Databanken

Tabel aanmaken

Een tabel aanmaken doe je met **CREATE TABLE:**

```
create table artikelen (
       artikel_id int not null auto_increment,
       beschr varchar(60) not null,
       voorraad smallint not null default 0,
       eenheid varchar(20),
       prijs decimal(5,2) not null,
       primary key (artikel_id)
) engine='innodb';
```

Elke veldnaam krijgt een kolomtype en een aantal kolom attributen
Een tabel verwijderen kan met **DROP:**

```
DROP table artikelen;
```

MySQL kolomtypes

Bepaald het soort gegevens dat in een bepaalde kolom kan bewaard worden.

vb.
Gehele getallen: TINYINT(m), SMALLINT(m), MEDIUMINT(m), INT(m) , BIGINT(m)
Floating point: FLOAT, DOUBLE(m, d), DECIMAL(m, d)
Character data: CHAR(n), VARCHAR(n), TEXT, BLOB
Datum en tijd: DATE, TIME, DATETIME, YEAR, TIMESTAMP
Set van waarden: ENUM, SET

Een volledig overzicht is te vinden op:
http://dev.mysql.com/doc/refman/5.6/en/datatypes.html

Gehele getallen

Gebruik het type waar het grootst verwachte getal in past.
Gebruik **UNSIGNED** waar geen negative getallen mogen voorkomen
De optionele parameter *m* in INT(m) staat voor de maximale breede. Bv. voor een postcode ->
SMALINT(4). = Bv. 9000
Voorloop nullen kunnen toegevoegd worden met **ZEROFILL** bv. artikel INT(5) ZEROFILL

Floating point

Gebruik **FLOAT** voor reële getallen waar precisie geen grote rol is. 8 cijfers na de komma
Gebruik **DOUBLE(m, d)** waar precisie wel van belang is. *m* en *d* zijn optioneel
Gebruik **DECIMAL(m, d)** waarbij allen het aaltal getallen voor en na de komma belangrijk zijn.

Character data

Gebruik **CHAR(n)** voor korte strings met vast lengte
Gebruik **VARCHAR(n)** voor korte strings zonder vaste lengte
Gebruik **TEXT** of **BLOB** voor grotere stukken text, ze zijn beiden gelijk. **TEXT** is niet case sensitive

Datum en tijd

DATE: YYYY-MM-DD
TIME: HH:MM:SS

Datum en tijd kunnen samen bewaard worden met **DATETIME:** YYYY-MM-DD HH:MM:SS
Wanneer je elk jaar wilt bijhouden kan je **YEAR** gebruiken
De huidige tijd en datum kan je opvragen met **NOW()**
bv.

```
mysql > insert into bestellingen(bestelling_id,
klant_id, datum) values(1, 2, now());
```

Tijd en datum moeten tussen ‘ ‘ ingegeven worden

Timestamp

TIMESTANMP: YYYMMDDHHMMSS
Wordt altijd bij wijzigen aangepast naar de huidige tijd en datum

Sets van waarden

Om de waarden van een kolom te beperken tot een gegeven verzameling kan men **ENUM** gebruiken.
een **ENUM** is een verzameling van strings;

```
skill ENUM(‘BASIC’, ‘INTERMEDIATE’, ‘PROFESSIONAL’)
```

De waarden worden ingegeven in HOOFDLETTERS

```
mysql> alter table employees add skill enum (‘BASIC’,
‘INTERMEDIATE’, ‘PROFESSIONAL’);
mysql> update employees set skill = 'BASIC' where name =
'Jones';
mysql> select * from employees order by skill;
```

Waarom is de keuze van datatypen belangrijk?!

- Opslagruimte
- Bepaalde bewerkingen worden onmogelijk en worden onzorgvuldig uitgevoerd
- Een goed gekozen type zal errors geven bij een foute ingaven.

MySQL kolom attributen

- **NULL of NOT NULL**: of al dan niet null waarden worden toegelaten
- **DEFAULTxxx:** De default waarde als er geen waarde wordt opgenomen
- **PRIMARY KEY:** duidt dit veld aan als primaire sluitel
- **AUTO_INCREMENT:** een sequentieel nummer zal automatisch ingevuld worden - kan enkel bij integer groepen
- **UNSIGNED:** integer waarden zullen zonder teken bewaard worden
- **UNIQUE:** De waarden van dit veld moeten onderling verschillen

```
mysql> create table klanten(klant_id int auto_increment not null
 primarykey, ... ) engine=‘innodb’;
```

- op NULL kan je enkel testen met IS NULL of IS NOT NULL, en dus niet met = of LIKE:
  - `mysql> select * from klanten where email is null; # ok`
  - `mysql> select * fromklantenwhereemail = null; # niet ok`
- by default laten velden NULL waarden toe
  - `name varchar(40) not null`

- voor een AUTO_INCREMENT veld geefje NULLop in een INSERT

```
    mysql> insert into klanten values (null, 'Smith', 'Mike', 10,
‘1976-03-18');
```

MySQL tabelypes

Er bestaan verschillende manieren waarom data in een bestand kunnen georganiseerd worden

- storage engine
  - transactional types: InnoDB, BDB (voor veel wijzigingen, Default InnoDB)
  - nontrasactionaltypes: MyISAM, HEAP, MERGE (voor zo weinig mogelijk wijzegingen)

Tabelstructuur wijzigen

Een kolom toevoegen

```
    mysql> alter table klanten add geboortedatum date;
```

De kolom definitie wijzigen

```
    mysql> alter table klanten change leeftijd leeftijd smallint;
```

Een tabel hernoemen

```
    mysql> alter table klanten rename to customers;
```

Een kolom verwijderen

```
    mysql> alter table klanten drop geboortedatum;
```

Insert, Update & Delete

Data invoegen

```
    mysql> insert into klanten(klant_id,naam, vnaam, adres, gemeente)
values(null, 'Smith', 'Mike','...', '...');
```

Shortcut

```
    mysql> insert into klanten values(null, 'Smith', 'Mike', '...',
'...');
```

! Data moet tussen “ “ staan

Data wijzigen of verwijderen

Updaten

```
    mysql> update artikelen set prijs= 4.0 where artikel_id=1;
```

Verwijderen

```
    mysql> delete from klanten where klant_id= 3;
```

! Beide zijn onomkeerbaar

Select

Data opvragen

```
    mysql> select klant_id, naam from klanten where naam= 'VAN BELLE';
    mysql> select * from artikelen where prijs between 5.0 and 12.0;
    mysql> select * from klanten where klant_id in (1, 3, 5);
```

Sorteren

```
    mysql> select * from klanten order by naam;
    mysql> select * from klanten order by naam DESC, vnaam;
    mysql> select beschr from artikelen where prijs < 10.0 order by
beschr limit 1, 2;
```

dubbels filteren

```
    mysql> select distinct naam from klanten;
```

Berekeningen uitvoeren

```
    mysql> select max(prijs) from artikelen;
    mysql> select count(*) from bestellingen;
    mysql> select sum(prijs* voorraad) from artikelen;
```

Met allias

```
    mysql> select count(*) as aantal_klanten from klanten;
```

Datum & tijd functies

Huidige datum en tijd: NOW, CURDATE, CURTIME, CURRENT_TIMESTAMP, UNIX_TIMESTAMP

Opmaak: DATE_FORMAT

```
    mysql> select date_format(curdate(), '%m/%d/%Y');
    08/28/2009
    mysql> select date_format(tijdstip, '%W %M %e %y') from items where
bestelling_id= 1
    ;Wednesday July 29 09
    mysql> select date_format(now(), '%a %D %b, %Y at %H hour');
    Mon 28th Aug, 2009 at 12 hour
```

Onderdelen: YEAR, MONTH, DAY, DAYOFxxx, WEEKDAY, HOUR, MINUTE, SECOND
Berekeningen: TO_DAYS, ADDDATE, SUBDATE

Berekeningen

Aantal dagen tussen 2 data

```
    mysql> select to_days(curdate()) - to_days(datum) from bestellingen;
```

Discrete data optellen of aftrekken

```
    mysql> select adddate(curdate(), interval1 day);
    mysql> select subdate(now(), interval‘1:1’ minute_second);
```

vb: iemand zijn leeftijd berekenen

```
    mysql> select name, year(curdate()) - year(dateofbirth) -
(right(curdate(),5) < right(dateofbirth,5)) as agefromemployees;
```

Joins

Om data uit meerdere tabellen te combineren worden joins gebruikt

```
    mysql> select naam, vnaam, bestelling_id from klanten inner join
bestellingen on klanten.klant_id=bestellingen.klant_id;
```

om tabellen te joinen worden de relaties gebruikt: in bovenstaand voorbeeld is klant_id de primary key in de klanten tabel en foreign key in de bestellingen tabel

Left join

Ongeacht de matches krijg je alle resultaten te zien

```
    mysql> select naam, vnaam, bestelling_id from klanten left join
bestellingen on klanten.klant_id= bestellingen.klant_id;
```

Right join is identiek maar de tabellen worden omgedraaid van volgorde

Natural join: shortcut voor de inner join in het geval dat de velden in beide tabellen dezelfde naam hebben:

```
    mysql> select naam, vnaam, bestelling_idfromklanten
naturaljoinbestellingen;
```

Group by

Nuttig om records uit een tabel te groepen.

```
    mysql> select datum, count(*) as aantal from bestellingen group by
datum;
```

Aggregatie functies

de meest frequent gebruikte aggregatie functies zijn:•
MIN en MAX: minimum en maximum waardevan de groep
AVG: gemiddelde waarde van de groep
SUM: totale waardevan de groep
COUNT: aantal records in de groep

bv. geef van elke bestelling het duurste artikel:

```
    mysql> select bestelling_id, max(prijs) from artikelen inner join
items on items.artikel_id= artikelen.artikel_idgroup by bestelling_id;
```

Having

Wat WHERE voor SELECT is, is HAVING voor GROUP BY. Zo kan je nog een extra voorwaarde opleggen.

```
    mysql> select artikelen.artikel_id, beschr from artikelen inner join
items on artikelen.artikel_id= items.artikel_id group by
artikelen.artikel_id having sum(aantal) > 5;
```

! Eerst word WHERE uitgevoerd op de volledige tabel dan pas HAVING op elke groep

Inner join & group by

```
    mysql> select artikelen.artikel_id, artikelen.beschr from items
inner join artikelen on items.artikel_id= artikelen.artikel_id group by
items.artikel_id having sum(items.aantal) > 5 and count(items.tijdstip)
>= 2;
```

Subqueries

= geneste queries

bv. geef alle bestellingen die op de meest recente datum werden geplaatst:

```
    mysql> select * from bestellingen where datum= (select max(datum)
from bestellingen);
```

Geef de gegevens van alle artikelen die duurder zijn dan het artikel met nummer 8:

```
    mysql> select * from artikel where prijs > (select prijs from
artikel where artikel_id= 8);
```

Geef alle bestellingen die op de meest recente datum werden geplaatst(alternatief):

```
    mysql> select * from bestellingen where datum >= all (select datum
from bestellingen);
```

Verwijder alle klanten die nog geen bestelling(en) hebben geplaatst:

```
    mysql> delete from klanten where klant_id not in (select klant_id
from bestellingen);
```

Wanneer een subquery ook als join geschreven kan worden is de join meestal te verkiezen want die is efficiënter.

```
    mysql> select bestelling_id from bestellingen inner join klanten on
bestellingen.klant_id= klanten.klant_id and naam= 'Lambert';
    mysql> select bestelling_id from bestellingen where klant_id=
(select klant_id from klanten where naam= 'Lambert');
```

Keys & Indexes

Een index op een veld kan je vergelijken met een index in een boek - het versnelt en vereenvoudigt het opzoeken.
Een sleutel (of key) is een index die tevens een constraint oplegt aan de waarden in die kolom

- Unique
- Primary
- Foreign

Unique: Je kan eisen dat een geïndexeerd veld geen dubbele waarden mag bevatten. Een unike index laat wel NULL waarden toe, tenzij anders aangegeven met het NOT NULL attribuut!

Primary key: Een unieke key waarin per definitie geen null waarden toegelaten worden. Elke tabel kan hoogstens één primary key bevatten. Meerdere velden in een tabel kunnen in aanmerking komen als primary key, dit noemen we candidate keys genoemd. Meestal id's (al dan niet met een AUTO_INCREMENT).

3-tier model: de daten de logica en de presentatie worden in 3 lagen van elkaar gescheiden

Redundancy

= het 2 x opslaan van de zelfde gegevens
Spreadsheets bevatten vaak dubbele gegevens om het geheel leesbaar te houden voor mensen.
Moeilijk te beheren en onderhouden van de data.

Normalisatie

is een proces waarmee redundancy uit een database geëlimineerd kan worden door tabellen op te splitsen.

- Een genormaliseerde database
—— Is minder onderhevig aan anomalieën
—— Is eenvoudiger aan te passen of uit te breiden
—— Is efficiënter
—— Vergt minder schijfruimte en resources
- het normalisatieproces bestaat uit een aantal stappen waarbij elke stap steunt op de voor gaande
—— Eerste normaalvorm 1NF
—— 2NF
—— 3NF
—— Hogere normaalvormen

0NF = alles in 1 entiteit

Eerste normaalvorm

Wanneer:

1. Alle gegevens in een kolom van hetzelfde logische type zijn, de waarden ‘Frank’ en ‘0467409834’ zijn wel allebei strings maar horen logisch daarom niet bij elkaar.
2. Elke cel uit de tabel een enkelvoudige waarde bevat, m a w 1 string, 1 getal, 1 datum
3. Er geen dubbele rijen in voorkomen, m a w de tabel moet een primary key hebben

Tweede normaalvorm

Wanneer:

1. ze in eerste normaalvorm staat
2. alle niet-sleutel velden afhankelijk zijn van de volledige sleutel

- splits gegevens die enkel horen bij een deel van de sleutel af in een nieuwe tabel - je kan dit gemakkelijk herkennen aan het feit dat deze gegevens herhaald voorkomen voor dat deel van de sleutel
- tabellen die je afsplits krijgen een primary key en zijn gelinkt met de originele tabel met een foreign key

Derde normaalvorm

Wanneer:

1. ze in tweede normaalvorm staat
2. er geen afhankelijkheden bestaan tussen niet-sleutel velden

- elke kolom moet bijdragen tot het beschrijven van het onderwerp waar de tabel voor staat - kijk opnieuw uit naar de herhalende gegevens

Doel van normaliseren is om herhalingen (redundancy) te elimineren