Evaluation

To judge whether the proposing of the database model and the database application design meet the user requirements in Chapter 4 and Chapter 5, four experiments are performed. The experiments are real-time data collecting, importing data from EDF file format, exporting data from the database to EDF file format, querying data from the database. In addition to the four experiments, a mixed of the experiments together with graphical visualization is also evaluated. When performing a database evaluation, the most important metrics under evaluating are the read/write speed and the size of the database. That is, time to retrieve data and space to store data must meet the user requirements. On the other hand, the database application must ensure no data lost, or data corrupted when storing and retrieving. With respect to mobile platforms, which are usually limited resources and battery driven, the database application must also use the resources in an efficient way. Since a database connection is shared for many activities while the application is used, it needs to evaluate time each activity uses the connection.

The extensibility for future use is clearly presented in the design chapter, and therefore not be evaluated in this section. As discussed in the design chapter, the database design is a platform independent, it can easily be implemented on a bio-signals collector application such as CESAR acquisition tool to minimize the sending and receiving overhead between collectors and the database application server. There are some common experiments when evaluating a database design and database application. The evaluated experiments in this chapter, therefore, inspire applications which want to implement the database design.

The goals of the experiments are to evaluate the feasibility of the database model, and to convince the reader that the functional and non-functional requirements for the database modeling and the database application are satisfied by measuring performance of the read/write on the database, performance of the database application, and storage capacity. In this section, each experiment is evaluated with respect to reasons it is performed, workloads used, evaluated metrics, procedure to evaluate, and the results from the experiment.

A BITalino plugged kit, an Android device, and a CESAR acquisition simulator run on macOS Sierra are using for the experiments. Table T1 presents specifications for the used devices in this chapter.

|  |  |  |  |
| --- | --- | --- | --- |
| Technology | Xiaomi Mi5 | MacBook Retina  15 Late 2013 | BITalino BT  plugged kit |
| OS | Android 7.0 | macOS Sierra | n/a |
| RAM | 3GB | 16GB | n/a |
| Chip | Snapdragon 820  4 core 1.8GHz | Intel core i7  4 core 2.3GHz | MCU with sampling rate 1, 10, 100, or 1000Hz |
| Wi-Fi | 802.11 a/b/g/n/ac | 802.11 a/b/g/ac | n/a |
| Bluetooth | v4.2 | v4.0 | v2.0, range 10m |
| Battery | 3000mAh | 8440mAh | 700mAh |

T1

1. Real-time data collecting experiment

The database application is required to collect and store bio-signal data from CESAR acquisition tool to the designed database model on a mobile device. The CESAR acquisition tool sends data with four different frequencies, that are 1Hz, 10Hz, 100Hz, and 1000Hz. The goal of the acquisition tool is to collect OSA data, hence, a duration of the experiment can be more than 12 hours. The current version of CESAR acquisition tool cannot delivery samples at 1000Hz; the tool stops sending data after 30min-40min. Therefore, a CESAR acquisition simulator is used for sending samples with frequencies 1000Hz or higher. Goals of the experiment are to evaluate the robustness of the database application, database read/write performances, database size when collecting data from many different sources with different frequencies. Since the application is run on a mobile platform where the resources are limited, usage resources such as CPU and battery consumption are also evaluated in this section.

* 1. Experiment workloads

The database application collects samples from multiple sources simultaneously, in which each source sends data at different frequencies. A mount of samples the database need to store at a certain time is depended on the number of current connected sources and the number of channels that belong to each source. In addition, when the frequencies are high, the number of arriving samples in a period time becomes very large; it has a huge impact on the storing process of the database application. Since the application uses batch processing to manage and store incoming samples, a batch size highly influents the performance of the application. The batch size is defined by multiplying the buffer duration with arrival rate. Therefore, the workloads for the experiment are a buffer duration in second, an arrival rate of a source in Hz, a number of channels (total channels from the connected sources).

* 1. Experiment metrics

Evaluation metrics must highlight the goals and the workloads of the experiment. In the real-time data collecting experiment, the application has to parse arrival packages into relevant objects for the database. Therefore, percent CPU usage and power consumption in mAh are considered the metrics used in the evaluation. Since the application uses batch processing to manage arrival data, when the temporary buffer is full, a SQLite transaction is requested for inserting data into the database. While the transaction is performed, it locks the entire database file. Hence, the SQLite usage time in millisecond for different batch size, and the number of insertions per millisecond are another metrics for the experiment. Nonetheless, the application is implemented on the mobile platform where resources are limited, it is good to see how big the database grows in megabytes for different workloads in a specific period time.

* 1. Experiment setup and results

As described above, there are two workload generators that are used in the experiment, they are the CESAR acquisition tool, and the CESAR acquisition simulator. In addition, a workload which is the buffer duration for the application can be adjusted before performing a collecting process. A workload that is monitored (its value is adjusted before a collecting process starts) is called the experimental workload. The other workloads that are unmonitored (their values must be keep the same for each collecting process) are called control workloads. Calculated results after each collecting process are called the responsive metrics.

Table T2 presents the responsive metrics for an experiment where the experimental workload is a buffer duration; control workloads are arrival rate and number of used channels. In this experiment, arrival rate is 100Hz, and all channels of the CESAR acquisition tool are used (6 channels). Likewise, in Table T3, the experimental workload is arrival rate; control workloads are buffer duration (5 second), and all channels of the CESAR acquisition tool are used (6 channels). The number of channels is chosen as the experimental workload in Table T4. In this case, the control workloads are the buffer duration (5s) and arrival rate (100Hz).

For each experiment, a collecting process is iterated three times. The first collecting iteration is 10 minutes; it is because the workload parameters are not changed under the collecting process, hence it is no need to perform long time evaluation for the CPU usage. Percent CPU usage is obtained by using command “adb shell busybox top –d 5”. This command generates an output which contains the information about CPU usage for each application, and the output is printed in the standard output in each five seconds. The outputs from the command are redirected to a text file, and the file are used for making visualization. Listing L1 presents a sample of five seconds fragment output from the command. As presented in the listing, the CPU usage of the application is 25.6%.

viettt$ adb shell busybox top d 5 > File1.txt

[H[JMem: 2720896K used, 39068K free, 0K shrd, 2496K buff, 537K cached

CPU: 27.2% usr 3.6% sys 0.0% nic 67.7% idle 0.2% io 0.6% irq 0.3% sirq

Load average: 6.89 6.25 5.88 3/2638 12713

[7m PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND[0m

11161 769 app\_199 S < 2310m 85.3 0 25.6 {i.viettt.mscosa} no.uio.ifi.viett

585 1 system R < 133m 4.9 1 1.8 /system/bin/surfaceflinger

1521 769 system S 2530m 93.4 0 1.1 system\_server

10096 2 root SW 0 0.0 0 0.3 [kworker/u8:3]

602 1 system S 68152 2.4 2 0.3 /system/vendor/bin/mm-pp-dpps

12592 2 root DW 0 0.0 0 0.3 [mdss\_fb0]

578 1 system S 9424 0.3 0 0.3 /system/bin/servicemanager

10861 2 root DW 0 0.0 0 0.3 [kworker/u8:5]

7 2 root SW 0 0.0 3 0.2 [rcu\_preempt]

2021 769 system S < 2417m 89.2 2 0.1 {ndroid.systemui} com.android.syst

12706 23916 shell R 1416 0.0 3 0.1 busybox top d 5

……

L1

The second and the third iterations are 6 and 12 hours, which are derived from the recommends sleep time from Sleep Foundation [https://sleepfoundation.org/press-release/national-sleep-foundation-recommends-new-sleep-times]. The metrics that are evaluated for these iterations are the power consumption, SQLite usage time, the writing speed (average insertion per millisecond), and the size of the database after the iterations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Responsive metrics  for different time periods | | Experimental workload  Buffer duration | | |
|  | | 5 second | 10 second | 15 second |
| CPU usage  (percent) | 10min | File1.txt | File2.txt | File 3.txt |
| Power consumption  (mAh) | 6 hours |  |  |  |
| 12 hours |  |  |  |
| SQLite usage time  (millisecond) | 6 hours |  |  |  |
| 12 hours |  |  |  |
| Database file size  (Megabytes) | 6 hours |  |  |  |
| 12 hours |  |  |  |
| Avg. nr. insertion per millisecond | 6 hours |  |  |  |
| 12 hours |  |  |  |

T2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Responsive metrics  for different time periods | | Experimental workload  Arrival rate | | | |
|  | | 1Hz | 10Hz | 100Hz | 1000Hz |
| CPU usage  (percent) | 10min | File1.txt | File2.txt | File 3.txt | File4.txt |
| Power consumption  (mAh) | 6 hours |  |  |  |  |
| 12 hours |  |  |  |  |
| SQLite usage time  (millisecond) | 6 hours |  |  |  |  |
| 12 hours |  |  |  |  |
| Database file size  (Megabytes) | 6 hours |  |  |  |  |
| 12 hours |  |  |  |  |
| Avg. nr. insertion per millisecond | 6 hours |  |  |  |  |
| 12 hours |  |  |  |  |

T3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Responsive metrics  for different time periods | | Experimental workload  Number of channels | | |
|  | | 3 | 6 | 12 |
| CPU usage  (percent) | 10 min | File1.txt | File2.txt | File 3.txt |
| Power consumption  (mAh) | 6 hours |  |  |  |
| 12 hours |  |  |  |
| SQLite usage time  (millisecond) | 6 hours |  |  |  |
| 12 hours |  |  |  |
| Database file size  (Megabytes) | 6 hours |  |  |  |
| 12 hours |  |  |  |
| Avg. nr. insertion per millisecond | 6 hours |  |  |  |
| 12 hours |  |  |  |

T4

* 1. Discussion

<WHEN I HAVE THE RESULTS, I WILL WRITE THIS SUBSECTION: HOW TO PARSE CPU USAGE FILE, HOW TO PLOT DATA….THEN, AS PRESENTED IN FIGURE …>

1. EDF importing experiment

The database model is designed in the way that opens for all bio-signals, and can store data from other database sources if the sources support EDF exporting. In other words, the data base application is required to import data from EDF and EDF+ file formats. Goals of the experiment are to evaluate database writing performance and usage resources such as CPU and battery consumption, especially, when the application reads files simultaneously.

* 1. Experiment workloads

The database application reads sample data from multiple files simultaneously. Ideally, each sample from the files is parsed to an object in memory, then inserted into the database. However, each INSERT statement in SQLite has its own transaction, and SQLite can only do a few dozen transactions per second. The samples are, therefore, grouped into a transaction before performing a SQLite insertion. The transaction locks entire database file when it performs inserting. Hence, if the sample group is too large, it can cause either memory problem, or poor performance since other transactions have to wait for it. Therefore, the workloads for the experiment are a size of the sample group and the number of file read.

* 1. Experiment metrics

Metrics that are chosen to highlight the goals and workload of the experiment are the percent CPU usage, the power consumption in mAh, the size of a EDF file in the database, total time the experiment holds the shared SQLite connection, and how many samples are inserted to the database for each hold-time. The percent CPU usage and the power consumption need to be evaluated since the application need to parse the samples in the EDF file format to objects that can be stored into the database. In addition, the number of samples are inserted for each hold-time can be used for evaluating the writing performance of the database.

* 1. Experiment setup and results

To generate workloads for the experiment, a function from Physionet which named mit2edf is used for exporting bio-signal data from Physionet databases to EDF file format. As presented in the manual page of the function, mit2edf creates an EDF file that contains the same data as the input files which are in form of Waveform Database format (header and signal files). However, the function is included in WFDB software package from Physionet, the user must install the package first. There are three options when using the function which are printing a brief usage summary, choosing a name for the exported edf (the exported file has the same name as the imported record), and printing debugging output. Listing L2 presents how to use the function to convert a01 record to EDF file format without options and with options on a Linux/Unix terminal.

(no options)Terminal $mit2edf –r a01

(options)Terminal $mit2edf –r a01 –o phy1.edf –h –v

L2

Workloads that are used for the experiment are a01.edf, a02.edf, and a03.edf files, which are generated from a01, a02, and a03 records from CinC2000 data sets. Since the implementation for the EDF-reader uses the same algorithm (batching processing) as real-time importing, there are two factors that influent the goals of the experiments. They are the number of simultaneously EDF files read, and the buffer size (batch size). Table T5 presents the results for the experiment where the experimental workload is the number of simultaneously EDF files read. In this experiment, the batch size is 50000 samples, and is kept the same for three iterations. Table T6 presents the results for the experiment after three iterations with batch size 50000, 100000 and 150000 samples respectively. Since the number of file read is not the experimental workload in each iteration, the number of file read and the used files must be the same for each iteration. a01.edf is chosen as the control workload for the experiment.

|  |  |  |  |
| --- | --- | --- | --- |
| Responsive metrics | Experimental workload  Number of files simultaneously read | | |
| 1 | 2 | 3 |
| CPU usage  (percent) | File1.txt | File2.txt | File 3.txt |
| Power consumption  (mAh) |  |  |  |
| SQLite usage time  (millisecond) |  |  |  |
| Database file size  (Megabytes) |  |  |  |
| Avg. nr. insertion per millisecond |  |  |  |

T5

|  |  |  |  |
| --- | --- | --- | --- |
| Responsive metrics | Experimental workload  Number of samples in batch | | |
| 50000 | 100000 | 150000 |
| CPU usage  (percent) | File1.txt | File2.txt | File 3.txt |
| Power consumption  (mAh) |  |  |  |
| SQLite usage time  (millisecond) |  |  |  |
| Database file size  (Megabytes) |  |  |  |
| Avg. nr. insertion per millisecond |  |  |  |

T6

* 1. Discussion

<WHEN I HAVE THE RESULTS, I WILL WRITE THIS SUBSECTION. AS PRESENTED IN FIGURE …>

1. EDF exporting experiment

The database application is required to export data to EDF file format for sharing or for backup data purposes. Since each record in the database is defined for a specific channel of a source (a.k.a channel record), and if records have the same patient id, physician id and timestamp, they are considered to be belonged to the same collecting process (a whole record). It is because a patient can use multiple sensor platforms to collect data. By choosing records with the same patient id, physician id and timestamp, a whole record can be retrieved. It is freely to choose which channel records to be included in the exported record; it is because sometimes a user needs data from some specific channels. EDF file format requires that a whole record need to be divided into small data records. A duration of the small data records is freely chosen. However, a data record size does not exceed 61440 bytes[http://www.edfplus.info/specs/edfplus.html]. In the implementation of the database application, a data record is defined as a buffer in memory, and this buffer is flushed to the exported file when it is full. Hence, the data record size is also the buffer size which is defined by a summary of all channel buffer sizes. A channel buffer size is defined by multiplying the data record duration with the frequency of a channel.

bufferSize=\sum\_{i = 1}^{n}\frac{duration}{frequency\ channel(i)}

Goals of the experiment are to evaluate the read performance of the database and resource usage such as percent CPU usage and battery consumption. In addition, time usage for exporting data is also evaluated in this subsection.

* 1. Experiment workloads

As presented in the high level design of the database application, the application continuously queries part of a record into a buffer, then writes the buffer to the exported EDF file. The application requests a database access for each selected record. Therefore, if the number of selected records for exporting is large, the application needs to access the database multiple times. In addition, although a large buffer may cause a memory problem if the memory of the used mobile device is small, it reduces the number database access of the application. In other word, a small buffer causes overhead for database accessing.

Therefore, experimental workloads for the experiment are the number of exporting records and the duration of a data record (a.k.a the buffer size).

* 1. Experiment metrics

A record can be very large, since it can be collected in long time. It takes time to export data, especially when a record is large, and a buffer size is small. In this case, the database application needs to query data multiple times to avoid memory corruption since the record cannot be kept in memory for exporting. Therefore, beside metrics such as CPU usage, power consumption, database read performance, and shared SQLite connect time usage as discussed in the experiments above, time it takes to export a record to EDF file format is also one of the chosen metrics in the experiment.

* 1. Experiment setup and results

Since the experimental workloads for the experiment are the number of exporting records and the duration of a data record, by changing values of the experimental workloads, the EDF exporting experiment can be evaluated by assessing the results of the experiment.

Table T7 presents the results for the experiment where the experimental workload is the number of exporting record. The experiment is performed with 5, 10, and 15 records for each iteration. The data record duration that is used in each iteration is five seconds. Table T8, on the other hand, presents the results for the experiment with different data record durations, which are one second, five seconds and ten seconds. The number of exporting records is 10, and is the same for each run in the experiment.

|  |  |  |  |
| --- | --- | --- | --- |
| Responsive metrics | Experimental workload  Number of exporting records | | |
| 5 | 10 | 15 |
| CPU usage  (percent) | File1.txt | File2.txt | File 3.txt |
| Power consumption  (mAh) |  |  |  |
| SQLite usage time  (millisecond) |  |  |  |
| Time to export  (millisecond) |  |  |  |

T7

|  |  |  |  |
| --- | --- | --- | --- |
| Responsive metrics | Experimental workload  Duration of a data record in EDF file | | |
| 1s | 5s | 10s |
| CPU usage  (percent) | File1.txt | File2.txt | File 3.txt |
| Power consumption  (mAh) |  |  |  |
| SQLite usage time  (millisecond) |  |  |  |
| Time to export  (millisecond) |  |  |  |

T8

* 1. Discussion

1. Querying data experiment

When processing a SQL query, a database management system (DBMS) goes through at least three steps. The first step is to parse SQL query into a data structure that can be processed by the DBMS. In this step, the syntax and semantic of the query are checked to determine if the SQL statement is syntactic validity, and if the statement is meaningful, i.e. whether the columns and tables are existed in the database. The second is to optimize the query. That is, by using a set of equivalence rules, the query optimizer can generate many equivalent plans in which a cost is assigned for each plan. The plan that has a lowest cost is taken to the third step which is to execute the query plan Although the SQL query is optimized before querying, the complexity of the SQL query and the results from the query are not changed.

Goals of the experiment is to evaluate the resource usage and performance of the database application with respect to the complexity of the queries, and the size of the results.

* 1. Experiment workloads

In SQLite, all queries are simple select statement. A simple select statement is often processed via four steps[https://sqlite.org/lang\_select.html#crossjoin].

* From clause processing: The input data of the statement are specified in this step.
* Where clause processing: The expression in the clause helps to filter the return data.
* Group by, having processing: groups of data are calculating and aggregating with respect to the filter expression in having.
* Distinct / ALL keyword processing: result rows are return (duplicate rows are removed if “distinct” is specified)

A compound query statement is defined via connecting multiple simple select statement by using UNION, UNION ALL, etc.

Workloads for the experiment are therefore depended on the FROM clause, and whether statements are compound statements. As presented in SQLite [https://sqlite.org/lang\_select.html#crossjoin], all join-operators are based on Cartesian product of the left hand side (N rows) and right hand side (M rows) of the datasets. The results from the join are therefore NxM rows. In other words, the complexity when joining two table is O(MxN). The complexity is increased by the number of tables that are stated in the FROM clause. Moreover, if the result from the query is too large to hold in the memory, the DBMS musts pass the database multiple times (multiple-passes).

|  |  |  |
| --- | --- | --- |
| Table | Number  of columns | Approximate records |
| SensorSource | 3 | < 100 |
| Patient | 7 | < 100 |
| Physician | 4 | < 100 |
| Clinic | 5 | < 100 |
| Person | 8 | < 100 |
| Channel | 11 | < 1000 |
| Record | 10 | < 1000 |
| RecordAnnotation | 2 | < 1000 |
| Annotation | 5 | < 1000 |
| Sample | 3 | > 1 million |

Datasets of the database T9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Query | Table | Input size | | Output size | |
| MB | # records | MB | # records |
| 1 | Channel |  |  |  |  |
| 2 | Sample |  |  |  |  |
| 3 | All tables |  |  |  |  |
| 4 | All tables |  |  |  |  |

SQLite queries on table Sample T10

As presented in T9, the heaviest table is the Sample table, and together with the complexity of queries, they are therefore considered the main workloads of the experiment. There are four queries that are used for evaluating the experiment; the queries are described as follow:

* Query 1: A relatively small dataset is scanned (a.k.a simple query with small result). Any tables presented in Table T9, as long as not the table Sample, can be chosen for this query.
* Query 2: A relatively large dataset is scanned (a.k.a simple query with large result).
* Query 3: All presented tables in Table T9 are joined, then an aggregate function, i.e. count, sum, max, etc., is applied to have a small result (a.k.a complex query with small result).
* Query 4: All presented tables in Table T9 are joined, no filters are applied in this query (a.k.a complex query with large result).
  1. Experiment metrics

Metrics that are used for evaluating the experiment must highlight the resource usage and performance of the database application when executing different queries with different complexity. Therefore, the metrics that are chosen for the experiment are percent CPU usage, power consumption in mAh, and response time for queries.

* 1. Experiment setup and results

The described queries in the subsection workload of this experiment are translated into SQLite code as follow:

* Query 1: The whole table Channel is chosen to be scanned.  
  select \* from Channel
* Query 2: Since the table Sample is the largest, it is chosen for this query.  
  select \* from Sample
* Query 3: This query counts number of rows return from Cartesian product all tables.  
  select count(\*) from SensorSource, Patient, Physician, Clinic, Person, Channel, Record, RecordAnnotation, Annotation, Sample
* Query 4: This query return all rows from Cartesian product all tables.  
  select \* from SensorSource, Patient, Physician, Clinic, Person, Channel, Record, RecordAnnotation, Annotation, Sample

Table T10 presents query statistics after executing the queries, while Table T11 presents the results for the experiment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Responsive metrics | Experimental workload | | | |
| Simple query-  small result | Simple query-  large result | Complex query-  small result | Complex query-  Small result |
| CPU usage  (percent) | File1.txt | File2.txt | File 3.txt | File 3.txt |
| Power consumption  (mAh) |  |  |  |  |
| Executing time  (millisecond) |  |  |  |  |

* 1. Discussion

1. Visualization with mixed tasks experiment

It is possible that a user collects data from multiple sensor sources to the database, and plots the collected data on a graphical view while there is an exporting process is running. In other words, the application usually performs multiple tasks simultaneously rather than finishes a task before beginning the others.

Goals of the experiment are to evaluate database read/write performances, database size when collecting data from many different sources (both EDF files and CESAR acquisition tool), and resource usage such as CPU and battery consumption.

* 1. Experiment workloads and metrics

Workloads for this experiment are borrowed from the experiments above, which are buffer duration, arrival rate, number of collecting channel (these workloads come from the real-time experiment), number of file simultaneously read, number of sample in a batch (these workloads are from EDF importing experiment), number of exporting records, duration of a data record in the exported EDF file (from the EDF exporting experiment), and the complexity of queries a user want to execute.

Metrics that are chosen in this experiment must highlighting the database application performance, the size of the database, and resource usage. Therefore, percent CPU usage, power consumption in mAh, database size in megabytes, and SQLite time usage for each task are chosen as metrics for the evaluation.

* 1. Results and discussion