Übung DI: Datenintegrität

Gruppe 8: Lukas Arnold, Patrick Bucher, Christopher James Christensen, Jonas Kaiser, Melvin Werthmüller

1. Selbststudium

Frage 1

Welche drei Arten von strukturellen Integritätsbedingungen gibt es?

1. Eindeutigkeitsbedingung: Jedes Tupel in einer Tabelle ist eindeutig mit einem Identifikationsschlüssel bestimmt.
2. Wertebereichsbedingung: Die Werte in einer Tabelle liegen innerhalb eines zuvor definierten Wertebereichs.
3. Referenzielle Integritätsbedingung: Zu jedem Fremdschlüssel gibt es in der referenzierten Tabelle ein Tupel mit dem entsprechenden Primärschlüssel.

Frage 2

Wie lautet die Definition für den Begriff “Referenzielle Integrität”?

- Referenzielle Integrität ist ein Zustand einer Datenbank, der vorliegt, wenn der Wert jedes Fremdschlüssels in der referenzierten Tabelle als Primärschlüssel vorliegt.

Frage 3

Welche acht verschiedenen Arten von deklarativen Integritätsbedingungen gibt es? Zählen Sie die acht Begriffe auf!

1. Primärschlüsseldefinition (`PRIMARY KEY`)
2. Fremdschlüsseldefinition (`FOREIGN KEY`)
3. Eindeutigkeit (`UNIQUE`)
4. Keine Nullwerte (`NOT NULL`)
5. Prüfregel (`CHECK`)
6. Ändern/Löschen mit Nullsetzen (`ON UPDATE/DELETE SET NULL`)
7. Restriktives Ändern/Löschen (`ON UPDATE/DELETE RESTRICT`)
8. Fortgesetztes Ändern/Löschen (`ON UPDATE/DELETE CASCADE`)

Frage 4

Können Sie die diese acht deklarativen Integritätsbedingungen aus Frage 3 den drei strukturellen Integritätsbedingungen aus Frage 1 zuordnen?

1. Eindeutigkeitsbedingung
    - Primärschlüsseldefinition
2. Wertebereichsbedngung
    - Eindeutigkeit (In der Einduetigkeitsbedingung geht es ausschliesslich um die *Bestimmung* der Tupel; `UNIQUE`-Werte sind zwar auch eindeutig, können jedoch ändern und somit nicht zur Bestimmung der Eindeutigkeit verwendet werden!)
    - Keine Nullwerte
    - Prüfregel
3. Referenzielle Identitätsbedingung
    - Fremdschlüsseldefinition
    - Ändern/Löschen mit Nullsetzen (`ON UPDATE/DELETE SET NULL`)
    - Restriktives Ändern/Löschen (`ON UPDATE/DELETE RESTRICT`)
    - Fortgesetztes Ändern/Löschen (`ON UPDATE/DELETE CASCADE`)

2. Referenzielle Integrität in SQL

```
alter table studenten
    add constraint pk_studenten
        primary key (matrnr);

alter table vorlesungen
    add constraint pk_vorlesungen
        primary key (vorlnr);

alter table professoren
    add constraint pk_professoren
        primary key (persnr);

alter table assistenten
    add constraint pk_assistenten
        primary key (persnr);

alter table hoeren
    add constraint pk_hoeren
        primary key (matrnr, vorlnr),
    add constraint fk_hoeren_studenten
        foreign key (matrnr)
        references studenten (matrnr)
```

```
        on update cascade
        on delete cascade,
    add constraint fk_hoeren_vorlesungen
        foreign key (vorlnr)
        references vorlesungen (vorlnr)
        on update cascade
        on delete cascade;

alter table pruefen
    add constraint pk_pruefen
        primary key (matrnr, vorlnr),
    add constraint fk_pruefen_studenten
        foreign key (matrnr)
        references studenten (matrnr)
        on update cascade
        on delete cascade,
    add constraint fk_pruefen_vorlesungen
        foreign key (vorlnr)
        references vorlesungen (vorlnr)
        on update cascade
        on delete cascade,
    add constraint fk_pruefen_professoren
        foreign key (persnr)
        references professoren (persnr)
        on update cascade
        on delete set null;

alter table voraussetzen
    add constraint pk_voraussetzen
        primary key (vorgänger, nachfolger),
    add constraint fk_voraussetzen_vorlesung_vg
        foreign key (vorgänger) references vorlesungen (vorlnr)
        on update cascade
        on delete cascade,
    add constraint fk_voraussetzen_vorlesung_nf
        foreign key (nachfolger)
        references vorlesungen (vorlnr)
        on update cascade
        on delete cascade;
```

3. Statische Integrity Constraints in SQL

```
alter table professoren
    add constraint ck_professoren_rang
    check (rang in ('C3', 'C4'));

alter table professoren
    add constraint uq_professoren_raum
    unique (raum);

alter table pruefen
    add constraint ck_pruefen_note
    check note between 1.0 and 5;

select * from studenten;

alter table studenten
    alter column name set not null;
alter table professoren
    alter column name set not null;
alter table assistenten
    alter column name set not null;
```

4. Trigger 1

Trigger:

```
create or replace function checkDegradierung()
returns trigger as $$ begin
    if old.rang is null
        or new.rang = 'C3' and old.rang <> 'C4'
        or new.rang = 'C4'
    then return new;
    else raise exception
        'degradierender Rang --> %', new.rang;
    return old;
    end if;
end; $$ language 'plpgsql';

create trigger keineDegradierung
before update on professoren
for each row execute procedure checkDegradierung();
```

Test:

```
update professoren set rang = 'C2' where name = 'Popper';
/*
ERROR:  degradierender Rang --> C2
CONTEXT:  PL/pgSQL function checkdegradierung() line 6 at RAISE
*/

update professoren set rang = 'C3' where name = 'Russel';
/*
ERROR:  degradierender Rang --> C3
CONTEXT:  PL/pgSQL function checkdegradierung() line 6 at RAISE
*/

update professoren set rang = 'C4' where name = 'Kopernikus';
/* Query returned successfully */
```

5. Trigger 2

Die Funktion:

```
create or replace function checkZugelassen()
returns trigger as $$
declare
    anz_vorgaenger int;
    anz_bestanden int;
begin
    anz_vorgaenger := (select count(vorgänger) as anz_vorgaenger
        from voraussetzen
        join pruefen on (voraussetzen.vorgänger = pruefen.vorlnr)
        where matrnr = new.matrnr and nachfolger = new.vorlnr);
    anz_bestanden := (select count(vorgänger) as anz_bestanden
        from voraussetzen
        join pruefen on (voraussetzen.vorgänger = pruefen.vorlnr)
        where matrnr = new.matrnr and nachfolger = new.vorlnr and pruefen.note <= 3);
    if anz_bestanden = anz_vorgaenger
    then return new;
    else raise exception
        'Nicht bestandene Vorgängerprüfung(en) für Student % und Vorlesung %',
new.matrnr, new.vorlnr;
    end if;
end; $$ language 'plpgsql';
```

Der Trigger:

```
create trigger pruefen_zugelassen
```

```
before insert on pruefen
for each row execute procedure checkZugelassen();
```

Abgelegte Prüfungen eines Studenten:

```
select matrnr, vorlnr, note from pruefen where matrnr = 28106;
```

```
matrnr  vorlnr  note
------  ------  ----
28106   5001    1.0
28106   5041    4.0
```

Nachfolger zu diesen geprüften Vorlesungen:

```
select vg.vorlnr as vgnr, nf.vorlnr as nfnr from vorlesungen as vg
inner join voraussetzen on (vg.vorlnr = voraussetzen.vorgänger)
inner join vorlesungen as nf on (voraussetzen.nachfolger = nf.vorlnr)
where vg.vorlnr in (5001, 5041);
```

```
vgnr    nfnr
----    ----
5001    5041
5001    5043
5001    5049
5041    5052
5041    5216
```

Der Student mit der Matrikelnummer `28106` darf beispielsweise die Prüfung zu der Vorlesungen mit der Vorlesungsnummer `5043`, nicht jedoch zu `5216`.

```
insert into pruefen (matrnr, vorlnr, persnr) values (28106, 5043, 2126);
/* Query returned successfully */

insert into pruefen (matrnr, vorlnr, persnr) values (28106, 5216, 2126);
/*
ERROR: Nicht bestandene Vorgängerprüfung(en) für Student 28106 und Vorlesung 5216
CONTEXT:  PL/pgSQL function checkzugelassen() line 18 at RAISE
*/
```

Alternative Lösung:

```
create or replace function checkPruefungen()
returns trigger as $$ begin
if (select count(*) from (
        select note from vorlesungen
        left join voraussetzen
            on vorlesungen.vorlnr = voraussetzen.nachfolger
        left join pruefen
            on voraussetzen.vorgänger = pruefen.vorlnr
```

```
        where vorlesungen.vorlnr = new.vorlnr) as vornoten
    where note > 3 or note is null) = 0
    then return new;
    else raise exception
        'Voraussetzungen nicht erfüllt';
    return old;
    end if;
end; $$ language 'plpgsql';
```

Trigger dazu:

```
create trigger checkPruefungen
    before insert on pruefen
    for each row execute procedure checkPruefungen();
```