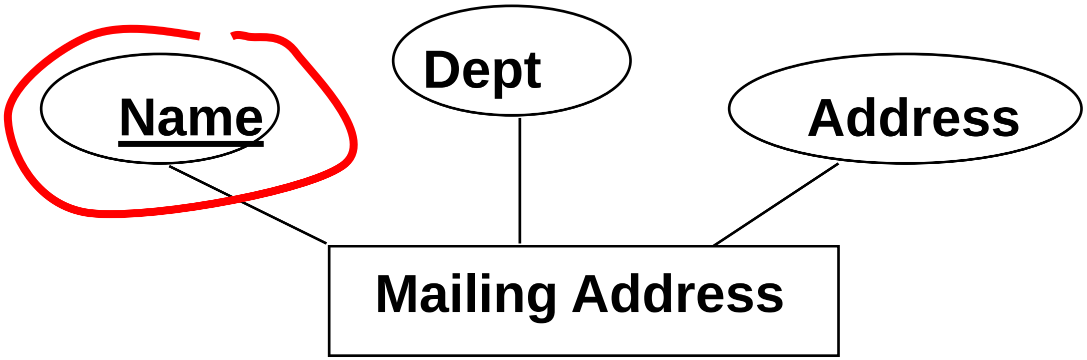
Functional Dependencies

R&G Chapter 19

Imagine that we've created a perfectly good entity for mailing addresses at Stevens



What would an instance look like?

Name	Dept	Address
Alice	CS	Lieb, Castle Point on Hudson
Bob	CS	Lieb, Castle Point on Hudson
Carol	CS	Lieb, Castle Point on Hudson
David	ECE	Burchard, Castle Point on Hudson

Observation: the same departments are always associated with the same address

BAD database design: as it contains **DATA REDUNDANCY**!

The Evils of Redundancy

- Redundancy
  - Some information is stored repeatedly in the database
  - The ROOT of several problems associated with relational schemas

Example of Redundancy

- **Consider the following relation and its instance: Person(<u>SSN</u>, Name, Address, Hobby)**

SSN	Name	Address	Hobby
12345678	Alan	123 Main street	Reading
12345678	Alan	123 Main street	Cooking
22345678	Bob	456 Main street	Sleeping
22345678	Bob	456 Main street	drinking

- **NOTE: some information is redundant in the instance**

What problems arise because of redundancy?

- Redundancy gives rise to ANOMALIES
  - **UPDATE ANOMALIES**
  - **INSERTION ANOMALIES**
  - **DELETION ANOMALIES**

Update Anomaly

If Alan moves to a new place (254 Main street), but only one tuple is updated?

SSN	Name	Address	Hobby
12345678	Alan	123 Main street	Reading
12345678	Alan	254 Main street	Cooking
22345678	Bob	456 Main street	Sleeping
32345678	Carol	456 Main street	Gardening

Data inconsistency problem!

Insertion Anomaly

A new hobby tuple of Alan is inserted, but with a different address…?

SSN	Name	Address	Hobby
12345678	Alan	123 Main street	Reading
12345678	Alan	123 Main street	Cooking
12345678	Alan	134 Main street	programming
22345678	Bob	456 Main street	Sleeping
32345678	Carol	456 Main street	Gardening

Data inconsistency problem (again)!

Deletion Anomaly

Delete Bob’s hobby by deleting his tuple

SSN	Name	Address	Hobby
12345678	Alan	123 Main street	Reading
12345678	Alan	123 Main street	Cooking
22345678	Bob	456 Main street	Sleeping
32345678	Carol	456 Main street	Gardening

Bob’s address info will not exist in the db!

Okay, that's bad. But how do I know if each person has only one address?

- Databases allow you say that one attribute determines another through a ***functional dependency (FD)***.
  - So if SSN determines Address, we say that there's a functional dependency from SSN to Address.

Name	Dept	Address
Alice	CS	Lieb, Castle Point on Hudson
Bob	CS	Lieb, Castle Point on Hudson
Carol	CS	Lieb, Castle Point on Hudson
David	ECE	Burchard, Castle Point on Hudson

Functional Dependencies (FDs)

- **A <u>functional dependency</u> $X \rightarrow Y$ holds over relation schema R if, for every instance *r* of R:**

  For any two tuples t1 , t2 $\in$ r,

  $$\pi_X(t1) = \pi_X(t2) \text{ implies } \pi_Y(t1) = \pi_Y(t2)$$

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

- **In other words: $X \rightarrow Y$ 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)

- **Can read "$\rightarrow$" as "determines"**

FD Example

SSN	Name	Address	Hobby
12345678	Alan	123 Main street	Reading
12345678	Alan	123 Main street	Cooking
22345678	Bob	456 Main street	Sleeping
44444444	Carol	456 Main street	Gardening

- We have: SSN $\rightarrow$ Address
- Do we have: Address $\rightarrow$ SSN?

X-> Y does not imply that Y -> X!

More FD Examples

- **A film has a unique title.**
  - FilmID -> title
- **The customerID uniquely identifies the customer and his/her address**
  - CustomerID -> name, address
- **On any particular day, a film copy can be rented to at most one customer**
  - Date, FilmID, copyNum -> CustomerID

FD's Continued

- **An FD is a statement about *all* allowable relations.**
  - Must be identified based on semantics of application.
- **Question: How to use FDs to determine keys?**
- **if "K $\rightarrow$ all attributes of R" then K is a *superkey* for R**

  (does not require K to be *minimal*.)
- **FDs are a generalization of keys.**
  - FDs are NOT necessarily to be key constraints.

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

What are the possible FDs on Hourly_Emps?

ssn is the key: S $\rightarrow$ SNLRWH

rating determines *wage_per_hr*: R $\rightarrow$ W

lot determines *lot*: L $\rightarrow$ L ("trivial" dependnency)

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

FDs:
S ->SNLRWH
R -> W

- ***Update anomaly*: Can we modify W in the 1st tuple only?**
- ***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

FDs:
S ->SNLRWH
R -> W

S ->SNLRWH infers that S-> W

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

Eliminating Redundancy by Relation Decomposition

- **Redundancy can be removed by “chopping” the relation into pieces.**
- **FD’s are used to drive this process.**

  R $\rightarrow$ W is causing the problems, so decompose SNLRWH into SNLRH and RW.

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

Reasoning About FDs

- **Given some FDs, we can usually infer additional FDs:**

  title $\rightarrow$ *(studio, star)* implies *title* $\rightarrow$ *studio* and *title* $\rightarrow$ *star*

  title $\rightarrow$ *studio* and *title* $\rightarrow$ *star* implies *title* $\rightarrow$ *studio, star*

  title $\rightarrow$ *studio*, *studio* $\rightarrow$ *star* implies *title* $\rightarrow$ *star*

- **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 (AA):**
  - 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 $X \rightarrow Z$

Common Mistakes of Inferences

If $(X, Y) \rightarrow Z$, then $X \rightarrow Z$ and $Y \rightarrow Z$

WRONG!!!

For example:

(title, star) $\rightarrow$ *studio* does NOT imply

title $\rightarrow$ *studio* or *star* $\rightarrow$ *studio*

Example

- **Contracts(*<u>cid</u>,sid,jid,did,pid,qty,value*), and:**
  - C is the key: $C \rightarrow CSJDPQV$
  - Proj purchases each part using single contract: $JP \rightarrow C$
  - Dept purchases at most 1 part from a supplier: $SD \rightarrow P$
- **Question: Prove that SDJ is a key for Contracts**
  - i.e., prove that SDJ -> CSJDPQV

1. $JP \rightarrow C$, $C \rightarrow CSJDPQV$ imply $JP \rightarrow CSJDPQV$ (by transitivity)
2. $SD \rightarrow P$ implies $SDJ \rightarrow JP$ (by augmentation)
3. $SDJ \rightarrow JP$, $JP \rightarrow CSJDPQV$ imply $SDJ \rightarrow CSJDPQV$ (by transitivity). Thus SDJ is a key.

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

No! FD inference is not like arithmetic multiplication.