Table Definition and Modification

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
create table [table name] (  
[attribute definition], ..., [attribute definition],  
[primary key definition],  
[candidate key definition], ..., [candidate key definition],  
[foreign key definition], ..., [foreign key definition])
[attribute name] [attribute type]

where the attribute type can be:

- **integer**
- **char\((n)\)** where \(n\) is an integer of your choice. This defines a string with at most \(n\) characters long.
- other types that depend on the concrete database system. In this course, we will work with the above types only.
create table PROF ( 
    pid char(20),
    name char(20),
    dept char(20),
    rank char(20),
    sal integer)
primary key ([attribute list])

A primary key functions in the same way as a candidate key, except that 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))
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))
foreign key ([attribute list]) references [table name]

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)
The statement in the previous slide does not allow the deletion of a tuple in PROF if it is references by a tuple in TEACH.

**Example:**

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Adam</td>
<td>CS</td>
<td>asst</td>
<td>6000</td>
</tr>
<tr>
<td>p2</td>
<td>Bob</td>
<td>EE</td>
<td>asso</td>
<td>8000</td>
</tr>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p4</td>
<td>Dorothy</td>
<td>EE</td>
<td>asst</td>
<td>5000</td>
</tr>
<tr>
<td>p5</td>
<td>Emily</td>
<td>EE</td>
<td>asso</td>
<td>8500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pid</th>
<th>cid</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>2011</td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>2012</td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2012</td>
</tr>
</tbody>
</table>

The first two tuples of PROF cannot be deleted.
foreign key ([attribute list]) references [table name] on delete cascade

If a referenced tuple is deleted, so are all the referencing 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)
Example:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>dept</th>
<th>rank</th>
<th>sal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Adam</td>
<td>CS</td>
<td>asst</td>
<td>6000</td>
</tr>
<tr>
<td>p2</td>
<td>Bob</td>
<td>EE</td>
<td>asso</td>
<td>8000</td>
</tr>
<tr>
<td>p3</td>
<td>Calvin</td>
<td>CS</td>
<td>full</td>
<td>10000</td>
</tr>
<tr>
<td>p4</td>
<td>Dorothy</td>
<td>EE</td>
<td>asst</td>
<td>5000</td>
</tr>
<tr>
<td>p5</td>
<td>Emily</td>
<td>EE</td>
<td>asso</td>
<td>8500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pid</th>
<th>cid</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>c1</td>
<td>2011</td>
</tr>
<tr>
<td>p2</td>
<td>c2</td>
<td>2012</td>
</tr>
<tr>
<td>p1</td>
<td>c2</td>
<td>2012</td>
</tr>
</tbody>
</table>

If the first tuple of PROF is deleted, so are the first and third tuples of TEACH.
insert into [table name] values ([value 1], [value 2], ...)

Example:

insert into PROF values (‘p1’, ‘Adam’, ‘CS’, ‘asst’, 6000)
delete from $T$ where $P$

- $T$ is a table name
- $P$ is a predicate (same as a 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 salary $\leq 5000$
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 $\leq$ 6000