Database Design: Functional Dependencies

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

October 8, 2018

Announcements

- HW 2 is due Oct 15 (23:59)
- Project Track 1 – Stage 2

And

- Project Track 2 – Stage 1

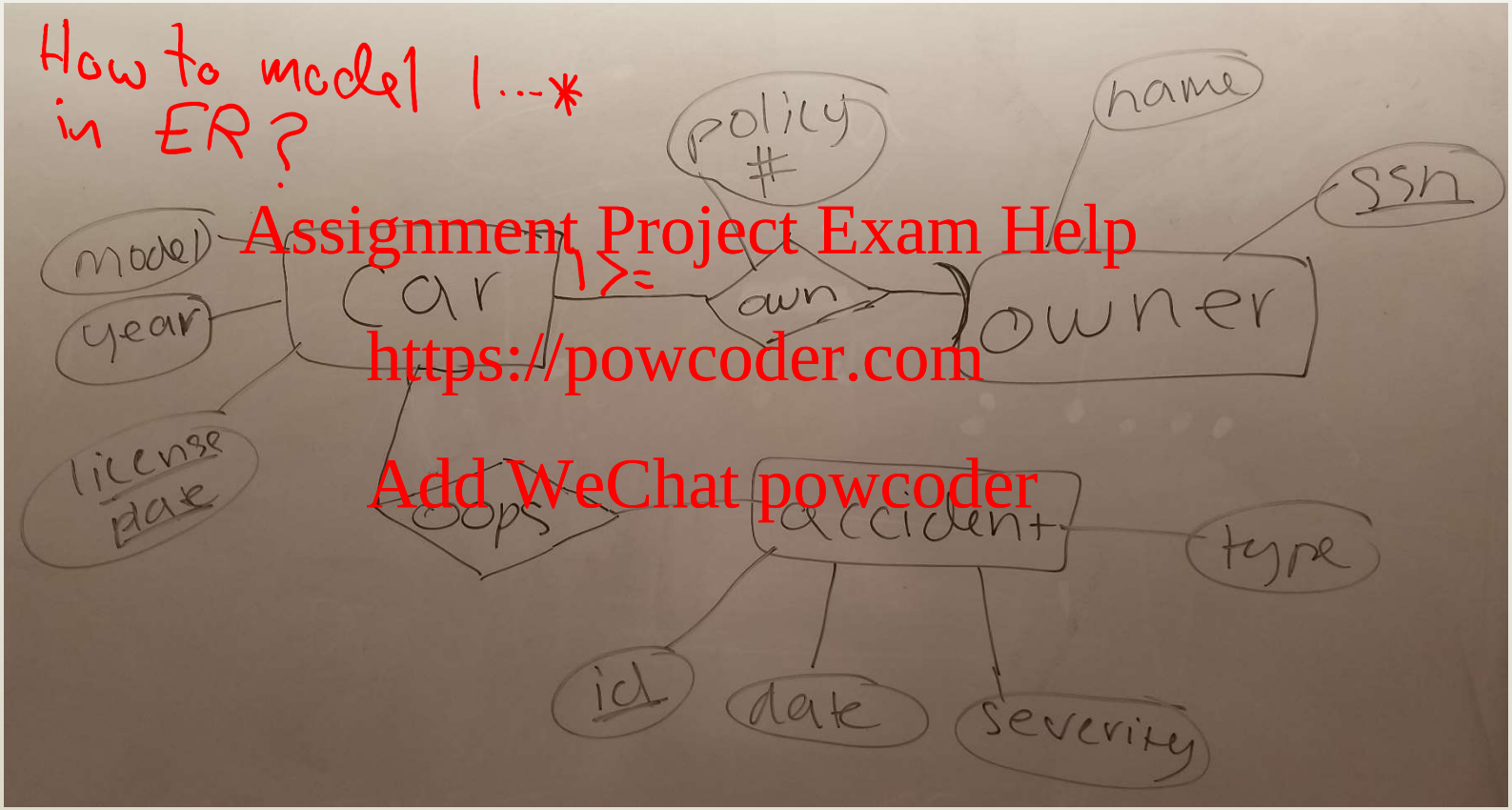
are due on Wednesday (10/10)

- **Midterm ➔ Oct, 29th in class (11-12:15)**
- **Final ➔ Dec, 12 in class (11-12:15)**

ERD Exercise

Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.

ERD Exercise Student Solution



Various notations for “one-to-many”

maximum cardinalities only

minimum and maximum cardinalities

Various notations for “many-to-many”

maximum cardinalities only

minimum and maximum cardinalities

Overview of Database Design

- Conceptual design: (ER & UML Models are used for this.)
  - What are the entities and relationships we need?
- Logical design:
  - Transform ER design to Relational Schema
- Schema Refinement: (Normalization)
  - Check relational schema for redundancies and related anomalies.
- Physical Database Design and Tuning:
  - Consider typical workloads; (sometimes) modify the database design; select file types and indexes.

We're here

Agenda

- How do we obtain a good design?
- Functional Dependencies and Keys
- Rules about Functional Dependencies
  - Splitting/Combination
  - Trivial Dependencies
  - Attribute Closure
  - FD Closure

Motivation

- We have designed ER diagram, and translated it into a relational db schema R = set of R1, R2, ... Now what?
- We can do the following
  - implement R in SQL
  - start using it
- However, R may not be well-designed, thus causing us a lot of problems
- OR: people may start without an ER diagram, and you need to reformat the schema R
  - Either way you may need to **improve** the schema

Q: Is this a good design?

Individuals with several phones:

Address	SSN	Phone Number
10 Green	123-321-99	(201) 555-1234
10 Green	123-321-99	(206) 572-4312
431 Purple	900-438-44	(908) 464-0028

Redundancy

address lost

Delete phone

Potential Problems

Address	SSN	Phone Number
10 Green	123-321-99	(201) 555-1234
10 Green	123-321-99	(206) 572-4312
431 Purple	909-438-44	(908) 464-0028

- Redundancy
- Update anomalies
  - maybe we'll update the address of the person with phone number '(206) 572-4312' to something other than '10 Green'. Then there will be two addresses for that person.
- Deletion anomalies
  - delete the phone number of a person; if not careful then the address can also disappear with it.

Better Designs Exist

Break the relation into two:

SSN	Address
123-321-99	10 Green
909-438-44	431 Purple

SSN	Phone Number
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028

Unfortunately, this is not something you will detect even if you did principled ER design and translation

How do We Obtain a Good Design?

- Start with the original db schema R
  - From ER translation or otherwise
- Transform it until we get a good design R*
- Some desirable properties for R*
  - must preserve the information of R
  - must have minimal amount of redundancy
  - must be "dependency preserving"
    - (we'll come to this later)
  - must also give good query performance

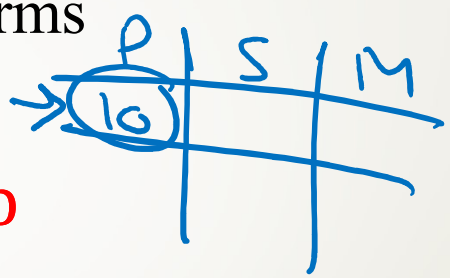
How do We Obtain a Good Design?

Normal Forms

- DB gurus have developed many "**normal forms**"
- These are basically schemas obeying certain rules
  - Converting a schema that doesn't obey rules to one that does is called "**normalization**"
  - This typically involves some kind of decomposition into smaller tables, just like we saw earlier.
  - (the opposite: grouping tables together, is called "**denormalization**")

Normal Forms

- DB gurus have developed many "normal forms"
- Most important ones
  - Boyce-Codd, 3rd, and 4th normal forms
- If R* is in one of these forms, then R* is guaranteed to achieve certain good properties
  - e.g., if R* is in Boyce-Codd NF, it is guaranteed to not have certain types of redundancies
- DB gurus have also developed algorithms to transform R into R* in these normal forms



Normal Forms (cont.)

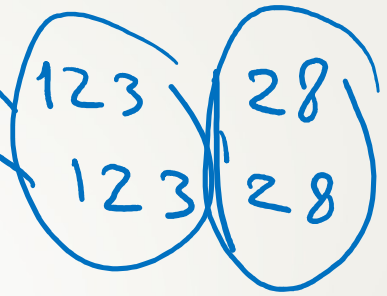
- There are also trade-offs among normal forms
- Thus, our goal is to:
  - learn these forms
  - transform R into R* in one of these forms
  - carefully evaluate the trade-offs
- To understand these normal forms we'll need to understand certain types of constraints
  - functional dependencies and keys

Functional Dependencies and Keys

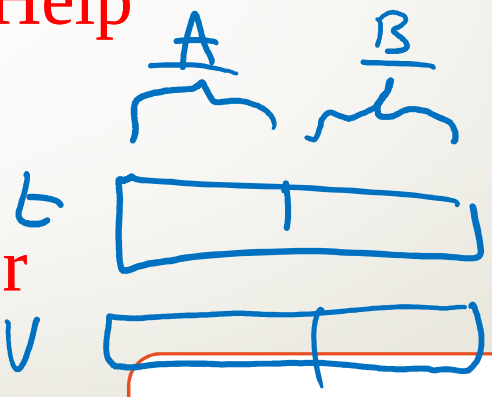
Functional Dependencies

Person(SSN, age)

SSN → age



- A form of constraint (hence, part of the schema)
- Finding them is part of the database design
- Used heavily in schema refinement
- Holds for ALL instances!



Definition:

If two tuples agree on the attributes

$A_1, A_2, \ldots A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots B_m$

Formally: $A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$

Where have we seen this before?

Examples

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	John	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

- EmpID ⟶ Name, Phone, Position
- Position ⟶ Phone
- but Phone ↛ Position: why?

What a FD actually means

- Knowing FD: A → B holds in R(A, B, C) means that
  - For ALL valid instances R(A, B, C):
    - A determines B
      - Or, if two tuples share A, then they share the same B
  - This is the property of the "world"
- Conversely, if: A /→ B, then there is no guarantee that the "A determines B" property holds in a given instance (though it might).
  - Trivially, it holds when you have only one tuple.

More examples

Product: name, manufacturer → price

Person: ssn → name, age

Company: name → stock price, president

Q: From this, can you conclude phone → SSN?

SSN	Phone Number
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028
909-438-44	(212) 555-4000

- No, you cannot. Like we discussed, this is a property of the world, not of the current data
- In general, you cannot conclude from a given instance of a table that an FD is true. An FD is not an observation made from a table's current tuples, it is an assertion that must always be respected.
- You can however check if a given FD ***is violated*** by the table instance.

Keys are a type of FD

- Key of a relation R is a set of attributes that
  - functionally determines all attributes of R
  - none of its subsets determines all attributes of R
- There could be many keys of a relation
  - Student (UIN, email, dept, age)
  - UIN → UIN, email, dept, age
  - email → UIN, email, dept, age
- Superkey
  - "Superset" of key
  - a set of attributes that contains a key
  - *Any examples for student?*

Many many FDs...

- MovieInfo (name, year, actor, director, studio)
  - Same movie can be remade multiple years, but a name, year pair uniquely determines a movie
  - A movie has a single director/studio but many actors
    - Name, year → director, studio
    - Name, year → director; Name, year → studio
    - Name, year /→ actor
  - A director works only with a single studio
    - Director → studio
  - An actor works on a given movie only once (never for remakes), but may work for many movies in a year
    - Actor, name → year; actor, year /→ name

Many many FDs...

- MovieInfo (name, year, actor, director, studio)
  - Name, year → director, studio
  - Name, year → director
  - Name, year → studio
  - Director → studio
  - Actor, name → year
  - ...
  - Actor, name, year → director, studio
  - Director, actor, name → studio, year
  - Director, name, year → studio
  - Studio, actor, name → year
  - ...

Any missing FDs?

Rules about Functional Dependencies

Outline of Rules

- Two examples of rules
  - Splitting/Combination
  - Trivial Dependencies
- Attribute Closure
  - Algorithm
  - Uses
- FD Closure
  - A complete set of rules:
    - Armstrong's axioms
  - Algorithm

The Splitting/Combining Rule

- $A_1A_2...A_n \rightarrow B_1B_2...B_m$
- Equivalent to:

  $A_1A_2...A_n \rightarrow B_1;$

  $A_1A_2...A_n \rightarrow B_2;$

  …

  $A_1A_2...A_n \rightarrow B_m$

- Can replace one for the other.

Trivial Functional Dependencies

- $A_1A_2...A_n \rightarrow A_1$

- In general,

  $A_1A_2...A_n \rightarrow B_1B_2\ldots B_m$

  if $\{B_1B_2\ldots B_m\} \subseteq \{A_1A_2...A_n\}$

  Example: name, UIN → UIN

Why does this make sense?

Closure of a Set of Attributes

Given a set of attributes $\{A1, \ldots, An\}$ and a set of FDs S.

Problem: find all attributes B such that:

for all relations that satisfy S, they also satisfy:

$A1, \ldots, An \rightarrow B$

The **closure** of $\{A1, \ldots, An\}$, denoted $\{A1, \ldots, An\}^+$, is the set of all such attributes B

We will discuss the motivations for attribute closures soon

Algorithm to Compute Closure

Split the FDs in S so that every FD has a single attribute on the right. (Simplify the FDs)

Start with $X=\{A_1A_2\ldots A_n\}$.

Repeat until X doesn't change **do**:

If $(B_1B_2\ldots B_m \rightarrow C)$ is in S,
such that $B_1,B_2,\ldots B_m$ are in X and C is not in X:
add C to X.

// X is now the correct value of $\{A_1A_2\ldots A_n\}^+$

Why does this algorithm converge?

Each relation has a finite set of attributes

Example

- Set of attributes A,B,C,D,E,F.
  - Functional Dependencies:

$AB \rightarrow C$

$AD \rightarrow E$

$B \rightarrow D$

$AF \rightarrow B$

Closure of {A,B}:

Closure of {A, F}:

Example

- Set of attributes A,B,C,D,E,F.
  - Functional Dependencies:

$A\ B \longrightarrow C$

$A\ D \longrightarrow E$

$B \longrightarrow D$

$A\ F \longrightarrow B$

Closure of {A,B}: X = {A, B, C, D, E}

Closure of {A, F}: X = {A, F, B, D, C, E}

Attribute Closure Exercise

Given:
Student_info(sid, name, crn, subj, cid, grade), and
F={sid ->name, crn -> cid, subj}, find A+ for A = {crn}

1. Decompose all FDs in F
   sid -> name, crn -> cid, crn -> subj

2. We start with with the attribute *crn* in the initial
   closure = {*crn*}

3. We look for all FDs that has crn, the 2nd FD (crn -> cid) has *crn*
   closure = *cid* U *closure* = {*crn, cid*}
4. We look for all FDs that has *cid* or *crn*, the 3rd FD (*crn -> subj*) has *crn*
   closure = *subj* U *closure* = {*crn, cid, subj*}

{crn}+ =

Is this algorithm correct?

- Yes. See Text (Section 3.2.5) for proof.

- Two parts of proof:
  - Anything determined to be part of $\{S\}^+$ deserves to be there:
    - *soundness*
  - There's nothing missing:
    - *completeness*

Uses for Attribute Closure

- Use 1: To test if X is a superkey
  - How?
  - compute X+, and check if X+ contains all attrs of R
- Use 2: To check if X → Y holds
  - How?
  - by checking if Y is contained in X+

An exercise

- Show that each of the following are not valid rules about FD's, by giving example relations that satisfy the given FDs (following the "If"), but not the FD that allegedly follows (after the "then").

(1) If **A --> B then B --> A**

(2) If **AB --> C and A --> C then B --> C.**

- **(1) A = SSN, B = Name**
- **(2) A = SSN, B = Phone, C = Name**