CMSC 424 – Database design
Lecture 10
Normalization

Mihai Pop

Midterm...

- Graded A & B – disappointing
- Outline of what I expected for A & B

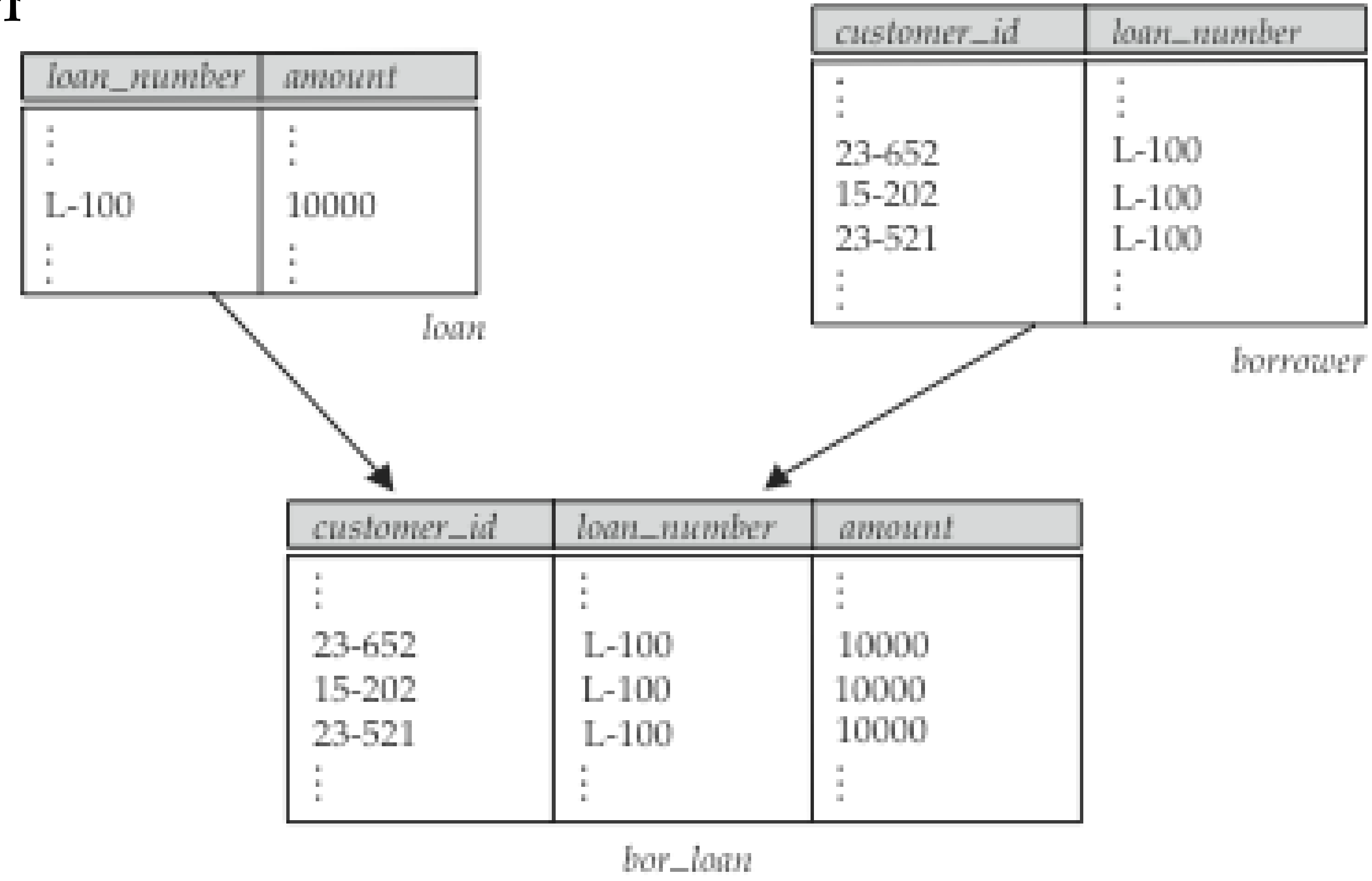
Normalization, functional dependencies....

- Large schemas: redundant information
- Small schemas: inefficient

- Normalization – a way to pick a "reasonable" compromise

Combine Schemas?

- Suppose we combine *borrower* and *loan* to get
  bor_loan = (*customer_id*, *loan_number*, *amount*)
- Result is possible repetition of information (L-100 in example below)



loan_number	*amount*
⋮	⋮
L-100	10000
⋮	⋮

loan

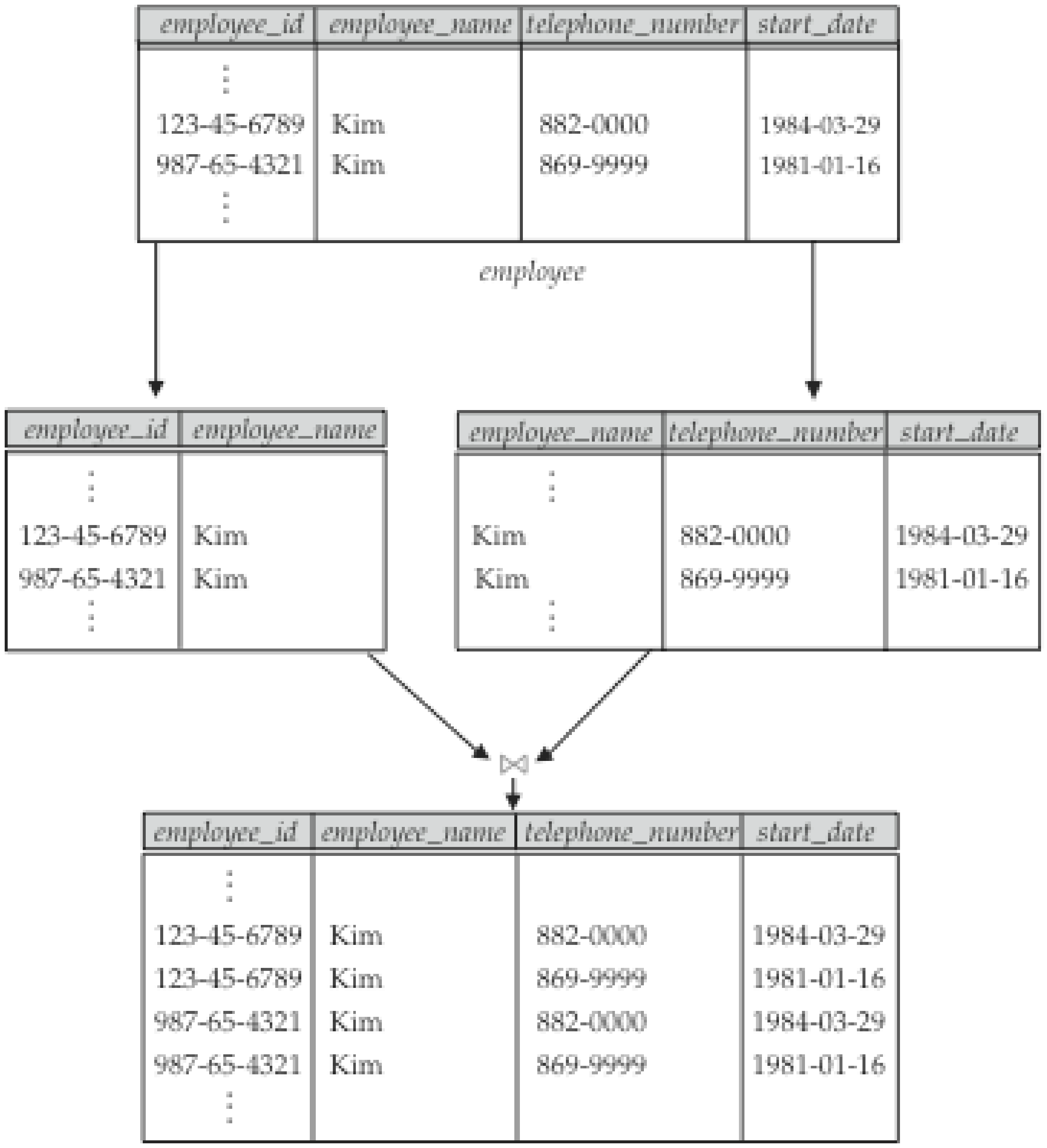
customer_id	*loan_number*
⋮	⋮
23-652	L-100
15-202	L-100
23-521	L-100
⋮	⋮

borrower

customer_id	*loan_number*	*amount*
⋮	⋮	⋮
23-652	L-100	10000
15-202	L-100	10000
23-521	L-100	10000
⋮	⋮	⋮

bor_loan

A Lossy Decomposition



employee_id	employee_name	telephone_number	start_date
⋮			
123-45-6789	Kim	882-0000	1984-03-29
987-65-4321	Kim	869-9999	1981-01-16
⋮			

employee_id	employee_name
⋮	
123-45-6789	Kim
987-65-4321	Kim
⋮	

employee_name	telephone_number	start_date
⋮		
Kim	882-0000	1984-03-29
Kim	869-9999	1981-01-16
⋮		

employee_id	employee_name	telephone_number	start_date
⋮			
123-45-6789	Kim	882-0000	1984-03-29
123-45-6789	Kim	869-9999	1981-01-16
987-65-4321	Kim	882-0000	1984-03-29
987-65-4321	Kim	869-9999	1981-01-16
⋮			

Functional dependencies

- How can we formally reason about when a decomposition is correct?
- Functional dependencies – how the attributes relate to each other:
  - basic idea: super-key – if I know the values of the attributes in a super-key I know the entire tuple

    super-key $\longrightarrow$ all attributes (functional dependency)
    super-key *implies* all attributes (note: "implies" is my term)
  - more generally – any set of attributes can "imply" any other set of attributes

$$\alpha \rightarrow \beta \text{ iff } \forall \text{ tuples } t_{1,} t_2$$
$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta] \text{ or}$$
$$t_1[\beta] \neq t_2[\beta] \Rightarrow t_1[\alpha] \neq t_2[\alpha]$$

FDs continued

- trivial dependencies: $\alpha \rightarrow \alpha$

  $\alpha \rightarrow \beta \text{ if } \beta \subseteq \alpha$

- closure
  - need all FDs
  - some logically implied by others e.g. if $A \rightarrow B$ & $B \rightarrow C$ then $A \rightarrow C$ is implied
- given F = set of FDs, find F+ (the closure) of all logically implied by F

- Why?
- See: http://www.schneier.com/blog/archives/2007/12/anonymity_and_t_2.html
- ***Given a user's public IMDb ratings, which the user posted voluntarily to selectively reveal some of his (or her; but we'll use the male pronoun without loss of generality) movie likes and dislikes, we discover all the ratings that he entered privately into the Netflix system, presumably expecting that they will remain private.***

FD formalism

Amstrong's axioms

- reflexivity: if $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$ (trivial FD)
- augmentation: if $\alpha \rightarrow \beta$ then $\gamma\alpha \rightarrow \gamma\beta$
- transitivity: if $\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$

More rules (can be inferred from Amstrong's axioms)

- union rule: if $\alpha \rightarrow \beta \wedge \alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta\gamma$
- decomposition rule: if $\alpha \rightarrow \beta\gamma$ then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$
- pseudotransitivity rule: if $\alpha \rightarrow \beta \wedge \gamma\beta \rightarrow \delta$ then $\alpha\gamma \rightarrow \delta$

Computing the closure of a set of FDs

Example: R(A,B,C,G,H,I)

```
 F = { A → B
       A → C
       CG → H
       CG → I
       B → H  }
F+ = { A → H                 /* A → B → H       transitivity
        CG → HI               /* CG → H, CG → I  union rule
         AG → I               /*  A → C  augmentation AG → CG → I
         AG → H              /*                               CG → H
...
```

- there is a non-trivial (exponential) algorithm for computing F+

Closure of Attribute Sets

- useful to find if a set of attributes is a superkey
- the closure $\alpha+$ of a set of attributes α under F is the set of all attributes that are functionally determined by α
- there is an algorithm that computes the closure

Example:

<u>Algorithm to Compute (AG)+</u>

start with		**result=(AG)**
$A \rightarrow B$	**expands**	**result=(AGB)**
$A \rightarrow C$	**expands**	**result=(AGBC)**
$CG \rightarrow H$	**“-”**	**result=(AGBCH)**
$CG \rightarrow I$	**“-”**	**result=(AGBCHI)**
$B \rightarrow H$	**no more expansion**	

Note that since G is not on any right hand side, no subset of the attributes can be a superkey unless it contains G for there is no FD to generate it.

Uses of Attribute Closure

There are several uses of the attribute closure algorithm:

- Testing for superkey:
  - To test if α is a superkey, we compute $\alpha+$, and check if $\alpha+$ contains all attributes of *R*.

- Testing functional dependencies
  - To check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in *F*+), just check if $\beta \subseteq \alpha+$.
  - That is, we compute $\alpha+$ by using attribute closure, and then check if it contains β.
  - Is a simple and cheap test, and very useful

Lossless Decompositions

- All attributes of an original schema (*R*) must appear in the decomposition (R_1, R_2):

$$R = R_1 \cup R_2$$

- Lossless-join decomposition.
  For all possible relations *r* on schema *R*

$$r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$$

- A decomposition of R into R_1 and R_2 is lossless join if and only if at least one of the following dependencies is in F^+:
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
  - anyone of these two FDs guarantees uniqueness in the mapping

Dependency Preservation

In a decomposition

- dependencies are preserved if
  - we do not need to join in order to enforce FDs
  - all FDs remain intra-relational and do not become inter-relational constraints
- to check if a decomposition is dependency preserving
  - we need to examine all FDs in F+
- there is an algorithm for testing dependency preservation
  - requires the computation of F+

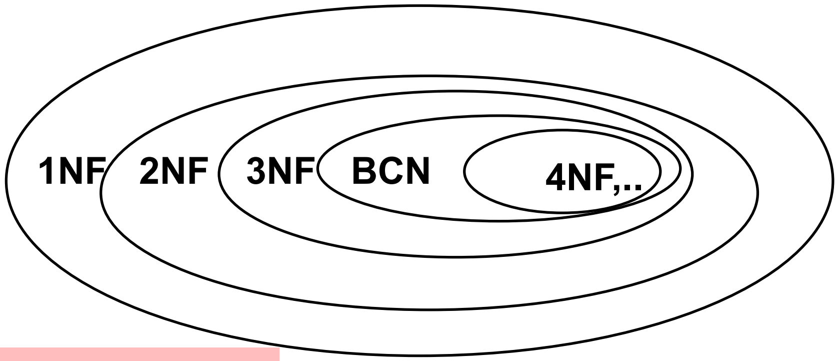
The Normal Forms

- 1NF: every attribute has an atomic value (not a set value)

- 2NF: we will not be concerned in this course

- 3NF: if for each FD $X \rightarrow Y$ either
  - it is trivial or
  - X is a superkey
  - Y-X is a proper subset of a candidate key

- BCNF: if for each FD $X \rightarrow Y$ either
  - it is trivial or
  - X is a superkey



- 4NF,…: we are not concerned in this course.