5 Die Datenbank zu Projekt 3

- Optional: Änderung an alter Tabelle
- Neue Tabellen
- Vorschlag: Generierung von Ids

5.1 Klage des Sysadmin

- Mails mit nicht existierenden Adressen
- Kopie landet beim Sysadmin
- bitte Feld

  `Reply-To: ich@war.der.esel.de`

  hinzufügen, damit der Sysadmin es nicht sieht
- es gibt auch Testdaten ohne 4-letter Words!

5.2 Daten aus Projekt 2

Zugangsdaten Nickname (1-40 Zeichen, alphanumerisch); Passwort (6-40 Zeichen, alphanumerisch)

Benutzerdaten Anrede (Herr/Frau); Name (1-40 Zeichen); Vorname (1-40 Zeichen); Email-Adresse (gültiges Email-Format); Benutzerbeschreibung optional; entweder Straße (1-40 Zeichen), Hausnummer (1-999) oder Postfach (1-999999); optional Land (beschränkte Auswahl); optional PLZ; optional Ort (1-40 Zeichen); optional Homepage-URL (gültiges HTTP-URL-Format); optional Geburtsdatum (TT/MM/YYYY, früher als heute); optional Bild; optional Selbstdarstellungstdarstellungstext (1-1000 Zeichen)

5.2.1 Änderungen an Projekt 2 Tabellen

```
CREATE TABLE friends(
  userid1 integer REFERENCES users_data(userid) NOT NULL,
  userid2 integer REFERENCES users_data(userid) NOT NULL);

CREATE TABLE comments(
  userid1 integer REFERENCES users_data(userid) NOT NULL,
  userid2 integer REFERENCES users_data(userid) NOT NULL,
  datum TIMESTAMP NOT NULL,
  comment text NOT NULL);
```

Zur Vermeidung von sinnlosen Tupeln.

5.2.2 Optionale Änderungen an `users_data`

- Alter Vorschlag: `bild` Attribute als Text, in dem der Dateiname des Bilds bzw seine URL abgelegt war
- Probleme mit diesem Vorschlag:
  - Migration auf anderen Webserver
  - Migration der Datenbank

- Vermeidung: Bilddaten direkt in der Datenbank ablegen
  - Moderne Datenbanken haben keine Probleme mit dem Speichern von Binärdaten (binary large objects, BLOBs)
  - Kompletter Datenbankansatz möglich: alle Daten in der Datenbank, im Extremfall sogar die Operationen auf den Daten (*stored procedures*)
  - Auslieferung des Bildes komplizierter
- Neuer Vorschlag:

```
CREATE TABLE users_data (
        ...
        bild bytea
        )
```

- Kodebeispiele verfügbar unter
  `http://www.postgresql.org/docs/pg_handbuch/html/jdbc-binary-data.html`

5.3 Neue Daten in Projekt 3

Posting Autor (Nickname), Datum, Zeit, Nachricht (1-1000 Zeichen), optional Eltern-Posting

Thread Besitzer (Nickname), Titel (1-40 Zeichen), Liste von Schlüsselworten (je 1-40 Zeichen), Liste von Zugriffspersonen (Nickname) oder unbeschränkter Zugriff, Liste von Postings

Müssen zur Vermeidung von Redundanzen auf fünf Tabellen verteilt werden!

5.3.1 Tabellen in Projekt 3

threads spezifische Daten für einen Thread: Besitzer, Titel, Zugriffskontrolle

keywords Schlüsselworte (nicht unbedingt erforderlich)

keythreads Zuordnung Schlüsselworte $\Leftrightarrow$ Threads

threadaccess Zugriffsliste

postings Daten für Posting (wie oben)

5.3.2 Tabelle `threads`

```
CREATE TABLE threads(
   tid                integer PRIMARY KEY,
   besitzer           integer REFERENCES users_data(userid) NOT NULL,
   ; kodiert durch userid, jeder Thread muss einen Besitzer haben
   titel              VARCHAR(40) NOT NULL
   ; jeder Thread muss einen Besitzer haben
   access_control     boolean NOT NULL
   ; false = keine Zugriffskontrolle, jeder Benutzer kann zugreifen
   ; true  = nur Benutzer, die durch threadaccess freigeschaltet sind
);
```

- `tid` ist künstlich generiert
- alle Attribute müssen angegeben werden: `NOT NULL`

5.3.3 Tabelle `threadaccess`

Zweck: Zugriffskontrolle

```
CREATE TABLE threadaccess(
   userid            integer REFERENCES users_data(userid) NOT NULL,
   tid               integer REFERENCES threads(tid) NOT NULL);
```

Falls Benutzer `u` auf Thread `t` zugreifen will und bei Thread `t` die Zugriffskontrolle aktiviert ist, dann muss das Tupel `(u, t)` in `threadaccess` vorhanden sein.

ZB. teste ob

```
SELECT count(*) FROM threadaccess WHERE userid=u and tid=t
```

als Ergebnis `1` liefert.

5.3.4 Tabelle `keywords`

Zweck: Abbildung von Schlüsselworten auf Zahlen
(Optimierung, kann man auch weglassen)

```
CREATE TABLE keywords (
        wid integer NOT NULL UNIQUE,
        word VARCHAR(40) PRIMARY KEY);
```

Typische Anwendung: Finde `id` von `word`

```
SELECT wid FROM keywords WHERE word='suchwort';
```

5.3.5 Tabelle `keythreads`

Zweck: Zuordnung von Schlüsselworten zu Threads

```
CREATE TABLE keythreads(
       wid                integer REFERENCES keywords(wid),
       tid                integer REFERENCES threads(tid),
       PRIMARY KEY(wid, tid));
```

- Jede Kombination aus `wid` und `tid` soll nur einmal vorkommen.
- `wid` und `tid` sind nie NULL, da Teil des Primärschlüssels.

5.3.6 Tabelle `postings`

Zweck: Ablegen von einzelnen Postings

```
CREATE TABLE postings(
       postid             integer PRIMARY KEY,
       author             integer REFERENCES users_data(userid) NOT NULL,
       datum              TIMESTAMP NOT NULL,
       message            VARCHAR(1000) NOT NULL,
       postid_father      integer REFERENCES postings(postid),
       tid                integer REFERENCES threads(tid) NOT NULL);
```

- `postid` ist künstlicher Primärschlüssel
- Autor, Datum, Nachricht und Threadid müssen definiert sein
- Bei Postings, die neuen Thread starten, gibt es kein Vater-Posting. Daher NULL Wert erlaubt.

5.4 Generierung von Ids

- Hierfür stellt SQL sog. `SEQUENCE`s zur Verfügung
- Jeder Zugriff auf eine Sequence erzeugt eine garantiert eindeutige Zahl
- Sequences sind effizient in der Datenbank implementiert
- Erzeugen einer SEQUENCE

```
CREATE SEQUENCE postid_seq;
```

- Auslesen eines Wertes aus einer SEQUENCE

```
SELECT nextval ('postid_seq');
```

- Genauso: `tid_seq` und `wid_seq`
- Ggf auch: `userid_seq`

Viel Erfolg