Checklist DB Design (EN)

The 5 Basic Rules

1. An **entity** is a person,thing or event. A number (for example weight) is never an entity but it is an attribute (characteristic) of an entity.
2. Every entity had excactly one **PK (primary key)**. The primary key is unique for the entity. For example social security number for a person or licence plate number for a car.
3. Entities can have the follwing **relations**: 1:1, !:N, N:1 of N:M.
   - 1:1 relations are rare. If they occur, you can most probably merge the two relations into one.
   - 1:N and N:1 are the same and will ocuur in most cases.
   - N:M relation is translated into two 1:N relations via a so called connection-entity/table.
4. A **1:N** relation is drawn via a line. The **line** has a **triangle** or rake on one side and just a line on the other side. The line is connected to the one-side and the triangle connects the many-side.

5. Every triangle (rake) 'belongs' to one FK. The FK is connected to the PK of the connecting entity and has the same data type (the name of the attribute, although the same in content, may differ).

Data types

The most common data-type are:

Datatype	Voorbeeld
int	-2 147 648 - 2 147 649
varchar(), bijv. varchar(20)	"Big Boss 12"
date	2022-04-01
datetime	2022-04-01 18:43:12
decimal(6,2)	1250,95

Relation(1:N) between entities

Een relatie tussen twee entiteiten is vrijwel altijd een 1-op-meer relatie en heeft daardoor aan één kant een 'harkje'.

A relation of two entities is almost always an 1 to many relation. The many side of the relation had the triangle or rake ('harkje' in Dutch).

> image-1643803461861.png
> ('harkje')

Examples

- One person owns more (0, 1 or more) cars ad one car belongs to exactly one person.
- One school class consists out of more (most commonly more than 1) students. One students belongs to one class.
- One home work assignment can be submitted 0,1 or more times (think about Canvas). One assignment submission belongs to one student.
- One football team has more players and one player belongs to one footlball team.
- One school has more students and one student is registered to one school.

Note that all these cases are describing the most common situation. Of course, you could think of one student who is registered at two schools, but that would be a rare case. If this still would not be a very rare case and you want to design your database so that one student can be registered at 2 or more schools, you would end up with an N:M relation which is ore complex and will be described below.

Example

Suppose you have two entities, **student** and **study coach**. In order to determine the relation, ask yourself what is applicable:

one study coach	more students
one student	more study coaches

Both are possible in theory, but in the situation at our school only the first line applies.

Thismeans that the more side will be the student, hence it will get the triangle. The relation between the two relations has the tringle on the student side and a plain line on the study coach side. The student will get a FK witch will connect to the PK of the study coach.

	one side	more side

PK	has one unique PK	has one unique PK
Lijntje	No triangle just a line	Triangle ('harkje')
FK	has no FK	one FK witch points to the PK off the one side

So the ERD will look like:

image-1643791334313.png

Both entities have gotten a unique PK. The FK and triangle are one the same side.

Summarized: the trinagle is situated at the may side and with every trinagle comes a FK.

Or in short:

> **Triangle= More= FK**

N:M (veel-op-veel)

Suppose you have two entities *product* en cleint (Dutch "*klant*").

The realtion between these entities is N:M, many to many. One client can buy more products and one product can be bought by many clients.

In order to put this kind of relation into an (relational) database, you have to create a connecting enitiy/table. The ERD will look like this:

image-1664052809552.png

(Note that the attribute data types are omitted in this ERD)

The entity product_klant (product-client) is the connecting entity. This entity connects the product entity and client (klant) entity in such a way that teh relation between client and product is a N:M relation.

De entiteit product_klant is de koppeltabel en deze verbind het product en de klant aan elkaar zodat er een N:M relatei ontstaat.

Note that the connecting table has two FK's (it also has two triangles). The combinations of these FK's are connecting a *product* to a *client.*

Suppose you have two clients, one with id (PK) 101 and one with id 102. Both clients bought product with the id 10 and 11. Th econtent of the connecting tabel will look like this:

id (PK)	klant_id (FK)	product_id (FK)
1	101	10
2	102	11
3	101	10
4	102	11

Creating an ERD step by step

All steps to create an ERD are listed below.

1. Determine the entities. These are persons, things, or events about which you want to store information in the database.
2. Determine from every entity what attributes (data items) you want to store in the database. Every attribute has an unique name in the entity (f.e. you can't have two "id" or two "names" in the same entity).
3. Determine the right data types of every attribute.
4. Create precisely one PK in every entity. When in doubt, you can always create an (unique) id.
5. Determine the relations between the entities. For an N:M relation, insert a connecting table.
6. Draw the relations with the triangle on the more side.
7. With every triangle comes a foreign key. This FK points to the PK of the entity it connects to via the relation.
8. The PK and FK connecting to each other don't necessarily need to have the same name, but they need to have the same data type.
9. Re-read the case and check if all data that needs to be stored is somehow part of your ERD/

What often goes wrong?

1. Does all entities has excaclty one PK (*Primar yKey*)?
2. Are all PK's unique and can only occur once in the data set?
   For example, surname cannot be PK, because there are people with the same surname.

Tip, you can always use (your own) id as PK.

3. Does every attribute has a datatype?
4. Phone number is not an int because the first 0 will disappear, nor can you use spaces. Since you don't need to make any calculations with phone numbers, there is no need to make it an integer.
5. For any date use *date* as datatype.
6. If you need to record the time next to the date, use *datetime*.
7. Every relation, line can have only one triangle. This triangle is situated on the more side of the relation.
8. For every triangle you need one FK. This FK connects the entity to the related entity via it's PK.
9. The PK and FK which are related have the same data type.
10. The length of an int doesn't matter. Nowadays, all int's will use the same length. You may omit the length.
11. Varchar always has a length. This length is the maximum length that can be used for this attribute. Using varchar(5) for the attribute surname is wrong because obviously it is too short. Varchar(300) for a phone number is wrong because it is too big and will waste database space.
12. String data types do not exist. Use *char()* or *varchar*.() Char has a fixed length and varchar has a maximum length.

--

Revision #5
Created 2022-09-24 18:07:29 UTC by Max
Updated 2025-11-12 19:19:57 UTC by Max