Week 9

(adapted from Evan Krul, Kyu-Sang Kim)

- ▼ Tutorial Questions

  1,2,3,4,5,7

RESCHEDULING WK10 TUTE

Functional Dependency

When we see something like X → Y

- X determines Y (or Y is functionally dependent on X)

What does this mean?

- E.g. Position → Salary
  - Position determines salary
  - For every unique position in our database, the salary for that given position is the same
  - Manager → $500 000
  - Ex Twitter Employee → $0
  - You → $50 000

Laws of Functional Dependencies

F1. Reflexivity e.g. $X \to X$

- a formal statement of *trivial dependencies*; useful for derivations

F2. Augmentation e.g. $X \to Y \;\Rightarrow\; XZ \to YZ$

- if a dependency holds, then we can freely expand its left hand side

F3. Transitivity e.g. $X \to Y,\, Y \to Z \;\Rightarrow\; X \to Z$

- the "most powerful" inference rule; useful in multi-step derivations

---

Inference Rules (cont)

While Armstrong's rules are complete, other useful rules exist:

F4. Additivity e.g. $X \to Y,\, X \to Z \;\Rightarrow\; X \to YZ$

- useful for constructing new right hand sides of *fd*s (also called union)

F5. Projectivity e.g. $X \to YZ \;\Rightarrow\; X \to Y,\, X \to Z$

- useful for reducing right hand sides of *fd*s (also called decomposition)

F6. Pseudotransitivity e.g. $X \to Y,\, YZ \to W \;\Rightarrow\; XZ \to W$

- shorthand for a common transitivity derivation

Closure

Given a set of functional dependencies (FDs), we can derive new ones (using the rules)

The largest collection of dependencies that can be derived from a set of FDs F is called the closure of F and is denoted F^+

{ A → B, B → C }

A+ = {ABC}

Keys

- **Super key** is any combination of columns/attributes that uniquely identifies a row in a table
- **Candidate key** is a super key which cannot have any columns removed from it without losing the unique identification property

R(A,B,C)

F = { A → B, B → C }

A+ = {ABC} = R → candidate key

AB+ = {ABC} = R

Normalisation

Normalisation is for reducing redundancies from your schema

▼ Example

Student (*zID, Name, Surname*, Address)

CourseEnrolments (*zID, Name, Surname*, Course, Course Name)

To normalise our databases, we put the schemas in a 'normal form'. We care only about two normal forms in this course:

- BCNF (Boyce Codd Normal form)
- 3NF

Although there are a lot more!

3NF Detection

To detect if a schema is 3NF, we must check for each functional dependency adheres to **one of the following**:

1. Is the RHS a subset of the LHS? AB → A, A → A
2. Is the LHS a **super key** (ie. super set of a candidate key)? AB+ ABC→D
3. Is the RHS a subset of a candidate key? AB+ D→A

BCNF Detection

To detect if a schema is BCNF, we must check for each functional dependency to **one of the following**:

1. Is the RHS a subset of the LHS?
2. Is the LHS a **super key** (ie. super set of a candidate key)?

Decomposition & Minimal Cover

If a relation is not in a desired normal form, it can be decomposed into multiple relations that each are in that normal form. To do this, we must take advantage of 'minimal cover'

> *''A set F of FDs is **minimal** if*
>
> - *every FD X → Y is simple*
>
>   *(Y is a single attribute)*
> - *every FD X → Y is left-reduced*
>
>   *(no Z $\subset$ X such that Z → Y could replace X → Y in F and preserve F^+)*
> - *every FD X → Y is necessary*
>
>   *(no X → Y can be removed without changes F^+)''*

So we want to **right-reduce, left-reduce and eliminate redundant FDs.**

The procedure to do this are:

1. Split the right-hand attributes of all FDs (a.k.a. canonical cover)

   eg. A → XY ⇒ A → X, A → Y
2. Find extraneous attributes and remove them

   eg. AB → C

   Either A or B or none can be extraneous.

   If A closure contains B then B is extraneous and it can be removed.

   If B closure contains A then A is extraneous and it can be removed.

3. Remove all redundant FDs

   eg. { A → B, B → C, A → C }

   Here A → C is redundant since it can already be achieved using Transitivity Property

3NF Decomposition

1. Find minimal cover
2. *Flatten* all FDs in the minimal cover (i.e. remove the arrows) and create new relation schemas.

   e.g. A → B, A → C, A → D becomes three new relation schemas R1(AB), R2(AC), R3(AD).

3. If the resulting set doesn't contain a candidate key, create a new relation schema.

   e.g. if we have candidate key AC but only have relation schemas R1(AB) and R2(AD), then create new relation schema R3(AC)

BCNF Decomposition

```
Initialize S = {R}
While S has a relation R' that is not in BCNF do:
   Pick a FD: X->Y that holds in R' and violates BCNF
   Add the relation XY to S
   Update R' = R'-Y
Return S
```