5.8 Bibliotheken für PostgreSQL

- Haskell/WASH: Modul `Dbconnect`
- PHP: pqsql-Funktionen
- Java/JSP: JDBC
- Perl: DBI database interface modul

5.9 Interaktives Arbeiten mit der PostgreSQL-Shell `psql`

- Meta-Befehle
  - Beenden der Shell: `\q`
  - Liste aller SQL-Befehle: `\h`
  - Syntax eines SQL-Befehls: `\h` *command*
  - Auflisten aller Tabellen: `\d`
  - Anzeigen der Spalten einer Tabelle: `\d` *relation*
  - Ausführen abgespeicherter Befehle: `\i` *filename*
- Ausführen eines SQL-Befehls:

```
SELECT ... FROM ... WHERE;
```

5.10 SQL – Eine Mini-Einführing

5.10.1 Erzeugen und Löschen von Tabellen

- Eingebaute Datentypen

  `int, smallint, real, char(N), varchar(N),`

  `date, time, timestamp, ...`

- `CREATE TABLE`

```
CREATE TABLE weather (
    city            varchar(80),
    temp_lo         int,           -- low temperature
    temp_hi         int,           -- high temperature
    prcp            real,          -- precipitation
    date            date
);
```

- CREATE TABLE

```
CREATE TABLE cities (
    name            varchar(80),
    location        point
);
```

- DROP TABLE

```
DROP TABLE weather;
```

5.10.2 Einfügen von Zeilen in eine Tabelle

- `INSERT INTO ... VALUES ...`

```
INSERT INTO weather VALUES ('San Francisco', 46, 50, 0.25,'1994-11

INSERT INTO weather (date, city, temp_hi, temp_lo)
    VALUES ('1994-11-29', 'Hayward', 54, 37);
```

5.10.3 Abfragen einer Tabelle

- SELECT ... FROM ... WHERE

```
SELECT * FROM weather;

SELECT city, (temp_hi+temp_lo)/2 AS temp_avg, date FROM weather;

      city       | temp_avg |    date
  ---------------+----------+------------
   San Francisco |       48 | 1994-11-27
   San Francisco |       50 | 1994-11-29
   Hayward       |       45 | 1994-11-29

SELECT * FROM weather
    WHERE city = 'San Francisco'
    AND prcp > 0.0;
```

5.10.4 Abfragen einer Tabelle

- ohne Duplikate, sortiert:

  ```
  SELECT DISTINCT ... FROM ... ORDER BY ...
  ```

- Beispiel

  ```
  SELECT DISTINCT city
      FROM weather
      ORDER BY city;
  ```

5.10.5 Abfragen mehrerer Tabellen — Joins

- Beispiel: Auflisten aller Wetterdaten inkl. Koodinaten

```
SELECT *
    FROM weather, cities
    WHERE city = name;

     city      | temp_lo | temp_hi | prcp |    date    |     name
---------------+---------+---------+------+------------+----------
 San Francisco |      46 |      50 | 0.25 | 1994-11-27 | San Franc
 San Francisco |      43 |      57 |    0 | 1994-11-29 | San Franc
(2 rows)
```

- besser:

```
SELECT city, temp_lo, temp_hi, prcp, date, location
    FROM weather, cities
    WHERE city = name;
```

- Self-Joins:

```
SELECT W1.city, W1.temp_lo AS low, W1.temp_hi AS high,
    W2.city, W2.temp_lo AS low, W2.temp_hi AS high
    FROM weather W1, weather W2
    WHERE W1.temp_lo < W2.temp_lo
    AND W1.temp_hi > W2.temp_hi;
```

5.10.6 Aggregationsfunktionen

- Berechnen eines Wertes aus mehrerern Zeilen
- Anzahl: `count`
- Summe: `sum`
- Maximum: `max`
- ...

```
SELECT city FROM weather
    WHERE temp_lo = (SELECT max(temp_lo) FROM weather);
```

5.10.7 Aktualisieren einer Tabelle

- `UPDATE ... SET ... WHERE ...`
- Beispiel

```
UPDATE weather
    SET temp_hi = temp_hi - 2,  temp_lo = temp_lo - 2
    WHERE date > '1994-11-28';
```

5.10.8 Schlüssel

- Primärschlüssel vs. Fremdschlüssel
- Beispieltabellen

```
CREATE TABLE cities (
 city     varchar(80) primary key,
 location point
);

CREATE TABLE weather (
city      varchar(80) references cities,
temp_lo   int,
temp_hi   int,
prcp      real,
date      date
);
```

- Beispielanfrage

```
INSERT INTO weather VALUES ('Berkeley', 45, 53, 0.0, '1994-11-28')

ERROR:  insert or update on table "weather" violates foreign key c
DETAIL:  Key (city)=(Berkeley) is not present in table "cities".
```

5.10.9 Transaktionen

- Ausführen mehrer Kommandos als atomare Einheit
- “all or nothing”
- `BEGIN; ... COMMIT;`
- Beispiel

```
BEGIN;
  UPDATE accounts SET balance = balance - 100.00
      WHERE name = 'Alice';
  UPDATE branches SET balance = balance - 100.00
      WHERE name = (SELECT branch_name FROM accounts WHERE name =
  UPDATE accounts SET balance = balance + 100.00
      WHERE name = 'Bob';
COMMIT;
```

- Abbruch der Transaktion: `ROLLBACK;` anstelle von `COMMIT;`
- Achtung: manche Bibliotheken machen implizit immer Transaktionen!