Anhang [A2]

Deploymentanalyse [DBAP2]

17.01.2013, Andreas Lockermann

Gefördert durch:

Bundesministerium
für Wirtschaft
und Technologie

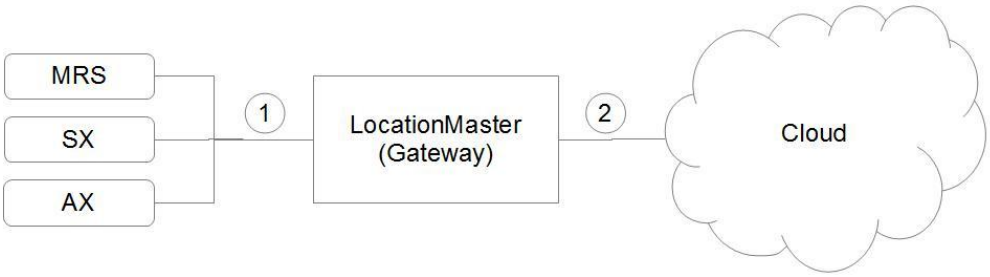
aufgrund eines Beschlusses
des Deutschen Bundestages

Auswertung der Deploymentanalyse

Laut dem Arbeitspaket DBAP2 des Projektantrags ist das Ziel der Deploymentanalyse „das Finden geeigneter DBMS sowohl für die Cloud DB als auch für die DB des LocationMaster. Innerhalb dieses Pakets ist für beide Rechnersysteme je ein optimales Produkt unter den Bedingungen einer public license, dem Umfang des darstellbaren Datenmodells und den Programmieranforderungen in Hinsicht auf die Verwaltung der Sensordaten auszuwählen."

Die verschiedenen DBMS, die zur Auswahl für den LocationMaster (LM) stehen, wurden gemäß bestimmter Untersuchungsmerkmale untersucht. Im Folgenden wird versucht, eine Entscheidung über die Wahl geeigneter DBMS mittels dieser Untersuchungsmerkmale zu treffen. Zu Untersuchungszwecken wurde eine Teststrecke errichtet. Diese wird im Folgenden erläutert:

Um neben den theoretischen Aspekten auch praktische Erfahrungen in die Deploymentanalyse mit einfließen lassen zu können, wurde eine Teststrecke aufgebaut:



MRS := Multi-Raum-Sensor misst Temperatur, Luftfeuchtigkeit, Luftgüte und Bewegung
SX := weitere Sensoren
AX := Aktoren

(1) := Sensordaten werden vom Sensor zum Gateway (LocationMaster) gesendet und in einer Datenbank abgelegt. Aktoren können angesteuert werden.
(2) := Daten aus dem Gateway werden in die Datenbank der Cloud übertragen.

Abbildung 1: Übersicht der Teststrecke

Auf dieser Teststrecke werden kontinuierlich Messdaten von Sensoren empfangen und lokal auf dem LocationMaster (Gateway) gespeichert (1). Dann werden die Messdatensätze in die Cloud (Laborrechner) übertragen (2), siehe dazu Bericht „Übertragung von Sensordaten unter Berücksichtigung von Transaktionskonzepten in der SensorCloud", Anhang A4. Der Multiraumsensor (MRS) in der Abbildung ist Beispiel eines Sensors, der ausprobiert wird. Weitere ausprobierte Sensoren SX sind z.B. ein Türkontaktsensor. Als Aktoren AX gibt es im Moment einen LEGO-Roboter und eine schaltbare Steckdose.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Zu Testzwecken wurden sieben verschiedene DBMS auf dem Gateway (LocationMaster) installiert und ausprobiert:

- SQLite
- db4o
- Apache Derby
- Berkeley DB
- HSQLDB
- PostgreSQL
- MySQL

Für die Cloud wurde Apache Cassandra in großen Umfang implementiert und getestet. Weiterhin wurden die DBMS db4o und PostgreSQL im begrenzten Umfang implementiert.

1.1 Relevante Kriterien

Die Kriterien, nach denen eine Entscheidung für geeignete DBMS getroffen werden soll, lassen sich in vier Kategorien unterteilen:

1. Kapazitätsverbrauch
2. Modellierungseigenschaften
3. Trusted-Aspekt
4. Kosten

Die folgenden acht Kriterien werden als relevant für die Auswahl geeigneter DBMS für den LocationMaster gesehen und diesen Kategorien wie folgt zugewiesen:

1. Kapazitätsverbrauch:
    a. Zeitverbrauch
    b. Speicherverbrauch
2. Modellierungseigenschaften:
    a. Datenmodell
    b. Erweiterbarkeit
3. Trusted-Aspekt:
    a. Sicherheit
    b. Transaktionen
    c. Kommunikation
4. Kosten:
    a. Lizenz

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

1.2 Bewertung der Kriterien

Um eine Bewertung vornehmen zu können, werden die Kriterien verschieden gewichtet und Szenarien zugeordnet. Ein Szenario ist z.B. ein Einfamilienhaus, in dem es eine geschätzte Menge X an Sensoren und Aktoren gibt, deren Messwerte zeitnah in die lokale Datenbank geschrieben werden müssen.

1. Kapazitätsverbrauch:
    a. Zeitverbrauch
    b. Speicherverbrauch

Ein LocationMaster soll mit einer bestimmten Menge von Sensoren und Aktoren kommunizieren können. Damit dies möglich ist, soll das DBMS viele Lese- und Schreibzugriffe zeitnah verarbeiten können.

Der LocationMaster hat beschränkte Betriebsmittel. Je nachdem welcher Typ oder welche Typen vom LocationMaster verwendet werden, ist es sinnvoll, wenn ein DBMS möglichst wenig Festspeicher verbraucht.

2. Modellierungseigenschaften:
    a. Datenmodell
    b. Erweiterbarkeit

Ein Datenbankmodell kann

- relational (z.B. SQLite)
- objektorientiert (z.B. db4o)
- key-value-orientiert (z.B. Berkeley DB)

sein. Es ist möglich, ein Modell A (z.B. key-value-orientiert) in der Cloud und ein anderes Modell B (z.B. relational) lokal auf dem LM zu verwenden. Demnach ist es also möglich, ein klassisches relationales DBMS lokal zu verwenden und in der Cloud eine key-value-DBMS zu benutzen. Der Grund für diese Modellvielfalt ist durch das Konzept der föderierten Datenbanksysteme (DBS) möglich:

Wenn in einem verteilten Datenbanksystem verschiedene Datenbankstrukturen integriert werden sollen, kann dazu das Konzept der föderierten DBS betrachtet werden: „Ein föderiertes Datenbanksystem ist ein schwach gekoppeltes verteiltes Datenbanksystem“ [SC TBFHK 11, S. 9]. Bei diesem Konzept besitzen die zu integrierenden DBS eigene Schemata, können jedoch insgesamt mit einem globalen Schema angesehen werden. (vgl. [SC TBFHK 11, S. 9])

Beim Datenmodell (2a) ist zu diskutieren, inwiefern sich die verschiedenen Datenbankmodelle für die Modellierung der lokalen FDBS-Segmente gut eignen, bzw. ob es Abstufungen gibt. Zum Beispiel ist zu überlegen, wie man eine referenzielle Integrität für eine objektorientiertes DBMS realisiert.

Über eine bestehende Menge von Sensoren und Aktoren sollen neue Sensoren und Aktoren einfach in das DBS des LM integriert werden können. Bei einem relationalen Modell kann man z.B. über den Befehl ALTER TABLE Tabellenstrukturen verändern.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

3. Trusted-Aspekt:
    a. Sicherheit
    b. Transaktionen
    c. Kommunikation

Im Projekt SensorCloud spielen die Trusted-Aspekte eine große Rolle. Demnach wird untersucht, ob ein Zugriffsschutz auf Attribute wie er z.B. in SQL mit den Kommandos GRANT und REVOKE ermöglicht wird und eine Verschlüsselung der Verbindung angeboten wird (z.B. mit SSL).

Bezüglich Transaktionen (3b) ist zu untersuchen, ob ACID-Bedingungen und eine XA-Unterstützung angeboten werden.

Das Kriterium Kommunikation (3c) spielt für die Kommunikation zwischen LocationMaster und Cloud eine große Rolle. Mit einem Client-Server-Mechanismus kann auf die lokale Datenbank auch von außerhalb zugegriffen werden. So kann man eine Übertragung von Messdaten in die Cloud oder die Übertragung von Daten aus der Cloud auf die lokale Datenbank z.B. mit einem Client-Server-Mechanismus auch von außerhalb durchführen.

4. Kosten:
    a. Lizenz

Bei einem DBMS verbunden mit einer kommerziellen Lizenz können weitere Kosten entstehen. Bei diesem Kriterium soll gezeigt werden, durch welche DBMS-Produkte Kosten entstehen können.

1.3 Untersuchung der DBMS

Im folgendem werden die DBMS gemäß der genannten Kriterien untersucht.

1.3.1 Kapazitätsverbrauch

1A) Zeitverbrauch gemessen auf der „SensorCloud“-Teststrecke der FH Köln:

Es wurden in die verschiedenen DBMS typische Datensätze geschrieben (Größe pro Datensatz ca. 76 Byte) und die Zeit gemessen, die ein DBMS für eine bestimmte Menge von Datensätzen braucht. Typische Datensätze sind in diesem Zusammenhang Datensätze, die durch den Aufbau der Teststrecke und der Erfassung von Messwerten gewonnen werden konnten. Gemäß eines FDBS-Schemas (siehe dazu Bericht zur „Anforderungsanalyse für das föderierte Datenbanksystem der SensorCloud“, Anhang A1) wurden diese Datensätze gemäß der Struktur MESSWERT gespeichert. Die Dauer für das Auslesen eine bestimmte Menge von Datensätzen wurde auch gemessen. Bei dieser Zeitmessung ging es um Zeitvergleiche der DBMS untereinander. Die absoluten Werte sind daher nicht von großer Bedeutung. Wichtig für den Test war es zu vergleichen, wie viel Zeit die verschiedenen DBMS für Schreib- und Leseoperationen benötigen, wenn man sie unter möglichst gleichen Bedingungen (gleiche Menge und Größe der Datensätze) betrachtet. Die Ergebnisse sind somit gültig bezogen auf die errichtete Teststrecke.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Die Tests wurden auf einem LocationMaster (Einplatinenrechner) durchgeführt. Dieser LocationMaster war vom Typ PANDA BOARD[1], für detaillierte Informationen, siehe [PABO GS 12] und [PABO RES 12]. Auf dem Einplatinenrechner wurde das Betriebssystem Ubuntu 10.10 ARM-OmapNetbook installiert.

Getestet wurde mit einem Java-Programm-System (zum Aufbau des Java-Programm-Systems siehe Anhang 1 „Java-Programm-System“) und es wurde über die folgenden Schnittstellen mit den Datenbanken kommuniziert:

- JDBC (bei SQLite, PostgreSQL, MySQL, Apache Derby, HSQLDB)
- Java Direct Persistence Layer (DPL) API (Berkeley DB)
- objektorientiert, Query By Example (Abfragemethode bei db4o)

Die DBMS wurden in den jeweiligen Standard-Konfigurationen verwendet.

Eine wichtige Frage ist die Folgende: Was wird genau gemessen? - Der Begriff „Zeitmessung“ bedeutet in diesem Zusammenhang das die Zeit gemessen wird, die benötigt wird, um einen Datensatz in eine Datenbank aus einem Java-Programm heraus zu schreiben. Dies beinhaltet die folgenden Arbeitsschritte (siehe dazu Anhang 1 „Java-Programm-System“):

- In einer Schleife wird jeweils ein Datensatz aus einer Array-Liste ausgelesen
- Eine UUID wird erzeugt
- SQL-Insert-String wird zusammengesetzt (im Fall von JDBC, sonst Objekterzeugung und Befüllung mit Inhalt)
- Datensatz wird an die Datenbank geschickt

Beim Lesen von Datensätzen werden die folgen Schritte ausgeführt (siehe dazu Anhang 1 „Java-Programm-System“):

- Objekt ResultSet wird angelegt bei JDBC; bei Berkeley DB wird ein Objekt Messwert und EntityCursor angelegt; bei db4o wird ein Objekt Messwert und eine Liste vom Typ Messwert angelegt
- SQL-Select-*-Anfrage wird an die Datenbank geschickt und das Ergebnis an die Instanz des ResultSets übergeben; bei Berkeley DB wird eine Instanz vom Typ EntityCursor zurückgegeben, mit dem man dann auf die Daten zugreifen kann; bei db4o wird eine Liste vom Typ Messwert zurückgegeben.

Es ist zu beachten, dass die Lesegeschwindigkeitsmessungen aufgrund unterschiedlicher Schnittstellen zu den DBs differenziert zu betrachten sind: Bei SQL-Datenbanken erhält man ein ResultSet zurück. Bei db4o erhält man eine Liste mit Objekten, und bei Berkeley DB ein Objekt vom Typ EntityCursor. Bei den verschiedenen Rückgabeobjekten sind gegebenenfalls verschiedene Verarbeitungsschritte notwendig, um mit den Daten weiterverarbeiten zu können. Diese Verarbeitungsschritte können dann unterschiedlich viel Zeit in Anspruch nehmen. Diese weiteren Schritte wurden bei den Messungen nicht berücksichtigt.

[1] PANDA BOARD REV A3, PCBA – 750-2152-021 REV A

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Die Ergebnisse dieses Tests sind in der folgenden Tabelle dargestellt:

DBMS	Test A1 (1000 Datensätze schreiben, in s)	Test A2 (5000 Datensätze schreiben, in s)	Test A3 (50000 Datensätze schreiben, in s)	Test B1 (1000 Datensätze lesen, in ms)	Test B2 3 (5000 Datensätze lesen, in ms)	Test B3 (10000 Datensätze lesen, in ms)
SQLite	42,1	213,9	1965,3	192,9	192,8	195,2
db4o	24,2	120,6	1478,5	277,4	290,8	346,8
Apache Derby	17,5	85,4	840,9	151,3	158,2	152,6
Berkeley DB	1,1	2,7	10,9	21	21	21
HSQLDB	0,9	3,0	28,4	98,4	169,8	282,6
PostgreSQL	8,7	40,5	386,2	121,1	213,4	293,2
MySQL	0,8	3,2	28,2	44,2	65,6	162

Die verschiedenen DBMS benötigen unterschiedlich viel Zeit, um zu schreiben oder zu lesen. Wenn man nach einem Punkteschema (10 Punkte für die DB, die die geringste Zeit beim schreiben bzw. lesen und 1 Punkte dementsprechend bei der meisten Zeit) versucht, die Datenbanken einzuteilen, sieht das Ergebnis dazu wie folgt aus:

DBMS	**Test A1**	**Test A2**	**Test A2**	**Test B1**	**Test B2**	**Test B3**
SQLite	3	3	4	9	9	9
db4o	6	6	6	8	8	7
Apache Derby	8	8	8	9	9	9
Berkeley DB	10	10	10	10	10	10
HSQLDB	10	10	10	10	9	8
PostgreSQL	9	9	9	9	8	8
MySQL	10	10	10	10	10	9

Als Bewertungskriterium beim Schreiben wird von folgendem ausgegangen: Wieviel Zeit benötigt ein DBMS um 1000 Datensätze zu schreiben, wenn das Zeitfenster maximal eine Minute groß ist (analog dazu 5 und 50 Minuten bei 5000 bzw. 50000 Datensätzen)? Demnach erhält bei einem 10 Punkteschema das DBMS, das z.B. 1 bis 6 Sekunden braucht 10 Punkte, ein DBMS das z.B. 7 bis 12 Sekunden braucht nur 9 Punkte bekommt und so weiter.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Als Bewertungskriterium beim Lesen wird von folgendem ausgegangen: Wieviel Zeit benötigt ein DBMS um 1000 (5000 bzw. 50000) Datensätze zu lesen, wenn das Zeitfenster maximal eine Minute groß ist? Da die Zeitunterschiede zwischen 1000 und 50000 gering sind, wird bei allen Tests (B1 bis B3) bei der Bewertung von maximal 1 Minute ausgegangen.

1B) Speicherverbrauch:

Speicherverbrauch meint den Verbrauch des Festspeichers. Der Festspeicher ist auf der Teststrecke eine SD-Karte, auf der auch das Betriebssystem liegt. Der Festspeicherverbrauch wurde aus der Literatur und aus einer eigenen Untersuchung auf der Teststrecke gegenübergestellt. Für manche DBMS sind keine Angaben zum Festspeicher bekannt. Der Festspeicher aus eigenen Untersuchungen bezieht sich auf die Größe bei dem jeweiligen DBMS zeigt den Verbrauch im Default-Zustand, d.h. ohne Schema oder Datensätze.

Ergebnisse zum Speicherverbrauch:

DBMS	**Festspeicherbedarf (Literatur)**	**Festspeicherbedarf (eigene Untersuchung)[2] „leere" DB, d.h. ohne Tabellen**	**Festspeicherbedarf (eigene Untersuchung) Tabellen mit 5000 Datensätzen[3]**
SQLite	kann < 350KB[4]	3132 KB (JAR-Datei)	ca. 632 KB (Größe der SQLite-Datei)
db4o	ca. 1 MB[5]	2588 KB (JAR-Datei) + 4 KB (Server-Programm, JAR-Datei) ergibt 2592 KB	ca. 1055 KB (Größe der db4o-Datei)
Apache Derby	ca. 2,6 MB[6]	2864 KB (3 JAR-Dateien)	ca. 5368 KB (Größe Ordner der DB), Ordner seg0 ist ca. 2240 KB und log ist ca. 3108 KB groß
Berkeley DB[7]	820 KB[8]	2360 KB	ca. 1030 KB (Größe Berkeley DB-Dateien)

[2] Benutzung des Programms df (vgl. http://wiki.ubuntuusers.de/df)
[3] Benutzung der Programme df (vgl. http://wiki.ubuntuusers.de/df) und du (vgl. http://wiki.ubuntuusers.de/du)
[4] [SQLI FP 12]
[5] [DB4O PI 12, S. 12]
[6] [ADB WD 12]
[7] Größe bei Java-Version
[8] [BDB OVJE 12], Größe bei Java-Version

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

HSQLDB	nicht angegeben	1368 KB (JAR-Datei) + 4 KB (Server-Programm, JAR-Datei) ergibt 1372 KB	ca. 552 KB (Größe der Dateien .lck, .log, .properties und .script), Datei .log ist ca. 550 KB und .script ist ca. 1 bis 2 KB groß
PostgreSQL	nicht angegeben	ca. 52796 KB (nach Installation mit apt-get)[9]	ca. 1008KB[10] (Größe var/lib/postgresql ca. 57080 KB)
MySQL	nicht angegeben	ca. 92612 KB (nach Installation mit apt-get)[11]	ca. 828 KB[12] (Größe var/lib/mysql ca. 22360 KB)

1.3.2 Modellierungseigenschaften

2A) Datenmodell:

Aus Sicht der Teststrecke ließen sich alle Entitäten auf jedem DBMS abbilden. Dabei konnte db4o direkt objektorientiert angesprochen werden. Bei Berkeley DB wurde die Java Direct Persistence Layer (DPL) API verwendet, um auf Java-Objekte zugreifen zu können[13].

DBMS	**Datenmodell**
SQLite	relational
db4o	objektorientiert[14]
Apache Derby	relational[15]
Berkeley DB	key/value[16]
HSQLDB	relational[17]
PostgreSQL	relational[18,19]
MySQL	relational[20]

[9] Differenz Festspeicher vor und nach Installation
[10] Differenz Festspeicher im Ordner var/lib/postgresql
[11] Differenz Festspeicher vor und nach Installation
[12] Differenz Festspeicher im Ordner var/lib/mysql
[13] [BDB OVJE 12]
[14] [DB4O PI 12, S. 1]
[15] [ADB GSD 12, S. 8]
[16] [BDB OVJE 12]
[17] [HSQL FS 12]
[18] [PG AB 12]
[19] PostgreSQL unterstützt ein objekt-relationales Datenmodell, auf der Teststrecke wurde nur das relationale verwendet

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

2B) Erweiterbarkeit:

Die Erweiterbarkeit wird bei den SQL-Datenbanken an den folgenden Kriterien bezüglich des SQL-ALTER TABLE Kommandos (vgl. SQL/92-Standard ([SQLS DD 99])) gemessen:

- Unterstützung von ADD und DROP COLUMN
- Unterstützung von ADD und DROP CONSTRAINT FOREIGN KEY

DBMS	Erweiterbarkeit	Konform zu ausgewählten Kriterien
SQLite	• ALTER TABLE nur eingeschränkt möglich. Folgende Befehle werden unterstützt: ○ RENAME TABLE ○ ADD COLUMN[21]	• kein DROP und COLUMN • kein ADD oder DROP CONSTRAINT FOREIGN KEY
db4o	• Änderung der Objektversionen (Modifizierung der Struktur einer Klasse) möglich. Dabei muss nicht die Datenbanknutzung unterbrochen werden[22]. Beispiele hierzu finden sich auf der Homepage[23]	• keine SQL-Datenbank
Apache Derby	• ALTER TABLE wird unterstützt. Unter anderem sind folgende Befehle möglich: ○ ADD COLUMN und CONSTRAINT ○ DROP COLUMN [CASCADE | RESTRICT] und CONSTRAINT für PRIMARY KEY, FOREIGN KEY, CHECK und UNIQUE ○ COLUMN CONTRAINTS: NOT NULL, PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK ○ DROP PRIMARY und FOEIGN KEY, UNIQUE und CHECK[24] • ALTER TABLE wird nicht bei temporären Tabellen unterstützt[25]	• alles wird unterstützt

[20] [MSQL RM 12, S. 4]
[21] [SQLI FTN 12]
[22] [DB4O RGDBB 12, S. 12]
[23] Beispiele und Erläuterungen dazu http://community.versant.com/documentation/Reference/db4o-8.1/java/reference/; Suchwort „rename a class“; Aufruf: 29.11.2012
[24] [ADB REF 12, S. 25-28, 76]
[25] [ADB REF 12, S. 52-54]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Berkeley DB	• Weiterentwicklung von Klassen wird in der Java Edition unterstützt (Felder hinzufügen oder erweitern)[26]	• keine SQL-Datenbank
HSQLDB	• ALTER TABLE wird unterstützt: ○ TABLE COLUMN TYPE kann z.B. verändert werden[27] ○ ADD, ALTER und DROP (RESTRICT oder CASCADE) COLUMN DEFINITION ○ ADD TABLE CONTRAINT DEFINITION ○ CONSTRAINTS: NOT NULL, UNIQUE (including PRIMARY KEY), PRIMARY KEY, FOREIGN KEY und CHECK[28]	• alles wird unterstützt
PostgreSQL	• ALTER TABLE wird unterstützt. Unter anderem sind folgende Befehle möglich: ○ ADD, DROP (RESTRICT oder CASCADE) und ALTER COLUMN ○ DROP und ADD CONSTRAINT ○ CONSTRAINTS: CHECK, NOT NULL, UNIQUE, PRIMARY und FOREIGN KEY[29]	• alles wird unterstützt
MySQL	• ALTER TABLE wird unterstützt: ○ ADD; ALTER und DROP COLUMN ○ ADD CONSTRAINT für PRIMARY KEY, UNIQUE, FOREIGN KEY ○ DROP PRIMARY KEY, FOREIGN KEY[30]	• alles wird unterstützt

[26] [BDB GS 11, S. 3f]
[27] [HSQL FS 12]
[28] [HSQL UG 12, S. 47f, S. 55f und S. 60ff]
[29] [PG DOC 12, S. 58 – 56, S. 1136 - 1145]
[30] [MSQL RM 12, S. 978f]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

1.3.3 Trusted-Aspekte

A) Sicherheit:

Die folgende Tabelle zeigt, ob ein Zugriffsschutz, wie er durch die SQL-Kommandos GRANT und REVOKE implementiert werden kann, und eine Verschlüsselung der Verbindung unterstützt wird.

DBMS	Zugriffsschutz	Verschlüsselung der Verbindung
SQLite	• Kein GRANT und REVOKE[31]	• keine SSL-Unterstützung bekannt
db4o	• *GrantAccess*-Methode (Benutzername und Passwort) für den Login zum Server[32]	• SSL-Unterstützung im Client-Server-Modus[33]
Apache Derby	• GRANT und REVOKE[34]	• Derby Network Server unterstützt SSL/TLS, Mode: ○ *off* (keine SSL-Verschlüsselung) ○ *basic* (SSL Verschlüsselung ohne peer authentication) ○ *peerAuthenication* (SSL Verschlüsselung mit peer authentication)[35]
Berkeley DB	• kein GRANT/REVOKE-Mechanismus bekannt	• keine SSL-Unterstützung bekannt
HSQLDB	• GRANT und REVOKE[36]	• SSL-Unterstützung bei Client-Server-Modus[37]
PostgreSQL	• GRANT und REVOKE[38]	• native Unterstützung für die Verwendung von SSL-Verbindungen zur Verschlüsselung der Client/Server-Kommunikation[39]
MySQL	• GRANT[40] und REVOKE[41]	• SSL-Unterstützung im Client-Server-Modus[42]

[31] [SQLI FTN 12]
[32] [DB4O RJ 12]
[33] [DB4O RJ 12]
[34] [ADB REF 12, S. 58 – 61 und S. 65 - 70]
[35] [ADB SAG 12, S. 48 - 51]
[36] [HSQL FS 12]
[37] [HSQL FS 12]
[38] [PG FM 12]
[39] [PG DOC 12, S. 403 – 406 und 636 – 639] und [PG AB 12]
[40] [MSQL RM 12, S. 1123f]
[41] [MSQL RM 12, S. 1131f]
[42] [MSQL RM 12, S. 591-599]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Darstellung demnach, ob beide Sicherheitsaspekte unterstützt werden mit „+“ und „-“:

DBMS	Zugriffsschutz	Verschlüsselung der Verbindung
SQLite	-	-
db4o	+	+
Apache Derby	+	+
Berkeley DB	-	-
HSQLDB	+	+
PostgreSQL	+	+
MySQL	+	+

B) Transaktionen:

ACID steht für die Begriffe Atomarität, Konsistenz, Isolation und Persistenz (Durability). Atomarität bedeutet in diesem Zusammenhang, dass „das Programm entweder vollständig oder gar nicht ausgeführt“ [GV DB 08, S. 643] wird. Konsistenz bedeutet, dass alle geforderten Konsistenzbedingungen (z.B. PRIMARY KEY) bei einer Transaktion eingehalten werden. Der Begriff Isolation stellt sicher, dass das Programm isoliert von anderen Transaktionen abläuft. Persistenz stellt sicher, dass die Daten die bei einer Transaktion z.B. neu eingefügt worden sind, dauerhaft gespeichert werden. Bei der Verarbeitung von Transaktionen ist es wichtig, dass die Transaktionsverarbeitung diese vier Eigenschaften sicherstellt. vgl. [GV DB 08, S. 643F]

XA ist ein standardisierte Schnittstelle des X/Open-DTP-(Distributed Transaction Processing) Modells, welches verteilte Transaktionen über mehrere Ressourcen (z.B. Datenbanken) durchführen zu können.

DBMS	ACID	XA-Unterstützung
SQLite	ACID[43]	Nein
db4o	ACID[44]	Nein
Apache Derby	ACID (bei Version 10.2.2.0) siehe http://db.apache.org/derby/releases/release-10.2.2.0.html	Ja[45]
Berkeley DB	ACID[46], Java Edition: ACID[47]	Nein

[43] [SQLI TR 12]
[44] [DB4O PI 12, S. 12]
[45] [ADB REF 12]
[46] [BDB OV2 12]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

HSQLDB	Nicht voll ACID[48]	Nein
PostgreSQL	ACID[49]	Ja[50]
MySQL	ACID (mit Engine InnoDB)[51]	Ja[52]

Darstellung demnach, ob beide Transaktionsaspekte unterstützt werden mit „+“ und „-“:

DBMS	**ACID**	**XA-Unterstützung**
SQLite	+	-
db4o	+	-
Apache Derby	+	+
Berkeley DB	+	-
HSQLDB		-
PostgreSQL	+	+
MySQL	+	+

C) Kommunikation:

Für das Kriterium Kommunikation ist es wichtig, ob ein DBMS einen Client-Server-Mechanismus unterstützt. Demnach wird in der zweiten folgenden Tabelle mit „+“ bzw. „-“ unterschieden, ob dieser Mechanismus unterstützt wird oder nicht.

Neben der Client-Server-Funktionalität sind in der folgenden Tabelle die Begriffe serverlos, Lokal und Embedded (bzw. Embeddable) bei den verschiedenen DBMS aufgeführt. Diese Begriffe können folgendermaßen verstanden werden: Serverlos bedeutet, dass auf diese Datenbank nur lokal und nicht über ein Netzwerk zugegriffen werden kann. DBMS können als Embedded bzw. Embeddable gesehen werden, wenn sie ressourcensparend betrieben werden können. Als Lokal können DBMS angesehen werden, die keinem Client-Server-Mechanismus besitzen und somit nicht über ein Netzwerk erreichbar sind. Ein DBMS kann die Eigenschaften Lokal bzw. Client-Server und Embedded besitzen.

[47] [BDB OVJE 12]
[48] [HSQL UG 12, S. 200f]
[49] [PG AB 12]
[50] PG DOC 12]
[51] [MSQL RM 12, S. 23f]
[52] [MSQL RM 12]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

DBMS	Modell	Client-Server-Funktionalität
SQLite	• serverlos[53]	• keine
db4o	• Client-Server • Lokal[54]	• wird unterstützt
Apache Derby	• Derby Network Client JDBC und Derby Network Server • Embedded JDBC Treiber[55]	• wird unterstützt
Berkeley DB	• Embeddable[56]	• keine
HSQLDB	• Client-Server (Protokolle: HSQL, HTTP, HSQL-BER mit SSL möglich) • Embedded (in Java Applikationen)[57]	• wird unterstützt
PostgreSQL	• Client-Server-Modell mit TCP/IP • Lokal[58]	• wird unterstützt
MySQL	• Client-Server • Embedded (in Applikation)[59]	• wird unterstützt

1.3.4 Kosten

A) Lizenz:

Durch Lizenzen können Kosten entstehen. Die folgende Übersicht zeigt die Lizenzbedingungen der untersuchten DBMS:

DBMS	Lizenz
SQLite	• Public Domain[60]
db4o	• Free General Public License • Commercial License[61]
Apache Derby	• The Apache License, Version 2.0[62]

[53] [SQLI SL 12]
[54] [DB4O PI 12, S. 12]
[55] [ADB WD 12]
[56] [BDB OV2 12] und [BDB OVJE 12]
[57] [HSQL FS 12]
[58] [PG DOC 12, S. 456 – 462 und 585 - 594]
[59] [MSQL RM 12, S. 4]
[60] [SQLI LC 12]
[61] [DB4O DL 12]
[62] [ADB REF 12, S. 11 - 14]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Berkeley DB	• Open Source License • Commercial License[63]
HSQLDB	• COPYRIGHTS AND LICENSES (based on BSD License)[64]
PostgreSQL	• The PostgreSQL License (PostgreSQL)[65], liberal open source license[66]
MySQL	• GPL (GNU General Public License) • Commercial License[67]

1.4 Szenarien

Anhand von konkreten Szenarien, die für einen LocationMaster entstehen können, wird im folgendem versucht, mit Hilfe von Anwendungsfällen die DBMS zu diskutieren, die die Auswahlkriterien der Deploymentanalyse gut erfüllen. Die Annahmen für die Szenarien sind frei gewählt.

Einfamilienhaus:

In einem Einfamilienhaus befinden sich viele Sensoren und Aktoren. Eine Annahme ist, dass sich diese Summe auf 50 (maximal 100) beläuft und jeder Aktor bzw. Sensor alle 60 Sekunden ein Messwert an den LocationMaster sendet. Demnach müssen pro Minute 50 (maximal 100) Datensätze in der lokalen Datenbank gespeichert werden.

Nach dem Kriterium 1a (Zeitverbrauch), kann jedes DBMS im Idealfall (laut Messung) 1000 Datensätze pro Minute in der DB speichern.

Sollen die Daten aus der lokalen Datenbank in die Cloud übertragen werden, ist die Lesegeschwindigkeit von großer Bedeutung. Nach dem Kriterium 1a besitzen alle untersuchten DBMS im Idealfall (laut Messung) eine Lesegeschwindigkeit, um in kurzer Zeit viele Datensätze auslesen zu können, die in der Summe mit der Schreibgeschwindigkeit bei maximal 100 Datensätzen pro Minute im Idealfall unter einer Minute bleibt.

Hotel:

Hotel steht hier als Synonym für ein Gebäude, in dem sich viele Sensoren befinden, also wesentlich mehr als in einem Einfamilienhaus. In einem Hotel befinden sich mehr Zimmer und

[63] [BDB LC 12]
[64] [HSQL LC 12]
[65] [PGL OSI 12]
[66] [PG AB 12]
[67] [MSQL RM 12, S. 1]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

somit auch mehr Sensoren. Mit der Annahme, das sich in einem Hotel 500 (maximal 1000) Sensoren befinden, die alle 60 Sekunden eine Messwert senden, ergibt eine Summe von 500 (maximal 1000) Datensätzen pro Minute.

Nach dem Kriterium 1a (Zeitverbrauch), kann jedes DBMS im Idealfall (laut Messung) 1000 Datensätze pro Minute in der DB speichern.

Auch für das Szenario „Hotel“ sind die Lesegeschwindigkeiten bei allen DBMS so hoch, das in einer Minute 1000 Datensätze pro Minute geschrieben und zum Schreiben in die Cloud gelesen werden können.

DBMS mit Trusted-Aspekten:

DBMS, die die Trusted-Aspekte Sicherheit (3a), Transaktionen (3b) und Kommunikation (3c) erfüllen, lassen sich aus der folgenden Übersicht entnehmen:

DBMS	**Zugriffsschutz**	**Verschlüsselung der Verbindung**	**ACID**	**XA-Unterstützung**	**Client-Server-Architektur**
SQLite	-	-	+	-	-
db4o	+	+	+	-	+
Apache Derby	+	+	+	+	+
Berkeley DB	-	-	+	-	-
HSQLDB	+	+		-	+
PostgreSQL	+	+	+	+	+
MySQL	+	+	+	+	+

Es zeigt sich, dass die DBMS Apache Derby, PostgreSQL und MySQL alle Trusted-Aspekte unterstützen.

Erweiterbarkeit des Datenmodells:

Ein Schema in einer Datenbank kann sich von Zeit zu Zeit ändern. Um ein DBMS im klassischen Sinn (ALTER TABLE) hinsichtlich der Spalten und der CONSTRAINTS ändern zu können, wurde unter 2b die Erweiterbarkeit untersucht. Die DBMS Apache Derby, HSQLDB, PostgreSQL und MySQL unterstützen die Erweiterbarkeitsfunktionen, nach denen die DBMS theoretisch untersucht

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

worden sind. Die DBMS db4o und Berkeley DB sind keine SQL-Datenbanken, besitzen aber die Möglichkeit Klassen zu modifizieren[68].

Ressoucensparend:

Auf der Teststrecke wurde ein LocationMaster verwendet, der hinsichtlich seines Festspeichers (1b) eine 16GB-Karte zur Verfügung hatte. Möchte man aus Kostengründen ein DBMS wählen, das möglichst wenig Speicher verbraucht, wird dieser Aspekt im folgendem betrachtet: Die DBMS, die weniger Festspeicher („leere DB“) als 10 MB benötigen sind, SQLITE, db4o, Apache Derby, Berkeley DB und HSQLDB. Wird das DBMS mit Daten befüllt, ergeben sich keine großen Unterschiede zwischen allen Untersuchten DBMS.

1.5 Zusammenfassung und Auswertung

Auf der Teststrecke ließen sich alle DBMS verwenden und demnach ist es möglich, jedes dieser DBMS auf dem LocationMaster zu verwenden.

Wie im vorherigen Kapitel beschrieben, ergeben sich aus verschiedenen Szenarien mit dem Fokus auf bestimmte Kriterien unterschiedliche Ergebnisse hinsichtlich der Auswahl geeigneter DBMS für den LocationMaster. In diesem Kapitel wird versucht, eine Auswertung aus den im vorherigen Kapitel diskutierten Szenarien verknüpft mit den Kriterien vorzunehmen.

Das Kriterium Datenmodell wurde auf der Teststrecke beachtet, hat jedoch zu dem aktuellen Datenmodell (gegeben durch die Anforderungsanalyse, siehe Anhang A1) keine besonderen Vor- oder Nachteile gegenüber den verschiedenen Modellen gezeigt. Das Kriterium Lizenzen wurde nur grob betrachtet, d.h. gibt es eine kommerzielle Lizenz oder gibt es keine und bedarf noch einer weiteren Untersuchung, die gemeinsam mit den industriellen Konsortialpartnern des Projekts SensorCloud durchzuführen ist.

Wenn man die Kategorie „Trusted-Aspekte“ (vgl. Szenario „DBMS mit Trusted-Aspekten“) mit einer hohen Priorität versieht, erfüllen die DBMS Apache Derby, PostgreSQL und MySQL diese Kategorie. Nimmt man dazu die Szenarien „Einfamilienhaus“ und „Hotel“ für die drei DBMS sind alle drei DBMS für diese Szenarien geeignet. Das Kriterium „Erweiterbarkeit des Datenmodells“ wird von allen drei DBMS gleich gut unterstützt. Kriterium „Ressourcensparend“ zeigt, dass Apache Derby weniger Festspeicher verbraucht als die anderen beiden DBMS. Dieses Szenario wird unter dem LocationMaster, der auf der Teststrecke verwendet wird vernachlässigt, da dort genug Festspeicher zur Verfügung steht.

Demnach stehen unter dem Aspekt, dass man einen LocationMaster dieser Art vorliegen hat und man die Szenarien „Trusted-Aspekte“ als besonders wichtig befindet die drei DBMS Apache Derby, PostgreSQL und MySQL zur Auswahl.

[68] db4o: [DB4O RGDBB 12, S. 12]; Beispiele und Erläuterungen dazu: http://community.versant.com/documentation/Reference/db4o-8.1/java/reference/; Suchwort „rename a class“; Aufruf: 29.11.2012; Berkeley DB: [BDB GS 11, S. 3f]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Alle DBMS sind unter Linux lauffähig. Es ist zu beachten, das Apache Derby zudem eine JVM (mind. JDK Version 1.4)[69] benötigt. MySQL erfordert möglicherweise eine kommerzielle Lizenz.

Wenn man die Kategorie „Ressoucensparend“ sowie die Szenarien „Einfamilienhaus“ und „Hotel“ in den Vordergrund stellt und „Trusted-Aspekte“ vernachlässigen würde, was im Kontext des TrustedCloud Programms eher nicht erwogen wird, sind die DBMS SQLite, db4o, Apache Derby, Berkeley DB und HSQLDB zu betrachten. Davon sind die DBMS SQLite und Berkeley DB ohne die wesentlichen „Trusted-Aspekte“, d.h. ohne Client-Server-Mechanismus und ohne XA-Unterstützung. SQLite ist das DBMS, das unter den SQL-Datenbanken wenig Erweiterbarkeitsmöglichkeiten bietet.

Es ist bei diesen letzten Aspekten zu beachten, dass man die DBMS-Produkte db4o, Apache Derby und HSQLDB auch in einem nicht Client-Server-Mechanismus betreiben kann. Somit kann es sein, dass im Laufe des Projekts durch verschiedene Anforderungen an das lokale DBMS sich Prioritäten verschieben können. Zum Beispiel wenn ein DBMS benötigt wird, dass einen Zugriffsschutz benötigt und nur lokal (nicht Client-Server-Mechanismus) funktionieren soll. Dann wäre z.B. db4o ein Kandidat.

Für ein minimales DBMS auf dem LocationMaster werden die beiden DBMS SQLite und Berkeley DB als Kandidaten für den LocationMaster vorgeschlagen. Berkeley DB erfordert möglicherweise eine kommerzielle Lizenz.

Abschließend wird PostgreSQL als Favorit vorgeschlagen, auf dem aktuell vorhandenen LocationMaster als DBMS zu installieren. Dieses DBMS bietet zusammen mit Apache Derby und MySQL die meisten gewünschten Eigenschaften und funktionierte bei den Tests auf der Teststrecke ohne Probleme. Dieser Vorschlag schließt nicht aus, dass sich im weiteren Verlauf des Projekts vielleicht eine andere Gewichtung der Anforderungen für den LocationMaster ergeben kann, wie z.B. ein DBMS wie SQLite oder Berkeley DB, die weniger Eigenschaften erfüllen, jedoch Ressourcensparender sind.

[69] [ADB GSD 12, S. 8]

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Quellen

[ADB GSD 12] Apache Derby: Getting Started with Derby:
Quelle: http://db.apache.org/derby/docs/dev/getstart/getstartderby.pdf
Aufruf: 19.04.2012

[ADB REF 12] Apache Derby: Derby Reference Manual Version 10.8:
Quelle: http://db.apache.org/derby/docs/10.8/ref/refderby.pdf
Aufruf: 29.10.2012

[ADB SAG 12] Apache Derby: Derby Server Administration Guide, Version 10.8:
Quelle:
http://db.apache.org/derby/docs/10.8/adminguide/derbyadmin.pdf
Aufruf: 19.04.2012

[ADB WD 12] Apache Derby: What is Apache Derby?:
Quelle:
http://db.apache.org/derby/index.html#What+is+Apache+Derby%3F
Aufruf: 19.04.2012

[BDB LC 12] Berkeley DB: Oracle Berkeley DB Licensing Information:
Quelle:
http://www.oracle.com/technetwork/database/berkeleydb/downloads/licensing-098979.html
Aufruf: 25.04.2012

[BDB GS 11] Berkeley DB: Oracle Berkeley DB, Java Edition
Getting Started with Berkeley DB Java Edition
11g Release 2 Library Version 11.2.5.0:
Quelle:
http://docs.oracle.com/cd/E17277_02/html/GettingStartedGuide/BerkeleyDB-JE-GSG.pdf
Aufruf: 23.05.2012

[BDB OV 12] Berkeley DB: Overview Oracle Berkeley DB 11g:
Quelle:
http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html
Aufruf: 25.04.2012

[BDB OVJE 12] Berkeley DB: Overview Oracle Berkeley DB Java Edition:
Quelle:
http://www.oracle.com/technetwork/products/berkeleydb/overview/index-093405.html
Aufruf: 25.04.2012

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

[DB4O DL 12] db4o: Download db4o:
Quelle: http://www.db4o.com/DownloadNow.aspx
Aufruf: 24.04.2012

[DB4O PI 12] db4o: Product Information, Version 8.0 Java and .NET:
Quelle:
http://www.db4o.com/about/productinformation/db4o%20Product20Information%20V8.0.pdf
Aufruf: 12.12.2012

[DB4O RGDBB 12] db4o: The Database Behind the Brains:
Rick Grehan:
Quelle:
http://www.db4o.com/deutsch/db4o%20Whitepaper%20-%20The%20Database%20Behind%20the%20Brains%28German%29.pdf
Aufruf: 8.05.2012

[DB4O RJ 12] db4o: Reference Docs Version 8.1 Java:
Quelle:
http://community.versant.com/documentation/Reference/db4o-8.1/java/reference/
Aufruf: 3.05.2012

[GV DB 08] Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme:
Gottfried Vossen:
München, 2008, 5. Auflage

[HSQL FS 12] HSQLDB: Features Summary:
Quelle: http://hsqldb.org/web/hsqlFeatures.html
Aufruf: 25.04.0212

[HSQL LC 12] HSQLDB: COPYRIGHTS AND LICENSES (based on BSD License):
Quelle: http://hsqldb.org/web/hsqlLicense.html
Aufruf: 25.04.2012

[HSQL UG 12] HSQLDB: HyperSQL Guide, HyperSQL Database Engine (HSQLDB) 2.2:
Blane Simpson, Fred Toussi:
Quelle: http://hsqldb.org/doc/2.0/guide/guide.pdf
Aufruf: 25.04.2012

[MSQL RM 12] MySQL: MySQL 5.5 Reference Manual
Including MySQL Cluster NDB 7.2 Reference Guide:
Quelle: http://dev.mysql.com/doc/
Aufruf: 29.05.2012

[PABO GS 12] Pandaboard: Resources: Getting Started:
Quelle: http://pandaboard.org/content/resources/getting-started
Aufruf: 18.07.2012

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

[PABO RES 12] Pandaboard: Resources – Board Refernces:
Quelle: http://pandaboard.org/content/resources/references
Aufruf: 18.07.2012

[PG AB 12] PostgreSQL: about:
Quelle: http://www.postgresql.org/about/
Aufruf: 17.04.2012

[PG DOC 12] PostgreSQL: 9.1.6 Documentation:
The PostgreSQL Global Development Group:
Quelle:
http://www.postgresql.org/files/documentation/pdf/9.1/postgresql-9.1-A4.pdf
Aufruf: 17.04.2012

[PG FM 12] PostgreSQL: Feature Matrix:
Quelle: http://www.postgresql.org/about/featurematrix
Aufruf: 8.05.2012

[PGL OSI 12] The PostgreSQL Licence (PostgreSQL):
Open Source Initiative,
Quelle: http://www.opensource.org/licenses/postgresql
Aufruf: 18.04.2012

[SC TBFHK 11] SensorCloud: Teilvorhabenbeschreibung FH-Köln

[SQLI FP 12] SQLite: Size Of The SQLite Library:
Quelle: http://www.sqlite.org/footprint.html
Aufruf: 18.04.2012

[SQLI FTN 12] SQLite: SQL Features That SQLite Does Not Implement:
Quelle: http://www.sqlite.org/omitted.html
Aufruf: 18.04.2012

[SQLI LC 12] SQLite: SQLite Copyright:
Quelle: http://www.sqlite.org/copyright.html
Aufruf: 19.04.2012

[SQLI SL 12] SQLite: SQLite Is Serverless:
Quelle: http://www.sqlite.org/serverless.html
Aufruf: 18.04.2012

[SQLI TR 12] SQLite: SQLite is Transactional:
Quelle: http://www.sqlite.org/transactional.html
Aufruf: 18.04.2012

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Anhang

In diesem Abschnitt sind die Anhänge erhalten.

Anhang 1 Java-Programm-System

Für jede Datenbank gibt es ein eigenes Testprogramm. Es gibt jeweils eine Hauptklasse mit der main, eine Klasse die Methoden für die Lese- und Schreibtests enthält und gegebenenfalls ein oder mehrere Klassen für die Verbindung zur Datenbank.

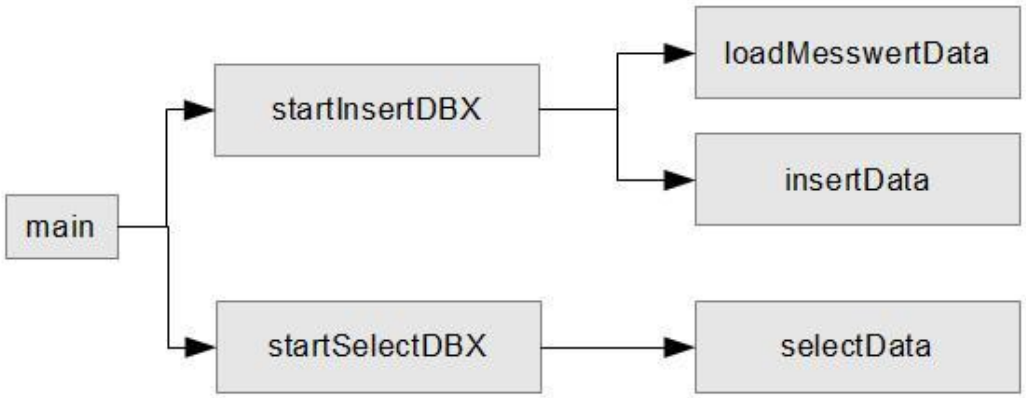


Abbildung 2: Programmübersicht

Jede main in jedem Testprogramm kann die Methode startInsertDBX oder startSelectDBX aufrufen. DBX steht z.B. für SQLiteJDBC. Dann werden entweder die Messwerte geladen (loadMesswertData) und der Insert der Daten gestartet (insertData) oder es werden Daten ausgelesen (selectData).

Es werden in dieser Abbildung nur die relevanten Methoden gezeigt. Weitere wie z.B. die für den Aufbau der Verbindung der Datenbank werden auch verwendet, jedoch hier nicht dargestellt.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages