

### 3. Versuch: Updates, Schemaänderungen, Referentielle Integrität, Zugriffsrechte, Transaktionen

PL/SQL ist hier noch nicht erlaubt!

#### Aufgabe 3.1 (Europa-Update, 40 P.)

In der Europäischen Union hat die Integration der Staaten so große Fortschritte gemacht, dass der Staatenbund zu einem Bundesstaat wird. Zu dem neuen Staat "Europe" schließen sich (fast) alle Vollmitglieder der alten Europäischen Union zusammen. Nur Griechenland erfüllt die Beitrittsbedingungen nicht, und ist nicht dabei. Auf der anderen Seite tritt die Schweiz dem neuen Bundesstaat bei.

Der Sitz der Europäischen Union wird dabei zur Hauptstadt, die Mitgliedsstaaten zu Provinzen. Dabei werden die Freistaaten Bayern und Sachsen ebenfalls zu Provinzen aufgewertet. Bei der Regierungsform orientierte man sich zu großen Teilen an der Bundesrepublik Deutschland mit seiner föderalen Struktur.

Bei Mittelungen gewichtet man wirtschaftliche Daten dabei sinnvollerweise mit dem Bruttoinlandsprodukt, andere relative Größen nach der Bevölkerungszahl als Gewichtungsfaktor.

Bei der Mitgliedschaft in den Organisationen wurde demokratisch vorgegangen: War die Mehrheit der alten Einzelstaaten Mitglied einer Organisation (wobei die verschiedenen Mitgliedsarten nicht unterschieden werden), so wird der neue Staat Vollmitglied.

Tip: Definieren Sie eine geeignete Tabelle, um die immer wieder benötigte Abbildung von alten Provinzen auf neue Provinzen wiederverwenden zu können.

#### Aufgabe 3.2 (Definition von Referentiellen Integritätsbedingungen, 20 P.)

Das Datenbankschema MONDIAL enthält in der einfachsten Version keine referentiellen Integritätsbedingungen. Erweitern sie die Schemadefinition um die entsprechenden Integritätsbedingungen einschließlich referentieller Aktionen.

Das Skript `create.sql`, mit dem die Erstellung der Datenbasis gesteuert wird, finden Sie in `/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial:`

```
start mondial-drop-tables;  
start mondial-schema;  
start mondial-inputs;
```

Zum "Abbau" des Schemas dient das Skript `mondial-drop-tables` (entfernt alle MONDIAL-Tabellen). Das MONDIAL-Schema *ohne* Integritätsbedingungen wird von `mondial-schema` erstellt.

- Erzeugen Sie sich auf dieser Basis ein eigenes `XYcreate.sql`.
- Fügen Sie an einer geeigneten Stelle ein Skript hinzu, das entsprechende `ALTER TABLE`-Befehle enthält,  
(wenn man die referentiellen Integritätsbedingungen direkt in die `CREATE TABLE`-Anweisungen aufnahme, müsste man sie zum Einfügen der Tupel aufgrund zyklischer Referenzen zumindest teilweise deaktivieren).
- Wenn ein Land oder eine Provinz gelöscht wird, soll alles, was in diesem Land(esteil) liegt, auch gelöscht werden. Das gleiche gilt für alle Daten, die nur in Zusammenhang mit einem Land(esteil), einem Kontinent oder einer Stadt relevant sind.
- eine Stadt kann nicht gelöscht werden, wenn sie Hauptstadt von einem Land ist, das nicht gelöscht wird, oder wenn eine Organisation ihren Sitz dort hat.
- Eine Organisation kann nur gelöscht werden, wenn sie keine Mitglieder besitzt.
- Nachbarschaftsbeziehungen entfallen, wenn eines der beteiligten Gebiete gelöscht wird.
- Mit der Löschung eines Berges o.ä. wird auch seine geographische Lage überflüssig.
- Es darf keine kaskadierenden Löschungen von Informationen geben, die von der vom Benutzer geforderten Löschung unabhängig sind.

#### Aufgabe 3.3 (3.1+3.2, 5 P.)

Lassen Sie das Skript, das die referentiellen Integritätsbedingungen aktiviert (Aufgabe 3.2) *nach*

dem Update (Aufgabe 3.1) laufen. Damit können Sie verifizieren, ob Sie alle notwendigen Referenzen nachgezogen haben.

Gibt es eine Möglichkeit, das *Update* durchzuführen, während die referentiellen Integritätsbedingungen bestehen?

### Aufgabe 3.4 (Zugriffsrechte, 20 P.)

Diese Aufgabe soll von allen Teilnehmern einer Gruppe gemeinsam bearbeitet werden.

Löschen Sie zuerst Ihre MONDIAL-Datenbasis (Aufruf des Skripts `mondial-drop-tables`). In Aufgabe 3.2 haben Sie ein Skript geschrieben, das die MONDIAL-Datenbasis mit referentiellen Integritätsbedingungen erstellt. Teilen Sie dieses Skript nun so auf, dass es drei getrennte Datenbanken erstellt. In der ersten sollen alle politischen Daten gespeichert werden, in der zweiten alle rein geographischen, und in der dritten alle Tabellen, die beide Bereiche verbinden und ergänzende Daten (zu den Ländern) enthalten (Hinweis: verwenden Sie Synonyme).

- 1. Politik: Country, Province, City, Organization, borders, isMember, Politics, und Economy. Person 1 entschließt sich, anstelle der Tabelle *Country* ein View zur Verfügung zu stellen, das die Spalten Name, Code, Population, Area, Capital, Province sowie eine Spalte Density (Einwohnerdichte) enthält.
- 2. Geografie: Sea, Lake, River, Mountain, Desert, Island, Continent, sowie alle weiteren rein geografischen Daten.
- 3. Verbindungs- und Zusatzdaten: geo\_..., located, encompasses, Population, Religion, Spoken und Ethnic\_Group.

Vergeben Sie die Zugriffsrechte so, dass die Betreiber der einzelnen Bereiche jeweils nur die Daten der anderen sehen/referenzieren können, die sie für eine sinnvolle Arbeit benötigen. Teilnehmer 1 darf die politischen Daten (Country, Province) in den Tabellen von Teilnehmer 3 verändern; Teilnehmer 2 darf die geografischen Daten (Namen) in den Tabellen von Teilnehmer 3 verändern.

Teilnehmer 4 repräsentiert a) eine übergeordnete Instanz, die auf semantische Konsistenz der DB achtet, und b) den klassischen Nutzer der Datenbank. Er darf alle Daten lesen, soll Kontrollanfragen stellen, und außerdem Objekte (Städte, Länder, Berge, ...), die er gerne in der Datenbank hätte, einfügen. Da er in einem solchen Fall wohl nicht alle Attribute kennt, ist das so gedacht, dass er etwa den Namen eines Berges ("Brocken") in die Relation *Mountain* einfügt, und die Betreiber der Relationen ggf. ihre Relationen (nach geeigneter Kommunikation) ergänzen.

Hinweis: In `/afs/informatik.uni-goettingen.de/group/dbis/public/Mondial` finden Sie Skripte, die zum Füllen der einzelnen Tabellen dienen (z. B. `country.sql` – Aufruf: `start country.sql`; die Filenamen sind jeweils komplett kleingeschrieben).

Aufgaben im einzelnen:

- Teilnehmer 1–3 erstellen ihren Anteil an der verteilten Datenbank, vergeben Zugriffsrechte.
- Teilnehmer 4 definiert Synonyme und überprüft die folgenden Bedingungen:
  - für alle Berge, die in mehreren Ländern liegen, sind diese Länder benachbart;
  - auf jedem Kontinent muss es mindestens ein Land geben;
  - jedes Land muss auf mindestens einem Kontinent liegen.
- Richten Sie es so ein, dass Teilnehmer 4 alle Aufgaben der Versuche 1 und 2 unverändert laufen lassen kann.
- In Aufgabe 3.1 haben Sie politische Veränderungen in in MONDIAL durchgeführt. Welche Teilnehmer müssen welche Veränderungen an der aufgeteilten Datenbank durchführen?
- Frage: Gibt es eine einfachere Möglichkeit als diese, das Update auf der aufgeteilten Datenbank durchzuführen?
- Überlegen Sie, welche Updates Teilnehmer 1, 2 und 3 ausführen müssen, wenn Teilnehmer 4
  - einen Berg (`INSERT INTO MOUNTAIN (Name) VALUES ('Schesaplana')`) (liegt bei 47.05 n. Br., 9.7 ö. L. in den Alpen an der Grenze zwischen der Schweiz und Österreich) oder
  - eine Stadt (`INSERT INTO City (Name, Country, Province) VALUES ('Hann. Münden', 'D', 'Niedersachsen')`) (liegt bei 51.4° n. Br., 8.55° ö. L., 23500 Einw., an den Flüssen Fulda, Werra und Weser).
- Erstellen Sie auch die referentiellen Integritätsbedingungen wie in Aufgabe 3.2, und probieren Sie aus, ob `ON DELETE CASCADE` von einem Teilnehmer zum anderen funktioniert.

### Aufgabe 3.5 Transaktionen (10 P.)

Untersuchen Sie (mindestens zu zweit) das Verhalten von Transaktionen in Oracle am folgenden Beispiel:

```
-- user1:
GRANT SELECT ON isMember TO user2;
GRANT UPDATE ON isMember TO user2;
ALTER TABLE isMember DISABLE CONSTRAINT memberkey;

-- user2
CREATE SYNONYM u1_isMember FOR user1.isMember;
SELECT * FROM u1_isMember WHERE organization IN ('EU','NATO');
UPDATE u1_isMember SET organization='EU' WHERE organization = 'NATO';
UPDATE u1_isMember SET organization='NATO' WHERE organization = 'EU';
COMMIT;
SELECT * FROM u1_isMember WHERE organization IN ('EU','NATO');      -- (*1a*)

-- user1
@consult (Eingabe: isMember)
COMMIT;

-- user2
UPDATE u1_isMember SET organization='EU' WHERE organization = 'NATO';

-- (*2*)
-- user1
SELECT * FROM isMember WHERE organization IN ('EU','NATO');      -- (*1b*)
UPDATE isMember SET organization='NATO' WHERE organization = 'EU';
COMMIT;

-- user2
COMMIT;
SELECT * FROM u1_isMember WHERE organization IN ('EU','NATO');      -- (*1c*)

-- user1
ALTER TABLE isMember ENABLE CONSTRAINT memberkey;
REVOKE SELECT ON isMember FROM user2;
REVOKE UPDATE ON isMember FROM user2;
```

Aufgabe:

1. Erklären Sie das Verhalten bei (\*1a\*), (\*1b\*) und (\*1c\*).
2. Machen Sie dasselbe, wenn user2 bei (\*2\*) ein "commit" ausführt.
3. Wie verträgt sich das Verhalten mit dem, was Sie in der Datenbank-Vorlesung zu "Serialisierbarkeit" gehört haben?  
(Hinweis: Das Transaktionsmodell für interaktive Transaktionen unterscheidet sich von demjenigen für interne Transaktionen.)

**Aufgabe 3.6 (ON DELETE SET NULL, 15 P.)** Vorbild für diese Aufgabe ist der auf den Folien der Vorlesung gezeigte Diamond mit ON DELETE CASCADE und ON DELETE SET NULL.

Erzeugen Sie eine Abhängigkeitsstruktur aus referentiellen Integritätsbedingungen, mit der sie zeigen können, dass das Verhalten von Oracle in diesem Fall nicht eindeutig ist. Finden Sie heraus, von was es tatsächlich abhängt, welche Aktion "sich durchsetzt" (das kann von Version zu Version unterschiedlich sein).

Abgabe bis 10.6.2022