Databases
Lecture 3

Timothy G. Griffin

Computer Laboratory
University of Cambridge, UK

Databases, Lent 2009

Lecture 03:

Outline

- Joining Tables
- Foreign Keys
- What is `NULL` in SQL?
  - The need for three-valued logic (3VL).
- Views

Product is special!

R

A	B
20	10
4	99

$\Longrightarrow$

$R \times \rho_{A \mapsto C,\ B \mapsto D}(R)$

A	B	C	D
20	10	20	10
20	10	4	99
4	99	20	10
4	99	4	99

- $\times$ is the only operation in the Relational Algebra that created new records (ignoring renaming),
- But $\times$ usually creates too many records!
- Joins are the typical way of using products in a constrained manner.

First, a wee bit of notation

Let $\mathbf{X}$ be a set of k attribute names.

- We will often ignore domains (types) and say that $R(\mathbf{X})$ denotes a relational schema.
- When we write $R(\mathbf{Z},\ \mathbf{Y})$ we mean $R(\mathbf{Z} \cup \mathbf{Y})$ and $\mathbf{Z} \cap \mathbf{Y} = \phi$.
- $u.[\mathbf{X}] = v.[\mathbf{X}]$ abbreviates $u.A_1 = v.A_1 \wedge \cdots \wedge u.A_k = v.A_k$.
- $\vec{\mathbf{X}}$ represents some (unspecified) ordering of the attribute names, $A_1,\ A_2,\ \ldots,\ A_k$
- If $\vec{\mathbf{W}} = B_1,\ B_2,\ \ldots,\ B_k$, then $\mathbf{X} \mapsto \mathbf{W}$ abbreviates $A_1 \mapsto B_1,\ \cdots A_k \mapsto B_k$.

Equi-join

Equi-Join

Given $R(\mathbf{X},\ \mathbf{Y})$ and $S(\mathbf{Y},\ \mathbf{Z})$, we define the equi-join, denoted $R \bowtie S$, as a relation over attributes $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ defined as

$$R \bowtie S \equiv \{t \mid \exists u \in R,\ v \in S,\ u.[\mathbf{Y}] = v.[\mathbf{Y}] \wedge t = u.[\mathbf{X}] \cup u.[\mathbf{Y}] \cup v.[\mathbf{Z}]\}$$

In the Relational Algebra:

$$R \bowtie S = \pi_{\mathbf{X},\mathbf{Y},\mathbf{Z}}(\sigma_{\mathbf{Y}=\mathbf{Y}'}(R \times \rho_{\vec{\mathbf{Y}} \mapsto \vec{\mathbf{Y}}'}(S)))$$

Join example

Students

name	**sid**	**age**	**cid**
Fatima	fm21	20	cl
Eva	ev77	18	k
James	jj25	19	cl

Colleges

cid	**cname**
k	King's
cl	Clare
q	Queens'
⋮	⋮

$$\pi_{\mathbf{name},\mathbf{cname}}(\text{Students} \bowtie \text{Colleges})$$

$\Longrightarrow$

name	**cname**
Fatima	Clare
Eva	King's
James	Clare

The same in SQL

```
select name, cname
from Students, Colleges
where Students.cid = Colleges.cid
```

```
+--------+--------+
| name   | cname  |
+--------+--------+
| Eva    | King's |
| Fatima | Clare  |
| James  | Clare  |
+--------+--------+
```


Keys, again

Relational Key

Suppose $R(\mathbf{X})$ is a relational schema with $\mathbf{Z} \subseteq \mathbf{X}$. If for any records u and v in any instance of R we have

$$u.[\mathbf{Z}] = v.[\mathbf{Z}] \Longrightarrow u.[\mathbf{X}] = v.[\mathbf{X}],$$

then $\mathbf{Z}$ is a superkey for R. If no proper subset of $\mathbf{Z}$ is a superkey, then $\mathbf{Z}$ is a key for R. We write $R(\underline{\mathbf{Z}}, \mathbf{Y})$ to indicate that $\mathbf{Z}$ is a key for $R(\mathbf{Z} \cup \mathbf{Y})$.

Note that this is a semantic assertion, and that a relation can have multiple keys.

Foreign Keys and Referential Integrity

Foreign Key

Suppose we have $R(\underline{\mathbf{Z}}, \mathbf{Y})$. Furthermore, let $S(\mathbf{W})$ be a relational schema with $\mathbf{Z} \subseteq \mathbf{W}$. We say that $\mathbf{Z}$ represents a Foreign Key in S for R if for any instance we have $\pi_{\mathbf{Z}}(S) \subseteq \pi_{\mathbf{Z}}(R)$. This is a semantic assertion.

Referential integrity

A database is said to have referential integrity when all foreign key constraints are satisfied.

Foreign Keys in SQL

```
create table Colleges
(   cid varchar(3) not NULL,
    cname varchar(50) not NULL,
    primary key (cid)    )

create table Students
(   sid varchar(10) not NULL,
    name varchar(50) not NULL,
    age int,
    cid varchar(3) not NULL,
    primary key (sid),
    constraint student_college
          foreign key (cid)
          references Colleges(cid)    )
```

An Example : Whatsamatta U

The entities of Whatsamatta U :

Person

name	pid	email
Fatima	fm21	ft@happy.com
Eva	ev77	eva@funny.com
James	jj25	jj@sad.com
Tim	tgg22	tgg@glad.com

College

cid	cname
k	King's
cl	Clare
q	Queens'
⋮	⋮

Course

csid	course_name	part
a1	Algorithms I	IA
a2	Algorithms II	IB
db	databases	IB
ds	Denotational Semantics	II

Term

tid	term_name
lt	Lent
ms	Michaelmas
er	Easter

An Example : Whatsamatta U

The relationships (more about this in Lecture 11) of Whatsamatta U :

InCollege

pid	cid
fm21	cl
ev77	k
ev77	q
jj25	cl
tgg22	k

Attends

pid	csid
ev77	a2
ev77	db
jj25	a1

OfferedIn

csid	tid
a1	er
a2	ms
db	lt
ds	ms

Lectures

csid	pid
a1	fm21
a2	fm21
a2	tgg22
db	tgg22

Example query

Query

All records of **name** and **term_name** associated with each lecturer and the terms in which they are lecturing.

$$\pi_{\textbf{name},\textbf{term\_name}}(\text{Person} \bowtie \text{Lectures} \bowtie \text{Course} \bowtie \text{OfferedIn} \bowtie \text{Term})$$

name	term_name
Fatima	Michaelmas
Fatima	Easter
Tim	Lent
Tim	Michaelmas

What is NULL in SQL?

What if you don't know Kim's age?

```
mysql> select * from students;
   +------+--------+------+
   | sid  | name   | age  |
   +------+--------+------+
   | ev77 | Eva    |   18 |
   | fm21 | Fatima |   20 |
   | jj25 | James  |   19 |
   | ks87 | Kim    | NULL |
   +------+--------+------+
```

What is NULL?

- NULL is a place-holder, not a value!
- NULL is not a member of any domain (type),
- For records with NULL for **age**, an expression like `age > 20` must unknown!
- This means we need (at least) three-valued logic.

Let $\perp$ represent **We don't know!**

$\wedge$	**T**	**F**	$\perp$
T	**T**	**F**	$\perp$
F	**F**	**F**	**F**
$\perp$	$\perp$	**F**	$\perp$

$\vee$	**T**	**F**	$\perp$
T	**T**	**T**	**T**
F	**T**	**F**	$\perp$
$\perp$	**T**	$\perp$	$\perp$

v	$\neg v$
T	**F**
F	**T**
$\perp$	$\perp$

NULL can lead to unexpected results

```
mysql> select * from students;
+------+--------+------+
| sid  | name   | age  |
+------+--------+------+
| ev77 | Eva    |   18 |
| fm21 | Fatima |   20 |
| jj25 | James  |   19 |
| ks87 | Kim    | NULL |
+------+--------+------+

mysql> select * from students where age <> 19;
+------+--------+------+
| sid  | name   | age  |
+------+--------+------+
| ev77 | Eva    |   18 |
| fm21 | Fatima |   20 |
+------+--------+------+
```

The ambiguity of NULL

Possible interpretations of NULL

- There is a value, but we don't know what it is.
- No value is applicable.
- The value is known, but you are not allowed to see it.
- ...

A great deal of semantic muddle is created by conflating all of these interpretations into one non-value.

On the other hand, introducing distinct NULLs for each possible interpretation leads to very complex logics ...

Not everyone approves of NULL

C. J. Date [D2004], Chapter 19

"Before we go any further, we should make it very clear that in our opinion (and in that of many other writers too, we hasten to add), NULLs and 3VL are and always were a serious mistake and have no place in the relational model."

age is not a good attribute ...

The **age** column is guaranteed to go out of date! Let's record dates of birth instead!

```
create table Students
     (  sid varchar(10) not NULL,
        name varchar(50) not NULL,
        birth_date  date,
        cid varchar(3) not NULL,
        primary key (sid),
        constraint student_college foreign key (cid)
        references Colleges(cid)    )
```


age is not a good attribute ...

```
mysql> select * from Students;
+------+---------+------------+-----+
| sid  | name    | birth_date | cid |
+------+---------+------------+-----+
| ev77 | Eva     | 1990-01-26 | k   |
| fm21 | Fatima  | 1988-07-20 | cl  |
| jj25 | James   | 1989-03-14 | cl  |
+------+---------+------------+-----+
```

Use a view to recover original table

(Note : the age calculation here is not correct!)

```
create view StudentsWithAge as
  select sid, name,
   (year(current_date()) - year(birth_date)) as age,
   cid
  from Students;
```

```
mysql> select * from StudentsWithAge;
+------+---------+------+-----+
| sid  | name    | age  | cid |
+------+---------+------+-----+
| ev77 | Eva     |   19 | k   |
| fm21 | Fatima  |   21 | cl  |
| jj25 | James   |   20 | cl  |
+------+---------+------+-----+
```

Views are simply identifiers that represent a query. The view's name

Contest!! Prizes!! Fame!!

Clearly the calculation of age does not take into account the day and month of year. Two prizes will be awarded in lecture for

SQL Contest

- the cleanest correct solution using standard SQL (no vendor-specific hacks),
- the most obfuscated (yet still correct) solution