GDB [HA] zum 14. 11. 2013

Tim Dobert, Kai Sonnenwald, Arne Struck

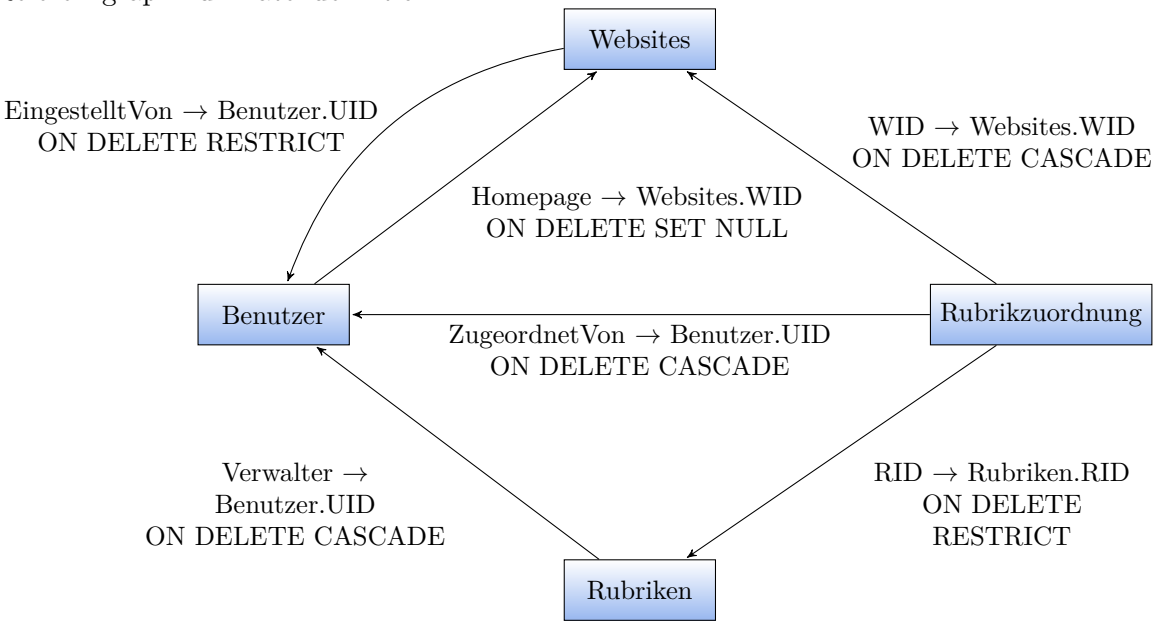
9. Januar 2014

1. a)

In einem bezüglich referentiellen Aktionen sicheren Schema sind Ergebnisse von SQL-Anfragen nicht von der Reihenfolge der Abarbeitung durch das DBMS abhängig.

b)

Referenzgraph zur Datendefinition:



c)

Wenn ein Benutzer gelöscht wird, ergeben sich 3 unterschiedliche Pfade für die Auswirkungen des Löschvorgangs, die bei der Rubrikzuordnung enden. Ebenso gibt es für den Fall, dass eine Website gelöscht wird, 3 verschiedene Pfade der Auswirkung, die ebenfalls bei der Rubrikzuordnung enden. Damit ist dort keine Sicherheit bezüglich referentiellen Aktionen gegeben.

d)

Mögliche Änderungen um das Schema sicher zu machen:

2. a)

Gegebenes Relationenmodell:

Raumschiffe(RNr, Name , Fraktion, Typ, Geschwindigkeit, Baujahr)
Besatzungsmitglieder(BNr, Name, Rang, Schiff → Raumschiffe.RNr)

i)

```
CREATE VIEW EnterpriseCrew AS
SELECT B.BNr, B.Name, B.Rang
FROM Besatzungsmitglieder B, Raumschiffe R
WHERE B.Schiff = R.RNr
AND R.Name = 'Enterprise'
WITH CASCADED CHECK OPTION;
```

CHECK OPTION, da für die Sicht mehrere Tabellen Verknüpft wurden.

ii)

```
CREATE VIEW Captains AS
SELECT Name
FROM Besatzungsmitglieder
WHERE Rang = 'Captain'
WITH CASCADED CHECK OPTION;
```

CHECK OPTION, da der Primärschlüssel nicht in der Sicht ist.

iii)

```
CREATE VIEW WarpFed AS
SELECT RNr, Fraktion, Baujahr
FROM Raumschiffe
WHERE Geschwindigkeit >= 1;
```

b)

Definierte Sichten:

```
CREATE VIEW Förderationsschiffe
  AS SELECT ? FROM Raumschiffe
WHERE Fraktion = ?Förderation?
WITH CASCADED CHECK OPTION;

CREATE VIEW Forschungsschiffe
AS SELECT ? FROM Förderationsschiffe
WHERE Typ = ?Forschungsschiff?;

CREATE VIEW GalaxyKlasse
AS SELECT ? FROM Forschungsschiffe
WHERE Geschwindigkeit = 9.8;

CREATE VIEW NebulaKlasse
AS SELECT ? FROM Forschungsschiffe
WHERE Baujahr > 2365
WITH CASCADED CHECK OPTION;
```

i))

```
UPDATE Förderationsschiff
  SET Geschwindigkeit = 9.8
  WHERE Geschwindigkeit = 9.7
AND Typ = ?Kriegsschiff?
AND Baujahr = 2350;
```

Die SQL-Anweisung ist zulässig, da die Änderungen nicht mit der Definition der Sicht 'Förderationsschiffe' kollidiert.
Die geänderten Daten sind in der Sicht 'Förderationschiffe' sichtbar.

ii)

```
INSERT INTO GalaxyKlasse
  VALUES (7, ?Sonnensegel?, ?Bajoraner?, ?Forschungsschiff?, 1, 1623);
```

Die SQL-Anweisung ist zulässig, da die eingefügten Daten zwar nicht mit der Definition der Sicht 'GalaxyKlasse' konform sind, aber keine Check-Option aktiviert ist.
Die eingefügten Daten sind in der Sicht 'GalaxyKlasse' sichtbar.

iii)

```
UPDATE Forschungsschiffe
  SET Baujahr = 2360
  WHERE Name = ?Enterprise?
AND Geschwindigkeit = 9.8;
```

Die SQL-Anweisung ist zulässig, da in der Sicht 'Forschungschiffe' keine Check-Option aktiviert ist.
Die geänderten Daten sind in der Sicht 'Forschungsschiffe' und 'GalaxyKlasse' sichtbar.

iv)

```
UPDATE NebulaKlasse
  SET Baujahr = 2360
  WHERE Geschwindigkeit = 9.6;
```

Die SQL-Anweisung ist unzulässig, da die Änderungen mit der Definition der Sicht 'NebulaKlasse' kollidieren und die Check-Option aktiviert ist.

v)

```
INSERT INTO Ga l a x yKl a s s e
  VALUES (88, ?DeltaFlyer?, ?Förderation?, ?Forschungsschiff?, 9.81, 2375);
```

Die SQL-Anweisung ist zulässig, da die eingefügten Daten zwar nicht mit der Definition der Sicht 'GalaxyKlasse' konform sind, aber keine Check-Option aktiviert ist.
Die eingefügten Daten sind in der Sicht 'GalaxyKlasse' sichtbar.

3

Gegebene Transaktionen:
T_1 liest B, liest A, schreibt (A+180+B) nach A.
T_2 liest A, schreibt A nach B, schreibt (A+110) nach A.

Gegebene Schedules:

S_1	=	$r_1(B)$	$r_1(A)$	$w_1(A)$	$r_2(A)$	$w_2(B)$	$w_2(A)$
S_2	=	$r_2(A)$	$r_1(B)$	$r_1(A)$	$w_2(B)$	$w_2(A)$	$w_1(A)$
S_3	=	$r_2(A)$	$w_2(B)$	$r_1(B)$	$w_2(A)$	$r_1(A)$	$w_1(A)$
S_4	=	$r_2(A)$	$w_2(B)$	$r_1(B)$	$r_1(A)$	$w_2(A)$	$w_1(A)$
S_5	=	$r_2(A)$	$r_1(B)$	$r_1(A)$	$w_1(A)$	$w_2(B)$	$w_2(A)$
S_6	=	$r_2(A)$	$w_2(B)$	$w_2(A)$	$r_1(B)$	$r_1(A)$	$w_1(A)$

Anfangswerte:
A = 5
B = 10

a)

Belegung von A und B nach der Ausführung von S?

S_1: A = 305, B = 195
S_2: A = 195, B = 5
S_3: A = 300, B = 5
S_4: A = 190, B = 5
S_5: A = 115, B = 5
S_6: A = 300, B = 5

b)

Abhängigkeiten zwischen den Transaktionen während des Schedules?

S_1:
S_2:
S_3:
S_4:
S_5:
S_6:

c)

Schedule seriell/serialisierbar/nicht serialisierbar?

S_1: seriell
S_2: nicht serialisierbar, da
S_3: serialisierbar als T_2T_1
S_4: nicht serialisierbar, da
S_5: nicht serialisierbar, da
S_6: seriell

4

3 Transaktionen: $T_1T_2T_3$
Schedule: $S = w_1(x)r_2(y)r_3(z)w_3(y)r_2(z)w_3(z)w_1(z)r_2(y)c_3c_1c_2$
RX-Sperrverfahren mit 2PL

LOCK-Table:

Zeitschritt	T_1	T_2	T_3	x	y	z	Bemerkung
0				NL	NL	NL	
1	lock(x,X)			X_1	NL	NL	
2	write(x)	lock(y,R)		X_1	R_2	NL	
3		read(y)	lock(z,R)	X_1	R_2	R_3	
4			read(z)	X_1	R_2	R_3	
5			lock(y,X)	X_1	R_2	R_3	T_3 wartet auf Freigabe von y
6		lock(z,R)		X_1	R_2	R_3 R_2	
7		read(z)	lock(z,X)	X_1	R_2	R_3 R_2	T_3 wartet auf Freigabe von z
8	lock(z,X)			X_1	R_2	R_3 R_2	T_1 wartet auf Freigabe von z
9		read(y)		X_1	R_2	R_3 R_2	
10		unlock(y)		X_1	X_3	R_3 R_2	Benachrichtigung von T_3
11		unlock(z)	write(y)	X_1	X_3	X_3	Benachrichtigung von T_3
12			write(z)	X_1	X_3	X_3	
13			unlock(y)	X_1	NL	X_3	
14			unlock(z)	X_1	NL	X_1	Benachrichtigung von T_1
15	write(z)		commit	X_1	NL	X_1	
16	unlock(x)			NL	NL	X_1	
17	unlock(z)			NL	NL	NL	
18	commit						
19		commit					