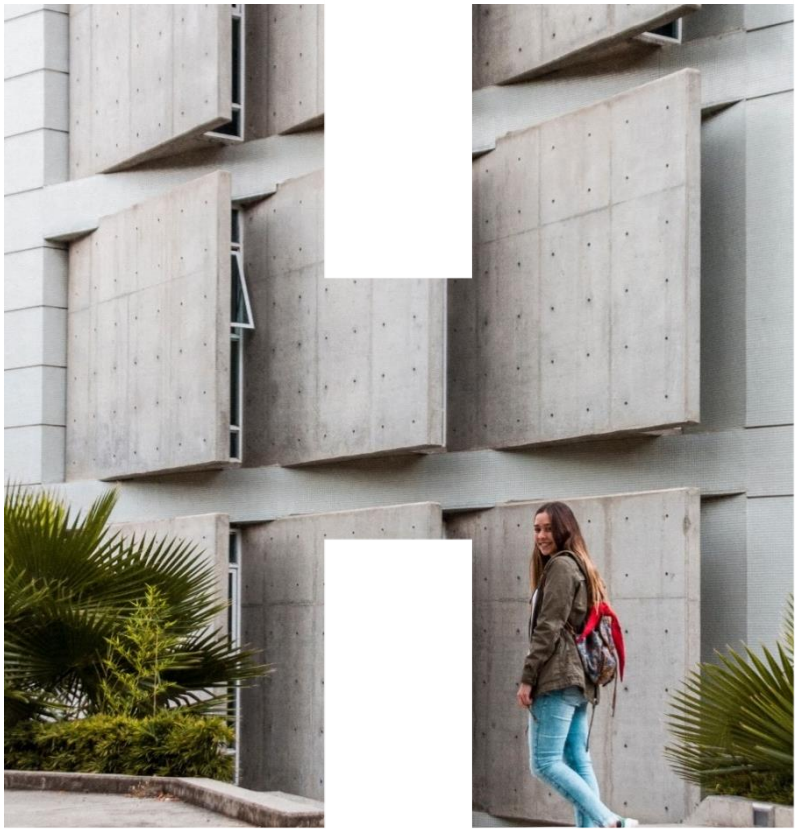


H10 SQL -- DDL.

Create

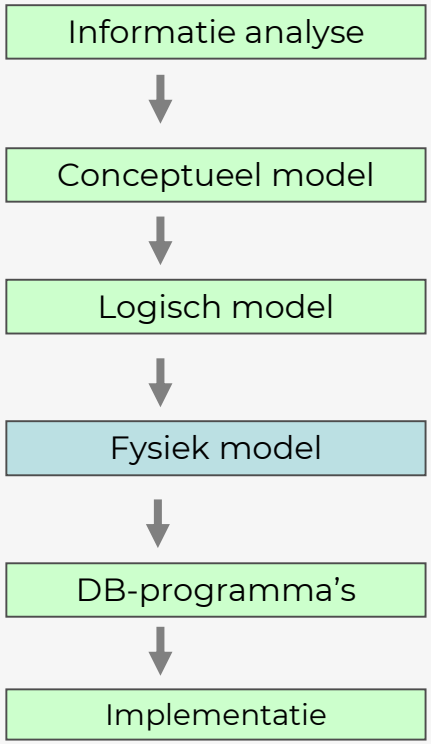
Alter

Drop

HO
GENT

Fysiek model

Stappen in de ontwikkeling van een DB:



HO
GENT

Fysiek model

- Een logisch model wordt omgezet naar een fysiek model dat nadien geïmplementeerd wordt in een DBMS.
- Het fysieke model bestaat uit volgende zaken:
  - de definitie van de tabellen
  - per tabel:
    - definitie van de primaire sleutel
    - definitie van de vreemde sleutels
    - definitie van de overige kolommen: not null-waarden, integriteitregels, …. (zie ook 2TI)
    - definitie van indexen (leerstof 2TI)
    - toewijzen aan tablespace (leerstof 2TI)

HO GENT

Fysiek model

- Voorbeeld:

  Relationeel model:

  Klant (klantID, naam, voornaam, adres, postcode, woonplaats)

  Faktuur (faktuurnummer, faktuurDatum, klant)
  klant: VS verwijst naar klantID in Klant, verplicht

  Fysiek model:

  Klant (

klantID	PRIMARY KEY, NOT NULL, auto_increment
naam	text, NOT NULL, length 20
voornaam	text, length 20
adres	text, length 30
postcode	integer
woonplaats	text, length 20

  );

HO GENT

Fysiek model

Fysiek model (vervolg):

Faktuur (

faktuurnummer	PRIMARY KEY, NOT NULL, auto_increment
faktuurdatum	date in format ‘yyyymmdd’
klant	NOT NULL, foreign key REFERENCES Klant(klantID)

);

HO
GENT

Fysiek model

- Definitie van vreemde sleutels
  - Referentiële integriteitsregel:

    Van een vreemde sleutel moet worden aangegeven of null-waarden zijn toegestaan. Als uit het conceptueel model blijkt dat de minimale cardinaliteit ten opzichte van de ‘parent’ gelijk is aan 1, zijn null-waarden niet toegestaan, en moet er een not-null-declaratie aan de vreemde sleutelkolom worden toegevoegd.
  - Dit wordt verder behandeld in Databanken II.

HO GENT

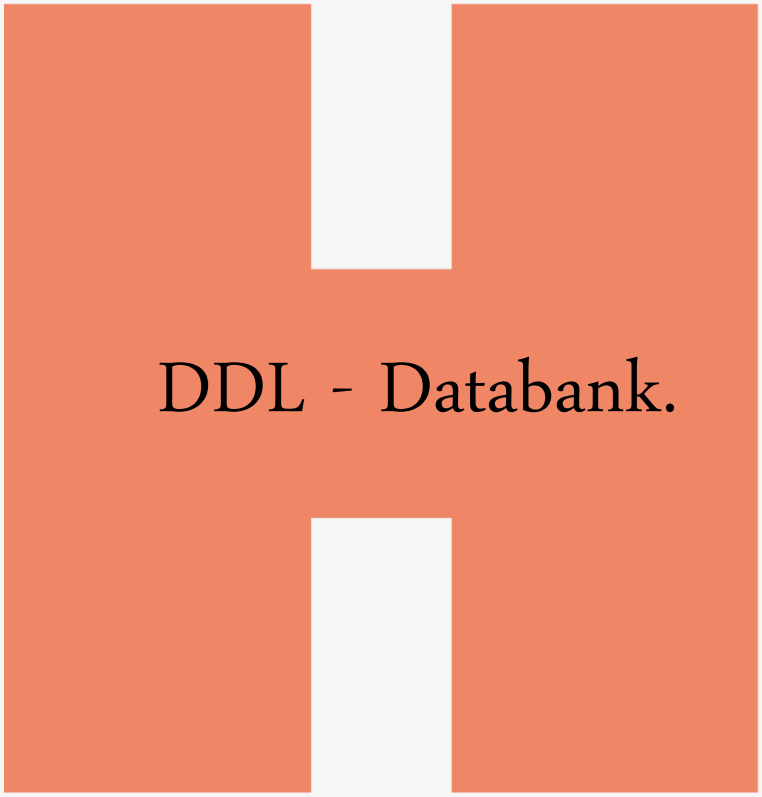
Inleiding DDL.

HO
GENT

Data Definition Language

- DDL wordt gebruikt voor
  - het definiëren van databanken
  - het definiëren van tabellen
  - het vastleggen van datatypes in MySQL
  - het definiëren van constraints - data integriteit (2TI)
  - het definiëren van indexen (2TI)

HO
GENT



HO
GENT

CREATE DATABASE

```
CREATE DATABASE database_name
```

create database simpelste vorm

```
CREATE DATABASE oefenDB
```

voorbeeld: aanmaken van de DB oefenDB

HO
GENT

DROP DATABASE

```
DROP DATABASE database_name
```

```
DROP DATABASE oefenDB
```

voorbeeld: verwijderen van de DB oefenDB

- merk op dat de systeemdatabank niet kan verwijderd worden…

HO GENT

DDL - Tabellen.

HO
GENT

Definiëren van tabellen

- tabellen creëren
- tabellen wijzigen
- tabellen verwijderen

HO
GENT

CREATE TABLE

```
CREATE TABLE table_name(
  {<column_definition> |
   <computed_column_definition> |
   <column_set_definition> }
  [<table_constraint>] [ ,...n ])
```

vereenvoudigde syntax van de create table opdracht

```
CREATE TABLE student(
  studentno int NOT NULL,
  lastname char(30) NOT NULL,
  firstname char(30) NOT NULL,
  gender char(1) NOT NULL
)
```

voorbeeld: toevoegen van een tabel 'student' aan de DB oefenDB

HO GENT

ALTER TABLE

```
ALTER TABLE table_name {
ALTER COLUMN column_name {type_name [({ precision[, scale] |
max})]}|
ADD  {<column_definition> |
        <computed_column_definition> | <table_constraint> |
        <column_set_definition> } [ ,...n ] |
DROP  {[CONSTRAINT] constraint_name  |
         COLUMN column_name } [ ,...n ]
```

vereenvoudigde syntax van de alter table opdracht

HO
GENT

ALTER TABLE

- Voorbeelden
  - Toevoegen van een kolom

```
ALTER TABLE student
ADD address char(40) NULL
```

voeg aan de tabel student de kolom address toe (tekst 40 posities variabele lengte)

  - Wijzigen van een kolom

```
ALTER TABLE student
ALTER COLUMN address char(50) NULL
```

pas de kolom address aan, vergroot het aantal posities tot 50

  - Verwijderen van een kolom

```
ALTER TABLE student
DROP COLUMN address
```

verwijder de kolom address

HO GENT

DROP TABLE

```
DROP TABLE table_name
```

vereenvoudigde syntax van de drop table opdracht

```
DROP TABLE student
```

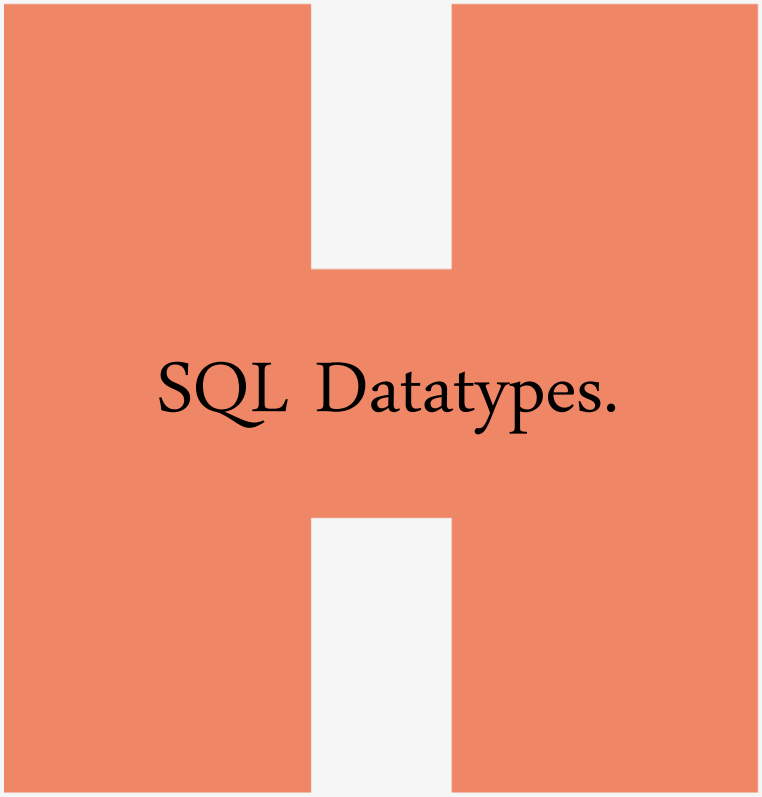
voorbeeld: verwijder de tabel student

HO GENT

Scripts

- verzameling SQL statements
- handig voor
  - batch processing
  - creatie van test- of productieomgeving
- kan op verschillende niveaus
    - databank, tabel, …
- voorbeelden: zie Chamilo script Planten, script Pubs, …

HO
GENT



SQL Datatypes.

HO
GENT

Datatypes

- overzicht van de verschillende categoriën van datatypes

Exact numerics	Unicode character strings
Approximate numerics	Binary strings
Date and time	Other data types
Character strings	

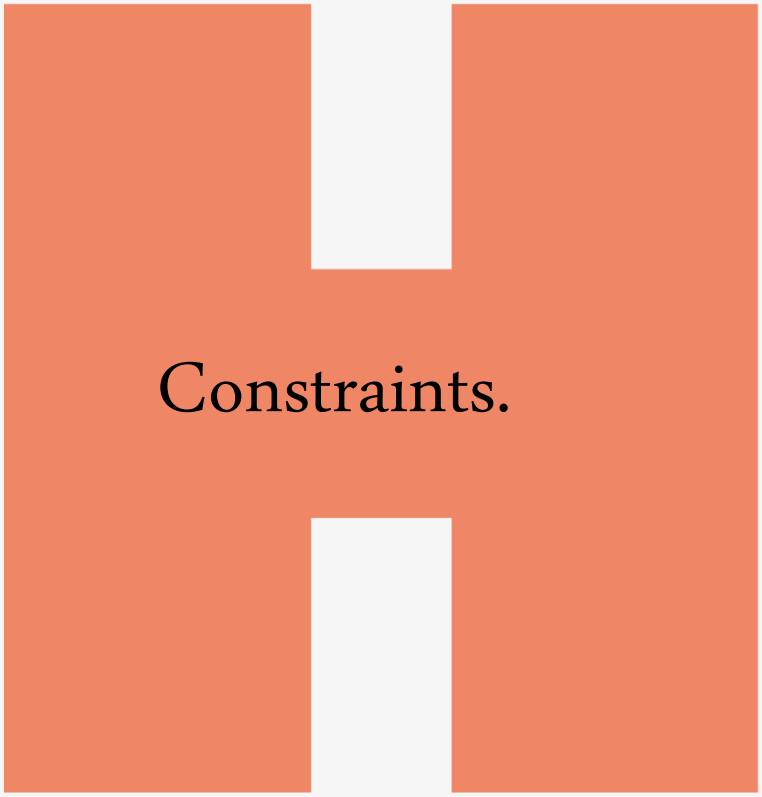
HO
GENT

Te gebruiken datatypes bij Projecten I

datatype	bereik	opslag
int(eger)	-2^31 (-2,147,483,648) tot 2^31-1 (2,147,483,647)	4 Bytes
decimal	- 10^38 +1 tot 10^38 - 1 bij maximale precisie	5 tot 7 Bytes (~precisie)
(var)char[(n)]	strings met niet meer dan n karakters	n Bytes
bool(ean)	0 of 1	1 Byte (column optimised)
date	January 1, 1753, through December 31, 9999	2 x 4 Bytes

- opmerkingen
  - bij **decimal/numeric** specificeer je precision (totaal aantal cijfers) en scale (aantal cijfers rechts van de decimale punt of komma)
    bv: decimal(5, 2) <-> 123.45
  - **boolean**: 1 is TRUE, 0 is FALSE
  - **char** bevatten non-unicode karakters; **n** kan gaan van 1 tot 8000
  - **date** geeft de datum in de vorm van yyyy-mm-dd

HO GENT



HO
GENT

AUTO_INCREMENT waarden

- een AUTO_INCREMENT kolom bevat
  - voor elke rij een unieke waarde
  - door het systeem gegenereerde (sequentiële) waarden
- slechts 1 AUTO_INCREMENT kolom per tabel mogelijk
- maakt gebruik van een integer datatype
- een AUTO_INCREMENT kolom kan geen NULL waarden bevatten
- een AUTO_INCREMENT kolom kan je niet zelf aanpassen
  - via LAST_INSERT_ID() kan je de laatst gecreëerde waarde opvragen

HO GENT

AUTO_INCREMENT waarden

```
CREATE TABLE studentVoorbeeldAutoIncrement(
  studentno int NOT NULL AUTO_INCREMENT,
  lastname char(30) NOT NULL,
  firstname char(30) NOT NULL,
  gender boolean NOT NULL
)
```

*voorbeeld: een tabel '*studentVoorbeeldAutoIncrement*' met een AUTO_INCREMENT kolom studentno aan de DB oefenDB toevoegen*

```
ALTER TABLE studentVoorbeeldAutoIncrement
 AUTO_INCREMENT = 100
```

voorbeeld: AUTO_INCREMENT start nu vanaf 100

HO GENT

Definitie van primaire sleutel

- Specificatie van de primaire sleutel
  - 1 primary key constraint per tabel
  - kan gedefinieerd worden op 1 of meerdere kolommen (samengestelde sleutel)
  - waarde (of combinatie van waarden) moet uniek zijn
  - NULL waarden zijn niet toegelaten
  - DBMS creëert een unieke index op de kolommen *(by default wordt een clustered index gecreëerd tenzij anders wordt opgegeven)*

```
studentno int primary key
```

voorbeeld: definitie van de primaire sleutel als deel van een kolom definitie

```
constraint studentno_PK primary key(studentno)
```

voorbeeld: definitie van de primaire sleutel als aparte regel in de tabel definitie (zie 2TI)

HO GENT

Definitie van primaire sleutel

```
CREATE TABLE users(
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(40),
   password VARCHAR(255),
   email VARCHAR(255)
);

CREATE TABLE userroles(
   user_id INT NOT NULL,
   role_id INT NOT NULL,
   PRIMARY KEY(user_id,role_id)
  );
```

HO
GENT

Definitie van vreemde sleutel

- Gebruikt om verbanden tussen relaties uit te drukken
  - 0, 1 of meerdere vreemde sleutels per tabel
  - NULL waarden kunnen al dan niet toegelaten zijn
  - de constraint waarborgt referentiële integriteit:
    - vreemde sleutels moeten verwijzen naar een *primaire sleutel* uit een tabel
    - de waarde van een NOT NULL vreemde sleutel moet voorkomen in de gerefereerde tabel
    - legt ook de trapsgewijze (cascading) referentiële integriteitsacties vast bij (2TI)
      - ON DELETE
      - ON UPDATE

HO GENT

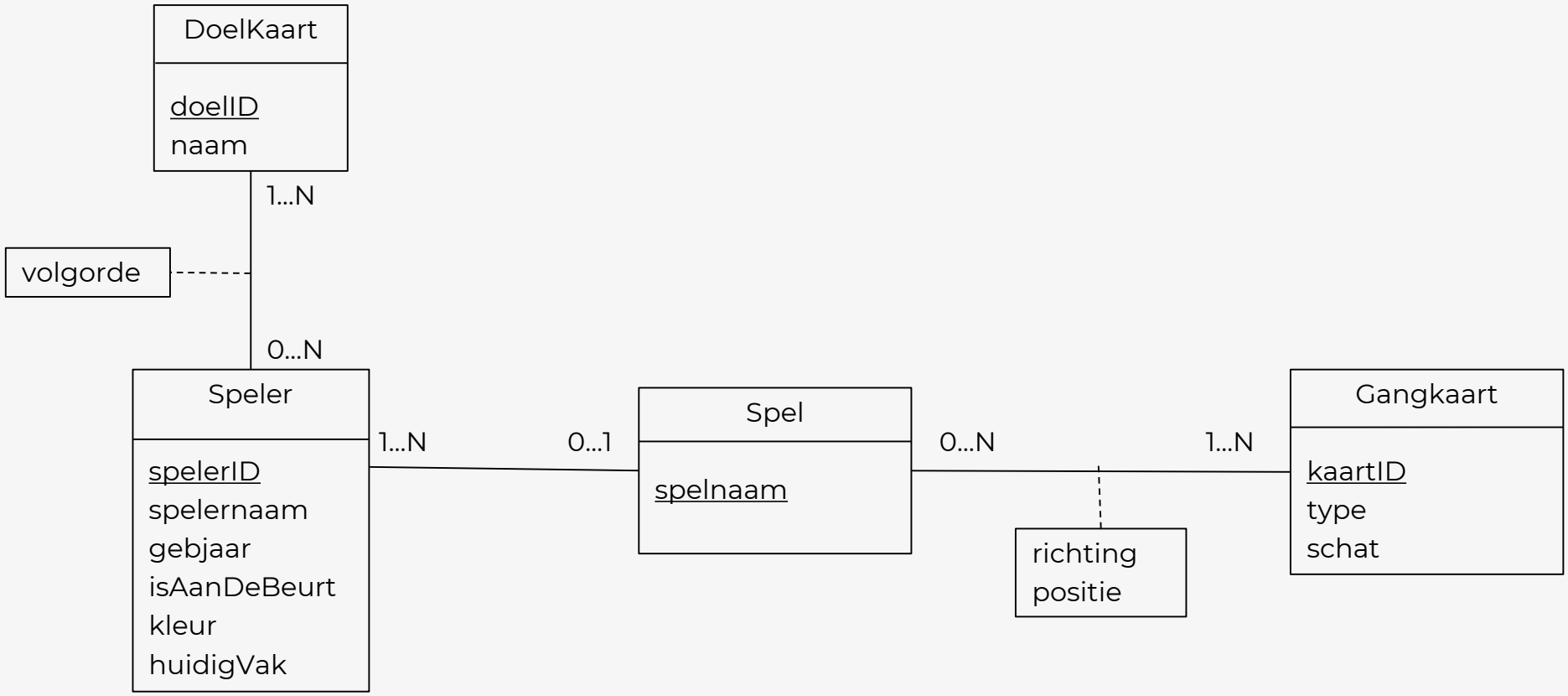
Definitie van vreemde sleutel

- Voorbeeld:

```
CREATE TABLE userroles(
    user_id INT NOT NULL,
    role_id INT NOT NULL,
    PRIMARY KEY(user_id, role_id),
    FOREIGN KEY(user_id) REFERENCES users(user_id),
    FOREIGN KEY(role_id) REFERENCES roles(role_id)
);
```

HO GENT

Voorbeeld: EERD



HO GENT

Relationeel model

Doelkaart (doelID, naam)

Speler/Doelkaart (spelerId, doelId, volgorde)

VS spelerID -> spelerID in Speler, verplicht

VS doelid -> doelID in Doelkaart, verplicht

Speler (SpelerID, spelernaam, gebJaar, kleur, huidigVak, isAanDeBeurt, spelnaam)

VS spelnaam -> spelnaam in Spel, optioneel

Spel (spelnaam)

Spel/Gangkaart (spelnaam, kaartId, richting, positie)

VS spelnaam -> spelnaam in Spel, verplicht

VS kaartId -> kaartID in Gangkaart, verplicht

Gangkaart (kaartID, type, schat)

HO GENT

DDL Databank

```
Create database mijndatabank;
Use mijndatabank;
```

HO
GENT

DDL Tabel

```
Create table doelkaart
  (
    doelID          char(5) not null,
    naam                    char(30)
);

Create table spelerdoelkaart
  (
    spelerID        int not null,
    doelID          char(5) not null,
    volgorde        int
);

Create table spel
  (
   spelnaam       char(20) not null
);
```

HO
GENT

DDL Tabel

```
Create table spelgangkaart
  (
    spelnaam        char(20) not null,
    kaartID         char(5) not null,
    richting        char(20),
    positie         char(10)
);

Create table speler
  (
    spelerID        int not null auto_increment,
    spelernaam      char(25),
    gebjaar         char(4),
    kleur           char(10) check kleur in ('rood', 'zwart'),
    huidigvak       char(20),
    isaandebeurt    char(1),
    spelnaam        char (5),
    primary key (spelerID)
);
```

HO
GENT

DDL Tabel

```
Create table gangkaart
  (
    kaartID          char(5) not null,
    type             char(8),
    schat            char(20)
);
```

HO
GENT

DDL Tabel – toevoegen primary key

```
Alter table doelkaart
        add constraint PK_doelkaart primary key (doelID);

Alter table spel
        add constraint PK_spel primary key (spelnaam);

Alter table gangkaart
        add constraint PK_gangkaart primary key (kaartID);

Alter table spelgangkaart
        add constraint PK_spelgangkaart primary key(spelnaam, kaartID);

Alter table spelerdoelkaart
        add constraint PK_spelerdoelkaart primary key (spelerID, doelID);
```

HO
GENT

DDL Tabel – toevoegen foreign key

```
Alter table spelerdoelkaart
        add constraint FK_speler foreign key (spelerID) references
                speler(spelerID);
Alter table spelerdoelkaart
        add constraint FK_doel foreign key (doelID) references
                doelkaart(doelID);
Alter table speler
        add constraint FK_spel foreign key (spelnaam) references
                spel(spelnaam);
Alter table spelgangkaart
        add constraint FK_gang foreign key (kaartID) references
                gangkaart(kaartID);
Alter table spelgangkaart
        add constraint FK_gangspel foreign key (spelnaam)
                references spel(spelnaam);
```

HO
GENT

DDL Tabel

Drop database mijndatabank;

Drop table speler;

HO
GENT

DDL extra voorbeelden

```
Alter table gangkaart
add column voordeel char(50);

Alter table gangkaart
alter column voordeel char(20);

Alter table gangkaart
drop column voordeel;
```

HO
GENT