FOREIGN KEY POLICY OPTIONS

e.g. ON DELETE CASCADE

So called "Foreign key policies" are not too familiar to the audience students. There are many choices (DNA, DCA, DSN, DSD for Delete and UNA, UCA, USN, USD for Update). Each relationship=association must be looked at separately, from the business point of view. Old times we used to write them to the database diagrams next to the child table, i.e. next to the foreign key end of the line.

1. Example [Category] 1..1 ---------- 0..* (DNA, UNA)[Idea]
DNA = ON DELETE NO ACTION = a bit funny way to say: NO DELETE. Deletion of Category is prohibited if children (Idea(s)) exist. This is most likely the correct way to go in that case. UNA = Also changing a reference category id cannot be changed (one choice we can make).

2. Example [Order] 1..1 ---------- 0..* (DCA, UNA) [OrderRow]
DCA = ON DELETE CASCADE = If Mother row deleted, delete the child rows automatically too. In case of Order and OrderRow it's the best. From business point of view.

UNA = ON UPDATE NO ACTION = RESTRICT = Updating mother table primary key not allowed. In the case of Order, the id is just a surrogate key, normally no reason to let anybody to update it.

3. Example [Employee] 1..1 ---------- 0..* (DSD, UNA) [Customer]
DSD = ON DELETE SET DEFAULT (This is just an example where this could be used, if wanted. Some customer contact person (Employee) will leave the company. We would have a Customer without contact person. To avoid that we can set Pekka to be the default customer contact person for that foreign key column. Combined with ON DELETE SET DEFAULT, Pekka would be contact person.

Extra special example [Client] 1..1 ---------- 0..* (DSD, UNA) [Project]
DSD = ON DELETE SET DEFAULT (This is just an example where this could be used, if wanted. In case of GDPR deleting client, we **might** still want to keep info about the project and its manning. Thus, there could be a dummy default client called e.g. "GDPR deleted client". Thus, no info about the client anymore. Though client might want to also make the project name disappear from Project table... But still an example of DSD (edited)

Everybody has now an idea about how each policy depends on the business case. The remaining policies explained:
DSN = ON DELETE SET NULL = On deletion of Mother, child rows' foreign key values are set to NULL. Of course possible only if NULLs allowed and make sense. Basically an Order might have a handler (SalesPerson). If SalesPerson quits, all his orders could be, not deleted, but the handler set to NULL. Though also DSD would work here, and also better. All orders could go to store managers responsibility so there would be always somebody responsible.
UCA = ON UPDATE CASCADE = Would maybe make sense if e.g. two e-shops merge and we need to update

the clashing (or potentially clashing) product ids. Then we often add something to all keys of one shop to make it 100% different from the other shop keys. Then any related child rows would get the new id.These rarely if ever used:

USN = ON UPDATE SET NULL. You'll get a bonus if can come up with a business case for this. Kinda makes no sense. Or at least my imagination is not enough.

USD = ON UPDATE SET DEFAULT = Same empty head here. Why on earth would we want to lose the connection to the mother just because mother's key changed?

Maybe in some strange game, or in some simulation of entropy? Or maybe just never these two. (Famous last words...)

That said, in the case we need to go through the foreign keys of all the tables and think which one to choose? It's per FK, so if FK1, FK2, FK3 you potentially have 3x a pair of Delete & Update policies. (always define it to the child, even if out of them the NO ACTION affects actually the Mother)

In SQL DDL, the policy is part of the FK definition. E.g.

```
 …
name  VARCHAR(200) NOT NULL,

CONSTRAINT FK_OrderRow_Order FOREIGN KEY (orderId) REFERENCES Order (id) ON DELETE
CASCADE ON UPDATE NO ACTION,
…
```

So no commas etc. Though we often continue on the next line to better **readability**:

```
CONSTRAINT FK_Idea_Category FOREIGN KEY (categoryId)
      REFERENCES Category (id)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
```

All in all **DNA** (1.) and **DCA** (2.) are the **most common Delete policies**. And **UNA** (1.) **absolutely the number one in Update policies**.

UCA could be, maybe, used *while* merging databases (though merges are usually run with 'batch run' by night separately, on service break, possibly with **many** or all **constraints switched off**). One good argument for UNA comes (and against UCA) to my mind though! What if some security etc. logs have been made in a snapshot manner, or into some separate text files? In snapshots and text files the UCA would not work anyway! Thus, we would lose information if we allow for updating the keys.

(((Now crossed my mind that in games and simulations we sometimes reuse objects for performance reasons. Building an object is heavy operation compared to just marking it "dead" "inactive" or so. And if the old id was used in some statistics or results, the "reborn/resurrected/recycled" objects might be given a new id. So maybe this kind of behavior of changing keys (UCA, USN, USD) is, in technical applications, needed also in database side. In math simulations, for example, the whole database might be run in RAM. Even if changing e.g. orderId doesn't make sense in business use.)))