Aantekeningen IRDB

Vak: IRDB -> Informatica Relationele DataBase

Onderstaande database scripts zijn gebaseerd op PostgreSQL.

Standaarden

- Schrijf SQL woorden altijd in hoofdletters, ook al werkt het met kleine letters;
    - (INSERT, UPDATE, CREATE)

- Er bestaan veel algemeen geaccepteerde afkortingen;
    - PK = PRIMARY KEY
    - FK = FORAIGN KEY

- Maak gebruik van duidelijke tabelnamen en kolomnamen. Desnoods met gebruik van een prefix;
    - Bijvoorbeeld: Kolom ‘student’ van de tabel ‘studenten’ wordt: ‘stu_student’;

Kolom typen

- (VAR)CHAR
    - Slaat alfanumerieke gegevens op tot maximaal 2000 karakters. De standaardbreedte is één karakter. Gegevenswaarden die korter zijn dan de gedefinieerde lengte worden aangevuld met spaties. Voorbeelddeclaraties: CHAR(5), CHAR(400).
        - LET OP: Bij sommige databases hoeft er geen aantal karakters opgegeven te worden. Bij PostgreSQL is dit wel verplicht!
- DATE
    - Sla een datum op in een kolom. Maakt gebruik van Y-m-d notatie;
- TEKST

    - Deze kolom kan grote stukken tekst aan en is bedoeld om tekst erin de plaatsen.

- INT
    - Geheel getal, bijvoorbeeld 8, 63, 835
- NUMERIC
    - Getal, kan ook met decimalen. Doe dan: NUMMERIC(2,1)
- BOOLEAN
    - Kan enkel bestaan uit TRUE of FALSE. Bij Java en andere talen wordt hele woordje TRUE of FALSE gebruikt, in SQL wordt '1' voor TRUE en '0' voor FALSE gebruikt.

SQL STATEMENTS

- INSERT
    - Door middel van de INSERT wordt een row / rij toegevoegd in een tabel van de database;
    - Voorbeeld: INSERT INTO student (naam, opleiding) VALUES (Bart, Informatica);
- UPDATE
    - Door middel van een UPDATE wordt een row / rij bijgewerkt uit een specifieke tabel in de database;
    - Voorbeeld: UPDATE student SET naam='Mathijs', opleiding='Informatica' WHERE naam='Bart';
- CREATE
    - Via het commando CREATE wordt er een tabel aangemaakt; (Geef hierbij altijd kolommen mee)!;
    - Voorbeeld: CREATE TABLE modulen (naam VARCHAR (50), opleiding VARCHAR (40));
- DELETE
    - Verwijder een row / rij uit de tabel. Let op, hier kunnen andere kollommen aan verbonden zijn, die dan ook automatische verwijderd zullen worden;
    - Voorbeeld: DELETE FROM studenten WHERE naam='Bart' AND opleiding='ICT'
- ALTER TABLE
    - Bewerk een huidige tabel, zonder deze opnieuw aan te maken. Handig om bijvoorbeeld een kolom te verwijderen;
    - Voorbeeld: ALTER TABLE modulen DROP COLUMN mod_cijfer;
- DROP

    - Verwijder een complete tabel uit de database;
    - Voorbeeld: DROP TABLE studenten;

Er zijn nog bepaalde statements die voorkomen, maar niet als eerste. Deze kunnen bijvoorbeeld later in de query worden gezet. De volgende statements kunnen verder nog gebruikt worden:

- AND / OR
    - Deze worden gebruikt na bijvoorbeeld een WHERE statement. Voorbeeld: Verwijder alle rijen van tabel studenten, WAAR naam='X' AND / OR opleiding='ICT';
    - Bij AND moeten beide voorwaarden goed zijn, bij OR hoeft er maar een goed te zijn.
- WHERE
    - Geef je bijna altijd mee aan een query. Doe iets met X, WHERE voorwaarden = X;
- HAVING
    - Lijkt veel op WHERE, maar kan worden gebruikt in de GROUP BY syntax;
- CONSTRAINT
    - Dit zijn bepaalde eigenschappen / voorwaarden welke aan een nieuw item worden gehangen. Bijvoorbeeld een primary key;
- GROUP BY
    - Laat de SELECT query worden gesorteerd op een bepaalde voorwaarden;
- JOIN
    - Betrek andere tabellen of kollommen in de query. Dit kan ook bij zichzelf om aan bepaalde voorwaarden te voldoen.

<u>De onderstaande query's zijn zoveel mogelijk gebaseerd op de volgende 'tabellen'.</u>

STUDENTEN					
stu_voornaam	**stu_achternaam**	**stu_nummer**	**stu_opleiding**	**stu_klas**	**stu_slb**
Bart	Mauritz	S12345678	informatica	Inf1G	Westveer
Mathijs	Bernson	S10852121	informatica	Inf1G	Westveer
Joshua	Beens	S52627312	informatica	Inf1G	Westveer

MODULEN

mod_naam	mod_code	mod_etc	mod_begeleider
Object Programmeren	IOPR1	4	Berend
Project 1	IPMEDT1	6	Patrick

CIJFER		
cij_stu_nummer	**cij_mod_code**	**cij_cijfer**
S12345678	IOPR1	8,2
S10852121	IPMEDT1	4,5
S52627312	IOPR1	9,0

Tabellen

Tabel aanmaken

```
CREATE TABLE osiris (
        gebruikersnaam VARCHAR(40),
        wachtwoord VARCHAR(40),
        laatste_login DATE,
        rechten INT
);
```

Geef na 'CREATE TABLE' de tabelnaam op. In de (HAKEN) worden alle kolom namen weergegeven, met het type erbij. Geef in PostgreSQL altijd een grootte aan VARCHAR mee!

Extra voorwaarden aan een nieuwe tabel

```
CREATE TABLE osiris_1 (
        gebruikersnaam VARCHAR(40) NOT NULL
                CONSTRAINT fk_osiris_1 REFERENCES osiris(wachtwoord),
        wachtwoord VARCHAR(40),
        laatste_login DATE NULL,
        rechten INT
);
```

De NOT NULL en de NULL geven aan dat iets verplicht WEL of NIET is ingevuld. Wanneer een kolom NOT NULL heeft, moet er verplicht iets instaan. Wanneer een kolom NULL bevat, staat er echt helemaal niks in. Zelfs geen '0' of een ander waarde / type.

De ‘CONSTRAINT fk_osiris_1 REFERENCES osiris(wachtwoord)’ is een PRIMARY KEY welke doorverwijst naar een andere tabel. Een PRIMARY KEY is een UNIEKE waarde. Een FOREIGN KEY is een waarde welke doorverwijst naar een PRIMARY KEY

Tabel bewerken

Verwijder een kolom uit een bepaalde tabel

```
ALTER TABLE osiris DROP COLUMN rechten;
```

Zet een nieuwe kolom erbij in een bepaalde tabel

```
ALTER TABLE osiris ADD COLUMN rechten VARCHAR(20);
```

Tabel verwijderen

```
DROP TABLE osiris;
```

Let op! Na het uitvoeren hiervan ben je direct de hele tabel kwijt. Wanneer hier voorwaarden aan hingen zoals bijvoorbeeld een FOREIGN KEY / PRIMARY KEY zullen andere rijen in andere tabellen ook verwijderd worden!

Database verwijderen

```
DROP DATABASE hogeschool;
```

Dit kan alleen nadat alle sessies met de database gestopt zijn. Ook hiervoor geldt dat direct alles verwijderd is uit de database.

Standaard query’s

INSERT

```
INSERT INTO studenten VALUES (‘Voornaam’, ‘Achternaam’, ‘S10293827’, ‘informatica’, ‘Inf1A’, ‘Pijnenburg’);
```

Bovenstaande query voert iets in de tabel studenten in. Hierbij wordt elke kolom naam gebruikt!

```
INSERT INTO studenten (stu_voornaam, stu_achternaam) VALUES (‘Voornaam’, ‘Achternaam’);
```

Deze query hierboven zet ook iets in de tabel, maar gebruikt alleen maar bepaalde kolommen. Deze worden voor de VALUES gedefinieerd.

UPDATE

```
UPDATE studenten SET voornaam=’Test’ WHERE achternaam=’achternaam’;
```

Update query. Zet de kolom voornaam om in een andere data. De WHERE geeft aan welke rij er bijgewerkt moet worden.

```
UPDATE studenten SET voornaam=’Test’, achternaam=’T’ WHERE achternaam=’achternaam’;
```

Hierboven worden meerdere kolommen aangepast.

DELETE

```
DELETE FROM studenten WHERE achternaam=’achternaam’;
```

Vergeet hier nooit de WHERE modulen. Anders verwijderd hij alle records uit de tabel studenten!

SELECT

SUM

```
SELECT SUM(leeftijd) from studenten WHERE achternaam=’Achternaam’;
```

Tel het aantal bij elkaar op. Gaat hier om de kolom leeftijd. Alles bij elkaar opgeteld is de SUM die eruit komt.

COUNT

```
SELECT COUNT(*) FROM studenten;
```

Telt het aantal rijen van elkaar op. De (*) staat voor alles! Dit kan ook altijd veranderd worden met een kolom naam.

SELECT

```
SELECT * FROM studenten;
```

Hier worden alle kolommen geselecteerd!

```
SELECT voornaam, achternaam FROM studenten;
```

KEYS

PRIMARY KEY

Dit is een sleutel met een unieke waarde. Deze is echt UNIEK.

Via een nieuwe tabel:

```
CREATE TABLE test (
        test_id CHAR(1) NOT NULL CONSTRAINT pk_test PRIMARY KEY
);
```

FOREIGN KEY

Dit is een sleutel waarvan er een verwijzing gaat naar een PRIMARY KEY.

Achteraf via commando ALTER TABLE:

```
ALTER TABLE gezinslid ADD CONSTRAINT fk_relatie FOREIGN KEY (gez_relatie) REFERENCES relatie(rel_id);
```

TEMP: (komt uit de lessen [aantekeningen] IRDB)

2 kollommen samenvoegen tot 1 PRIMARY KEY

```
ALTER TABLE osiris ADD CONSTRAINT pk_cijfers PRIMARY KEY (osi_fk_studentnummer, osi_fk_mod_code)
```

Foreign key toewijzen (achteraf aan kolom)

```
ALTER TABLE studenten ADD CONSTRAINT fk_modulen FOREIGN KEY (stu_modulecode) REFERENCES modulen(mod_code)
```

Foreign key aanmaken inclusief kolom

```
ALTER TABLE studenten ADD stu_modulecode CHAR(10) CONSTRAINT fk_modulen REFERENCES modulen(mod_code)
ALTER TABLE gezinslid ADD CONSTRAINT fk_relatie FOREIGN KEY (gez_relatie) REFERENCES relatie(rel_id);
```

Temp aantekeningen IRDB

Zoeken in SQL

SELECT * FROM gezinslid WHERE gez_geslacht='V' AND gez_naam LIKE '%i%';

Foreign key toewijzen (achteraf aan kolom)

ALTER TABLE studenten ADD CONSTRAINT fk_modulen FOREIGN KEY (stu_modulecode) REFERENCES modulen(mod_code)

Foreign key aanmaken inclusief kolom

ALTER TABLE studenten ADD stu_modulecode CHAR(10) CONSTRAINT fk_modulen REFERENCES modulen(mod_code)

ALTER TABLE gezinslid ADD CONSTRAINT fk_relatie FOREIGN KEY (gez_relatie) REFERENCES relatie(rel_id);

2 kollommen samenvoegen tot 1 PRIMARY KEY

ALTER TABLE osiris ADD CONSTRAINT pk_cijfers PRIMARY KEY (osi_fk_studentnummer,

osi_fk_mod_code)

SELECT SUM (mdw_salaris) FROM medewerker;

SELECT AVG (mdw_salaris) FROM medewerker;

SELECT mdw_achternaam, MIN (mdw_salaris) FROM medewerker WHERE mdw_geslacht='V' GROUP BY mdw_achternaam;

SELECT proj_locatie, proj_afd_nr, COUNT (*) AS "Count"

FROM project

GROUP BY proj_afd_nr, proj_locatie

HAVING proj_locatie = 'Groningen' AND proj_afd_nr='7';

Join statement

LEFT JOIN links moet persee bestaan

RIGHT JOIN rechts moet persee bestaan

SELECT mdw_achternaam, mdw_voornaam, gez_naam, gez_relatie FROM medewerker m LEFT JOIN gezinslid g ON m.mdw_sofi_nr = g.gez_mdw_sofi_nr;

SELF JOIN

```
SELECT a.mdw_voornaam “Medewerkernaam”, b.mdw_voornaam “Managernaam” FROM
medewerker a, medewerker b WHERE a.mdw_mgr_sofi_nr = b.mdw_sofi_nr;


SELECT mdw_voornaam, mdw_salaris FROM medewerker WHERE mdw_salaris > '25000';


SELECT mdw_voornaam, mdw_salaris
        FROM medewerker
                WHERE mdw_salaris > '25000' AND mdw_sofi_nr
                IN(SELECT DISTINCT gez_mdw_sofi_nr FROM gezinslid);



SELECT mdw_voornaam, mdw_salaris
        FROM medewerker
                WHERE mdw_salaris > '25000' AND mdw_sofi_nr
                IN(SELECT DISTINCT gez_mdw_sofi_nr FROM gezinslid);



SELECT count(g.gez_naam), m.mdw_voornaam
        FROM gezinslid g, medewerker m
                WHERE g.gez_mdw_sofi_nr = m.mdw_sofi_nr
                        GROUP BY m.mdw_voornaam;


SELECT count(a.mdw_mgr_sofi_nr), b.mdw_achternaam
        FROM medewerker a, medewerker b
                WHERE a.mdw_mgr_sofi_nr = b.mdw_sofi_nr
                        GROUP BY b.mdw_sofi_nr;
```

```
SELECT SUM(o.opd_uren), m.mdw_voornaam
    FROM opdracht o, medewerker m
        WHERE o.opd_mdw_sofi_nr = m.mdw_sofi_nr
            GROUP BY m.mdw_voornaam
                ORDER BY SUM(o.opd_uren) ASC;
```

Antwoorden opdracht

1. SELECT count(afd_naam) FROM afdeling, locatie WHERE afd_naam LIKE ‘%o%’ OR loc_plaats LIKE ‘%o%’;
2. WHERE
3. SELECT mdw_achternaam FROM medewerker WHERE mdw_mgr_sofi_nr IS NULL
4. SELECT mdw_voornaam, count(gez_naam) FROM medewerker, gezinslid WHERE mdw_sofi_nr = gez_mdw_sofi_nr GROUP BY mdw_voornaam;
5. ALTER TABLE
6. SELECT t1.mdw_voornaam Manager, count(t2.mdw_voornaam) Medewerker FROM medewerker t1, medewerker t2 WHERE t1.mdw_sofi_nr = t2.mdw_mgr_sofi_nr GROUP BY t1.mdw_voornaam;
7. SELECT proj_nr FROM project, afdeling WHERE proj_afd_nr = afd_nr AND afd_naam = ‘Hoofdvestiging’;
8. SELECT SUM(opd_proj_nr)uren, mdw_voornaam FROM opdracht, medewerker WHERE opd_mdw_sofi_nr = mdw_sofi_nr GROUP BY mdw_voornaam ODER BY uren.