CMSC 424 – Database design
Lecture 9
Normalization

Mihai Pop

Administrative

- SQL assignment questions – Sharath
- Project – please pair up – submit pairs by Monday, March 4.
- For midterm – chapters 1-4, 6
- Anything you'd like me to go over now?

Accessing databases from software

- Embedded SQL (special commands within C, Java, etc. code)

SQL APIs

- ODBC
- JDBC
- Perl::DBI
- Ruby on Rails

Basic protocol

- connect to server
- run SQL commands – tuples returned as cursors/iterators (allows you iterate over each tuple in result table)
- disconnect from server

- Read chapter 4!!! You'll need this for project.

SQL...last thoughts

- You learn best through practice
- Every database system is different (syntax, conventions, etc.)
- READ THE REFERENCE MANUALS!

Relational Database Design

Where did we come up with the *schema* that we used ?

E.g. why not store the actor names with movies ?

Or, store the author names with the papers ?

Topics:

Formal definition of what it means to be a "good" schema.

How to achieve it.

Movies Database Schema

Movie(*title, year*, length, inColor, studioName, producerC#)

StarsIn(movieTitle, movieYear, starName)

MovieStar(*name*, address, gender, birthdate)

MovieExec(name, address, *cert#,* netWorth)

Studio(*name*, address, presC#)

Changed to:

Movie(*title, year*, length, inColor, studioName, producerC#, starName)

<merged into above>

MovieStar(*name*, address, gender, birthdate)

MovieExec(name, address, *cert#,* netWorth)

Studio(*name*, address, presC#)

Example Relation

Movie(*title, year*, length, inColor, studioName, producerC#, starName)

<merged into above>

MovieStar(*name*, address, gender, birthdate)

MovieExec(name, address, *cert#,* netWorth)

Studio(*name*, address, presC#)

Title	Year	Length	StudioName	prodC#	StarName
Star wars	1977	120	Fox	128	Hamill
Star wars	1977	120	Fox	128	Fisher
Star wars	1977	120	Fox	128	H. Ford
King Kong	2005	..	Studio_A	150	Naomi
King Kong	1940	..	Studio_B	20	Faye

What we're looking for in a schema

- Low/no redundancy
- Easy to understand structure
- Easy to write queries
- Efficient to answer queries
- Ease of maintaining integrity of the data

- Difficult to do this “by hand”

- Normalization – formal algorithms for creating a “reasonable” schema

Combine Schemas?

- Suppose we combine *borrow* 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
⋮	⋮

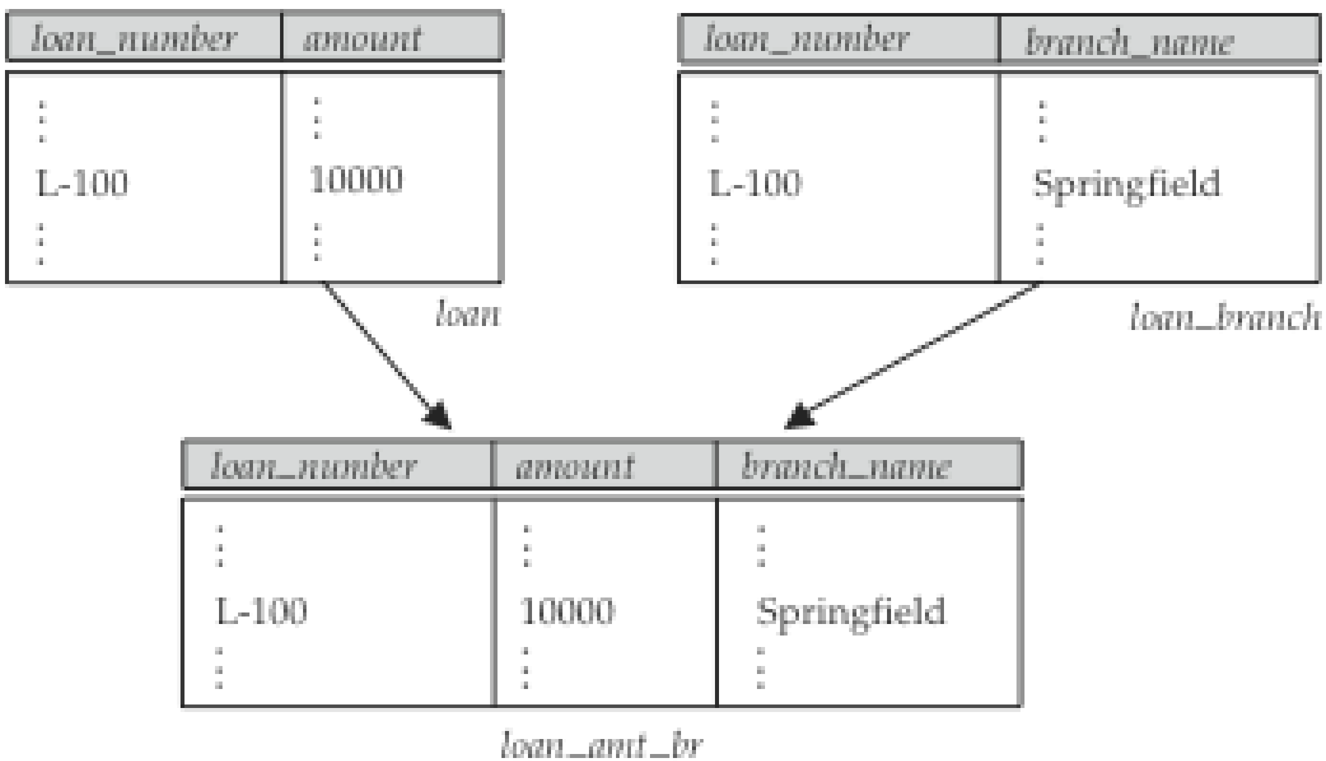
borrower

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

bor_loan

A Combined Schema Without Repetition

- Consider combining *loan_branch* and *loan*
  - *loan_amt_br* = (*loan_number*, *amount*, *branch_name*)
- No repetition (as suggested by example below)



What About Smaller Schemas?

- Suppose we had started with *bor_loan*. How would we know to split up (decompose) it into *borrower* and *loan*?

- Write a rule "if there were a schema (*loan_number, amount*), then *loan_number* would be a candidate key"

- Denote as a functional dependency:

  loan_number $\rightarrow$ *amount*

Functional Dependencies

- set of attributes whose values uniquely determine the values of the remaining attributes e.g. a key defines an FD:

  e.g. in EMP(<u>eno</u>,ename,sal) key FDs: eno $\rightarrow$ ename

  DEPT(<u>dno</u>,dname,floor) eno $\rightarrow$ sal

  WORKS-IN(<u>eno</u>,<u>dno</u>,hours) other FDs: {eno,dno} $\rightarrow$ hours

  for every pair of values of eno,dno there exists exactly one value for hours
- in general if $\alpha \subseteq R$ and $\beta \subseteq R$, then $\alpha \rightarrow \beta$ holds in the extension r(R) of R

  iff for any pair t1 and t2 tuples of r(R) such that t1(α) = t2(α),

  then it is also true that t1(β) = t2(β) (uniqueness of β values)

- we can use the FDs as
  - constraints that we want to enforce (e.g. keys)
  - for checking if the FDs are satisfied in the database

R(A B C D)

A	B	C	D
1	1	1	1
1	2	1	2
2	2	2	2
2	3	2	3
3	3	2	4

A $\rightarrow$ B satisfied? no

A $\rightarrow$ C -”- yes

C $\rightarrow$ A -”- no

AB $\rightarrow$ > D -”- yes

FDs continued

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

  $\alpha \rightarrow \beta$ 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

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 $\alpha \rightarrow \beta$ & $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$

More FD Rules

- union rule: if $\alpha \to \beta$ & $\alpha \to \gamma$ then $\alpha \to \beta\gamma$
- decomposition rule: if $\alpha \to \beta\gamma$ then $\alpha \to \beta$ & $\alpha \to \gamma$
- pseudotransitivity rule: if $\alpha \to \beta$ & $\gamma\beta \to \delta$ then $\alpha\gamma \to \delta$

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

F={ $A \to B$

$A \to C$

$CG \to H$

$CG \to I$

$B \to H$ }

F+= { $A \to H$ /* $A \to B \to H$ transitivity

$CG \to HI$ /* $CG \to H$, $CG \to I$ union rule

$AG \to I$ /* $A \to C$ augmentation $AG \to CG \to I$

$AG \to H$ } /* $CG \to H$

- 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:

Algorithm to Compute (AG)+

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.