4. Versuch: Mehrbenutzerbetrieb

Aufgabe 4.1 (Zugriffsrechte, 50 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, und Politics
  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. Geographie: Sea, Lake, River, Mountain, Desert, Island, merges_with und Continent
- 3. Verbindungs- und Zusatzdaten: geo_..., located, encompasses, is_Member, Population, Economy, Religion, Language 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 geographischen 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`).

Aufgabe 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 3.5 haben Sie politische Veränderungen in Europa in MONDIAL integriert. Teilnehmer 1 und 3 führen diese Veränderungen an der aufgeteilten Datenbank durch.
- Frage: Gibt es eine einfachere Möglichkeit als diese, das Update auf der aufgeteilten Datenbank durchzuführen?
- Überlegen Sie, welche Updates Teilnehmer 1 und 2 ausführen müssen, wenn Teilnehmer 4
  - einen Berg (`INSERT INTO MOUNTAIN (Name) VALUES ('Pico de Aneto')`) (liegt bei 0.6 ö. L., 42.6 n. Br. in den Pyrenäen an der Grenze zwischen Frankreich und Spanien) oder

– eine Stadt (`INSERT INTO City (Name, Country, Province) VALUES ('Offenburg', 'D', 'Baden Wurttemberg')`) (8 ö. L., 48.5 n. Br., 70000 Ew) einfügt.

Aufgabe 4.2 Transaktionen (25 P.)
Untersuchen Sie (mindestens zu zweit) das Verhalten von Transaktionen in Oracle am folgenden Beispiel, das auf Aufgabe 2.8 aufbaut:

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

-- user2
CREATE SYNONYM u1_is_member FOR user1.is_member;
SELECT * FROM u1_is_member WHERE organization IN ('EU','NATO');
UPDATE u1_is_member SET organization='EU' WHERE organization = 'NATO';
UPDATE u1_is_member SET organization='NATO' WHERE organization = 'EU';
COMMIT;
SELECT * FROM u1_is_member WHERE organization IN ('EU','NATO');      -- (*1a*)

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

-- user2
UPDATE u1_is_member SET organization='EU' WHERE organization = 'NATO';

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

-- user2
COMMIT;
SELECT * FROM u1_is_member WHERE organization IN ('EU','NATO');      -- (*1c*)

-- user1
ALTER TABLE is_member ENABLE CONSTRAINT memberkey;
REVOKE SELECT ON is_member FROM user2;
REVOKE UPDATE ON is_member 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.)