On_delete Behaviors

source: stackoverflow

This is the behaviour to adopt when the *referenced* object is deleted.

It is not specific to django, this is an **SQL standard**.

There are 6 possible actions to take when such event occurs:

- `CASCADE`: When the referenced object is deleted, also delete the objects that have references to it (When you remove a blog post for instance, you might want to delete comments as well). SQL equivalent: `CASCADE`.
- `PROTECT`: Forbid the deletion of the referenced object. To delete it you will have to delete all objects that reference it manually. SQL equivalent: `RESTRICT`.
- `SET_NULL`: Set the reference to NULL (requires the field to be nullable). For instance, when you delete a User, you might want to keep the comments he posted on blog posts, but say it was posted by an anonymous (or deleted) user. SQL equivalent: `SET NULL`.
- `SET_DEFAULT`: Set the default value. SQL equivalent: `SET DEFAULT`.
- `SET(...)`: Set a given value. This one is not part of the SQL standard and is entirely handled by Django.
- `DO_NOTHING`: Probably a very bad idea since this would create integrity issues in your database (referencing an object that actually doesn't exist). SQL equivalent: `NO ACTION`.

Note: `on_delete` will become a required argument in Django 2.0. In older versions it defaults to `CASCADE`.