How to Update/Delete Records in child table when Parent table records are updated/deleted ?

Syntax: Foreign key(column_name) References table_name(column_name) ON UPDATE CASCADE ON DELETE CASCADE

STEP BY STEP Explanation with an example:

Step 1: Create Parent table(i.e Primary key table)

```
Create table Table1
(ID_VAL int AUTO_INCREMENT PRIMARY KEY,
NAME_VAL varchar(255),
DESCRIPTION text,
Picture varchar(255));
```

Step 2: Insert some records

```
Insert into Table1(NAME_VAL,DESCRIPTION,Picture)
values('abcd','bla bla bla','pic.jpg'),
('xyz','bla bla bla','pic1.jpg');

select * from Table1;
```

Now the table will be

Name:Table1

ID_VAL	NAME_VAL	DESCRIPTION	Picture
1	abcd	bla bla bla	pic.jpg
2	xyz	bla bla bla	pic1.jpg

Step 3: Create Child Table(i.e Foreign key table)

```
Create table Table2
(ID_val int AUTO_INCREMENT PRIMARY KEY,
NAME_val varchar(255),
Table1_ID int,
FOREIGN KEY(Table1_ID) REFERENCES Table1(ID_VAL) ON UPDATE
CASCADE ON DELETE CASCADE);
```

Step 4: Insert records in Table2 with valid foreign key values

```
Insert INTO table2
(NAME_val,Table1_ID)
Values('name1',1), ('name2',1), ('name3',2), ('name4',2);

Select * from Table2;
```

Now the Table2 will be

Name:Table2

ID_val	NAME_val	Table1_ID
1	name1	1
2	name2	1
3	name3	2
4	name4	2

Step 5: Update some record in Parent table(Table1)

```
Update table1 set ID_VAL=5
where NAME_VAL='abcd';
```

Step 6: View the tables

```
Select * from Table1;

Select * from Table2;
```

Name:Table1

ID_VAL	NAME_VAL	DESCRIPTION	Picture
5	abcd	bla bla bla	pic.jpg
2	xyz	bla bla bla	pic1.jpg

Name:Table2

ID_val	NAME_val	Table1_ID
1	name1	5
2	name2	5
3	name3	2
4	name4	2

We did not wrote any update query to the table2 but from the above two tables you will find the effected records in Table2 by the parent(Table1) table update query.

Step 7: Delete any one record in Parent table.

```
Delete from Table1 where ID_VAL='5';
```

Step 8: View the tables

```
Select * from Table1;

Select * from Table2;
```

Name:Table1

```
-----------------------------------------------------------------
ID_VAL | NAME_VAL  | DESCRIPTION   |Picture
-----------------------------------------------------------------
2          | xyz              | bla bla bla          |pic1.jpg
-----------------------------------------------------------------
```

Name:Table2

```
-----------------------------------------------
ID_val | NAME_val  | Table1_ID
-----------------------------------------------
3     | name3     | 2
4     | name4     | 2
-----------------------------------------------
```

From the above two tables in Table2, Two records are deleted which had **5** value in Table1_ID column.

you can create table like **on update cascade / on delete cascade / on update cascade on delete cascade**

ex: foreign key(col_name) references table_name(col_name) on delete cascade
ex: foreign key(col_name) references table_name(col_name) on update cascade
ex: foreign key(col_name) references table_name(col_name) on update cascade on delete cascade

```
CREATE TABLE Albums
(
AlbumID INT PRIMARY KEY,
Name VARCHAR(50)
);
CREATE TABLE Tracks
(
TrackID INT PRIMARY KEY,
 Title VARCHAR(50),
AlbumID INT REFERENCES Albums(AlbumID)
ON DELETE SET NULL
ON UPDATE CASCADE,
Duration TIME(0)
);
```

```
INSERT INTO dbo.Albums (AlbumID, Name)
VALUES (1,'Death Magnetic'), (4,'Master Of Puppets')

INSERT INTO dbo.Tracks (TrackID, Title, AlbumID, Duration)
VALUES(1, 'That Was Just Your Life' , 1, '00:07:08'),
(2, 'The End Of The Line', 1, '00:07:52'),
 (3, 'The Day That Never Comes', 1, '00:07:56'),
 (4, 'Battery', 4, '00:05:12')
```

In the above example, **AlbumID** in **Tracks** references **AlbumID** in **Albums**. There are two cascading actions specified here.

ON DELETE SET NULL = When a row is deleted from **Albums**, **AlbumID** will be set to **NULL** for all matching rows in **Tracks**.

ON UPDATE CASCADE = When **AlbumID** is updated in **Albums**, all matching rows in **Tracks** will also have the updated **AlbumID**.

Imp:

If you like the `Parent` and `Child` terms and you feel they are easy to be remembered, you may like the translation of `ON DELETE CASCADE` to `Leave No Orphans!`

Which means that when a `Parent` row is deleted (killed), no orphan row should stay alive in the `Child` table. All children of the parent row are killed (deleted), too. If any of these children has grandchildren (in another table through another foreign key) and there is `ON DELETE CASCADE` defined, these should be killed, too (and all descendants, as long as there is a cascade effect defined.)

The `FOREIGN KEY` constraint itself could also be described as `Allow No Orphans!` (in the first place). No `Child` should ever be allowed (written) in the child table if it hasn't a `Parent` (a row in the parent table).

For consistency, the `ON DELETE RESTRICT` can be translated to the (less aggresive) `You Can't Kill Parents!` Only childless rows can be killed (deleted.)

28 down vote

> For example, if I have two tables - Parent and Child - where Child records are owned by Parent records, which table needs the ON DELETE CASCADE?

ON DELETE CASCADE is an optional clause in a foreign key declaration. So it goes *with* the foreign key declaration. (Meaning, in the "child" table.)

> ...it could mean delete the Parent record when the Child record is deleted, or it could mean delete the Child record when the Parent is deleted. So which is it?

One way to interpret a foreign key declaration is, "All valid values for this column come from 'that_column' in 'that_table'." When you delete a row in the "child" table, nobody cares. It doesn't affect data integrity.

When you delete a row from the "parent" table--from "that_table"--you remove a valid value from the possible values for the "child" table. To maintain data integrity, you have to do *something* to the "child" table. Cascading deletes is one thing you can do.

There are five options for `ON DELETE`, and `ON UPDATE` that can apply to the `FOREIGN KEY`. These are called `<referential actions>`, directly from the SQL:2011 spec

- `ON DELETE CASCADE`: if a row of the referenced table is deleted, then all matching rows in the referencing table are deleted.
- `ON DELETE SET NULL`: if a row of the referenced table is deleted, then all referencing columns in all matching rows of the referencing table to be set to null.
- `ON DELETE SET DEFAULT`: if a row of the referenced table is deleted, then all referencing columns in all matching rows of the referencing table to be set to the column’s default value.
- `ON DELETE RESTRICT`: it is prohibited to delete a row of the referenced table if that row has any matching rows in the referencing table.
- `ON DELETE NO ACTION` **(the default)**: there is no referential delete action; the referential constraint only specifies a constraint check.

The foreign key establishes the dependent relationship. The `<referential action>` determines what happens when the relationship is dissolved.