数据库系统原理

陈岭

浙江大学计算机学院

5 SQL语言（3）

- 视图
- 索引

视图

- 在某些情况下，让所有用户看到整个逻辑模型是不合适的
- 考虑一个职员需要知道教师的标识、姓名和所在系名，但是没有权限看到教师的工资值。此人应该看到的关系由如下SQL语句所描述：

```
select ID, name, dept_name
from instructor
```

- 视图就提供了这种机制：向用户隐藏特定的数据
- SQL允许通过查询来定义“虚关系”，它在概念上包含查询的结果，但并不预先计算并存储。像这种作为虚关系对用户可见的关系称为视图（view）

视图定义

- 在SQL中，我们用create view命令定义视图，命令的格式为：

  create view *v* as < query expression >

  - < query expression >可以是任何合法的查询表达式
  - *v* 表示视图名
  - 使用视图的目的：安全及易于使用
  - 对应地，删除视图，使用命令： drop view *v*

- 例，重新考虑需要访问*instructor*关系中除*salary*之外的所有数据，此视图的定义如下：

```
create view faculty as
select ID, name, dept_name
from instructor
```

视图定义

- 例，创建一个视图，列出Physics系在2009年秋季学期开设的所有课程，以及每个课程在哪栋建筑的哪个房间授课的信息

```
create view physics_fall_2009 as
select course.course_id, sec_id, building, room_number
from course, section
where course.course_id = section.course_id
        and course.dept_name = 'Physics'
        and section.semester = 'Fall'
        and section.year = '2009';
```

视图定义

- 视图的属性名也可以按下述方式显示指定
  - 例，列出每个系中所有教师的工资总和

```
create view departments_total_salary(dept_name, total_salary) as
select dept_name, sum (salary)
from instructor
group by dept_name;
```

- 当我们定义一个视图时，数据库系统存储视图定义本身，而不存储定义该视图的查询表达式的执行结果

SQL查询中使用视图

- 一旦定义了一个视图，就可以用视图名指代该视图生成的虚关系。由于数据库只存储视图定义本身，那么当视图关系出现在查询中时，它就会被已存储的查询表达式代替
- 例，使用视图physics_fall_2009，找到所有于2009年秋季学期在Watson大楼开设的Physics课程

```
select course_id, room_number
from physics_fall_2009
where building = 'Watson';
```

SQL查询中使用视图

- 例，定义视图physics_fall_2009_Watson，列出于2009年秋季学期在Watson大楼开设的所有Physics课程的标识和教室号

```
create view physics_fall_2009_watson as
select course_id, room_number
from physics_fall_2009
where building = 'Watson';
```

等价于：

```
create view physics_fall_2009_watson as
(select course_id, room_number
 from (select course.course_id, building, room_number
       from course, section
        where course.course_id = section.course_id
              and course.dept_name = 'Physics'
              and section.semester = 'Fall'
              and section.year = '2009')
where building= 'Watson';
```

视图更新

- 假设我们向视图*faculty*插入一条新元组，可写为：

```
insert into faculty values ('30765', 'Green', 'Music');
```

但这个插入必须表示为对instructor关系的插入，即必须给出salary的值。存在两种合理的解决方法：

- 拒绝插入，并向用户返回一个错误信息
- 向instructor关系插入元组('30765', 'Green', 'Music', null)

视图更新

- 通过视图修改数据库的另一类问题发生在这样的视图：

```
create view instructor_info as
select ID, name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

这个视图列出了大学里每个教师的ID、name和建筑名。若执行以下视图插入：

```
insert into instructor_info values ('69987', 'White', 'Taylor');
```

  - 假设没有标识为69987的教师，也没有位于Taylor大楼的系，唯一的解决办法是（显然不可行）：
    - 向instructor中插入元组（'69987', 'White',null,null）
    - 向department中插入元组（null, 'Taylor',null）

视图更新

- 一般地，如果定义视图的查询对下列条件都能满足，我们称SQL视图是可更新的（updatable），即视图上可以执行插入、更新或删除
  - from子句中只有一个数据库关系
  - select子句中只包含关系的属性名，不包含任何表达式、聚集或distinct声明
  - 任何没有出现在select子句中的属性可以取空值；即这些属性上没有not null约束，也不构成主码的一部分
  - 查询中不含有group by或having子句

视图更新

- 在上面条件的限制下，下面的视图上允许执行udpate、insert和delete操作：

```
create view history_instructors as
select *
from instructor
where dept_name = ’ History’ ;
```

- 即便是在可更新的情况下，仍然存在问题。假设向视图 *history_instructors* 中插入元组（‘25566’，‘Brown’，‘Biology’，100000），但这个元组并不满足视图所要求的选择条件
- 在默认情况下，SQL允许执行上述更新。但是，可以在视图定义的末尾加上with check option子句来定义视图，对更新操作进行检查

视图更新

- SQL-99对于何时可以在视图上执行插入、更新和删除有更复杂的规则集，我们就不在这里讨论了

物化视图

- 物化视图（materialized view）：特定数据库系统允许存储视图关系，但是它们保证，如果用于定义视图的实际关系改变，视图也跟着修改
  - 例，如果视图department_total _salary是物化的，则它的结果就会存放在数据库中
- 物化视图维护（materialized view maintenance），通常简称视图维护（view maintenance）：保持物化视图一直在最新状态的过程。视图维护的三种方式：
  - 当构成视图定义的任何关系被更新时，进行视图维护
  - 当视图被访问时，才进行视图维护
  - 周期性地进行视图维护（在这种情况下，访问的数据可能是过时的）

索引

- 我们用create index命令，为关系中的某些属性创建索引，它允许数据库系统高效地找到关系中那些在索引属性上取给定值的元组，而不用扫描关系中的所有元组
  - CREATE INDEX <*i-name*> ON <*table-name*> (<*attribute-list*>);
  - CREATE INDEX *ins_index* on *instructor*(*ID*);
  - CREATE INDEX *ins_ID_name_index* on *instructor*(*ID*, *name*);
- 我们用create unique index命令，为关系中的某些属性创建唯一索引
  - CREATE UNIQUE INDEX *uni_stu_index* on *student*(*ID*);
- 我们用drop index命令，删除一个索引
  - DROP INDEX <*i-name*>

总结

- 视图可以定义为包含查询结果的关系。视图可以隐藏不需要的信息，可以把信息从多个关系收集到一个单一的视图中
- 视图更新问题及规则
- 物化视图
- 索引的作用与创建

谢谢！