SQL 4 : Table Definition and Modification

Gary KL Tam

Department of Computer Science
Swansea University

Overview

Recall, SQL provides:

- Data Manipulation Language (DML)
- Data Definition Language (DDL)
- Data Control Language (DCL)

- In this lecture, we will learn DDL - how to add, update, delete database objects.
- And constraints that enforce database integrity

Table Creation

```
create table [table name] (
    [attribute definition], ..., [attribute definition],
    [primary key definition],
    [candidate key definition], ..., [candidate key definition],
    [foreign key definition], ..., [foreign key definition])
```

Attribute Definition

[attribute name] [attribute type]

where the attribute type can be:

- integer 32bit integer 2147483648 to 2147483647
- double double precision number, e.g., 3.14159
- char(*n*) where *n* is an integer of your choice. This defines a string with at most *n* characters long.
- many other types depend on the concrete database system.
  In this course, we will mostly work with the above types only.

```
create table PROF (
    pid char(20),
    name char(20),
    dept char(20),
    rank char(20),
    sal integer)
```

Tuple Insertion

insert into [table name] values ([value 1], [value 2], $\cdots$)

Example

insert into PROF values ('$p1$', 'Adam', 'CS', 'asst', '6000')

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000

Tuple Deletion

delete from *T* where *P*

- *T* is a table name
- *P* is a predicate (same as predicate in the where clause of an SQL statement)

The statement removes all the tuples of *T* that satisfy *P*.

Example

delete from PROF where sal <= 8000

PROF

pid	name	dept	rank	sal
*p*1	Adam	CS	asst	6000
*p*2	Bob	EE	asso	8000
*p*3	Calvin	CS	full	10000
*p*4	Dorpthy	EE	asst	5000
*p*5	Emily	EE	asso	8500

PROF

pid	name	dept	rank	sal
*p*3	Calvin	CS	full	10000
*p*5	Emily	EE	asso	8500

Tuple Update

update T set $A = v$ where P

- T is a table name
- A is an attribute and v is the new value of the attribute
- P is a predicate

The statement updates the A values to v for all the tuples of T that satisfy P.

Example:

update PROF set salary = 6000 where salary = 5000

update PROF set salary = salary * 1.05 where salary <= 6000

Updating the Table

- Alter: alter a table

```
alter table PROF add column room integer;
alter table PROF drop column room;
```

- Drop: remove a whole table

```
drop table PROF;
```

- How is that different from?

```
delete from PROF;
```

Integrity and Validation

- Entity Integrity Contraint : Primary key
- Referential Integrity Constraint : Foreign key
- Validation

Integrity and Validation

- Entity Integrity Contraint : Primary key
- Referential Integrity Constraint : Foreign key
- Validation

Entity Integrity Constraint

- No component of the primary key of a base relation is allowed to accept nulls.

- NULL: information is missing for some reasons.
- a base relation: one which permanently exists in the database.
- a transient relation: temporary results (which is also a relation) of a query.
- every relation entities in the database should be uniquely identifiable
  (you do not want to be confused with someone else, do you?)

Key constraint - revisit

An attribute (or set of attributes, i.e. composite) K of a relation R is a candidate key for R if and only if it satisfies the following two time-independent properties:

- Uniqueness: At any given time, no two rows of R have the same value for K.
- Minimality: If K is composite, then no component of K can be eliminated without destroying the uniqueness property. (Otherwise, it is a superkey)

Primary Key is chosen from candidate keys, rest are known as alternate keys.

Primary Key in SQL

- A primary key for each table is defined through a constraint
- Primary Key also automatically adds UNIQUE and NOT NULL to the relevant column definition.
- Big hint to the DBMS: optimize for searches by this set of attributes!

primary key([attribute list])

every table should have exactly one primary key

```
create table PROF (
    pid char(20),
    name char(20),
    dept char(20),
    rank char(20),
    sal integer,
    primary key (pid)
)
```

Candidate Key Definition

unique ([attribute list])

You can define as many candidate keys as you want.

```
create table PROF (
    pid char(20), name char(20), dept char(20), rank char(20), sal integer,
    primary key (pid),
    unique (name),
    unique (dept, rank))
```

Integrity and Validation

- Entity Integrity Contraint : Primary key
- Referential Integrity Constraint : Foreign key
- Validation

Recap

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

Natural Join $\bowtie$ Prof $\bowtie$ Teach

pid	name	dept	rank	sal	cid	year
$p1$	Adam	CS	asst	6000	$c1$	2011
$p2$	Bob	EE	asso	8000	$c2$	2012
$p1$	Adam	CS	asst	6000	$c2$	2012

Delete Problem

What if I delete the first tuple in PROF? → P1 has been deleted

PROF

pid	name	dept	rank	sal
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
~~$p1$~~	~~$c1$~~	~~2011~~
$p2$	$c2$	2012
~~$p1$~~	~~$c2$~~	~~2012~~

Natural Join $\bowtie$

pid	name	dept	rank	sal	cid	year
$p2$	Bob	EE	asso	8000	$c2$	2012

Information Loss

What is $p1$ in TEACH refers to now?

Update Problem

What if I update the first tuple in PROF?

Similar to other Slide.

PROF

pid	name	dept	rank	sal
$p6$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

Natural Join $\bowtie$

pid	name	dept	rank	sal	cid	year
$p2$	Bob	EE	asso	8000	$c2$	2012

Information Loss

What is $p1$ in TEACH refers to now?

Must be in Corresponding table.

Referential Integrity Constraint

- The database must not contain any unmatched foreign key values.

- For every non-null value for a foreign key, there is a matching primary key

Foreign Key Definition 1

foreign key (attribute list) references tablename (attribute list)

The attributes in the attribute list must have the same types as those in the primary key in the table referenced.

```
create table prof(
    pid char(20), name char(20),
    dept char(20), rank char(20),
    sal integer, primary key (pid)
);

create table teach (
    pid char(20),  cid char(20),
    year integer,   primary key (pid, cid),
    foreign key (pid) references prof (pid)
);
```

The statements in the previous slide

- requires pid be declared either primary key or unique in PROF.
- does not allow the update/deletion of a tuple in PROF if it is referenced by a tuple in TEACH.

Example:

PROF

pid	**name**	**dept**	**rank**	**sal**
***p*1**	Adam	CS	asst	6000
***p*2**	Bob	EE	asso	8000
*p*3	Calvin	CS	full	10000
*p*4	Dorothy	EE	asst	5000
*p*5	Emily	EE	asso	8500

TEACH

pid	**cid**	**year**
***p*1**	*c*1	2011
***p*2**	*c*2	2012
***p*1**	*c*3	2012

The first two tuples of PROF cannot be deleted.

ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key constraint fails

Another example

- employee's DeptNo references to dept table.

```
create table dept (
    DeptNo integer not null,
    Name char(20) not null,
    Budget integer not null,
    primary key (DeptNo)
};
create table employee (
    EmpNo integer not null,
    LastName char(20) not null,
    FirstName char(20) not null,
    DeptNo integer not null,
    MgrNo integer,
    primary key ( EmpNo ),
    foreign key (DeptNo) references dept (DeptNo)
);
```

- DeptNo need not be a primary key in employee

Self Referential Integrity

```
create table employee (
    EmpNo integer not null,
    LastName char(20) not null,
    FirstName char(20) not null,
    DeptNo integer not null,
    MgrNo integer,
    primary key (EmpNo),
    foreign key (MgrNo) references employee (EmpNo)
);
```

reference itself

- Manager is also an employee
- Note in this example:
- - an employee may not have a manager (it can be null).

Self Referential Integrity

- employee manager is an employee.
- What about managing director?
- null allowed in MgrNo foreign in employee.

Employee

EmpNo	FirstName	...	MgrNo
p5	Arnold (Head Of Department)	...	NULL
p2	Stephen	...	p5
p7	Gary	...	p5

Find all who are being supervised.

$T1 \leftarrow \rho_{e1}(employee) \times \rho_{e2}(employee)$

$\Pi_{e1.EmpNo, e1.FirstName, e2.FirstName}(\sigma_{e1.MgrNo=e2.EmpNo}(T1))$

EmpNo	e1.FirstName	e2.FirstName
p2	Stephen	Arnold (Head Of Department)
p7	Gary	Arnold (Head Of Department)

Reaction Policy

- What if we really want to modify a tuple in PROF which is referenced by some tuple in TEACH?
- We need to provide reaction policies for update/delete:

Restrict Restrict to the case where there are no such matching entities (otherwise it is not carried out) [default].

Cascades The delete/update operation cascades to delete/update those matching entities.

Nullifies The foreign key is set to null in all such matching entities and the item is then deleted/updated (should not apply if the foreign key cannot accept nulls in the first place – big problem)

Foreign Key Definition 2

foreign key ([attribute list]) references [table name] on delete cascade

If a referenced tuple is deleted, so are all the referencing tuples. → refrence deleted ↓ So are all refrence tuples

```
create table PROF (
    pid char(20), name char(20), dept char(20), rank char(20), sal integer,
    primary key (pid))

create table TEACH (
    pid char(20), cid char(20), year integer
    primary key (pid, cid),
    foreign key (pid) references PROF on delete cascade)
```

defining the foreign Key

Delete example

foreign key (attribute list) references tablename (attribute list)
on delete cascade → definition

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

No inconsistency

- If the first tuple of PROF is deleted, so are the first and third tuples of TEACH.

pid	name	dept	rank	sal
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

pid	cid	year
$p2$	$c2$	2012

So first deletion so are the matching references.

Similarly... for update

foreign key (attribute list) references tablename (attribute list)
on update cascade

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

No inconsistency

- If the first tuple of PROF is updated, so are the first and third tuples of TEACH.

pid	name	dept	rank	sal
$p6$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

pid	cid	year
$p6$	$c1$	2011
$p2$	$c2$	2012
$p6$	$c2$	2012

updates all tuples

Note the asymmetry

So works as

Froiegn Key back to table to update or delete.

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p3$	$c3$	2012

- Suppose table R (e.g. TEACH) references table S (PROF).
- You can define "fixes" that propogate changes backwards from S to R. So ∴ reverses the change R->S => S->R
- You define them in table R because it is the table that will be affected.
- You cannot define fixes that propogate forward from R to S.

Integrity and Validation

- Entity Integrity Contraint : Primary key
- Referential Integrity Constraint : Foreign key
- Validation

Validation

- May involve extra constraints, for examples,
  - Supplier numbers must be of the form Snnnn (where nnnn stands for up to four decimal digits);
  - Part numbers must be of the form Pnnnnn (5 digits);
  - Supplier status values must be in the range 1-100;
  - Supplier and part cities must be drawn from a certain list;
  - Part colours must be drawn from a certain list;
  - Part weights must be greater than zero;
  - Shipment quantities must be a multiple of 100;
  - If the supplier city is London, then the status must be 20;
- implementable but database specific: *check, trigger, procedure etc.*
  - `http://www.w3schools.com/sql/sql_check.asp`
  - the details of these commands are beyond CS-250