第 19 章 触发器

触发器可以看作由指定事件激活而自动执行的存储过程。激活事件一般是表的 DML 操作或数据库的指定操作，一般完成对数据的约束检查或审计功能。

本章主要内容包括:

- 触发器与存储过程及约束的差异
- Oracle 的触发器
- SQL Server 的触发器
- 管理触发器

19.1 触发器与存储过程及约束的差异

触发器可看作由指定事件激活而自动执行的存储过程，但触发器与存储过程有以下区别：

- 存储过程的执行不需要其他事件触发。
- 存储过程一般有输入或输出参数，触发器没有参数。
- 触发器与其触发事件处于同一事务，在触发器中执行 commit 或 rollback 命令，会连同其触发事件一起提交或回滚。

一般由触发器完成数据约束检查或数据库审计的功能，表上的约束显然也会检查数据是否违反某些限制条件，但触发器的功能更强：

- 与 check 约束不同，触发器可以引用其他表的数据，使得新添加到表中的数据或更新后的数据可以与其他表的数据进行对比，然后根据对比结果执行相关操作。
- 约束使用标准错误信息，而触发器可以自定义错误处理及错误信息。
- 触发器可以对比更新前后的数据值，根据对比结果执行指定任务。

19.2 触发器中引用的两种临时结构

为了能在触发器中对比修改前后的数据差异，Oracle 和 SQL Server 各自定义了两种临时表(Oracle 中称为临时结构，我们在这里统一称为临时表)，Oracle 称为 old 和 new，SQL Server 称为 deleted 和 inserted。

两种临时表中的数据由以下操作产生：

- 执行 insert 操作时，新记录也会把一份拷贝添加到 new 或 inserted 临时表。
- 执行 delete 操作时，删除记录的一份拷贝会添加到 old 或 deleted 临时表。
- 执行 update 操作时，更新记录的原值所在行会添加到 old 或 deleted 临时表，更新后的新值所在行会添加到 new 或 inserted 临时表。

19.3 Oracle 中的触发器

相比 SQL Server，Oracle 触发器在功能方面要强大很多，本节对其类型和语法分别说明。

19.3.1 触发器类型

根据激活触发器的事件可以分为：

- DDL 触发器：执行 DDL 或 DCL 操作时触发。
- DML 触发器：对表或视图执行 update、insert、delete 操作时触发。
- system 或 database 触发器：由系统或数据库事件触发。

根据触发器执行的次数，DML 触发器又分为：

- row-level 触发器：对触发事件所影响的每行记录执行一次触发器。
- statement-level 触发器：对触发事件只执行一次触发器。
- compound 触发器：以上两种方式的综合。

如果一个 update 语句修改了 10 条记录，则 row-level 触发器会执行 10 次，而 statement-level 只执行 1 次，另外，row-level 触发器可以使用 new 和 old 引用修改前后的旧值与新值，statement-level 触发器则不能。

根据触发器执行的时机，可以分为：

- instead of 触发器：触发器代替触发事件，触发事件未执行，但可以使用 new 及 old。
- before 触发器：在触发事件执行之前执行。
- after 触发器：在触发事件执行之后执行。

创建触发器时，要注意触发器的主体不能超过 32760 个字节，因为其主体存储在 long 类型的列中，如果触发器主体超过这个大小限制，可以考虑把触发器的主要代码编写为函数或存储过程，然后在触发器中调用。

19.3.2 DDL 触发器

创建 DDL 触发器的语法结构如下：

create [or replace] trigger *trigger_name*
{before | after | instead of } *ddl_event* on {database | schema}
[when (*logical_expression*)]
[declare]
    declaration_statements;
begin
    execution_statements;
end [*trigger_name*];

其中的 ddl_event 可以取下面的关键字：

表 19-1 DDL 事件可取关键字

ddl_event	说明
alter	对数据库对象执行 alter 语句。
analyze	对数据库对象执行 analyze 命令以得到其统计信息。
associate statistics	对列、序列、函数、包等对象关联某个统计类型。
audit	激活数据库对象的审计功能。
comment	对表或列附加注释。
create	创建表、视图、序列等数据库对象。
ddl	此表列出的所有 ddl 语句。
disassociate statistics	撤销对列、序列、函数、包等对象的某个统计类型的关联。
drop	删除数据库对象。
grant	对用户授予权限。
noaudit	撤销对数据库对象的审计功能。
rename	更改数据库对象名称。
revoke	撤销用户权限。

truncate	清空表的数据。

只能在执行 create 命令创建对象时，DDL 触发器才能使用 instead of 关键字。

下面给出一个审计 create 操作的示例，当用户创建数据库对象时，create 操作的信息会记录在 audit_creation 表中，其第一个列 audit_id 的值由序列 sq_audit_creation 自动填充。

创建表 audit_ddl 存储 DDL 操作的信息：

```
SQL> create table audit_creation
  2  (
  3      audit_id number,
  4      audit_owner varchar2(30),
  5      audit_obj_name varchar2(30),
  6      audit_date date,
  7      constraint pk_audit_ddl primary key(audit_id)
  8  )
  9  /
```

创建序列：

```
SQL> create sequence sq_audit_creation;
```

最后创建触发器：

```
SQL> create or replace trigger audit_creation
  2  before create on schema
  3  begin
  4     insert into audit_creation values
  5     (
  6        sq_audit_creation.nextval,
  7        ora_dict_obj_owner,
  8        ora_dict_obj_name,
  9        sysdate
 10     );
 11  end audit_creation;
 12  /
```

其中的系统函数 ora_dict_obj_owner 以及 ora_dict_obj_name 分别用于得到触发事件所操作的对象属主及名称。

触发器创建完毕后，可以创建一个表 t 激活触发器：

```
SQL> create table t(a int);
```

查询 audit_creation，可以发现其中记录的建表操作信息：

```
SQL> select * from audit_creation;

  AUDIT_ID  AUDIT_OWNER     AUDIT_OBJ_NAME      AUDIT_DATE
---------- ---------------- ------------------- -------------------
         1 SCOTT            T                   2016-07-09 15:26:52
```

其他触发事件，这里不再赘述。

下面示例说明 instead of 关键字的作用。

首先删除之前创建的触发器：

```
SQL> drop trigger audit_creation;
```

创建 instead of 触发器，其中只给出一行提示信息：

```
SQL> create or replace trigger audit_instead_creation
```

```
  2  instead of create on schema
  3  begin
  4     dbms_output.put_line('Creation operation not allowed now.');
  5  end audit_instead_creation;
  6  /
```

执行下面创建表的操作：

```
SQL> create table tt(a int);
Creation operation not allowed now.
```

虽然最后给出了“表已创建”的提示信息，如果查询此表，可以发现其并不存在，说明建表操作被替换为触发器中的代码，从而建表操作并未执行：

```
SQL> select * from tt;
select * from tt
              *
第 1 行出现错误:
ORA-00942: 表或视图不存在
```

19.3.3 DML 触发器

创建 DML 触发器的语法结构如下：

create [or replace] trigger *trigger_name*
{before | after}
{insert | update | update of *column1* [, *column2*, [...]] | delete}
on *table_name*
[for each row]
[when *logical_expression*]
[declare]
    declaration_statements;
     begin
    execution_statements;
end [*trigger_name*];

下面示例创建 row-level 类型的 DML 触发器，审计对 emp 表的更新操作，如果 sal 列值更新前后的差超过 2000，则把更新操作的信息记入 emp_sal_audit 表。

创建 emp_sal_audit 表，用于存放更新 emp 表的 sal 列的审计结果：

```
SQL> create table emp_sal_audit
  2  (
  3     empno number(4),
  4     old_sal number(7,2),
  5     new_sal number(7,2)
  6  )
  7  /
```

创建触发器：

```
SQL> create trigger before_emp_sal_update
  2  before update of sal
  3  on emp
  4  for each row when(abs(new.sal-old.sal)>2000)
  5  begin
  6     insert into emp_sal_audit values(:old.empno, :old.sal, :new.sal);
  7  end before_emp_sal_update;
```

```
  8  /
```

上述代码中的 old 及 new 即是对两个临时结构的引用。

对 emp 表执行 update 操作：

```
SQL> select empno,sal from emp where deptno=10;

     EMPNO        SAL
---------- ----------
      7782       2450
      7839       5000
      7934       1300

SQL> update emp set sal=1500 where deptno=10;
```

然后查询 emp_sal_audit 表，可以发现符合条件的更新操作信息已经被记入：

```
SQL> select * from emp_sal_audit;

     EMPNO     OLD_SAL    NEW_SAL
---------- ---------- ----------
      7839       5000       1500
```

下面示例创建 statement-level 类型的 DML 触发器，审计用户对 emp 表的 update 操作。为了避免混淆，在进行下面步骤之前，请先删除之前创建的 before_emp_sal_update 触发器。

创建表 emp_update_audit 记录对 emp 表的更新信息：

```
SQL> create table emp_update_audit
  2  (
  3     audit_id number,
  4     user_id varchar2(30),
  5     update_date date,
  6     constraint pk_emp_update_audit primary key(audit_id)
  7  )
  8  /
```

创建序列以自动填充 emp_update_audit 的 audit_id 列：

```
SQL> create sequence sq_emp_update;
```

创建触发器：

```
SQL> create trigger emp_update_audit
  2  after update of sal on emp
  3  begin
  4     insert into emp_update_audit values
  5     (sq_emp_update.nextval, user, sysdate);
  6  end emp_update_audit;
  7  /
```

对 emp 的 sal 列执行 update 操作，检查触发器是否生效：

```
SQL> update emp set sal=sal+100 where deptno=10;
```

查询 emp_update_audit，可以发现以上更新操作的信息已被记入：

```
SQL> select * from emp_update_audit;

  AUDIT_ID USER_ID                UPDATE_DATE
---------- ---------------------- -------------------
         1 SCOTT                  2016-07-09 18:00:48
```

19.3.4 系统触发器

系统触发器可以审计服务器启动、关闭，服务器错误以及用户的登录、退出等操作，适合于记录用户连接服务器的总时间或服务器运行的总时间。

系统触发器的语法结构如下：

create [or replace] trigger *trigger_name*
{before | after } *database_event* on {database | schema}
[declare]
    declaration_statements;
begin
    execution_statements;
end [*trigger_name*];

下面示例使用系统触发器记录用户的登录：

```
SQL> conn system/oracle
已连接。
SQL> create table connection_audit
  2  (
  3     audit_id number,
  4     user_name varchar2(30),
  5     event_type varchar2(20),
  6     event_date date,
  7     constraint pk_conn_audit primary key(audit_id)
  8  )
  9  /
```

创建序列：

```
SQL> create sequence sq_conn_audit;
```

创建触发器：

```
SQL> create or replace trigger conn_audit
  2  after logon on database
  3  begin
  4     insert into connection_audit values
  5     (sq_conn_audit.nextval, user, 'CONNECT', sysdate);
  6  end conn_audit;
  7  /
```

注意，数据库的 logon 事件只能使用 after 关键字。

在另一个客户端以 scott 连接数据库：

```
SQL> conn scott/tiger
已连接。
```

查询 connection_audit 表，可以发现登录信息已记入：

```
SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';
会话已更改。
SQL> select * from connection_audit;

  AUDIT_ID  USER_NAME            EVENT_TYPE        EVENT_DATE
---------- -------------------- ----------------- -------------------
         1 SCOTT                CONNECT           2016-07-09 18:27:32
```

19.4 SQL Server 的触发器

在版本更新过程中，SQL Server 的触发器功能不断加强，特别是在 2005 版本加入了 DDL 触发器类型，与 Oracle 的差距进一步降低。

19.4.1 SQL Server 触发器类型

按照触发事件可分为：

- DDL 触发器
- DML 触发器
- logon 触发器

按照触发器或执行的时机可分为：

- alter 触发器：在触发事件执行之后，触发器执行。
- instead of 触发器：触发事件并未执行，而是被替换为触发器中的代码。

与 Oracle 对比，SQL Server 的触发器有以下特点：

- 不支持 Oracle 的 before 触发器功能。
- 不支持 Oracle 的 row-level 触发器，只支持类似 Oracle 的 statement-level 形式的触发器，即 DML 语句只会激活触发器执行一次而不管其影响了多少行记录。
- DDL 触发器只支持 after 形式。
- DDL 触发器除与 Oracle 的 DDL 触发器类似外，还包含其系统触发器的部分功能。

19.4.2 DDL 触发器

激活 DDL 触发器的事件包括 DDL、DCL 以及 update statistics 语句，DDL 语句主要包括 create、alter、drop，DCL 语句主要包括 grant、deny、revoke。对临时表和存储过程的 DDL 操作不会激活 DDL 触发器。与 DML 触发器不同，DDL 触发器不属于任何架构，也不存在 object_id、object_name 等属性。

SQL Server 的 DDL 触发器可以在数据库或服务器范围创建，分别由数据库或服务器范围发生的事件激活，用于审计数据库或服务器范围的事件，也可以在满足指定条件情况下执行特定任务，如在某些条件下禁止建表。

创建 DDL 触发器的语法结构为：

create trigger *trigger_name*
on {all server | database}
for *event_type(s)*
as
execution_statements;

其中，all server 表示服务器范围的触发器，database 表示当前数据库范围的触发器。for 关键字也可以用 after 代替。

下面示例创建的触发器禁止对数据库执行 drop table 与 alter table 操作：

```
1> create trigger deny_alter_table
2> on database
3> for drop_table, alter_table
4> as
5>    print 'Drop or alter tables not allowed now.'
6>    rollback
7> go
```

执行 drop table 操作，测试以上触发器的效果：

```
1> drop table emp
2> go
Drop or alter tables not allowed now.
消息 3609，级别 16，状态 2，服务器 LAW_X240，第 1 行
事务在触发器中结束。批处理已中止。
```

下面示例创建服务器范围的触发器，禁止执行 create login 操作：

```
1> create trigger deny_login_creation
2> on all server
3> for create_login
4> as
5>    print 'Login creation not allowed now.'
6>    rollback
7> go
```

测试其效果：

```
1> create login login1 with password='login1login1'
2> go
Login creation not allowed now.
消息 3609，级别 16，状态 2，服务器 LAW_X240，第 1 行
事务在触发器中结束。批处理已中止。
```

19.4.3 DML 触发器

创建 DML 触发器的语法结构为：

create trigger *trigger_name*

on *table_name*

[instead of | after] *dml_operation(s)*

as

execution_statements

创建触发器时，名称不能与相同架构下的表相同，这与 Oracle 不同，在 Oracle 中，触发器与表处于不同的名称空间(namespace)，触发器的名称与表的名称可以相同。

after 触发器与其他事件的执行先后顺序如下：

- 约束检查。
- 创建 inserted 表及 deleted 表。
- 执行触发事件。
- 执行触发器。

下面示例审计对 emp 表的 sal 列的 update 操作，如果对 emp 表的 sal 列执行了 update 操作，则在 emp_sal_audit 表中记录下 update 操作的相关信息。

首先创建 emp_sal_audit：

```
1> create table emp_sal_audit
2> (
3>    empno numeric(4),
4>    old_sal numeric(7,2),
5>    new_sal numeric(7,2),
6>    user_name varchar(15),
7>    update_time datetime
8> )
9> go
```

其中 empno 记录被修改记录的 empno，old_sal 记录 sal 列修改之前的值，new_sal 记录 sal 列修改之后的新值，user_name 记录执行 update 操作的数据库用户名称，update_time 记录 update 操作执行的时间。

创建 after 触发器：

```
1> create trigger tri_emp_sal_audit
2> on emp
3> after update
4> as
5> if update(sal)
6>     insert into emp_sal_audit
7>     select i.empno, d.sal, i.sal, user, getdate()
8>     from inserted i, deleted d
9>     where i.empno=d.empno
10> go
```

SQL Server 不支持 Oracle 创建触发器的 update of column 语法形式，在上述代码中，通过 update(sal)系统函数判断 sal 列的值是否被修改，若被修改则把更新记录的 empno、更新前后的 sal 值、执行 update 操作的用户名称以及 update 操作的执行时间添加入 emp_sal_audit 表，以供以后审计之用。

对 emp 表执行 update 操作：

```
1> update emp set sal=sal+100 where deptno=10
2> go
```

查询审计表，可以发现以上 update 操作的信息已经添加进来：

```
1> select * from emp_sal_audit
2> go
empno  old_sal   new_sal   user_name       update_time
------ --------- --------- --------------- -----------------------
  7934   1300.00   1400.00 dbo             2016-07-10 11:20:42.563
  7839   5000.00   5100.00 dbo             2016-07-10 11:20:42.563
  7782   2450.00   2550.00 dbo             2016-07-10 11:20:42.563
```

如果要完成与 Oracle 中的示例类似的功能，则可以把条件改为：

if(select max(abs(i.sal-d.sal)) from inserted i, deleted d where i.empno=d.empno)>2000

如果读者对比一下上述示例与对应 Oracle 示例的功能以及语法形式，可以发现两者的功能还是有所差别，修改条件后，只要一个 update 操作导致的修改结果有满足上述条件的记录，就会把这个 update 操作的信息记录下来，包括不满足上述条件的其他记录，也就是说，SQL Server 支持持类似 Oracle 中的 statement-level 触发器，而不支持 Oracle 中的 row-level 触发器。

instead of 触发器执行时：

- 约束还未检查。
- inserted 表与 deleted 表已经创建。

如下面示例使用 instead of 触发器禁止用户在每天晚上 18:00 到 23:59:59 之间以及凌晨 00:00:00 到 06:59:59 之间对 emp 表执行 update 操作。因为 instead of 触发器的内容会代替 update 操作，其实现的关键是如何处理不在禁止时间范围内的 update 操作。

首先删除之前创建的 tri_emp_sal_audit 触发器：

```
1> drop trigger tri_emp_sal_audit
2> go
```

创建 instead of 触发器：

```
1> create trigger tri_deny_emp_update
2> on emp
3> instead of update
4> as
5> if (datepart(hour, getdate())>=18 and datepart(hour, getdate())<=23)
6>     or (datepart(hour, getdate())>=0 and datepart(hour, getdate())<=6)
7>    print 'Update not allowed now.';
8> else
9> begin
10>      delete from emp where empno in(select empno from deleted);
11>      insert into emp select * from inserted;
12> end
13> go
```

修改计算机系统时间，使其处于禁止时间范围，查看当前时间以确认：

```
1> select getdate() as now
2> go
now
-----------------------
2016-07-10 19:25:29.070
```

查看 deptno 为 30 的记录的 sal 值：

```
1> select empno,sal from emp where deptno=30
2> go
empno  sal
------ ---------
  7499   1600.00
  7519   1250.00
  7654   1250.00
  7698   2850.00
  7844   1500.00
  7900    950.00
```

对 emp 表执行 update 操作，修改其 sal 列的值：

```
1> update emp set sal=3000 where deptno=30
2> go
Update not allowed now.
```

重新执行对 emp 表的查询，可以发现 deptno 为 30 的记录的 sal 值并未被修改：

```
1> select empno,sal from emp where deptno=30
2> go
empno  sal
------ ---------
  7499   1600.00
  7519   1250.00
  7654   1250.00
  7698   2850.00
  7844   1500.00
  7900    950.00
```

修改计算机的系统时间，使其不在禁止范围，执行下面查询进行确认：

```
1> select getdate() as now
2> go
now
-----------------------
2016-07-10 11:29:25.693
```

再执行相同的 update 操作：

```
1> update emp set sal=sal+10 where deptno=30
2> go
```

查询 emp 表，可以发现这时 update 操作生效：

```
1> select empno, sal from emp where deptno=30
2> go
empno  sal
------ ---------
  7499   1610.00
  7521   1260.00
  7654   1260.00
  7698   2860.00
  7844   1510.00
  7900    960.00
```

19.4.4 logon 触发器

用户登录时，在安全认证过程结束、建立连接之前，激活 logon 触发器，若认证过程失败，则不会激活 logon 触发器。

下面示例禁止登录帐号 login1 在 18:00:00 至 23:59:59 之间登录服务器：

```
1> create trigger denylogon
2> on all server
3> for logon
4> as
5> begin
6>      if original_login()='login1' and
7>         cast(getdate() as time) between '18:00:00' and '23:59:59'
8>              rollback
9> end
10> go
```

若登录时间在禁止范围，则给出如下错误：

```
C:\>sqlcmd -U login1 -P login1login1
Sqlcmd: 错误: Microsoft ODBC Driver 11 for SQL Server : 由于执行触发器, 登录名 'login1' 的登录失败。
```

19.5 管理触发器

管理触发器主要包括查询和修改触发器定义，删除触发器、启用和禁用触发器等操作。

19.5.1 查询触发器定义

在 Oracle 中要得到触发器的系统信息，可以查询 dba_triggers 数据字典视图，其 trigger_body 列保存了触发器定义的主体。

下面示例查询 emp 表上的触发器及其定义：

```
SQL> set line 300
SQL> set pagesize 20
SQL> set long 1000
SQL> column trigger_name for a18
SQL> column description for a30
SQL> select trigger_name, description, trigger_body
```

```
  2  from dba_triggers
  3  where table_name='EMP'
  4  /

TRIGGER_NAME       DESCRIPTION                 TRIGGER_BODY
------------------ --------------------------- ------------------------------------
EMP_UPDATE_AUDIT   emp_update_audit            begin
                   after update of sal on emp      insert into emp_update_audit values
                                                   (sq_emp_update.nextval,user,sysdate);
                                               end emp_update_audit;
```

在 SQL Server 中与 Oracle 的 dba_triggers 功能类似的目录视图是 sys.triggers(服务器范围是 sys.server_triggers)，但 sys.triggers 并未保存触发器定义，得到其定义需查询 sys.sql_modules 目录视图，它除了保存触发器定义外，还保存了存储过程与函数的定义。

查询 emp 表上的触发器：

```
1> select name from sys.triggers
2> where object_name(parent_id)='emp'
3> go
name
--------------------
tri_deny_emp_update
```

查询上述触发器的定义，注意启动 sqlcmd 时，设置-y 的值，否则定义不能完整显示：

```
C:\>sqlcmd -d law -y 800
1> select definition from sys.sql_modules
2> where object_id=object_id('tri_deny_emp_update')
3> go
definition
------------------------------------------------------------------------
create trigger tri_deny_emp_update
on emp
instead of update
as
set nocount on
if (datepart(hour, getdate())>=18 and datepart(hour, getdate())<=23)
    or (datepart(hour, getdate())>=0 and datepart(hour, getdate())<=6)
   print 'Update not allowed now.'
else
begin
     delete from emp where empno in(select empno from deleted);
     insert into emp select * from inserted;
end
```

19.5.2 删除触发器

Oracle 删除触发器使用 drop trigger 命令，如删除 emp_update_audit 触发器：

```
SQL> drop trigger emp_update_audit;
```

SQL Server 删除 DML 触发器与 DDL 触发器使用不同的命令。

删除 DML 触发器与 Oracle 使用相同的命令：

drop trigger *trigger_name*

如删除 emp 表上的触发器 tri_deny_emp_update：

```
1> drop trigger tri_deny_emp_update
2> go
```

删除数据库范围的 DDL 触发器使用命令：

drop trigger *trigger_name* on database

删除服务器范围的 DDL 触发器使用命令：

drop trigger *trigger_name* on all server

19.5.3 修改触发器定义

Oracle 中修改触发器定义，可以使用附带 or replace 关键字的 create trigger 命令，其余部分与创建触发器时的命令相同，当其后的触发器名称不存在时，则创建触发器，若触发器存在，则修改触发器：

create or replace *trigger_name*

trigger_body

SQL Server 中修改触发器定义，只要用 alter 关键字替换创建触发器时的 create 关键字，其他语法形式与创建触发器时相同。

19.5.4 启用和禁用触发器

Oracle 使用 alter trigger 命令执行启用和禁用指定触发器。

下面命令禁用触发器 before_emp_sal_update enable：

```
SQL> alter trigger before_emp_sal_update disable;
```

下面命令启用触发器 before_emp_sal_update enable：

```
SQL> alter trigger before_emp_sal_update enable;
```

也可以使用 alter table 命令启用或禁用表上的所有触发器。

下面命令启用 emp 表的所有触发器：

```
SQL> alter table emp enable all triggers;
```

下面命令禁用 emp 表的所有触发器：

```
SQL> alter table emp disable all triggers;
```

SQl Server 使用 enable/disable trigger 命令启用或禁用触发器。

禁用触发器使用下面命令：

disable trigger { [*schema_name* .] *trigger_name* [,...n] | all }

on { object_name | database | all server }

启用触发器使用下面命令：

enable trigger { [*schema_name* .] *trigger_name* [,...n] | all }

on { object_name | database | all server }

下面命令禁用 emp 表上的所有触发器：

```
1> disable trigger all on emp
2> go
```

下面命令禁用 emp 表上的所有触发器：

```
1> enable trigger all on emp
2> go
```

下面命令启用 emp 表上的触发器 tri_deny_emp_update：

```
1> enable trigger tri_deny_emp_update on emp
```

```
2> go
```

下面命令启用服务器范围的所有 DDL 触发器：

```
1> enable trigger all on all server
2> go
```