Lecture 16

Principles of Relational Database Design (cont.)

Week 9

To learn how to design good quality database schemata we study the theory of ***functional dependencies***

Overview

- Functional dependencies
- Inference rules for FDs
- The closure of a functional dependency set
- Functional dependencies and keys
- Closure of set of attributes under an FD set
- Minimal FD set
- Equivalence of FD sets

Functional dependencies

Employee_p (ssn, pno, hours, ename, pname, plocation)

In *any* instance of this relation it will be true that

if two tuples agree on the value of **ssn**,

then these two tuples also must agree on

the value of **ename**!

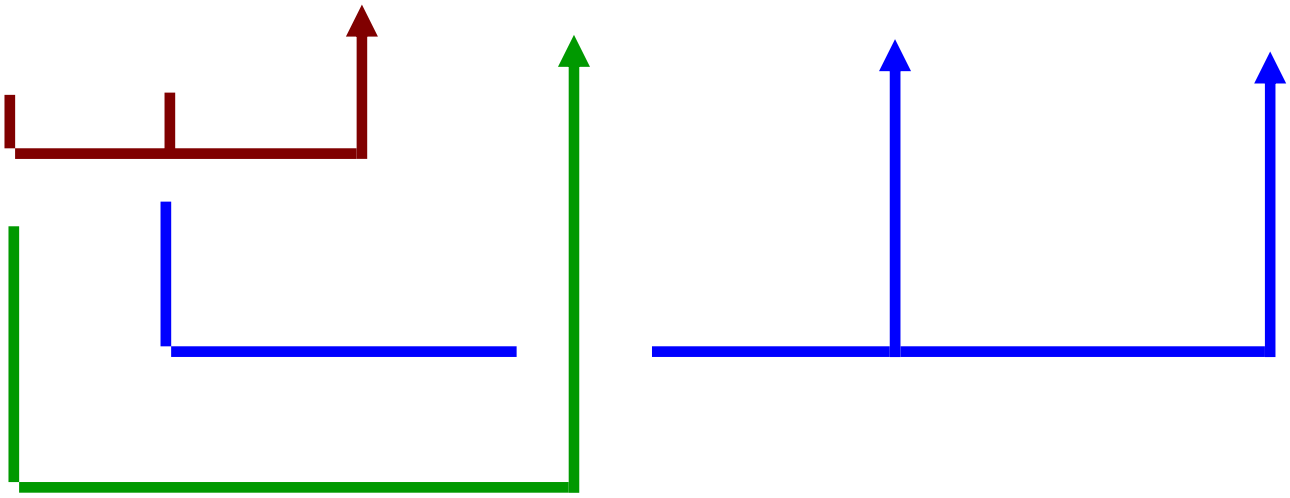
We say that ssn functionally determines ename

We write **ssn → ename**

Dependencies in Employee_p

Employee_p

(<u>ssn, pno</u>, hours, ename, pname, plocation)

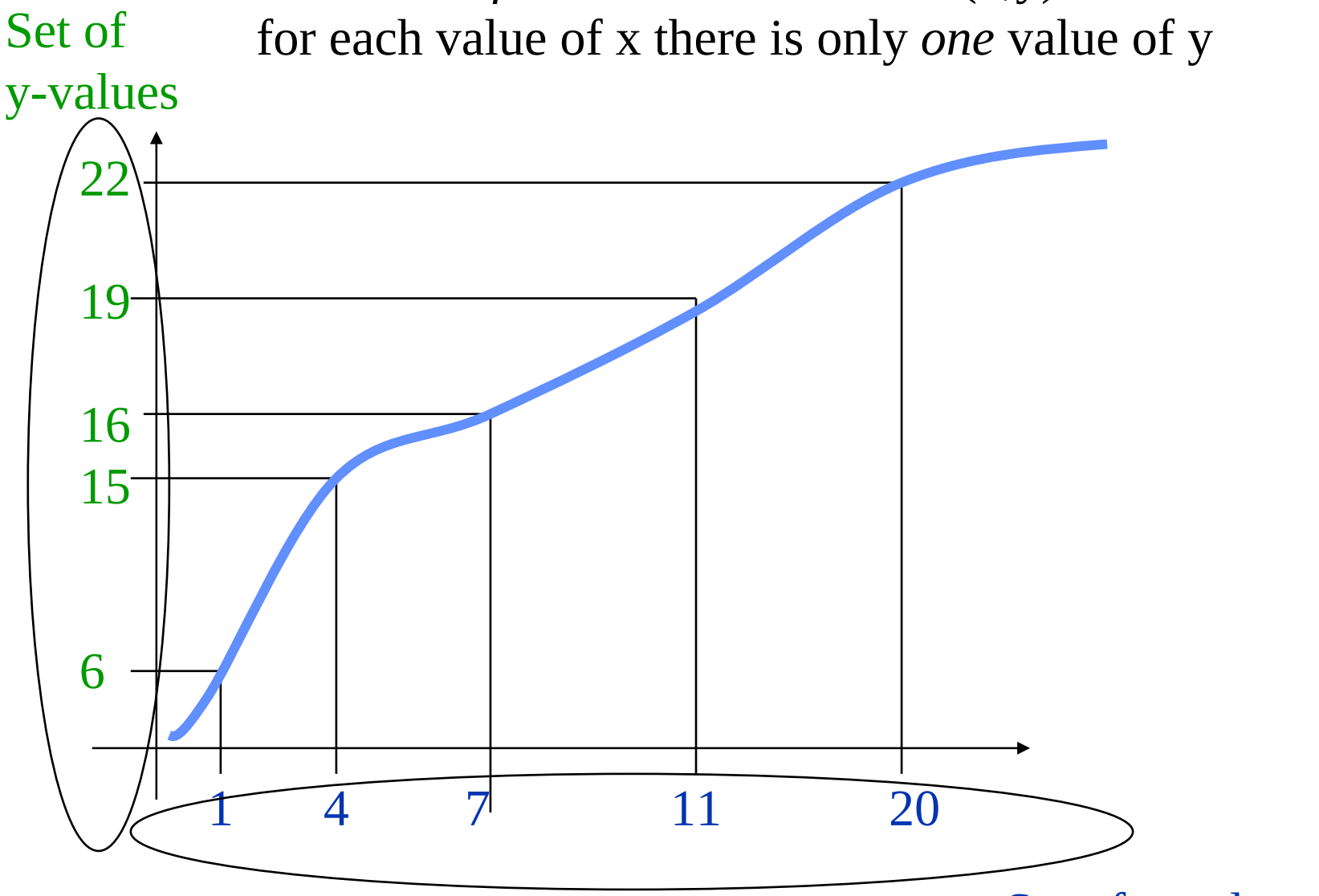


ssn → ename

pno → pname, plocation

ssn, pno → hours

Why is it called ‘functional’ dependency?
Because a *function* is a relation R(x,y) where
for each value of x there is only *one* value of y



Inference Rules for FDs

- Consider the well known company database, in which we know:

$$\frac{\begin{array}{l} ssn \rightarrow dno \\ dno \rightarrow dname \end{array}}{ssn \rightarrow dname}$$

1. transitivity

Inference Rules for FDs

- ssn,pno → ssn — **2.** reflexivity
- ssn → dno / ssn,pno → dno,pno — **3**. augmentation
- pno → pname, plocation / pno → pname (and also pno → plocation) — **4**. decomposition

Inference Rules for FDs

- pno → pname
  pno → plocation
  ---
  pno → pname,plocation **5.** additive (union)

Several inference rules exist, but rules 1,2, and 3 are enough* - the rest follows.

* 1,2,3 known as Armstrong's Axioms

The Closure of a Functional Dependency Set

Given a set of functional dependencies **F** , the **closure F+** of this set is obtained by applying these inference rules to derive all possible consequences of **F.**

(The result is usually a ***very large*** set, with many trivial dependencies. We never actually calculate the closure of an FD set, but use the **concept** in the sequel)

Functional Dependencies and Keys...

- Superkey

  If a set of attributes **X** functionally determines **all** other attributes in a relation schema **R**, then that set is a **superkey of R.** Eg. in Employee_p

  {ssn, pno, ename} is (one) superkey

  ssn, pno, ename $\rightarrow$ all attributes of Employee

Functional Dependencies and Keys...

- Of course we could have left out **ename** from {ssn, pno, ename} and **still** the result would be a superkey.

- If we keep leaving out attributes (but *still have a superkey*) such that we get a **minimal superkey**, then we obtain a **candidate key**:

- **Definition: Candidate key** is a minimal superkey

  Eg. **{ssn, pno} is a candidate key** of Employee_p since neither {ssn} nor {pno} alone determine all attributes of this relation

Algorithms to determine all candidate keys of a relation:

- top-down: eliminate attributes from trivial superkey sets while **preserving** the superkey feature;
- bottom up: start from a necessary set of attributes which **must** be part of a candidate key (ck), test if they are **ck** themselves, if not try **extending** the set until superkeys are found

Closure of a Set of Attributes under a Set of FDs

- Given a set of attributes **X** we often would like to determine what other attributes are functionally determined by these?

  We call this determined set the **closure of X under a set of FDs F**, noted X_F^+ .

Closure of Attributes under FDs (cont.)

- Provided the set of FDs is given, this is an easy task: all we have to do is try to apply the known functional dependencies and according to them, extend the determined set

- Let R = ABCD and F = $A \rightarrow B$, $B \rightarrow C$

  Question

  if X= {A,D} what is the closure X_F+ ?

Closure of Attributes under FDs (cont.)

$$R = ABCD \text{ and } F = \{A \rightarrow B;\ \ B \rightarrow C\}$$

- In other words $\{AD\} \rightarrow$?

Apply inference rules:

$\{AD\} \rightarrow \{AD\}$
$\{AD\} \rightarrow \{ADB\}$ by FD1
$\{AD\} \rightarrow \{ADB\} \rightarrow \{ADBC\}$ by FD2

therefore

$$\{AD\}_{\mathbf{F}}+ = \{ABCD\}$$

Minimal Set of Functional Dependencies

- Consider the FD set

This FD is redundant in itself - why ?

$$F = \{AB \rightarrow B, B \rightarrow C, C \rightarrow BA\}$$

What about **B** $\rightarrow$ **A** ? Would **it** be redundant ?

- Clearly, this is redundant, because we could simplify F without losing any information.

  Eg. $H = \{B \rightarrow C, C \rightarrow B, C \rightarrow A\}$ is equivalent with F, ie. $F^+ = H^+$

Minimal Set of Functional Dependencies (minimal Cover)

Definition: A set of functional dependencies is *minimal* if:

- All FDs have *one* attribute on the right hand side
- No FD can be omitted without losing information
- All left hand sides are minimal (no attribute can be omitted without losing information)

Minimal Set of Functional Dependencies (minimal Cover)

- The outline of the algorithm to determine a **minimal cover** (a minimal set of dependencies) of an FD set

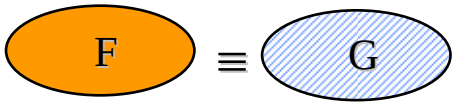
  1. Split all FDs into atomic ones, instead of
     $A \rightarrow BC$ write $A \rightarrow B$, $A \rightarrow C$
  2. Try to omit FDs which are redundant
  3. Try to omit attributes from left hand sides of the remaining FDs (this is usually possible when other FDs may be identified **within** the left side, e.g.:
     $AB \rightarrow C$ is equivalent to $A \rightarrow C$ if A also $\rightarrow B$)

Equivalence of FD sets

- Suppose that two analysts go out and try to establish the FDs of a database schema. If they come back with two different results **F** and **G**, which one is right?

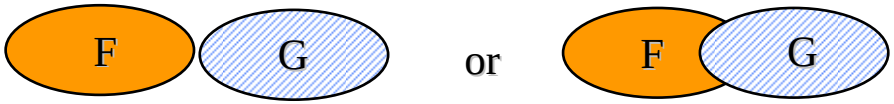
Possibilities:

- Both are right, because F and G are equivalent.
  (ie F+ = G+)



- One covers the other .
  (eg F+ ⊇ G+ : F says at least as much as G, and **more**)

- None covers the other or partially cover each other
  (need their union)



Equivalence of FD sets

- There is a simple algorithm to test if two FD sets are equivalent.
- We test if **F covers G**, i.e. if F+ $\supseteq$ G+ and if **G covers F**, i.e. if G+ $\supseteq$ F+
- If both are true, then F+ = G+ i.e. F and G **are equivalent**.

Outline of algorithm

- For each FD $X \rightarrow Y$ in F determine the closure of X under F and under G * :

  If for each X $X_F+ \subseteq X_G+$ then $F+ \subseteq G+$

- For each FD $Z \rightarrow Q$ in G determine the closure of Z under G and under F :
  If for each Z $Z_G+ \subseteq Z_F+$ then $G+ \subseteq F+$

* That is, what attributes are functionally determined by the set X under the FD sets F and G. X_F+ and X_G+ are in the form of {..attributes..}

Conclusion

- Implemented relational schemata should be non-redundant and free from update anomalies;
- functional dependency, superkeys and candidate keys have been defined;
- useful concepts related to functional dependencies which assist relational database design, have also been defined:

  Closure of a **set** of FDs F (F+);

  Closure of a **set of attributes** under a set of FDs F (X_F+)

  Minimal set of FDs (or **minimal cover** of a set of FDs);

  Equivalence of FD sets (F+ = G+ implies F equiv. to G)

The end