Structured Query Language (SQL)-2

陆伟

Database Systems

March 26, 2021

Outline

- Index
- View
- Transaction control
- Integrity Constraints
- Access control
- Procedure and Trigger

Index

- An index is a structure that provides accelerated access to the rows of a table based on the values of one or more columns.
- We will discuss the index in detail in another topic.

Index

- Creating an Index (CREATE INDEX)

  CREATE [UNIQUE] INDEX IndexName
  ON TableName(columnName[ASC|DESC][,…])

```
CREATE UNIQUE INDEX StudentInd
ON Student(sNo);
```

Index

- DROP INDEX IndexName

  DROP INDEX StudentInd

View

- The dynamic result of one or more relational operations on the base relations to produce another relation.
- A view is a *virtual relation* that does not necessarily exist in the in the database but can be produced upon request by a particular user, at the time of request.
- The DBMS stores the definition of the view in the database.

View

- Creating a View (CREATE VIEW)

```
CREATE VIEW ViewName[(newColumnName[,…])]
AS subselect [WITH [CASCADED|LOCAL] CHECK OPTION];
```

- If WITH CHECK OPTION is specified, SQL ensures that if a row fails to satisfy the WHERE clause of the defining query of a view, it is not added to the underlying base table of the view.

View

- 例子

```
CREATE VIEW IS_Student
AS SELECT sNo,sName,sex,age
    FROM student
    WHERE dNo IN(select dNo
                 from department
                 where dName='信息学院');
```

View

- Removing a View (DROP VIEW)

  DROP VIEW ViewName [RESTRICT|CASCADE];

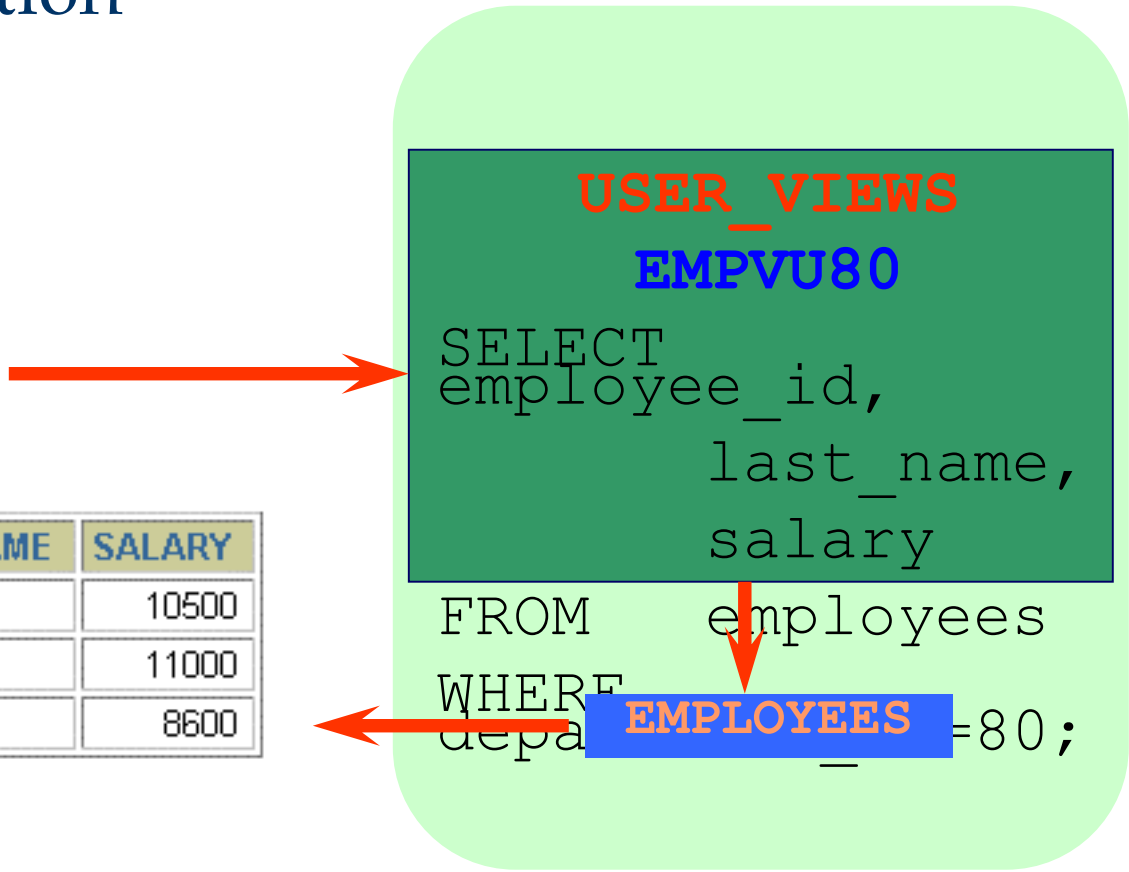
- If CASCAE is specified, DROP VIEW deletes all related dependent objects, in other words, all objects that reference the view.

View

- View Resolution

```
SELECT *
FROM empvu80;
```

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600



View

- Restriction on Views
  - If a column in the view is based on an aggregate function, then the column may appear only in SELECT and ORDER BY clauses of queries that access the view. In particular, such a column may not be used in a WHERE clause and may not be an argument to an aggregate function in any query based on the view.
  - A grouped view may never be joined with a base table or a view.

View

- View Updatability
  - All updates to a base table are immediately reflected in all views that encompass that base table.
  - Similarly,we may except that if a view is updated then the base table(s) will reflect that change.
  - Consider that if any view is updatable.

View

```
UPDATE IS_Student
SET sName='马虎'
WHERE sNo='070115';
```

```
UPDATE Student
SET sName='马虎'
WHERE sNo='070115'
   AND dNo IN();
```

```
INSERT
INTO IS_Student
VALUES('070116', '麻烦', '男', 20);
```

View

- WITH CHECK OPTION
  - The rows that enter or leave a view are called **migrating rows**.
  - The WITH CHECK OPTION clause of the CREATE VIEW statement prohibits a row migrating out of the view.

View

```
CREATE VIEW IS_Student
AS SELECT *
   FROM student
   WHERE dNo IN(select dNo
                from department
                where dName='信息学院')
With check option;

INSERT
INTO IS_Student
VALUES('070117', '烦人' , '男', 20, '02');
```

View

- Advantages of using view
  - Data independence
  - Currently
  - Improved security
  - Reduced complexity
  - Convenience
  - Customization
  - Data integrity

View

- Disadvantages of using view
  - Update restriction
  - Structure restriction
  - Performance

Transaction

- A transaction is a sequence of database statements that needs to execute atomically .
- A database transaction consists of one of the following:
  - **DML statements which constitute one consistent change to the data.**
  - **One DDL statement**
  - **One DCL statement**

Transaction

- Beginning and end of transaction
  - Implicitly declare
  - Explicitly declare (begin transaction, commit / rollback)
  - Programmatic SQL aborts.

Transaction

- Beginning and end of transaction in PostgreSQL through interactive terminal

```
SELECT * FROM librarian;
INSERT INTO librarian VALUES('Mary');

BEGIN TRANSACTION (implicit)
SELECT * FROM librarian;
commit (implicit)
BEGIN TRANSACTION (implicit)
 INSERT INTO librarian VALUES('Mary');
COMMIT(implicit)
```

Transaction

```
BEGIN TRANSACTION; (explicit)
 SELECT * FROM librarian;
 INSERT INTO librarian VALUES('Mary');
COMMIT; (explicit)
```

- Configure certain aspects of the transaction

```
SET TRANSACTION
[READ ONLY|READ WRITE]|
[ISOLATION LEVEL READ UNCOMMITTED|
                 READ COMMITTED|
                 REPEATABLE READ|
                 SERIALIZABLE]
```

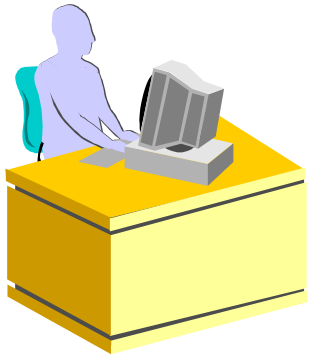
Integrity Constraints

- Immediate and Deferred Integrity Constraints
  - In some situations, we do not want integrity constraints to be checked immediately, that is after every SQL statement has been executed, but instead at transaction commit

```
SET CONSTRAINTS
{ALL|constraintName[,…]}{[NOT]DEFERRABLE}
[INITIALLY IMMEDIATE|DEFEREED]
ALTER TABLE
ADD CONSTRAINT constraintName
FOREIGN KEY(columnName) REFERENCES
tableName.columnName
DEFERRABLE INITIALLY DEFERRED/IMMEDIATE
```

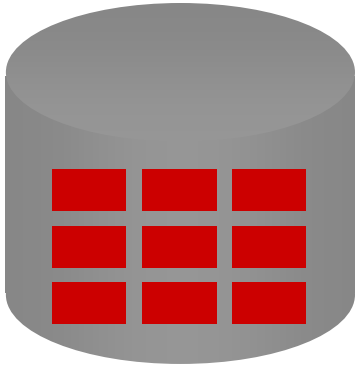
Access control

Database administrator



Username and password

Privileges



Access control

- Database security
  - System security
  - Data security
- System privileges: Gaining access to the database
- Object privileges: Manipulating the content of the database objects

Access control

- System privileges
  - There are many system privileges available according to different DBMSs.
- The database administrator has high-level system privileges for tasks such as:
  - Creating new users
  - Removing users
  - Removing tables

Access control

- An application developer, for example, may have the following system privileges:
  - CREATE SESSION
  - CREATE TABLE
  - CREATE VIEW
  - CREATE PROCEDURE

Access control

- Grant and revoke system privileges

```
GRANT {system_privilege|role}
      [, {system_privilege|role} ]…
TO {user|role|PUBLIC}
   [, {user|role|PUBLIC}]
[WITH ADMIN OPTION];

REVOKE {system_privilege|role}
       [, {system_privilege|role} ]…
FROM {user|role|PUBLIC}
     [, {user|role|PUBLIC}]...;
```

Access control

- Object previliges
  - Object privileges vary from object to object.
  - An owner has all the privileges on the object.

Access control

Object Privilege	Table	View	Sequence	Procedure
ALTER	√		√	
DELETE	√	√		
EXECUTE				√
INDEX	√			
INSERT	√	√		
REFERENCES	√	√		
SELECT	√	√	√	
UPDATE	√	√		

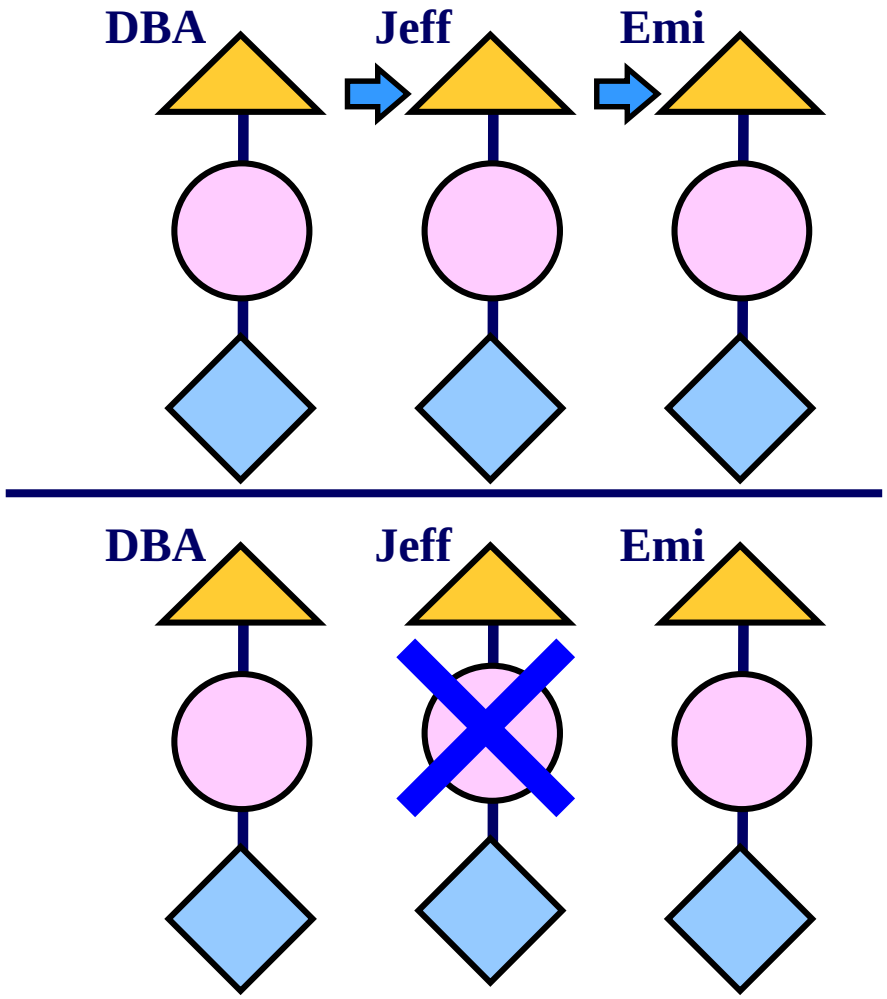
Access control

- Grant and revoke object privileges

```
GRANT {object_privilege[(column_list)]
      [, object_privilege[(column_list)]…
      |ALL [PRIVILEGES]}
ON [schema.]object
TO {user|role|PUBLIC}[,{user|role|PUBLIC}]
[WITH GRANT OPTION];
REVOKE {object_privilege
   [, object_privilege]…|ALL [PRIVILEGES]}
ON [schema.] object
FROM {user|role||PUBLIC}
      [, {user|role||PUBLIC}]...
      [CASCADE CONSTRAINTS];
```

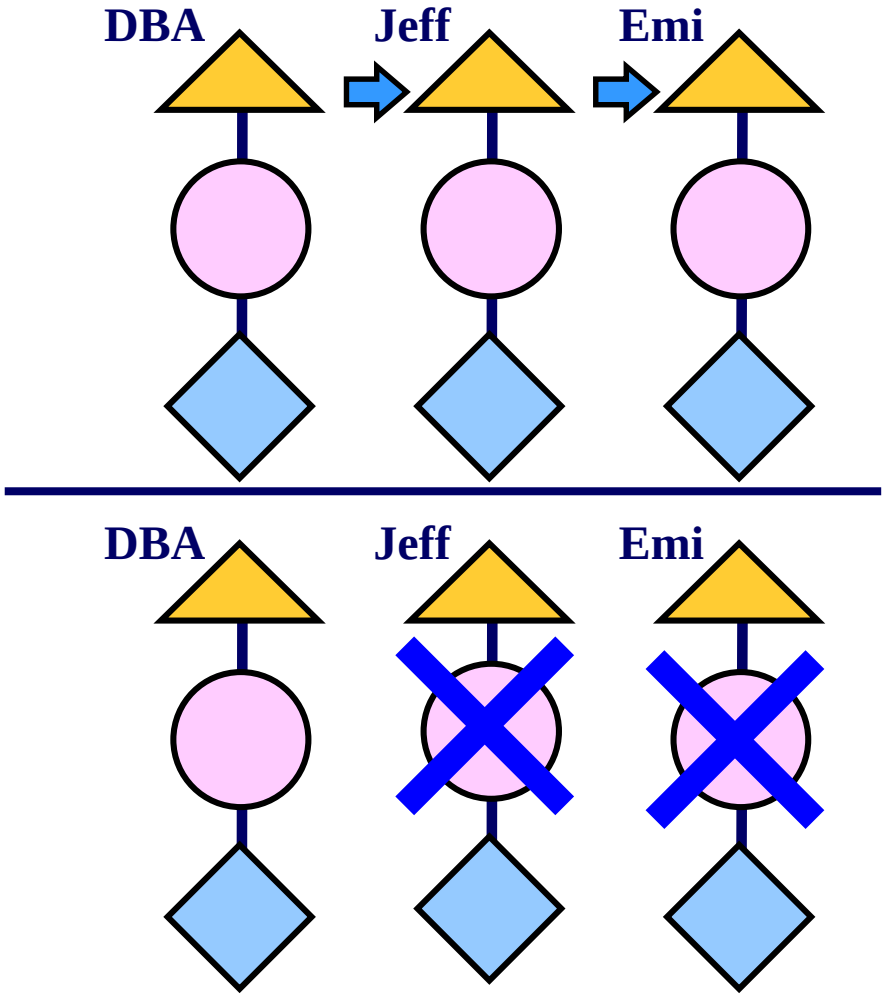
Access control

- About WITH ADMIN OPTION in system privilege



Access control

- About WITH GRANT OPTION in object privilege



Procedure and Trigger

- Procedure:将执行计划存储于数据库中的数据库对象。

```
CREATE OR REPLACE PROCEDURE procedureName
  [(argument[{IN|OUT|IN OUT}] type,
   …
   argument[{IN|OUT|IN OUT}] type)]{IS|AS}
Procedure_body
```

Procedure and Trigger

- Trigger: procedure that starts automatically if specified changes occur to the DBMS
- Three parts:
  - Event (activates the trigger)
  - Condition (tests whether the triggers should run)
  - Action (what happens if the trigger runs)

Procedure and Trigger

- CREATE TRIGGER语法格式

  CREATE TRIGGER <触发器名>
  {BEFORE | AFTER} <触发事件> ON <表名>
  REFERENCING NEW|OLD ROW AS<变量>
  FOR EACH {ROW | STATEMENT}
  [WHEN <触发条件>]<触发动作体>

触发器又叫做事件-条件-动作（event-condition-action）规则。当特定的系统事件发生时，对规则的条件进行检查，如果条件成立则执行规则中的动作，否则不执行该动作。规则中的动作体可以很复杂，通常是一段SQL存储过程。

Procedure and Trigger

- 定义触发器的语法说明
  - （1）表的拥有者才可以在表上创建触发器
  - （2）触发器名
    - 触发器名可以包含模式名，也可以不包含模式名
    - 同一模式下，触发器名必须是唯一的
    - 触发器名和表名必须在同一模式下
  - （3）表名
    - 触发器只能定义在基本表上，不能定义在视图上
    - 当基本表的数据发生变化时，将激活定义在该表上相应触　发事件的触发器

Procedure and Trigger

（4）触发事件

- 触发事件可以是INSERT、DELETE或UPDATE
  也可以是这几个事件的组合
- 还可以UPDATE OF<触发列，…>，即进一步指明修改哪些列时激活触发器
- AFTER/BEFORE是触发的时机
  - AFTER表示在触发事件的操作执行之后激活触发器
  - BEFORE表示在触发事件的操作执行之前激活触发器

Procedure and Trigger

(5) 触发器类型

- 行级触发器（FOR EACH ROW）
- 语句级触发器（FOR EACH STATEMENT）

例如,在例5.11的TEACHER表上创建一个AFTER UPDATE触发器，触发事件是UPDATE语句：

```
UPDATE TEACHER SET Deptno=5;
```

假设表TEACHER有1000行

- 如果是语句级触发器，那么执行完该语句后，触发动作只发生一次
- 如果是行级触发器，触发动作将执行1000次

Procedure and Trigger

（6）触发条件

- 触发器被激活时，只有当触发条件为真时触发动作体才执行;否则触发动作体不执行。
- 如果省略WHEN触发条件，则触发动作体在触发器激活后立即执行

注意：不同的RDBMS产品触发器语法各部相同