Table of Contents

We discussed views earlier when we used views to retrieve data from the base tables. We can also, in some circumstances, use views to update the data in the base tables. Since the view does not contain any data, the issue of doing updates or deletes via a view is more complex than with a base table.

1. DML Queries and Views

Some restrictions for updating (any change) though a view

- If the view is based on all of the columns of a single table and the view was not created with check option, then you can update through the view
- A with check option means that you cannot do an update that would violate the check option
- You cannot update a column in a view which is based on a calculated column. If the view has several columns and only some columns are calculated, you can update the other columns
- If you have any calculated columns then you cannot insert via the view. However you can delete via the view.
- You cannot update views that are defined with aggregate functions or Group By
- You cannot update views that are defined with Distinct
- If the view is based on multiple tables, you cannot do an update on a column which maps to a non key-preserved table. (Essentially this means that if the view definition does not contain the pk – as a pk for the view set, then the view cannot be updated. So if we had a parent and child situation and the view definition includes the pk of the parent we would, in general, be able to update the parent. But we could not update the child even with the child table pk since it could occur more than once in the view rowset)

Demo 01: In order to not change our basic data these are two inserts that you can use to test the updatability of the views. I am going to do the work from the a_testbed database and refer to the base tables by the two table name pattern.

An Insert for the employees table

```
Use a_testbed;
delete from a_emp.employees where  emp_id = 1995;
insert into a_emp.employees values
   (1995,'Zahn', 'Joe', '111111111', 100, 10, '2009-09-09', 500, 2);
```

An Insert for the products table

```
delete from  a_prd.products  where  prod_id = 1995;
insert into a_prd.products values
(1995, 'book','Train your cat book',29.95, 'PET', null);
```

1.1. Determining which columns in a view can be changed

Demo 02: Simple, single table view. This view can be used for updates

```
CREATE OR REPLACE VIEW vw_emp AS
    select   emp_id, name_last, name_first
    from     a_emp.employees;
```

```
select name_first, name_last
from   a_emp.employees
where emp_id = 1995;
+------------+-----------+
| name_first | name_last |
+------------+-----------+
| Joe        | Zahn      |
+------------+-----------+
```

Demo 03: This update via the view works

```
Update vw_emp
set name_first = 'Susan'
where emp_id = 1995;

select name_first, name_last
from   a_emp.employees
where emp_id = 1995;
+------------+-----------+
| name_first | name_last |
+------------+-----------+
| Susan      | Zahn      |
+------------+-----------+
```

Demo 04: This one does not work; the attribute salary is not accessible through the view because it was not in the select clause.

```
Update vw_emp
set salary = 12500
where emp_id = 1995;

ERROR 1054 (42S22): Unknown column 'salary' in 'field list'
```

Demo 05: This update does not work; the attribute name_last is accessible through the view, but that attribute is not nullable in the base table. The base table determines what data can be inserted in that table.

```
Update vw_emp
set name_last = null
 where emp_id = 5199;
ERROR 1048 (23000): Column 'name_last' cannot be null
```

Can you do an insert via this view? No- there are several attributes in the table which are not nullable and not accessible via the view.

Can you do a delete via this view? Yes, unless there is another constraint that would disallow the delete.

```
Delete from vw_emp
where emp_id = 1995;
```

Demo 06: This is a view that shows summary data

```
CREATE OR REPLACE VIEW vw_high_price_by_category AS
    select   catg_id, MAX(prod_list_price) As HighPrice
    from     a_prd.products
    group by catg_id;
```

Demo 07: If we try to change data through the view , we get an error because the attribute HighPrice does not tie back to a specific row in the underlying table

```
update  vw_high_price_by_category
set     HighPrice = 250
where   catg_id = 'PET'
;
ERROR 1288 (HY000): The target table vw_high_price_by_category of the UPDATE is
not updatable
```

Demo 08: This is a view that includes a calculated column

```
CREATE  VIEW vw_new_price AS
    select   catg_id, prod_name, Round(prod_list_price * 1.05,2) As NewPrice
    from     a_prd.products;
```

Demo 09: Trying to update the calculated column is blocked.

```
Update vw_new_price
set NewPrice = 45
where prod_name like '%Cat%';
ERROR 1348 (HY000): Column 'NewPrice' is not updatable
```

Demo 10: This is another view that includes a calculated column

```
CREATE  VIEW vw_new_price_2 AS
    select   catg_id, prod_name, prod_list_price * 1 As NewPrice
    from     a_prd.products;
```

If you think about this mathematically, prod_list_price * 1 does not really change the value, but this is still a calculated column and cannot be updated via the view.

```
Update vw_new_price_2
set NewPrice = 45
where prod_name like '%Cat%';

ERROR 1348 (HY000): Column 'NewPrice' is not updatable
```

1.2. Updatable view using a join

Create a view that joins the department table (the parent) and the employee table (the child). Can we update columns in the parent? in the child? There are several different ways to do the join. The next two views use different join techniques.

Demo 11: Creating the join with the ON clause including the dept_id column from the child table.

```
CREATE OR REPLACE VIEW vw_em_dept_1 AS
    select  E.emp_id, E.name_last
    ,       E.dept_id
    ,       D.dept_name
    from    a_emp.employees   E
    join    a_emp.departments D on  E.dept_id = D.dept_id;
```

Demo 12: Creating the join with the ON clause including the dept_id column from the parent table.

```
CREATE OR REPLACE VIEW vw_em_dept_2 AS
    select  E.emp_id, E.name_last
    ,       D.dept_id
    ,       D.dept_name
    from    a_emp.employees   E
    join    a_emp.departments D on  E.dept_id = D.dept_id;
```

If you do a Select * for any of these two views you get the same result set.

```
+--------+-----------+---------+-----------------+
| emp_id | name_last | dept_id | dept_name       |
+--------+-----------+---------+-----------------+
|    100 | King      |      10 | Administration  |
|   1995 | Zahn      |      10 | Administration  |
|    201 | Harts     |      20 | Marketing       |
|    101 | Koch      |      30 | Development     |
|    108 | Green     |      30 | Development     |
|    109 | Fiet      |      30 | Development     |
|    110 | Chen      |      30 | Development     |
|    203 | Mays      |      30 | Development     |
|    204 | King      |      30 | Development     |
|    205 | Higgs     |      30 | Development     |
|    206 | Geitz     |      30 | Development     |
|    162 | Holme     |      35 | Cloud Computing |
|    200 | Whale     |      35 | Cloud Computing |
|    207 | Russ      |      35 | Cloud Computing |
|    145 | Russ      |      80 | Sales           |
|    150 | Tuck      |      80 | Sales           |
|    155 | Hiller    |      80 | Sales           |
|    103 | Hunol     |     210 | IT Support      |
|    104 | Ernst     |     210 | IT Support      |
|    102 | D'Haa     |     215 | IT Support      |
|    146 | Partne    |     215 | IT Support      |
|    160 | Dorna     |     215 | IT Support      |
|    161 | Dewal     |     215 | IT Support      |
+--------+-----------+---------+-----------------+
```

In both of these we can update the Emp_id and name_last which come from the child table. For each Emp_id or Emp_name, that value ties back to a single definable row in the employees table. There is only one row in the employees table for each row in the view results table. We cannot update the dept_name which comes from the parent table; there are multiple rows in the result set for a row in the departments table.

The dept_id can be updated in **vw_em_dept_1** - where that attribute in the view comes uniquely from the child table. The dept_id cannot be updated in **vw_em_dept_2** where that attribute in the view comes from the parent table and is not unique.

Demo 13: An attempt to update the dept_id using the second view. This fails. Before you try this, check that row for employee 5 is in the employees table.

```
update   vw_em_dept_2
set     dept_id = 80
where   emp_id = 1995
;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key
constraint fails (`a_emp`.`employees`, CONSTRAINT `dept_emp_fk` FOREIGN KEY
(`dept_id`)REFERENCES `departments` (`dept_id`))
```

Discussion: We can do the update via the first view. In that case we are changing the dept id for child table- (the employee table). So this changes a single row. If we try to do the change via the second view, we would be changing the dept_id in the department table- so presumably we would be changing dept_id 10 to 80 in the department table- that would result in two rows in the department table with ID 80. Allowing this type of change would create problems.

2. Views that are more limiting

There are some options that can be used in the creation of a view that provide protection of the data in the base table.

2.1. Making some columns not updatable

Demo 14: The list price and the description can be updated but not the id or name. because these are calculated columns

```
CREATE OR REPLACE VIEW product_list AS
    select   prod_id + 0     as prod_id
           , prod_list_price as prod_list_price
           , prod_name + ''  as prod_name
           , prod_desc
    from     a_prd.products;
```

Demo 15: This one succeeds

```
Update product_list
set prod_list_price = 15
where prod_id = 1995;
```

Demo 16: This one fails since this is a calculated column

```
Update product_list
set prod_name = 'Cats and me'
where prod_id = 199;

ERROR 1348 (HY000): Column 'prod_name' is not updatable
```

2.2. Updates with check constraints

The next few SQL statements show the purpose of the With Check Option clause. Suppose I create a view that has a WHERE clause. I can use the view to update the data in such a way that the data no longer satisfies that WHERE clause. The With Check Option clause restricts this behavior.

Demo 17: Creating a view with a check constraint

```
CREATE OR REPLACE VIEW emp_dept_80 AS
    select   emp_id, name_last
    ,        dept_id
    from     a_emp.employees
    where    dept_id = 80
WITH CHECK OPTION;
```

Demo 18: It is OK to change the last name value through this view. Check that emp 1995 is in dept 80

```
update   emp_dept_80
set      name_last = 'Hoyer'
where    emp_id = 1995;
```

Demo 19: It is not OK to change the department to a value that does not meet the check constraint.

```
update   emp_dept_80
set      dept_id = 20
where    emp_id = 1995;

ERROR 1369 (HY000): CHECK OPTION failed 'a_testbed.emp_dept_80'
```

Similarly you cannot do an Insert via a view with a check constraint if that insert would not be displayed with the view- i.e. if the insert data does not meet the Where clause criterion.

Demo 20: Creating a view with a check constraint

```
CREATE OR REPLACE VIEW EmpHighSalary AS
    select   emp_id, name_last, salary
    ,        dept_id
    from     a_emp.employees
    where    salary > 70000
WITH CHECK OPTION;

-- the record set
select *
from EmpHighSalary;
+--------+-----------+----------+---------+
| emp_id | name_last | salary   | dept_id |
+--------+-----------+----------+---------+
|    101 | Koch      | 98005.00 |      30 |
|    146 | Partne    | 88954.00 |     215 |
|    155 | Hiller    | 80000.00 |      80 |
|    162 | Holme     | 98000.00 |      35 |
|    206 | Geitz     | 88954.00 |      30 |
+--------+-----------+----------+---------+
```

Demo 21: We can update employee 1995 salary via the employees table and then that employee will appear via the view.

```
update  a_emp.employees
set     name_last = 'Prince',
        salary = 75000
where   emp_id = 1995;

select *
from EmpHighSalary
where dept_id = 80;
+--------+-----------+----------+---------+
| emp_id | name_last | salary   | dept_id |
+--------+-----------+----------+---------+
|    155 | Hiller    | 80000.00 |      80 |
|   1995 | Prince    | 75000.00 |      80 |
+--------+-----------+----------+---------+
```

Demo 22: Updating through the view. This will do the update.

```
update  EmpHighSalary
set     name_last = 'Prince',
        salary = 85000
where   emp_id = 1995;
```

Demo 23: Updating through the view. This update will fail due to the check constraint

```
update   EmpHighSalary
set      name_last = 'Koch',
         salary = 10000
where    emp_id = 1995;
;
```