Discrete Mathematics

Version 1.0, SQLITE manual.

Date 17-03-2019



**Table of contents**

[Introduction](#_iqgu142le703)2

[System create](#_9j8x0z5isvuw) 3

[SQL INSERT queries](#_3hxikznyidat) 4

[SQL SELECT queries](#_6haf4nwjwfzp) 5

[Wiskundige relaties](#_66opahkvb0r2) 7

# **Introduction**

This document contains instructions to create our database, query our SQL statements in SQLite3 and the mathematical explanation of the queries done in SQL.

We advise the reader to have the SQLite tool open alongside the document in order to easily understand, and perform the written queries.

## **System create**

Below is given the SQL script which will create all the tables and their corresponding attributes that we will be using in this assignment. As we can see below it consists of four tables: Kingdom, City, Leader and Equator\_country. We have assigned attributes as not null and unique where we think it is necessary.

CREATE TABLE [Kingdom] (

[Kingdom\_id] INTEGER PRIMARY KEY AUTOINCREMENT,

[Country] NVARCHAR(50) NOT NULL,

[Flag] NVARCHAR(50) NOT NULL

);

CREATE TABLE [City] (

[City\_name] NVARCHAR(50) NOT NULL,

[Economy] NVARCHAR(50),

[Money] NVARCHAR(50),

[Kingdom\_id] INTEGER NOT NULL, FOREIGN KEY (Kingdom\_id) REFERENCES Kingdom(Kingdom\_id)

);

CREATE TABLE [Leader] (

[Leader\_name] NVARCHAR(50) NOT NULL,

[Authority] NVARCHAR(50),

[Delegate] NVARCHAR(50)

);

CREATE TABLE [Equator\_country] (

[Equator\_country\_name] NVARCHAR(50) NOT NULL,

[Sunny\_climate] NVARCHAR(50) NOT NULL,

[Good\_temperature] NVARCHAR(50) NOT NULL,

UNIQUE (Equator\_country\_name)

);

## **SQL INSERT queries**

Here we insert the values into the tables that we will be using for our Select queries. We also add some entries that will not be used to show that the queries do indeed take the correct values from the database and not the undesired values.

INSERT INTO Kingdom (Country, Flag) VALUES (‘Netherlands’, ‘Red\_white\_blue’), (‘France’, ‘Blue\_white\_red’), (‘Kenya’, ‘Black\_red\_green’), (‘Romania’, ‘Blue\_yellow\_red’);

INSERT INTO City (City\_name, Economy ,Money, Kingdom\_id) VALUES (‘Amsterdam’, ‘Good’ ,’Rich’, 1), (‘Paris’, ‘Excellent’, ‘Rich’, 2), (‘Nairobi’, ‘Low’, ‘Poor’, 3), (‘Bucharest’, ‘Average’, ‘Modal’, 4);

INSERT INTO Leader(Leader\_name ,Authority, Delegate) VALUES (‘Rutte’, ‘Low’, ‘Pietje\_Bel’) ,(‘Kim’, ‘High’, ‘Fouw\_Lou\_Ping’);

INSERT INTO Equator\_country(Equator\_country\_name, Sunny\_climate, Good\_temperature) VALUES (‘Netherlands’, ‘Low’, ‘Low’), (‘Central\_African\_Republic’, ‘High’, ‘High’);

## **SQL SELECT queries**

The following 10 database queries are the ones that **∃**

1. SELECT City\_name FROM City WHERE City\_name = ‘Amsterdam’;

**{t.City\_name |(t ∈ City ∧ t.City\_name = Amsterdam)}**

*The first query is a simple control query in which we want to return the city name of the city object which has the name “Amsterdam”, so this will simply just return “****Amsterdam****”.*

1. SELECT Country, Flag FROM Kingdom WHERE Country = ‘Netherlands’;

**{k.Country, k.Flag | (k ∈ Kingdom ^ k.Country = “Netherlands”) }**

*In this query we want to select the country name and the combination of flag colors of the country “Netherlands”. This should return* ***Netherlands | Red\_white\_blue***

1. SELECT City\_name, Economy, Money FROM City WHERE City\_name= ‘Paris’ AND Money = ‘Rich’;

**{ c.City\_name, c.Economy, c.Money | (c ∈ Kingdom ^ c.City\_name = ’Paris’ ^ c.Money = ‘rich’ )}**

*Here we want to return the three attributes (name, economy and money) of a city with name Paris and which money status is “Rich”. So this should return* ***Paris | Excellent | Rich***

1. SELECT Leader\_name, Authority, Delegate FROM Leader WHERE NOT Authority = ‘High’;

**{ l.Leader\_name, l.Authority, l.Delegate | (l ∈ Leader ^ l.Authority ¬= ‘High’) }**

*Here we want to return the three attributes again by using a not-statement to select the values we want of the Leader table. This should return* ***Rutte | Low | Pietje\_Bel***

1. SELECT Flag, Country FROM Kingdom WHERE NOT Country = ‘France’ AND NOT Flag = ‘Blue\_white\_red’;

**{ k.Flag, k.Country | ( k ∈ Kingdom ^ k.Country ¬= ‘France’ ^ k.Flag ¬= ‘Blue\_white\_red’)}**

*In this fifth query we use an and-statement and an not-statement to select the flag and the country name of the kingdom that corresponds to these requirements. This should return:****lag = 'Blue\_white\_red';***

***Red\_white\_blue|Netherlands***

***Black\_red\_green|Kenya***

***Blue\_yellow\_red|Romania***

1. SELECT \* FROM City WHERE Economy='Average' OR Money=’Modal’;

**{ c.City\_name, c.Economy, c.Money, c.kingdom\_id | (c ∈ City ^ c.Economy = ‘Average’ ∨ c.Money = ‘c.Money‘ )}**

*Here we want to return all the information of Cities that have an average economy and modal money. The result should be :*

***Bucharest|Average|Modal|4***

1. SELECT Leader\_Name FROM Leader WHERE Authority= ‘Low’ AND Delegate = ‘Pietje\_Bel’;

**{ l.Leader\_Name | ( l ∈ Leader ^ Auhority = ‘Low’ ^ Delegate = ‘Pietje\_Bel’ ) }**

*Here we select the name of the leader that has low authority and who’s name is Pietje\_Bell. The expected result is :*

***Rutte***

1. SELECT \* FROM Equator\_country WHERE Sunny\_Climate = ‘Low’ AND Good\_temperature =’Low’;

**{ E.Equator\_country\_name, E.Sunny\_climate, E.Good\_temperature |** (**E.Sunny\_climate = ‘Low’ AND E.Good\_temperature =’Low’)};**

*In query 8 we select all the attributes of equator countries that have a “low” sunny climate and have a “low” good temperature. The result should be:*

***Netherlands|Low|Low***

1. SELECT Kingdom.Country, City.City\_name FROM Kingdom INNER JOIN City ON Kingdom.Kingdom\_id = City.Kingdom\_id;

**{** k.Country, c.City\_name**|( k ∈ Kingdom ∧ c ∈ City ^ k.Kingdom\_id = c.Kingdom\_id) }**

*Query 9 and 10 are a bit more complicated because here we INNER JOIN two tables on the value of the kingdom id. Afterwards we want to select the country and the city that correspond to each other in query 9 and in query 10 we ask for the flag and the money value. For query 9 we expect the following result :*

***Netherlands|Amsterdam***

***France|Paris***

***Kenya|Nairobi***

***Romania|Bucharest***

1. SELECT Kingdom.Flag, City.Money FROM Kingdom INNER JOIN City ON Kingdom.Kingdom\_id = City.Kingdom\_id;

**{k.Flag, c.money |( k ∈ Kingdom ∧ k.kingdom\_id = c.kingdom\_id)}**

*And for query 10 this is the expected result:*

**Red\_white\_blue|Rich**

**Blue\_white\_red|Rich**

**Black\_red\_green|Poor**

**Blue\_yellow\_red|Modal**

## **Wiskundige relaties**

Here we have described how to mathematical relations behave

1. *Select all values X of collection A with value “Amsterdam”.*
2. *Select all pairs x and y that are elements of collection A where Y has the value “Rood\_wit\_blauw”.*
3. *Select all pairs X,Y and Z where X has value ‘Paris’ and Z has value ‘rich’.*
4. *Select all pairs X,Y and Z where Y does not have the value ‘High’.*
5. *Select all pairs of X and Y where X does not have the value ‘*Blue\_white\_red*’ and Y does not have the value ‘France’.*
6. *Select all pairs of W, X, Y, and Z where X has the value ‘Average’ or Y is ‘Modal’.*
7. *Select all X values where Y has the Low value and Z has value ‘Pietje\_Bell’.*
8. Select all values of X, Y and Z where Y is low and Z is low as well.
9. *Display the pairs of country and city names where in the product of all attributes the kingdom id of the kingdom and the kingdom id of the city are the same.*
10. *Select all pairs of X and Z where in the product of all attributes the kingdom id of the*

*kingdom and the kingdom id of the city are the same.*