better?

- Redundancy?
- Update anomaly?
- Delete anomaly?
- Insert anomaly?

What are "good" *decompositions?*

Functional Dependencies

Functional Dependency

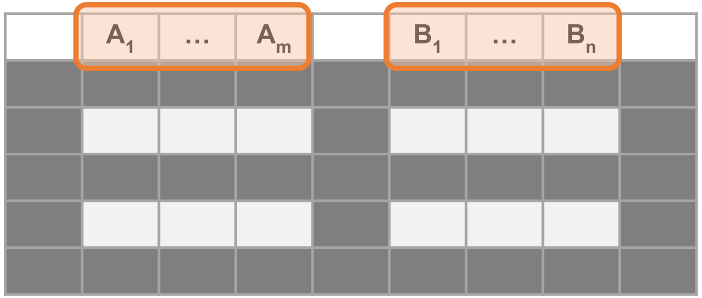
Def: Let A,B be *sets* of attributes
We write $A \rightarrow B$ or say A ***functionally determines*** B if, for any tuples t_1 and t_2:

$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B]$$

and we call $A \rightarrow B$ a **functional dependency**

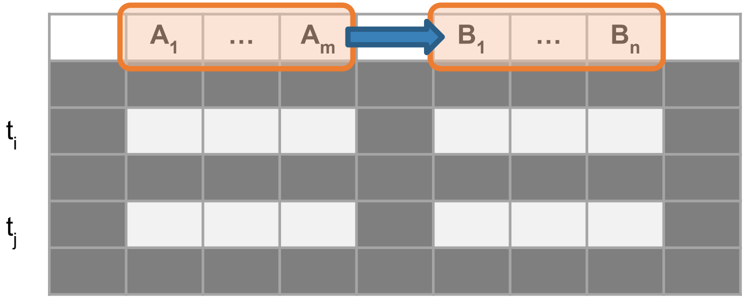
A->B means that
“whenever two tuples agree on A then they agree on B.”

A Picture Of FDs



Defn (again):
Given attribute sets **A={A_1,…,A_m}** and **B = {B_1,…B_n}** in **R**,

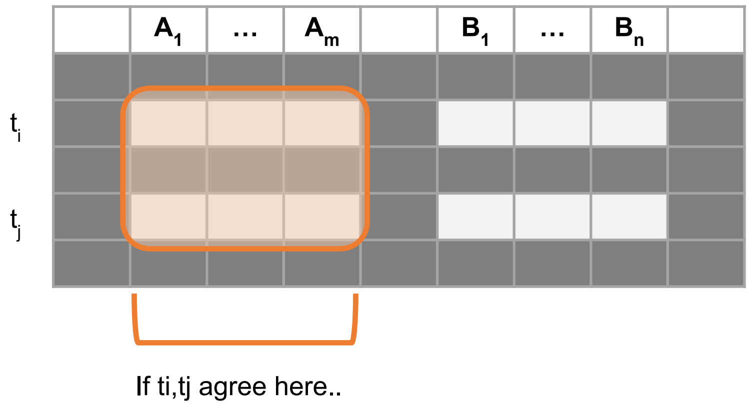
A Picture Of FDs



<u>Defn (again):</u>
Given attribute sets $\mathbf{A=\{A_1,\ldots,A_m\}}$ and $\mathbf{B = \{B_1,\ldots B_n\}}$ in **R,**

The ***functional dependency*** **A→ B on R** holds if for ***any*** t_i,t_j in R:

A Picture Of FDs

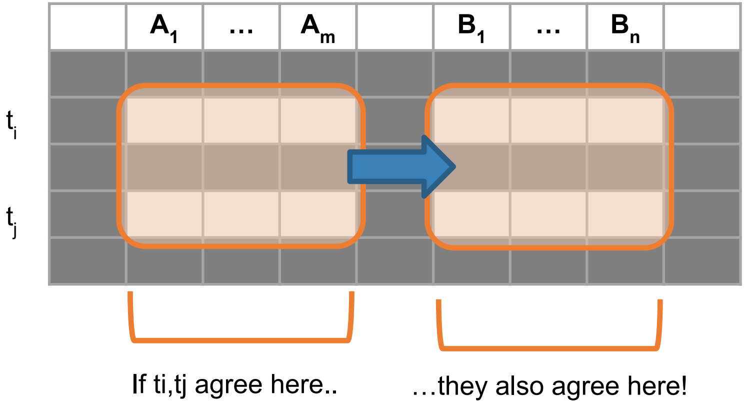


<u>Defn (again):</u>
Given attribute sets $\mathbf{A=\{A_1,\dots,A_m\}}$ and $\mathbf{B = \{B_1,\dots B_n\}}$ in **R,**

The ***functional dependency*** $\mathbf{A \rightarrow B}$ **on R** holds if for ***any*** t_i,t_j in R:

<u>**if**</u> $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

A Picture Of FDs



<u>Defn (again):</u>
Given attribute sets $\mathbf{A=\{A_1,\dots,A_m\}}$ and $\mathbf{B = \{B_1,\dots B_n\}}$ in **R**,

The ***functional dependency*** $\mathbf{A \rightarrow B}$ **on R** holds if for ***any*** t_i, t_j in R:

<u>if</u> $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

<u>then</u> $t_i[B_1] = t_j[B_1]$ AND $t_i[B_2]=t_j[B_2]$ AND … AND $t_i[B_n] = t_j[B_n]$

FDs for Relational Schema Design

High-level idea: **why do we care about FDs?**

1. Start with some relational *schema*
2. Find *functional dependencies (FDs)*
3. Use these to *design a better schema*
   One which minimizes the possibility of anomalies

Functional Dependencies as Constraints

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

Note: The FD {Course} -> {Room} ***holds on this table instance***

However, cannot *prove* that the FD {Course} -> {Room} holds on all instances. That is, FDs are for an instance and not for ***schema***

Functional Dependencies as Constraints

Note that:

- You can check if an FD is **violated** by examining a single instance;
- However, you **cannot prove** that an FD is part of the schema by examining a single instance.
  - *This would require checking every valid instance*

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

More Examples

An FD is a constraint which holds, or does not hold on an instance:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

More Examples

EmpID	Name	Phone		Position
E0045	Smith	1234		Clerk
E3542	Mike	9876	$\leftarrow$	Salesrep
E1111	Smith	9876	$\leftarrow$	Salesrep
E9999	Mary	1234		Lawyer

{Position} $\rightarrow$ {Phone}

More Examples

EmpID	Name	Phone		Position
E0045	Smith	1234	→	Clerk
E3542	Mike	9876		Salesrep
E1111	Smith	9876		Salesrep
E9999	Mary	1234	→	Lawyer

but *not* {Phone} → {Position}

Example (mega) Enrollment table - “v0”

~375 cs145 students

~300 cs245 students

SID	Class	Room	Time	Lat	Lng
4749732	cs 145	Nvidia Aud	T/R 4:30-6	37.4277° N	122.1742° W
2720942	cs 145	Nvidia Aud	T/R 4:30-6	37.4277° N	122.1742° W
4823984	cs 145	Nvidia Aud	T/R 4:30-6	37.4277° N	122.1742° W
4287594	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2984994	cs 145	Nvidia Aud	T/R 4:30-6	...	...
8472374	cs 145	Nvidia Aud	T/R 4:30-6	...	...
4723663	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2478239	cs 145	Nvidia Aud	T/R 4:30-6	...	...
4763268	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2364532	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2364573	cs 145	Nvidia Aud	T/R 4:30-6	...	...
3476382	cs 145	Nvidia Aud	T/R 4:30-6	...	...
2347623	cs 145	Nvidia Aud	T/R 4:30-6	...	...
...	...	...	...	...	...
2364579	cs 245	Nvidia Aud	T/R 3-4:30	37.4277° N	122.1742° W
3476343	cs 245	Nvidia Aud	T/R 3-4:30	37.4277° N	122.1742° W
2322232	cs 245	Nvidia Aud	T/R 3-4:30	37.4277° N	122.1742° W

Problems
Repeats?
Room/time change?
Deletes?

FDs
Class -> Room,Time
Room -> Lat, Lng

(more compact)