Matrikel-Nr: 12181407

Aufgabe 1

1. Vorteile & Nachteile von DBS die mit/ohne die Mengen Operationen INTERSECT & EXCEPT:

* Mit den Mengen Operationen kann eine Tabelle als Mathematisches Objekt betrachtet. (Vorteil)
* INTERSECT verändert manchmal die Reihenfolge der Tabelle – man kann es mit ORDER BY Lösen (Nachteil)
* Mengen Operationen anders als JOIN brauchen keine vergleich Operationen. (Vorteil)
* EXCEPT Operationen anders als NOT IN gibt keine Duplikaten zurück - nur DISTINCT Ergebnisse (Nachteil wenn man die Duplikaten braucht)

Bsp. Mit EXCEPT: (Tabelle Kunde mit attribute Kund\_nr)

SELECT kund\_nr FROM Kunde

EXCEPT SELECT kund\_nr FROM Verkauf;

Simulation – Mit NOT IN :

SELECT kund\_nr FROM Kunde

WHERE kund\_nr NOT IN (SELECT kund\_nr FROM Verkauf);

Bsp. Mit INTERSECT:

(2 Einfache Tabellen Hicking und Traveling Mit SerialNummer und Name – siehe unten)

Text

Description automatically generated with low confidenceSELECT \* FROM Hiking INTERSECT SELECT \* FROM Traveling;

Simulation – Mit NATRUAL JOIN

A picture containing table

Description automatically generatedText

Description automatically generated with low confidenceSELECT \* FROM Hiking NATURAL JOIN Traveling;

b)i) in Worten:

Wir suchen ein Produkt b, dessen **Preis größer 209** ist. Falls dieser b eine **kleinere Größe** und **kleineres Gewicht** als ein beliebiges anderes Produkt (c) dann muss auch gelten, dass der **Preis kleiner** ist jedoch die **Menge größer** ist.

ii) Wir wissen von der Mengenlehre dass A=>B ist äqualent zu ¬A ∨ B

In dem Fall definieren wir:

* A = (c.Größe > b.Größe) ∧ (c.Gewicht > b. Gewicht)
* ¬A = ¬ ((c.Größe > b.Größe) ∧ (c.Gewicht > b. Gewicht)) =

= ¬(c.Größe > b.Größe) ∨ ¬ (c.Gewicht > b. Gewicht)

* B = (c.Preis > b. Preis) ∧ (c.Menge > b. Menge)

Daraus erfolgt ¬A ∨ B =

( ¬(c.Größe > b.Größe) ∨ ¬ (c.Gewicht > b. Gewicht) ) ∧ ( (c.Preis > b. Preis) ∧ (c.Menge > b. Menge))

Aufgabe 2: ER Diagramm

Diagram

Description automatically generated

Aufgabe 3:

a)

CREATE TABLE Teilnehmer(

TNr INTEGER PRIMARY KEY,

Vorname VARCHAR(100),

Nachname VARCHAR(100) NOT NULL,

Geburtsdatun CHAR(10),

Wohnort VARCHAR(100)

);

CREATE TABLE Gruppe(

GNr INTEGER PRIMARY KEY,

Gruppenname VARCHAR(100) NOT NULL,

Leiter INTEGER REFERENCES Teilnehmer(TNr);

);

CREATE TABLE Mitglied (

Teilnehmer INTEGER REFERENCES Teilnehmer (TNr),

Gruppe INTEGER REFERENCES Gruppe (GNr)

);

1. Da TNr eine PrimaryKey ist, können wir die Einträge nicht ändern.

SELECT GNr, Gruppenname

FROM Gruppe NATURAL JOIN Hackathon

WHERE Hackathon.Ort = "Hannover" AND Hackathon.Kapazität > 15

-- Zusätzlich Namen der entsprechenden Hackathon

SELECT Name

FROM Hackathon

WHERE Hackathon.Ort = "Hannover" AND Hackathon.Kapazität > 15

CREATE VIEW JAD1 AS

SELECT Datum, COUNT(DISTINCT HNr) AS anzahlHack

FROM Hackathon

GROUP BY Datum;

SELECT Datum FROM JAD1

WHERE anzahlHack = (SELECT MAX(anzahlHack) FROM JAD1);

Create VIEW Jad2 AS

SELECT Gruppe, SUM(Punkte) AS allPunkte

FROM Teilnahme

GROUP BY Gruppe;

SELECT Gruppe, IF(allPunkte = NULL, 0, allPunkte) FROM Jad2

1. Rationale Algebra:

πGNr, Gruppenname(σTeilnahme. Hackathon = 17 ∨ Teilnahme. Hackathon = 94 (Teilnahme⨝Gruppe))

Tupelkalkül:

Schema(gr) = Schema(Gruppe);

{ [gr.GNr, gr.Gruppenname] | Gruppe(gr) ∧ ( ∃ t ∈ Teilnahme) ∧ (gr.GNr = t.Gruppe) ∧

(t.Hackathon = 17 ∨ t.Hackathon = 94)

}

Aufgabe 4:

S (A, B, C, D, E, F, G, H )

• G → H • F, G → E, C • F, G → D

• B, A → C • E → A, H • F, G, A → B

1. {F, G} einzige Schlüsselkandidat!

S 2.ter NormalForm?

Die Bedingungen der zweiten Normalform sind:

* Jedes Attribut ist prim oder
* voll funktional abhängig von jedem Schlüssel Kandidaten!

Wir betrachte die Attribute in S die unter verdacht sind: (A,B,C,D,E)

* A voll funktional abhängig von E (E → A, H ) aber E von F,G (F, G → E, C ) 🡪 auch von (F,G)
* B voll funktional abhängig von F,G (F, G, A → B)
* C voll funktional abhängig von B,A (BA sind abhängig von F,G)
* D voll funktional abhängig von F,G (F, G → D)
* E voll funktional abhängig von F,G (F, G → E, C)
* F Prim
* G Prim

Da Alle Fälle entweder Prim oder voll funktional abhängig von F,G sind, erfüllt S die Eigenschaften von dem 2ter Normalform.

1. Relationsschema R(A, B, C, D, E, F, G, H, I)

• C, B 🡪 A • E 🡪 C, F. • B, F 🡪 I

• H 🡪 C. • C, F 🡪 B. • F 🡪 D, G • B, I 🡪 E

Nur auf der linken Seite?

{H}+  = {H, C, A, B, I, E, F, D, G} = {A, B, C, D, E, F, G, H; I}

H bestimmt ganz R 🡪 H ist ein Superschlüssel

Noch zu beweisen wäre, dass dieser Minimal ist.

* Sei K ⊆ R, und K ein Schlüsselkandidat für R wenn K🡪R also R ⊆ K+  und K Minimal ist, also ∀ **α** ∈ K: **α**🡪R (Keine Teilmenge von K bestimmt R)



In unserem Fall es existiert überhaupt kein Teilmenge von {H} „eine Menge mit einem einzigen Element H“!!

* Damit ist {H}minimal aber auch eindeutig (+ es gibt keinen anderen) ☺ 🡪 Es handelt sich hier um einen Schlüsselkandidat!

Aufgabe 5:

1. Das einzige Objekt , auf dem 2 verschiedenen Transaktionen gescheht ist w.

r2(w) – r1(W) – w2(W) Keine Anamolie!

S2 = (w 1( w), r 2 ( v ), r 1 ( y ), w 1 ( v ), r 1 ( v ), w 2 ( v ), w 1 ( y ), r 2 ( w ), r 2 ( y ), r 1 ( v ), w 1 ( y ), w 2 ( w))

* R2(v) – w1(v) – w2(v) – Lost Update
* R1(v)- w2(v)-r1(v). – Non Repeatable Read
* w 1 ( y ), r 2 ( y ), w 1 ( y ) Dirty Read

S3 = ( r 2 ( y ), r 2 ( x ), w 1 ( x ), r 1 ( x ), w 1 ( y ), r 2 ( w ), w 2 ( x ), r 1 ( x ), r 2 ( y ), w 2 ( y ), w 2 ( w ), w 2 ( w))

* r 2 ( y ), w 1 ( y ), r 2 ( y ) – NON Repeatable Read