Lecture 9: Database Design

Wednesday, February 18, 2015

Agenda

- Review HW #2
- Discuss Normalization Theory
- Take Quiz #3

Normal Forms

- 1st Normal Form (1NF): will discuss briefly
- 2nd Normal Form (2NF): will discuss briefly
- 3rd Normal Form (3NF): will spend time on

Warning: We will use a more formal method from the book to derive 3NF

Optional References

- “A Simple Guide to Five Normal Forms in Relational Database Theory”, William Kent, Sept 1982.
  http://www.bkent.net/Doc/simple5.htm

- “5 Rules of Data Normalization” (Poster), Marc Rettig, Database Programming and Design 2006.
  http://tinyurl.com/khsabww

Relational Schema Design

Main idea:

- Start with some relational schema
- Find out its ***functional dependencies***
- Use them to design a better relational schema

Functional Dependencies

Definition:

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_n$

Formally:

$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n$

FD Examples

Which FDs hold or do not hold on this table:

EmpID	Name	Phone	Title
E0045	Alice	1234	Developer
E3542	Bob	9876	DBA
E0045	Alice	6666	Developer
E9999	Carol	1234	Researcher

EmpID → Name ? Ans = Yes

EmpID → Phone ? Ans = No

EmpID → Title ? Ans = Yes

Title → Phone ? Ans = No

Phone → Title ? Ans = No

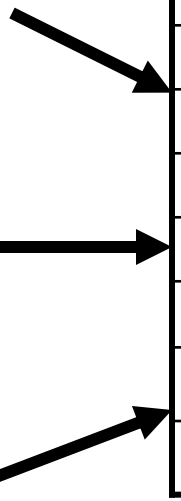
Unnormalized to 1NF

Rule: A database schema is in 1NF if all attributes have scalar values

Students

Student	Semester	GPA	Courses
Alice	Spring15	3.9	Math DB Alg
Bob	Spring15	3.7	DB Alg
Carol	Spring15	3.5	Math Alg

unnormalized



Students'

Student	Semester	GPA	Course
Alice	Spring15	3.9	Math
Alice	Spring15	3.9	DB
Alice	Spring15	3.9	Alg
Bob	Spring15	3.7	DB
Bob	Spring15	3.7	Alg
Carol	Spring15	3.5	Math
Carol	Spring15	3.5	Alg

1NF

1NF examples from book

Original invoice table

	VENDOR_NAME	INVOICE_NUMBER	ITEM_DESCRIPTION
1	Cahners Publishing	112897	VB ad, SQL ad, Library directory
2	Zylka Design	97/522	Catalogs, SQL Flyer
3	Zylka Design	97/533B	Card revision

Invoice tables unnested

	VENDOR_NAME	INVOICE_NUMBER	ITEM_DESCRIPTION_1	ITEM_DESCRIPTION_2	ITEM_DESCRIPTION_3
1	Cahners Publishing	112897	VB ad	SQL ad	Library directory
2	Zylka Design	97/552	Catalogs	SQL flyer	(null)
3	Zylka Design	97/553B	Card revision	(null)	(null)

	VENDOR_NAME	INVOICE_NUMBER	ITEM_DESCRIPTION
1	Cahners Publishing	112897	VB ad
2	Cahners Publishing	112897	SQL ad
3	Cahners Publishing	112897	Library directory
4	Zylka Design	97/522	Catalogs
5	Zylka Design	97/522	SQL flyer
6	Zylka Design	97/533B	Card revision

1NF to 2NF

Rule: A database schema is in 2NF *iff* it is in 1NF and there are no partial FDs on the primary key (i.e. all non-key attributes must be dependent on the entire PK)

Students

Student	Semester	Course	GPA
Alice	Spring15	Math	3.9
Alice	Spring15	DB	3.9
Alice	Spring15	Alg	3.9
Bob	Spring15	DB	3.7
Bob	Spring15	Alg	3.7
Carol	Spring15	Math	3.5
Carol	Spring15	Alg	3.5

1NF

Enrolls

Student	Course	Semester
Alice	Math	Spring15
Alice	DB	Spring15
Alice	Alg	Spring15
Bob	DB	Spring15
Bob	Alg	Spring15
Carol	Math	Spring15
Carol	Alg	Spring15

2NF

GPA

Student	Semester	GPA
Alice	Spring15	3.9
Bob	Spring15	3.7
Carol	Spring15	3.5

2NF

Assumptions:

1. Student, Semester → GPA
2. GPA is not functionally determined by course

2NF example from book

	INVOICE_NUMBER	VENDOR_NAME	INVOICE_ID
1	11287	Cahners Publishing	1
2	97/522	Zylka Design	2
3	97/533B	Zylka Design	3

	INVOICE_ID	INVOICE_SEQUENCE	ITEM_DESCRIPTION
1	1	1	VB ad
2	1	2	SQL ad
3	1	3	Library directory
4	2	1	Catalogs
5	2	2	SQL flyer
6	3	1	Card revision

2NF to 3NF

Rule: A database schema is in 3NF *iff* it is in 2NF and there are no transitive dependencies

Students

EID	Name	Major	College
100	Alice	Math	Natural Sciences
200	Bob	CS	Natural Sciences
300	Carol	Finance	Business

2NF

Assumptions:

EID → Name, Major

Major → College

By transitivity, EID → College

Students'

EID	Name	Major
100	Alice	Math
200	Bob	CS
300	Carol	Finance

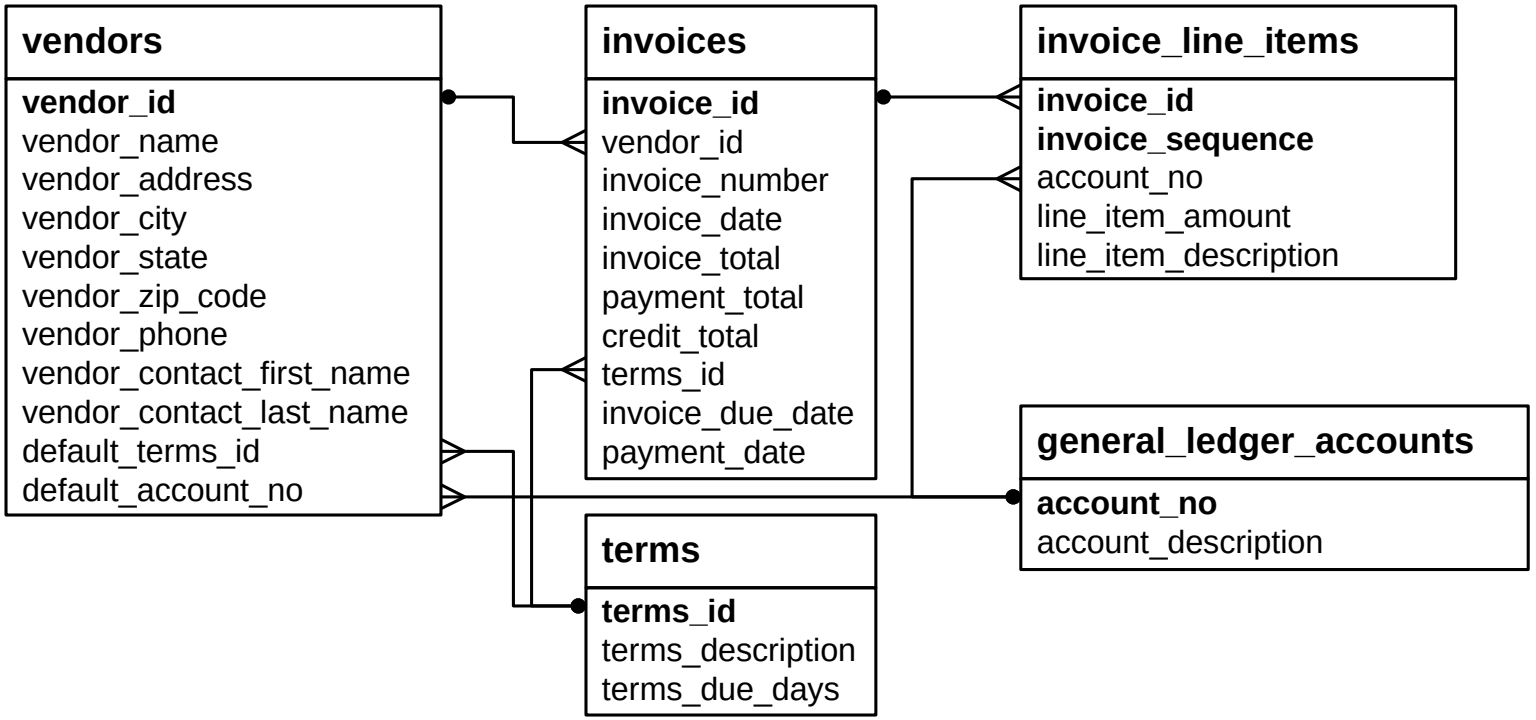
3NF

Majors

Major	College
Math	Natural Sciences
CS	Natural Sciences
Finance	Business

3NF

3NF example from book



Data Anomalies

Assume: One person can have multiple phones, but lives in only one city

SSN	Name	Phone	City
123-45-6789	Alice	512-555-1234	Austin
123-45-6789	Alice	512-555-6543	Austin
987-65-4321	Bob	201-555-2121	San Antonio

Data anomalies:

- Redundancy = repeated data
- Update anomalies = Alice moves to San Marcos
- Deletion anomalies = Bob deletes his phone number

Relation Decomposition

T1

SSN	Name	PhoneNumber	City
123-45-6789	Alice	512-555-1234	Austin
123-45-6789	Alice	512-555-6666	Austin
987-65-4321	Bob	201-555-2121	San Antonio

T2

SSN	Name	City
123-45-6789	Alice	Austin
987-65-4321	Bob	San Antonio

T3

SSN	PhoneNumber
123-45-6789	512-555-1234
123-45-6789	512-555-6666
987-65-4321	201-555-2121

Data anomalies are resolved:

- No more repeated data
- Easy to move Alice to “San Marcos” (how?)
- Easy to delete Bob’s phone number (how?)

Finding ALL Functional Dependencies

- Data anomalies occur when certain “bad” FDs hold
- We know some of the FDs
- Want to find *all* FDs, then look for the “bad” ones

Armstrong's Rules (1/3): Splitting and Combining Rule

$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n$

Is equivalent to:

$A_1, A_2, \ldots, A_n \rightarrow B_1$
$A_1, A_2, \ldots, A_n \rightarrow B_2$
. . . . .
$A_1, A_2, \ldots, A_n \rightarrow B_n$

Armstrong's Rules (2/3): Trivial Rule

$A_1, A_2, \ldots, A_n \rightarrow A_i$ where i = 1, 2, ..., n

Armstrong's Rules (3/3)
Transitivity Closure Rule

If $A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n$

and $B_1, B_2, \ldots, B_n \rightarrow C_1, C_2, \ldots, C_n$

then $A_1, A_2, \ldots, A_n \rightarrow C_1, C_2, \ldots, C_n$

Applying Armstrong's rules to infer new FDs

Start with these FDs:

1. name → maker
2. description → category
3. maker, description → price

Inferred FD	**Rule applied**
4. name, description → name	
5. name, description → maker	
6. name, description → description	
7. name, description → maker, description	
8. name, description → price	

Applying Armstrong's rules to infer new FDs

Start with these FDs:

1. name → maker
2. description → category
3. maker, description → price

Inferred FD	**Rule applied**
4. name, description → name	Trivial rule
5. name, description → maker	Transitivity rule
6. name, description → description	Trivial rule
7. name, description → maker, description	Combining rule
8. name, description → price	Transitivity rule

Eliminating Data Anomalies

Relation R is in Third Normal Form (3NF) if:

- There exists an $A \longrightarrow B$ in R where A is the key of R
- Any other FD in R must be a subset of $A \rightarrow B$
- There are no $A \rightarrow C$ in R such that $B \rightarrow C$

Correct 3NF Decomposition

T1

SSN	Name	PhoneNumber	City
123-45-6789	Alice	512-555-1234	Austin
123-45-6789	Alice	512-555-6543	Austin
987-65-4321	Bob	201-555-2121	San Antonio

T2

SSN	Name	City
123-45-6789	Alice	Austin
987-65-4321	Bob	San Antonio

T3

SSN	PhoneNumber
123-45-6789	512-555-1234
123-45-6789	512-555-6543
987-65-4321	201-555-2121

Lossless decomposition =

1) Attr(T2) ∪ Attr(T3) = Attr(T1)

2) JOIN(T2, T3) = T1

Problematic 3NF Decomposition

T1

SSN	Name	PhoneNumber	City
123-45-6789	Alice	512-555-1234	Austin
123-45-6789	Alice	512-555-6543	Austin
987-65-4321	Bob	201-555-2121	San Antonio

T2

SSN	Name	City
123-45-6789	Alice	Austin
987-65-4321	Bob	San Antonio

T3

Name	PhoneNumber
Alice	512-555-1234
Alice	512-555-6543
Bob	201-555-2121

Lossy decomposition =

1) Attr(T2) ∪ Attr(T3) <> Attr(T1) or

2) JOIN(T2,T3) <> T1

Quiz #3

Q1: List 3 types of database constraints

Q2: Identify every constraint in the CREATE TABLE statement below. Also, specify which column of the table the constraint is defined on. For example, PRIMARY KEY is on the column id. Remember that NOT NULL is also a valid constraint type.

```
CREATE TABLE Individual (
   id NUMBER(8) PRIMARY KEY,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  name_suffix CHAR(3) CHECK (name_suffix IN ('Jr.', 'Sr.', 'Dr.', 'MD')),
  dob DATE CHECK (dob BETWEEN '01-JAN-1900' AND '01-JAN-2000'),
  customer_id NUMBER (8) NOT NULL,
  CONSTRAINT customer_id_fk FOREIGN KEY (customer_id)
   REFERENCES Customer(id) ON DELETE CASCADE
)
```

Quiz #3 Continued

Q3: What does the ON DELETE CASCADE option mean when declaring a foreign key constraint?

Q4: Given the two entities Product(product_id, product_name, category, price) and Supplier(supplier_id, supplier_name, address, city, phone):

a) Briefly describe the type of relationship you would expect to see between these two entities.

b) Based on the relationship you identified in a), come up with the create table statements in SQL that represent these two entities and their relationship.