Principles of Database Systems

Intermediate SQL（2）

主讲教师：

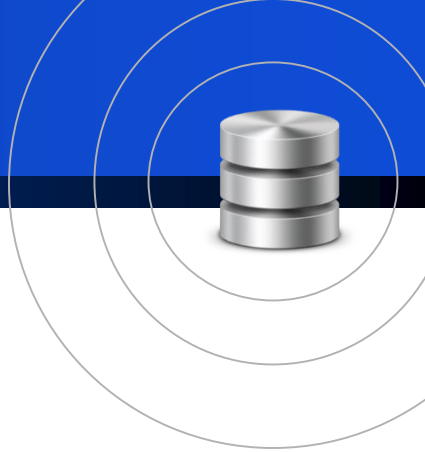
薛昕惟

办公室：信息楼320B

xuexinwei@dlut.edu.cn

Constituent Parts of SQL (SQL组成部分)

- The SQL language has several parts:
  - Data-definition language (DDL)
  - Data-manipulation language (DML)
  - **Integrity (完整性) (included in DDL)**
  - **View definition (视图定义) (included in DDL)**
  - **Transaction control (事务控制)**
  - Authorization (授权)
  - Embedded SQL and dynamic SQL (嵌入式SQL及动态SQL)



Views

Views

- In some cases, it is not desirable for all users to see the entire logical model.

- Consider a person who needs to know an instructors name and department, but not the salary. This person should see a relation described, in SQL, by

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

- **Any disadvantages?**

Views

- A **view** provides a mechanism to hide certain data from the view of certain users.

- Any relation that is not of the conceptual model but is made visible to a user as a "**virtual relation**" is called a **view**.(不是概念模型的一部分，对用户可见的“虚关系”)

Views

- SQL provides the **create view** command to create a view, which is stored in the database **in the long run**
  - **create view** v **as** <query expression>
  - where
    - <query expression> is any legal expression
    - the **view** name is represented by v

- View definition is not the same as creating a new relation by evaluating the query expression. (创建视图与创建关系不同)
  - Rather, a view definition causes the **saving of an expression**; the expression is substituted into queries using the view. (存储的是表达式)

Try…

- A view of instructors without their salary

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

Try…

- Find all instructors in the Biology department

```
select name
  from faculty
  where dept_name = 'Biology'
```

Try…

- Create a view of department salary totals

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

Views Defined Using Other Views

- Explain the following views:

- **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';

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

Views Defined Using Other Views

- Expand use of a view in a query/another view

```
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';
```

Materialized Views

- **Materializing a view**(物化视图): **create a physical table** containing all the tuples in the result of the query defining the view

- If relations used in the query are updated, the materialized view result becomes out of date
  - Need to **maintain** the view(维护视图), by updating the view whenever the underlying relations are updated.

Drop View

- The **Drop View** command deletes the definition the view from the **data dictionary**.

drop view view_name;

Other views depending on this dropped view should be deleted explicitly.

Update of a View

- Add a new tuple to faculty view which we defined earlier

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

- Two reasonable approaches:
  - Reject the insertion
  - Insert the tuple

    ('30765', 'Green', 'Music', null)

    into the instructor relation

    (必须转化为对实际关系的修改)

Some Updates cannot be Translated Uniquely

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

- **insert into** instructor_info **values** ('69987', 'White', 'Taylor');
  - which department, if multiple departments in Taylor?
  - what if no department is in Taylor?

Some Updates cannot be Translated Uniquely

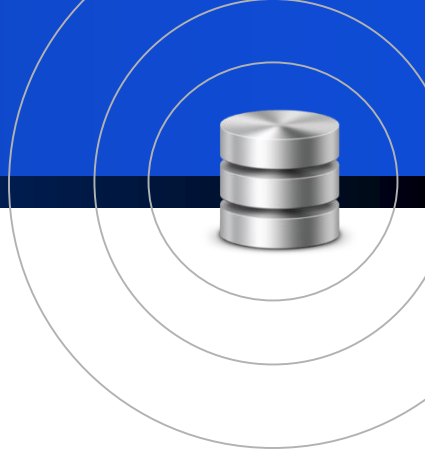
- Most SQL implementations allow updates only on simple views
  - The **from** clause has only **one** database relation.
  - The **select** clause contains only attribute names of the relation, and does **not have any expressions**, **aggregates**, or **distinct** specification.
  - Any attribute not listed in the **select** clause can be set to null.
  - The query does **not have** a **group** by or **having** clause.

And Some Not at All

- **create view** history_instructors **as**
  select *
  from instructor
  where dept_name= 'History';

- What happens if we insert ('25566', 'Brown', 'Biology', 100000) into history_instructors?

- **with check option:** if a tuple inserted into the view does not satisfy the view's **where clause** condition, the insertion is rejected by the database system



Transactions

Transactions

- A transaction is a sequence of queries and update statements on DB, executed as a single, and are started implicitly and terminated by one of commit work(提交)or rollback/abort work
  - **Commit work :** makes the updates performed by the transaction become permanent in the database.
  - **Rollback work:** undoes all the updates performed by the SQL statements in the transaction.

Transactions

- Unit of work
- Atomic transaction
  - either fully executed or rolled back as if it never occurred
- Transactions begin implicitly
  - Ended by **commit work** or **rollback work**
- But default on most databases: each SQL statement commits automatically
  - Can turn off auto commit for a session (e.g. using API)
  - In SQL:1999, can use: **begin atomic** …. **end**
    - Not supported on most databases

Integrity Constraints

Integrity Constraints

- Integrity constraints guard **against accidental damage** to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.
  - An instructor name cannot be *null*.
  - No two instructors can have the same instructor ID.
  - The budget of a department must be greater than $0.00.

Integrity Constraints on a Single Relation

- primary key
- not null
- Unique
- foreign keys
- check (P), where P is a predicate

Not Null

- Declare name and budget to be **not null**
  - name **varchar**(20) **not null**
  - budget **numeric**(12,2) **not null**

Unique Constraints

- **unique** (A_1, A_2, …, A_m)
  - The unique specification states that the attributes A_1, A_2, … A_m form a **candidate key**.
  - Candidate keys are **permitted to be null** (in contrast to primary keys).

The check clause

- The **check** clause is applied to relation declaration
  - check (P), where P is a predicate which must be satisfied by every tuple in the relation.

- Example: ensure that the budget of a department must be greater than $0.00
  - **create table** department
    (dept name **varchar (20),**
    building **varchar (15),**
    budget **numeric (12,2),**
    primary key (dept name**)**
    check(budget>0**));**

Try…

- Ensures that semester is one of fall, winter, spring or summer:

```
create table section (
course_id  varchar (8),
sec_id  varchar (8),
semester  varchar (6),
year  numeric (4,0),
building  varchar (15),
room_number  varchar (7),
time slot id  varchar (4),
primary key (course_id, sec_id, semester, year),
check (semester  in ('Fall', 'Winter', 'Spring', 'Summer'))
);
```

Referential Integrity

- Ensures that a value that appears in one relation for a given set of attributes **also appears** for a certain set of attributes in another relation.

  - Example: If "Biology" is a department name appearing in one of the tuples in the *course* relation, then there exists a tuple in the *department* relation for "Biology". -- **foreign key**

```
create table course (
    course_id   char(5) primary key,
    title       varchar(20),
    dept_name varchar(20) references department
)
```

Cascading Actions in Referential Integrity

- When the DB is modified by **Insert, Delete,** and **Update**, the tests must be made in order to preserve the referential integrity constraint.

- **create table** course (

  …

  dept_name **varchar**(20),

  foreign key (dept_name) **references** department

  on delete cascade

  on update cascade,

  . . .

)

- alternative actions to **cascade**: **set null**, **set default**

Integrity Constraint Violation During Transactions

- E.g. **create table** person (
  ID **char**(10) **primary key** ,
  name **char**(40),
  spouse **char**(10),
  foreign key spouse **references** person)

- How to insert a tuple without causing constraint violation ?
  - set spouse to null initially, update after inserting all persons (not possible if spouse attribute declared to be **not null**)
  - OR **defer**(延迟) constraint checking

Defer Constraint Checking

- The SQL standard allows a clause **initially deferred** to be added to a constraint specification.

- For constraints declared as **deferrable**, executing a statement **set constraints** constraint-list **deferred** as part of a transaction causes the checking of the specified constraints to be deferred to the end of that transaction.

Complex Check Conditions and Assertions

- **check** (time_slot_id
  in (**select** time_slot_id **from** time_slot))

- Unfortunately: subquery in check clause not supported by pretty much any database.

Assertions

- An **assertion** is a predicate expressing a condition that we wish the database always to satisfy.
  - e.g. domain constraints, referential-integrity constraint

- An assertion in SQL takes the form

create assertion <assertion-name> **check** <predicate>

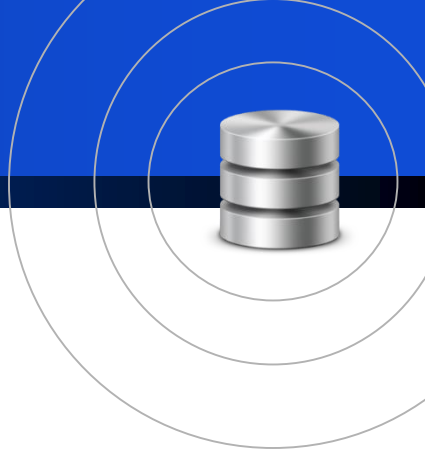
Assertions

- E.g. The value of the attribute tot_cred for each student must equal the sum of credits of courses that the student has completed successfully.

```
create assertion credits_earned_constraint check
  (not exists (select ID from student
    where tot_cred < > (
          select sum(credits)
          from takes join course
          on takes.course_id= course.course_id
          where student.ID=takes.ID and grade is not
                                null and grade < > 'F')
```

Assertions

- When an assertion is made, the DBMS tests it for validity. Any modification to DB is allowed only if it does not cause that assertion to be violated.
  - This testing may introduce a significant amount of overhead, hence **assertions should be used with great care.**
- Not supported by every DBMS.



Review

Review

- Join Expressions
  - left outer join, right outer join, full outer join
  - inner join = join
  - Join types and join conditions
- Views
  - Create view
  - Use views in SQL queries
  - Update view: with check option
- Transactions
  - Atomic
  - Commit work, Rollback work

Review

- Integrity Constraints
  - **Not null**
  - **unique** (A_1, A_2, …, A_m), candidate key, null
  - **Check**(P)
  - Referential Integrity, foreign key, **on delete/update cascade**, on delete/update set null, on delete/update set default
  - defer constraint checking
  - **create assertion** <assertion-name> **check** <predicate>