SQLite - eine schlanke Datenbank

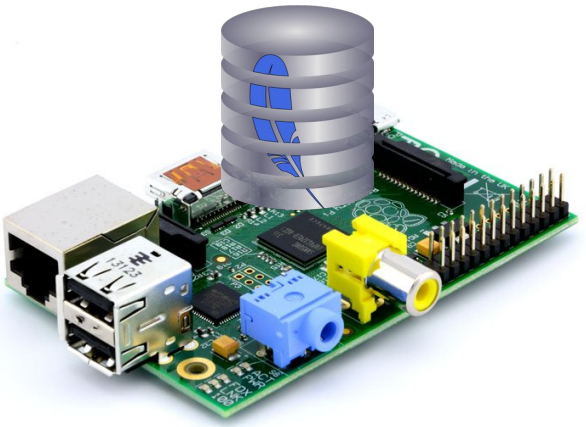
Uwe Berger
bergeruw@gmx.net

Foto: olga meier-sander / pixelio.de

Uwe Berger

- Beruf: Softwareentwickler
- Freizeit: mehr oder weniger sinnvolle Dinge mit Hard- & Software veranstalten
- Linux seit ca. 1995
- BraLUG e.V.
- bergeruw@gmx.net

oder...



Inhalt

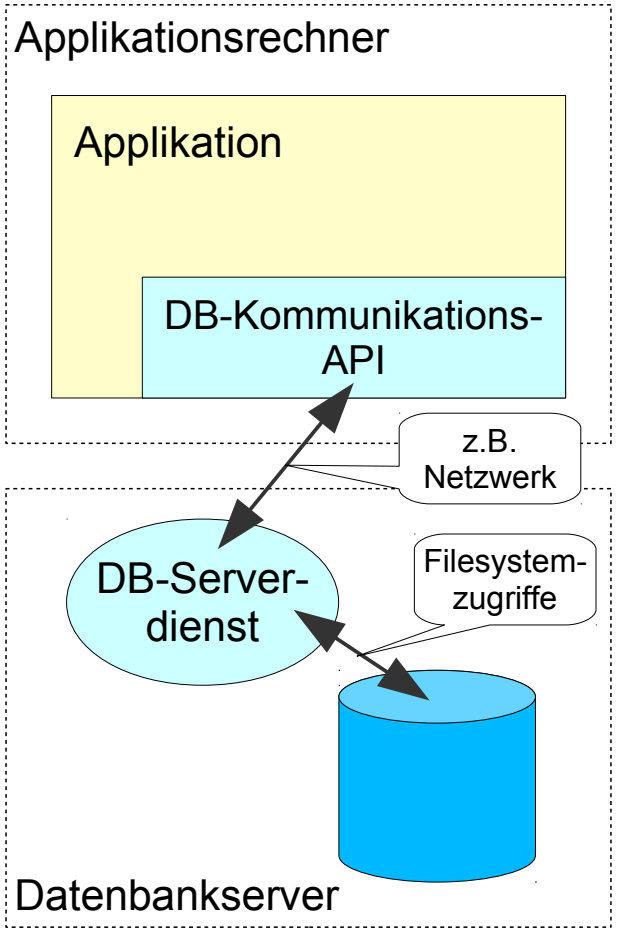
- Ein paar Definitionen...
- SQLite
  - Eigenschaften, Einordnung, Limits, Einschränkungen, Performance
  - Tools
  - SQLite in eigenen Programmen verwenden
- Ein konkretes Anwendungsbeispiel

Ein paar Definitionen...

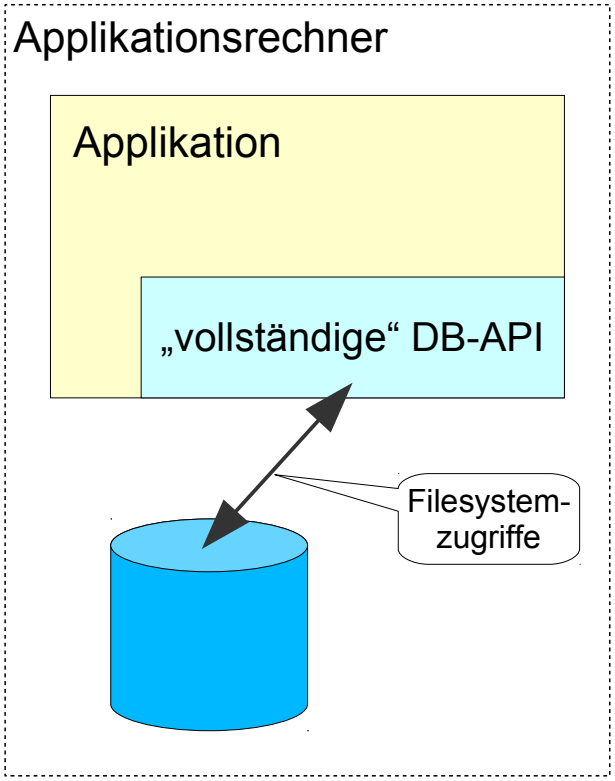
- Datenbanksystem (DBS)
- DBS-Komponenten:
  - Datenbankmanagementsystem (DBMS)
  - Datenbank (DB)
- Datenbankmodell
- Datenbanksprache

Client/Server DBS, eingebettetes DBS

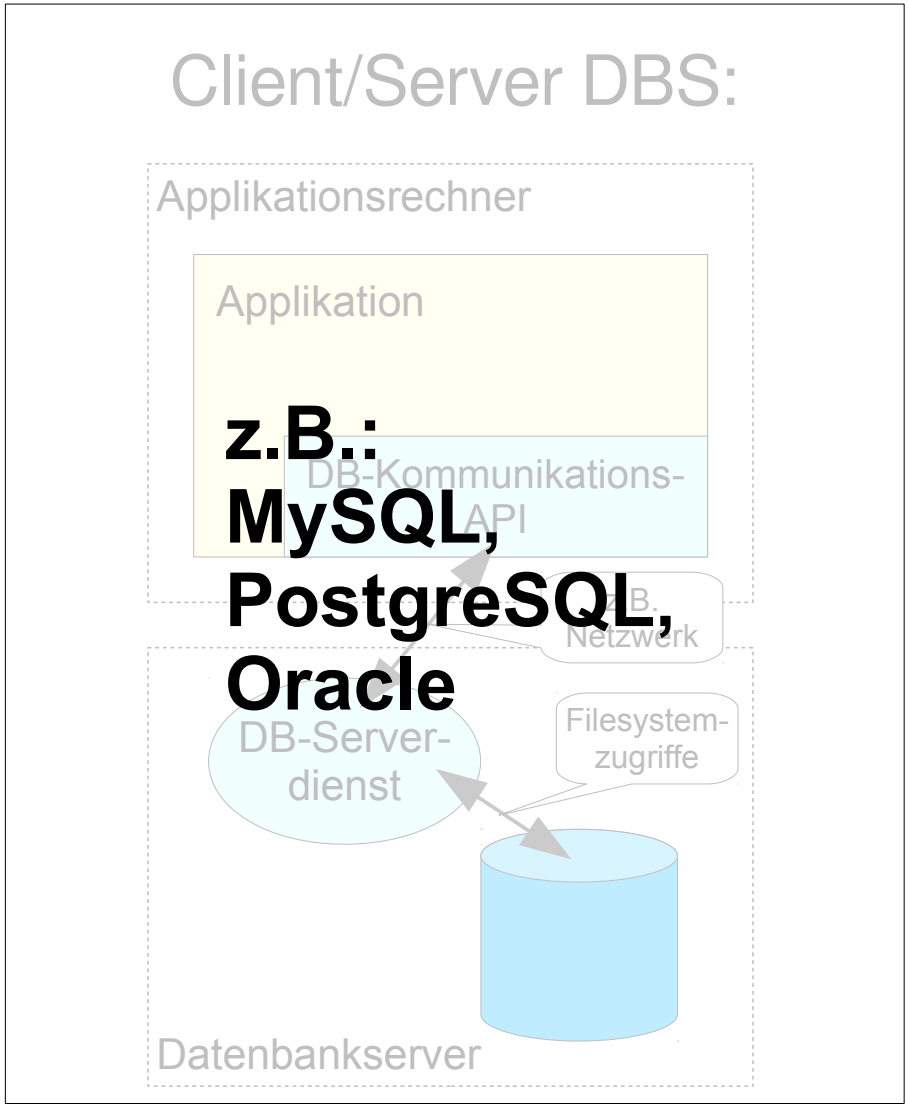
Client/Server DBS:

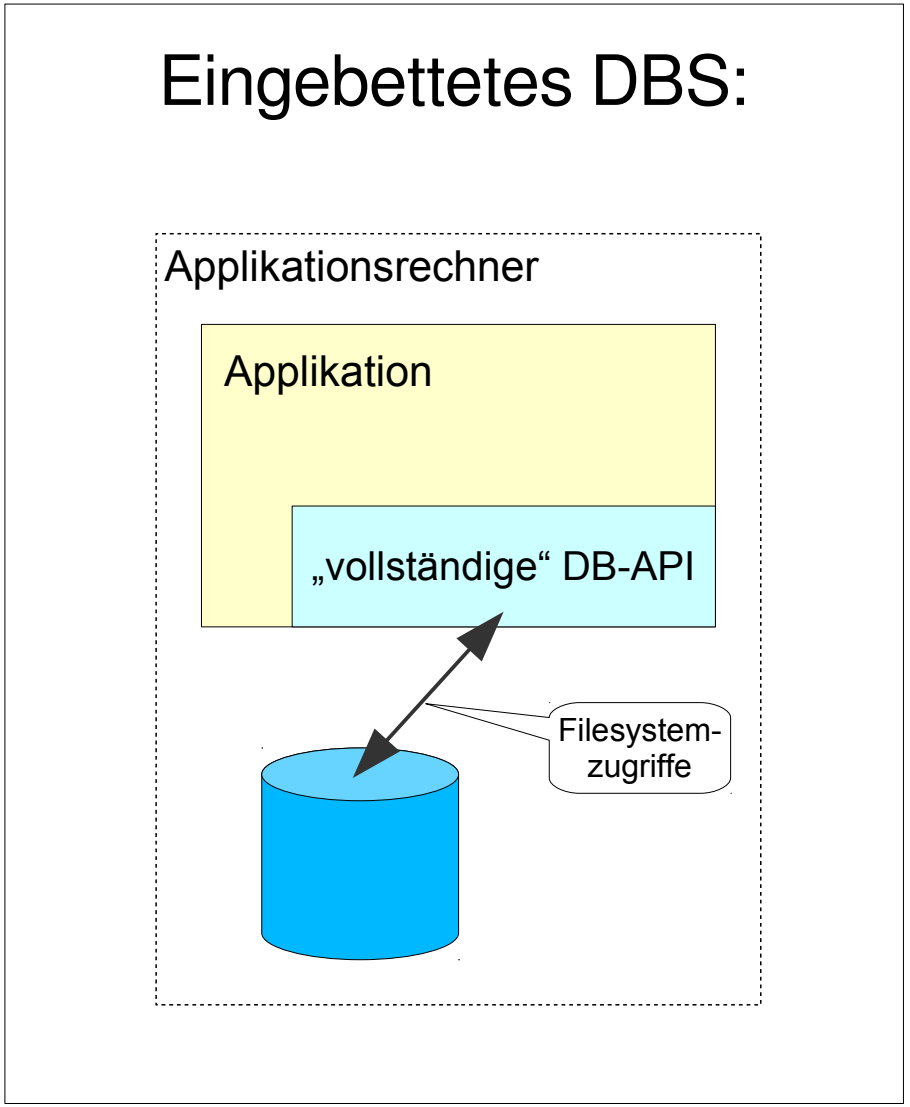


Eingebettetes DBS:

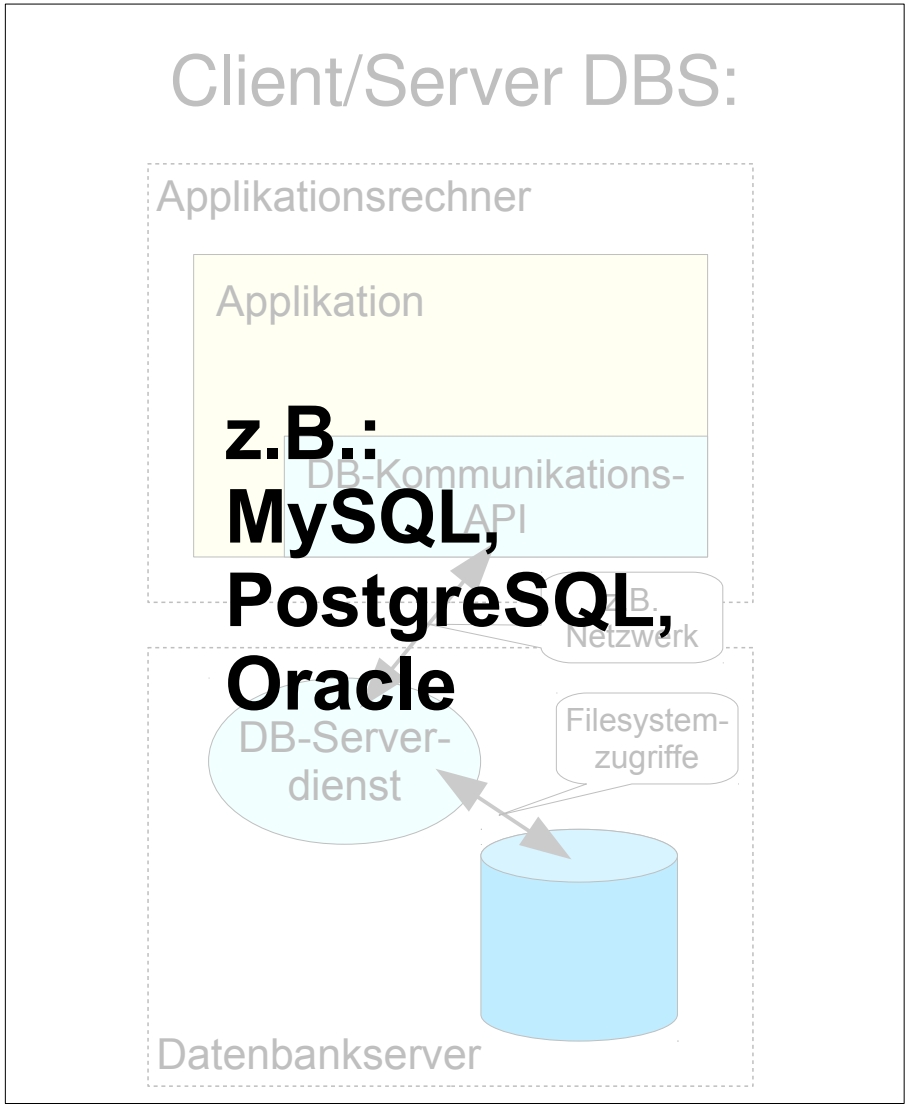


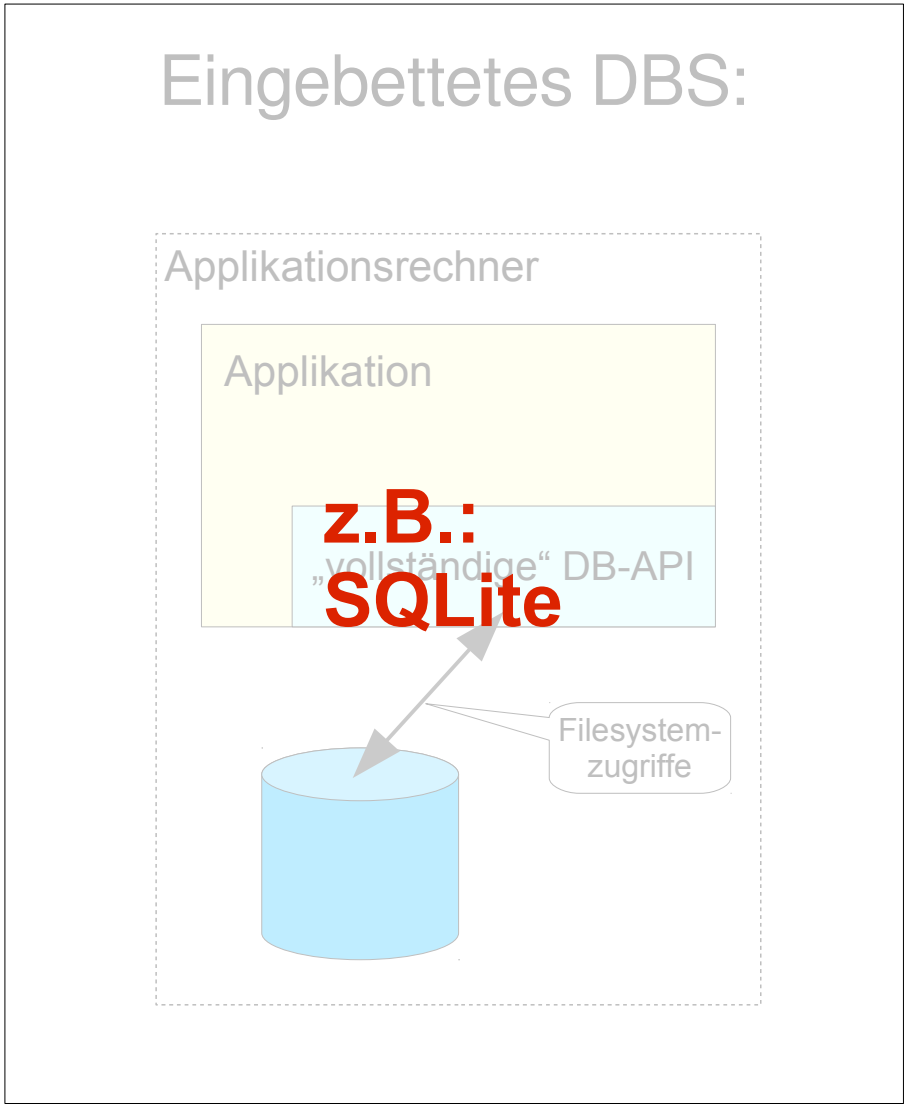
Client/Server DBS, eingebettetes DBS





Client/Server DBS, eingebettetes DBS





SQLite

→ relationales, „eingebettetes“ DBS, Datenbanksprache: SQL

- Projektstart: 2000, Richard Hipp
- Lizenz: Public Domain
- „SQLite“ → in den USA eingetragene Marke
- Aktuelle Version: 3.8.8
- Betriebssysteme: Linux, Windows, MacOS (plus einiger UNIX-Ableger; auch Android, iOS etc.)
- Komponenten:
  - SQL-Interpreter (Kommandozeiletool sqlite3)
  - C-Bibliothek inklusive Headerdatei

Einige SQLite-Eigenschaften

- Komplette DB in einer Datei
- „Zero-configuration“
- SQL-Standard (SQL-92) größtenteils implementiert
- Kleine, konfigurierbare, schnelle APIs mit sehr wenigen Abhängigkeiten (Objektcode max. 500kByte)
- Transaktionen: ACID („atomar“, „konsistent“, „isoliert“, „dauerhaft“)
- Benutzerdefinierte Funktionen möglich
- Zur Laufzeit nachladbare Erweiterungen...
  → http://sqlite.org/loadext.html

Einige SQLite-Limits

- Grenzen des Rechners, Filesystems, Betriebssystems, auf dem die SQLite-Anwendung läuft!
- Max. Länge von Strings/BLOBs: theoretisch (2^{31}-1) Byte
- Max. Anzahl Spalten einer Tabelle: 32767
- Max. Anzahl Zeilen pro Tabelle: 2^{64}
- Max. Anzahl Tabellen in einem JOIN: 64
- Max. Länge SQL-Statement: theoretisch 2^{30} Byte
- Max. DB-Größe: theoretisch (2^{31}-2) Pages * 2^{16} Byte
- Max. Anzahl geöffnete DBs in einer Applikation: 125

→ http://sqlite.org/limits.html

Abweichungen zum „Standard“-SQL?

Hauptsächlich sind dies:

- OUTER JOIN → nur LEFT OUTER JOIN implementiert
- ALTER TABLE → nur RENAME TABLE und ADD COLUMN implementiert
- Trigger → es wird FOR EACH ROW unterstützt
- Views → nur lesende Views möglich
- Berechtigungskonzept → nicht vorhanden, also z.B. auch kein GRANT und REVOKE

Performance: SQLite vs. MySQL

Randbedingungen:

- Hardware, OS, DBS-Versionen:
  - CPU: Intel Atom N450 (1.66GHz), 1GB RAM
  - Linux 3.13.0
  - SQLite 3.8.2
  - MySQL 5.5.41 (RESET QUERY CACHE vor jedem SELECT)
- Datenbank mit 7 Tabellen (insg. 6.159.331 Datensätze):
  - **adc_log: 1.235.489**
  - heatmap_ts: 420.768
  - humidity_log_neu: 1.067.108
  - last_values: 6
  - pressure_log: 964.939
  - **temp_log: 1.879.992**
  - voltage_log: 591.029

Performance: SQLite vs. MySQL

SQL-Befehl/Aktion	SQLite	MySQL
Import von 6.159.331 Datensätzen über Textdatei mit INSERTs	9m20.907s	46m24.495s
SELECT COUNT(*), AVG(lm75) FROM temp_log;	0m2.017s	0m3.396s
SELECT ts, lm75 FROM temp_log ORDER BY lm75 DESC;	0m11.477s	0m4.363s
SELECT ts, **tmp36** FROM **temp_log** ORDER BY **tmp36** DESC;	0m0.013s	0m0.026s
SELECT t.ts, t.lm75, a.adc0 FROM temp_log t LEFT JOIN adc_log a ON t.ts = a.ts ORDER BY lm75 DESC LIMIT 10;	0m29.693s	0m10.277s
SELECT t.ts, **t.tmp36**, a.adc0 FROM **temp_log t** LEFT JOIN adc_log a ON t.ts = a.ts ORDER BY **tmp36** DESC LIMIT 10;	0m0.223s	0m0.323s

→ Feld **tmp36** in Tabelle **temp_log** indiziert

SQLite-Tool: sqlite3

- Kommandozeileninterface (CLI); ein „SQL-Interpreter“
- Funktionen:
  - Manuelle Eingabe und Ausführung von SQL-Kommandos
  - Weitere Spezial-Kommandos („dot-commands“)
- Aufrufvarianten u.a.:
  - sqlite3 db_datei
  - sqlite3 db_datei ‘kommando‘
  - sqlite3 db_datei < query.sql
  - echo ‘kommando‘ | sqlite3 db_datei> | ...

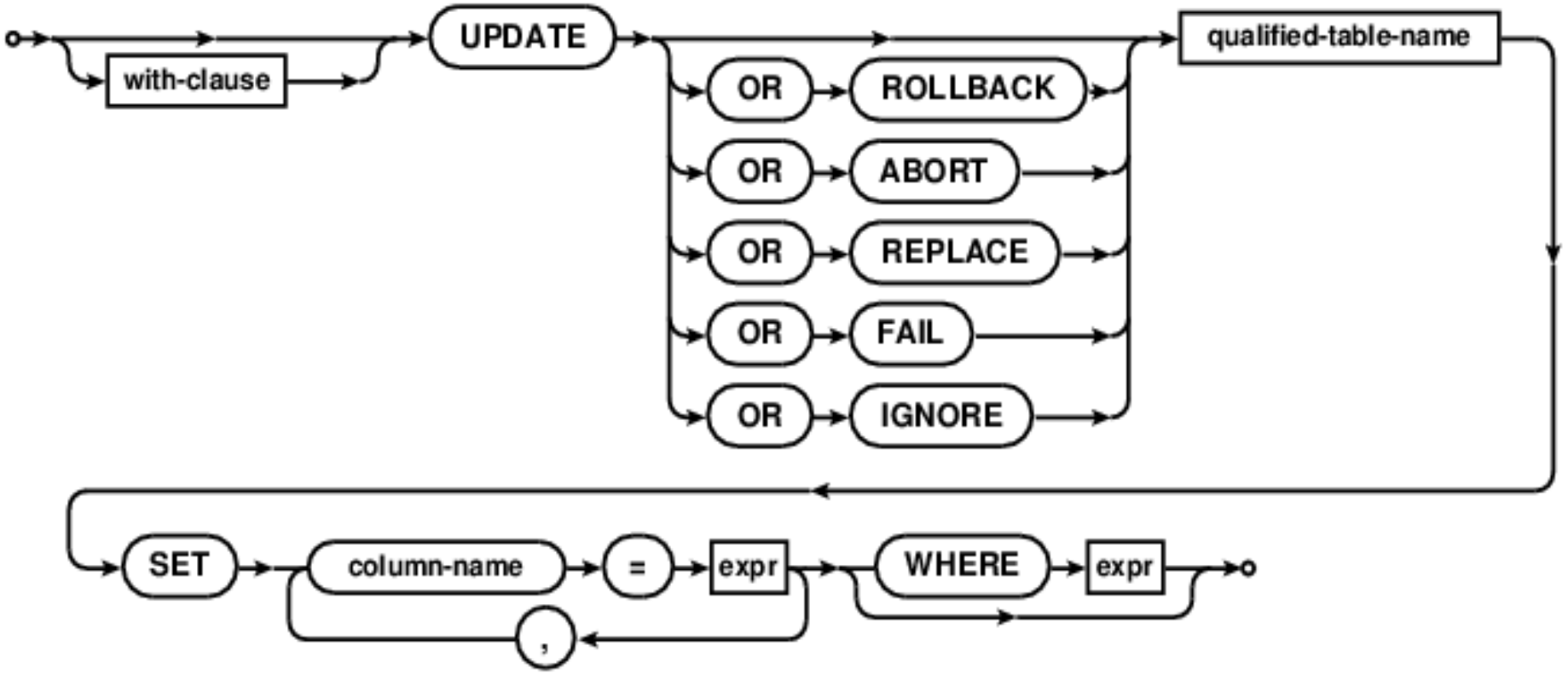
SQLite - weitere Managment-Tools

- SQLite-Manager: Firefox-Add-on (https://addons.mozilla.org/de/firefox/addon/sqlite-manager/)
- SQLite Database Browser (http://sqlitebrowser.org/)
- SQLiteman (http://sqliteman.yarpen.cz/)
- SQLiteStudio (http://sqlitestudio.pl/index.rvt)
- diverse WEB-Frontends (...analog phpMyAdmin)
- ...

SQLite - Dokumentation

→ https://www.sqlite.org

U.a. auch eine sehr gute SQL-Befehlsdokumentation:



SQLite in Programmen verwenden

Unzählige APIs für diverse Sprachen...

C/C++

Java

C#

php

tcl\tk

Lua

Haskell

ERLANG

python™

Ruby

Perl

SQLite in C-Programmen verwenden (1)

→ http://sqlite.org/cintro.html

```
/*
 *  SQLite-DB oeffnen und schliessen
 */

#include <sqlite3.h>

sqlite3 *db;
...
if (sqlite3_open("./db_file", &db)) {
   printf("%s\n", sqlite3_errmsg(db));
   Exit(1);
}
...
if (sqlite3_close(db) != SQLITE_OK) {
   /* Fehlerbehandlung */
}
```

SQLite in C-Programmen verwenden (2)

```
/*
 *   SQL-Kommando ausfuehren
 */

char *err = NULL;
char *sql;

...

sql = sqlite_mprintf("insert into tab1 values ('%s', %d)",
                     "zahl42", 42);

if (sqlite3_exec(db, sql, NULL, NULL, &err) != SQLITE_OK) {
    printf("%s\n", err);
    sqlite3_free(err);
    return;
}

...
```

SQLite in C-Programmen verwenden (3)

```
/*
 *   SQL-Abfrageergebnis abarbeiten
 */

char *sql;
sqlite3_stmt *vm;
...
sql = "select * from tab1";
if (sqlite3_prepare(db, sql, -1, &vm, NULL)) {
    /* Fehlerbehandlung */
} else {
    while (sqlite3_step(vm) != SQLITE_DONE){
        printf("%s, %d",
               sqlite3_column_text(vm, 0),
               sqlite3_column_int(vm, 1));
    }
}
sqlite3_finalize(vm);
...
```

SQLite in Tcl-Programmen verwenden

→ http://sqlite.org/tclsqlite.html

```
package require sqlite3

sqlite3 db ./db_file

db eval {create table tab1 (f1 text, f2 int)}
db eval {insert into tab1 values ('z23', 23)}
db eval {insert into tab1 values ('z42', 42)}

db eval {select * from tab1 order by f2} {
   puts "$f1, $f2"
}

db function bin {format 0b%b}
set result [db eval {select bin(f2) from tab1 where f1='z42'}]
puts $result

db close
```

Best practice – DB-Definitionen abfragen

Tabellendefinitionen etc. in Programmen abfragen...

- SQLite-spezifisches SQL-Kommando
  pragma table_info(<table_name>);
  - Liefert eine Liste alle Tabellenspalten, deren Typ etc.
  - Leere Liste → Tabelle ist nicht vorhanden
  - Es können auch Views abgefragt werden
- *paragma*-Anweisung weitere Möglichkeiten
  → http://www.sqlite.org/pragma.html

Best practice – Parallele DB-Zugriffe

SQLite bietet keine expliziten Mechanismen zur Verwaltung von parallelen Zugriffen...

- Aber:
  - Jedes schreibende SQLite-Programm setzt das Busy-Flag in der DB-Datei
  - Jede entsprechende SQLite-API-Funktion fragt dieses Flag ab und generiert ggf. einen entsprechenden Fehler, auf den reagiert werden kann!
  - Es kann eine Wartezeit (timeout) eingestellt werden
- Konkurrierende Lesezugriffe sind kein Problem

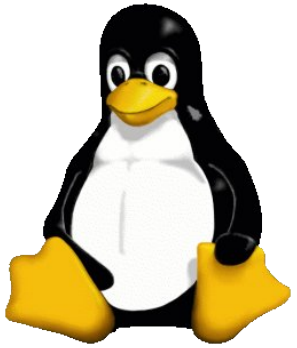
Best practice – DB-Backup

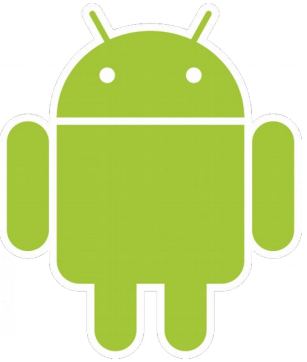
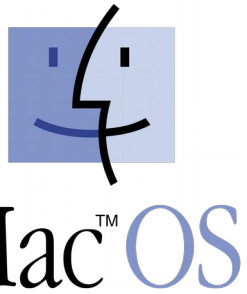
Wichtige Daten sollten auch mal gesichert werden...!

3 Methoden für SQLite-DBs:

- DB-File einfach kopieren:
  - Voraussetzung → es darf kein Prozess das DB-File geöffnet haben!
- API-Funktion *backup* bzw. sqlite3-Kommando *.backup*
  - Es wird eine 1:1 Kopie der DB „online“ erstellt
- sqlite3-Kommando *.dump*:
  - Generiert eine Text-Ausgabe mit INSERTS plus Tabellen-Schema(s) → Pipe...

Anwendungen, die SQLite benutzen









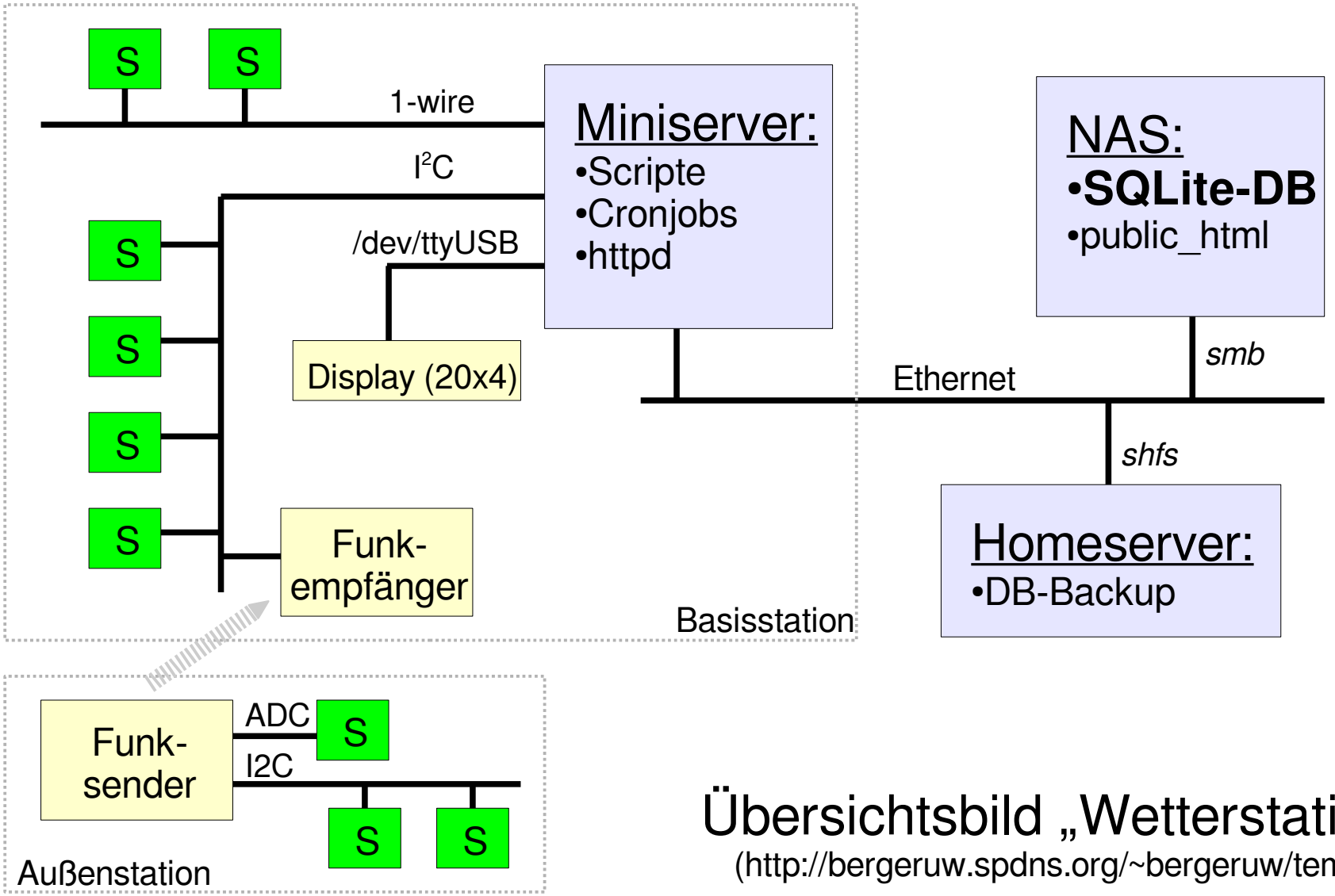
Anwendungen, die SQLite benutzen



...eigentlich ist SQLite
das am weitesten verbreitete DBS!

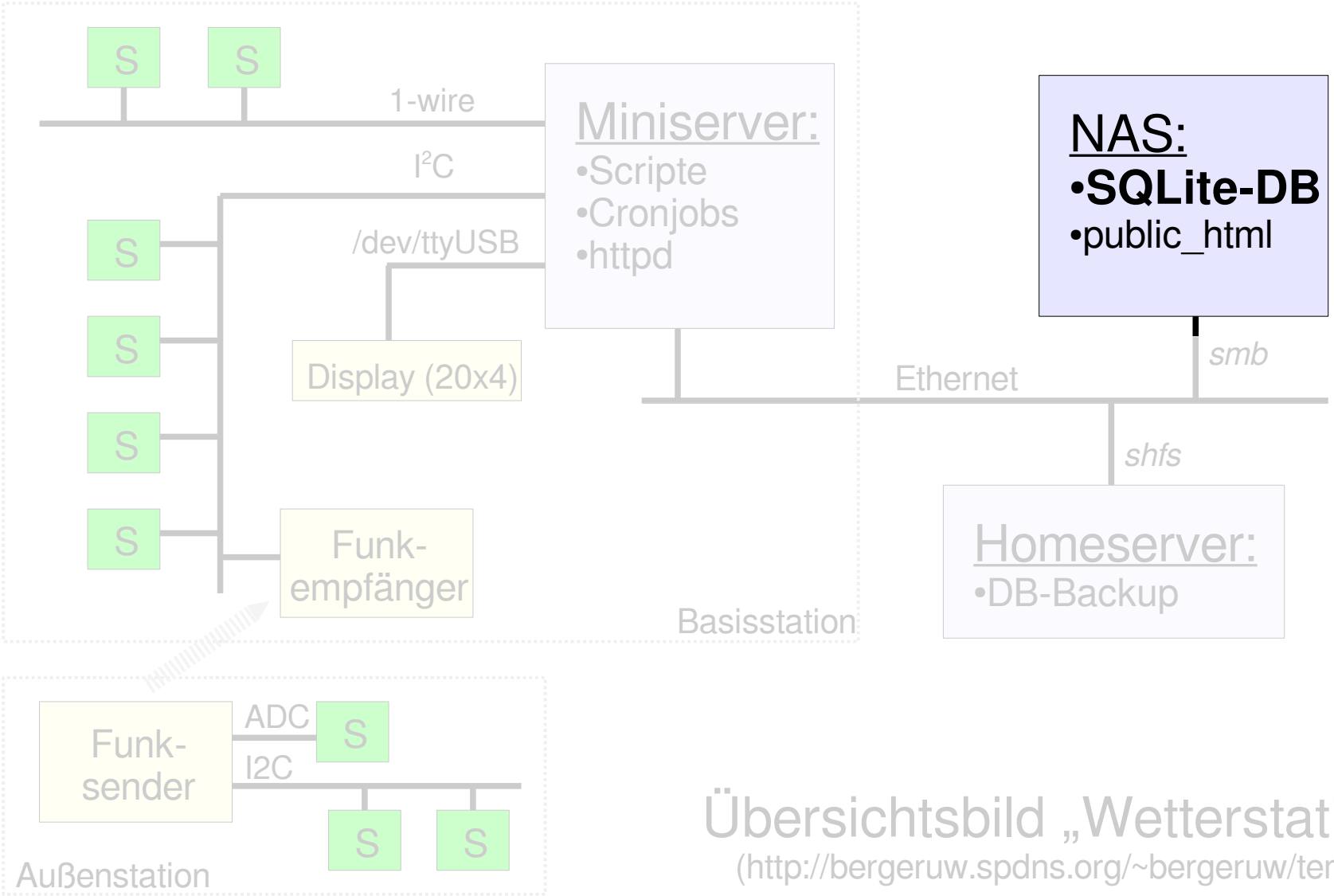
(→ http://sqlite.org/mostdeployed.html)

SQLite → konkretes Anwendungsbsp.



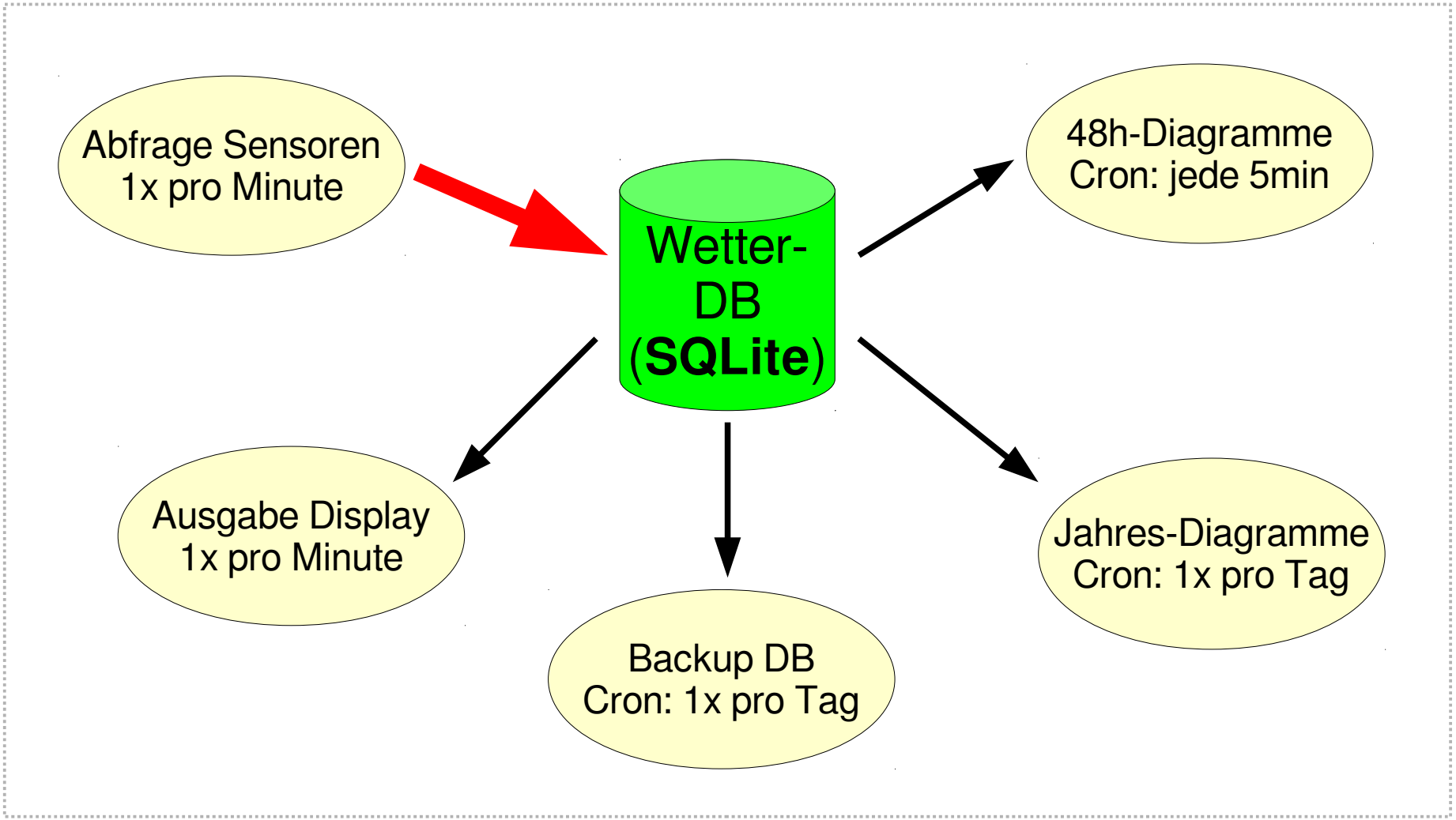
Übersichtsbild „Wetterstation“
(http://bergeruw.spdns.org/~bergeruw/temp/)

SQLite → konkretes Anwendungsbsp.



Übersichtsbild „Wetterstation“
(http://bergeruw.spdns.org/~bergeruw/temp/)

SQLite → konkretes Anwendungsbsp.



SQLite → konkretes Anwendungsbsp.

- Wetterdatenbank (Stand 17.11.2014):
  - Datenbankgröße: ca. 449MByte
  - Tabellen/Views/Trigger: 7/11/5
  - Anzahl Datensätze: ca. 6,2 Millionen
- Verwendete APIs:
  - C
  - Tcl
  - „Konsole“ (echo ‘sql-cmd‘ | sqlite3 <db-file> | ...)
- DB-Backup:
  - echo '.dump' | sqlite3 $source_file | gzip -c > $dest_file
  - komprimiert derzeit ca. 40MByte
  - Dauer ca. 40min (auf ein Netzwerklaufwerk)

Links

- http://sqlite.org
- http://www.linux-magazin.de/Ausgaben/2004/09/Datenspeicher-fuer-Sparsame
- http://www.tcl.tk/community/tcl2004/Presentations/D.RichardHipp/slides/slides-all.html
- http://www.linuxjournal.com/content/mariadbmysql-postgresql-and-sqlite3-comparing-command-line-interfaces?page=0,0
- http://db-engines.com/de/system/MySQL%3BPostgreSQL%3BSQLite
- http://www.sql-workbench.net/dbms_comparison.html
- http://bralug.de/wiki/Wetterdaten_mit_Linux_aufzeichnen_und_verarbeiten

Ende!

Hinweis: morgen 15:00 Uhr im V2;
„Raspberry Pi als Musikabspielgerät“