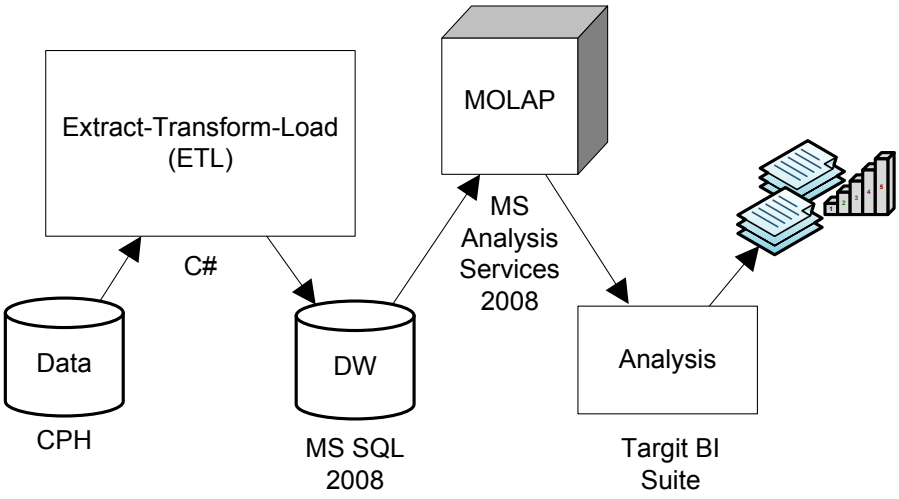
A Data Warehouse Solution for Analysis on Indoor Tracking Data

Department of Computer Science @ Aalborg University



Group d521a
Jonas T. Hansen
Stig Jørgensen
Simon Nicholas M. Tinggaard
Rune L. Wejdling

Department of Computer Science

Title:
A Data Warehouse Solution for Analysis on Indoor Tracking Data

Project period:
DAT5:
1st of September to 19th of December, 2008

Project Group:
Computer Science, d521a

Members:
Jonas T. Hansen
Stig Jørgensen
Simon Nicholas M. Tinggaard
Rune L. Wejdling

Supervisor:
Torben Bach Pedersen
Hua Lu

Copies:
7

Pages in Report:
38

Pages in Appendix:
1

Pages in total:
39

Abstract:

This project concerns the design and implementation of a data warehouse for analysis on tracking data collected at Copenhagen Airports A/S. The project is done in cooperation with BLIP Systems A/S who has a Bluetooth based tracking system implemented in Copenhagen Airports A/S. First, we introduce a number of technologies that can be applied for indoor tracking, then we describe how BLIP Systems A/S stores the collected tracking data and propose a data warehouse design that enables answering some business intelligence related questions provided by BLIP Systems A/S. As a part of the Extract Transform Load, some data cleansing has been performed on the source data. Additionally, we introduce two case-specific algorithms, *FlyerFrequencyCalculation*, used for calculating measures and *BounceDetection*, which normalizes the tracking data. By utilizing Targit BI Suite, we address some of the questions presented and provide answers through graphs. Lastly, we conclude on the project and suggests directions for further work.

Contents

1 Introduction

It is becoming increasingly more common for mobile devices to have at least a positioning device or network module. This fact combined with the continuously growing number of mobile devices and Location Based Services (LBS) [1] [8], has resulted in various advancements within the area of tracking and monitoring of mobile devices, and thereby also the users. Most LBS utilize Global Positioning System (GPS) to track the users and rely on the users submitting their location periodically.

Using GPS modules in mobile devices for tracking in an outdoor environment provides a good estimation of the position of the device, but it is too inaccurate for indoor tracking. Instead Bluetooth [3] and other wireless networks or Radio Frequency Identification (RFID) can be used to obtain positions in indoor environments. However, these solutions require an underlying infrastructure within the immediate area to work, i.e. a number of strategically placed Bluetooth access points or RFID readers. An advantage of these tracking methods is the ability to track devices without the need of special software running on the devices, e.g., it is possible to track and monitor the location of a device if it has its Bluetooth device enabled. Because a Bluetooth infrastructure supports creating a Personal Area Network (PAN) between the Bluetooth access points and the tracked devices, it can also provide users with LBS, such as navigation guidance, mobile-mapping and location specific content, by pushing information to the devices. To support this, extra software installed on the devices could be required.

The motivation behind the tracking of devices is often related to Business Intelligence (BI) [9]. The topic BI is very broad and covers a variety of subjects, where the common denominator is a transformation of data into knowledge. This report focuses primarily on data analysis. Many organizations can benefit by redefining their business processes according to the knowledge derived from BI related analysis.

The knowledge that can be derived from tracking people within an organization's physical environment can be used to increase its revenues. In the case of shopping malls or airports, this knowledge of people's behavior can be put to use when deciding the physical placement of shops and services. Furthermore, the mix of tenants within the mall or airport can be optimized using the data gathered. Other business decisions, such as rent per square meter based on the location of the store, can be improved, using the knowledge gathered to assist the decision making. Supermarkets can use analysis of the flow of the people in the store to decide where to place their products, e.g., they might discover that 95% of the people seen at section A, at some time later in their visit is also seen at section B. This knowledge could be used for placing advertisements in section A, targeting products in section B, to influence the shopper's shopping behavior.

This report, in cooperation with BLIP Systems A/S [16], focuses on defining and implementing a data warehouse and analyzing data gathered from a Bluetooth network infrastructure. To provide an efficient and effective analysis solution, we propose the system presented in Figure 1. The architecture shows how our Extract, Transform, and Load (ETL) application extracts data from the source data set and pushes the transformed data to an MS SQL Server 2008 RDBMS. MS Analysis Services 2008 is used to pull the data from the RDBMS,

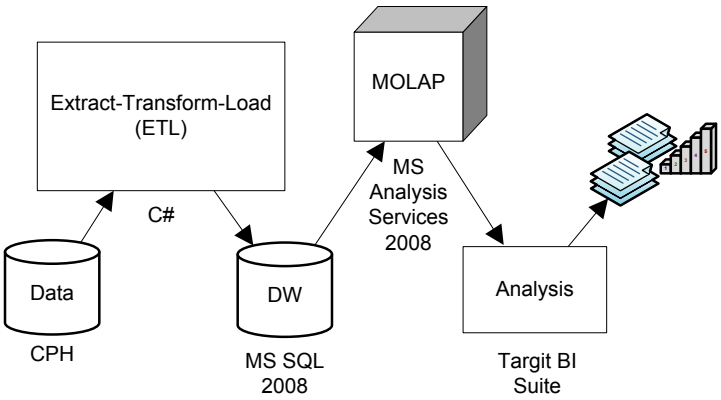


Figure 1: Proposed system architecture.

define multidimensional cubes, and store them physically on disk in a proprietary format. Targit BI Suite is used to analyze the data by slicing and dicing the cubes and presenting graphical results.

In order to answer a series of BI related questions, a specific site, namely Copenhagen Airports A/S, is chosen. For the remainder of this report we will refer to BLIP Systems A/S as BLIP and Copenhagen Airports A/S as CPH.

BLIP is working in collaboration with CPH's administration on the SPOPOS project [1] to develop a tracking and LBS system. The goal of SPOPOS is, amongst others, to offer a range of services to the passengers, helping them make their flights in time and thereby reducing the number of delayed flights. This is done by tracking the passengers and send them a text message or give them a call when it is time for them to board. If the gate personnel can tell by a passengers position in the airport that the passenger will not be able to make the flight, they make arrangements for the passengers baggage to be taken off the flight. Before the SPOPOS project was launched, tracking of people in the airport was done manually using video surveillance. The manual method allows the airport to track up to 200 people a day, without the ability to recognize recurring visitors. With the SPOPOS project, and BLIPs implementation of automatic tracking, even across subsequent visits, this number is up to 6,500 people a day. Another goal is to enable the administration to compare the current visitor movement in the airport with historical data. Some BI related questions relevant for the airport administration based on historical data would include the following.

1. *How often have visitors been seen in the airport?*
2. *How is the visitor load distributed over the weekdays?*
3. *How much time do visitors spend at the different locations in the airport?*
4. *How is the distribution of time spent at a specific location?*
5. *How is the distribution of time spent at different times of the day?*

These questions all refer to the time spent by visitors in the airport or the amount of visits to the airport. More complex questions would include flow analysis of the visitors in the airport to detect movement patterns and to classify different visitor types. This topic is left for future work.

The structure of this report is as follows. Section 2 introduces existing technologies for tracking of moving objects and presents a comparison of these. Section 3 covers how BLIP tracks objects and, specifically, how it is implemented in CPH. A data warehouse design is presented in Section 4 to answer the questions proposed by BLIP. Section 4.1 describes which dimensions are created. In Section 4.2, the creation of the OLAP cube is described. Once the data warehouse is in place, the process of ETL is described in Section 5. Section 5.1 presents the selection criteria from which the source data is selected. A set of case specific problems were encountered and two algorithms, namely *Flyer Frequency Calculation* and *Bounce Detection* are proposed in Section 5.2 and 5.3, respectively, to solve the problems. Section 6 presents experimental results and analysis of the BI related questions, performed on the BLIP data set. Lastly, we conclude on our work and elaborate on future work in Section 7 and 8.

2 Technological Background

This section introduces BLIPs tracking method as well as two other projects whose focus areas are similar to BLIPs. These are introduced in order to provide sufficient grounds for evaluating and comparing different ways of collecting tracking data in an indoor environment. We will compare and evaluate the possibilities of each technology with respect to the needs of our specific case. All three projects involve tracking devices indoors, as well as transforming the gathered data into knowledge that can be used to support business decisions within an organization or provide users with LBS. The major difference between the projects is the way data is collected - one project tracks the Temporary Mobile Subscriber Identity (TMSI) in mobile phone signals, the second tracks Radio Frequency Identification (RFID) tags, and the third tracks Bluetooth devices.

2.1 Temporary Mobile Subscriber Identity

A company called Path Intelligence Ltd. [11] has specialized in tracking mobile phones by listening in on the network communication between the mobile phones and base stations using strategically placed monitoring units. They use an identification number contained in the communication packages called the Temporary Mobile Subscriber Identity [14] to isolate and track the individual phones within the area. TMSI is an ID assigned to a cell phone when it enters a new location (base station) on the GSM and UMTS network[1]. The TMSI is periodically transmitted between the device and the base station and is also used when creating communication channels for receiving or making calls. The ID is temporary and changes over time and location. Path Intelligence Ltd. has developed a method allowing them to track the devices and present results from

[1] Global System for Mobile communications and Universal Mobile Telecommunications System

data analysis, such as shopper flow and densities in a mall, to the customer. An overview of the pros and cons of this approach can be seen in the following:

Pros: Because the system is able to track any cell phone as long as it is turned on, the penetration rate is very high. The range of a cell phone is rather large up to 3-20 miles depending on the amount of interference and obstructions in the area, therefore the number of nodes needed to monitor signals in the airport can be kept at a minimum depending on the accuracy wanted. The system provided by Path Intelligence has an accuracy of down to 1-2 meters using triangulation [10].

Cons: Because the TMSI is only broadcast periodically, e.g. when the cell phone is being used or changes base station due to low signal strength, the time between updates varies from a couple of seconds up to a number of minutes. Additionally, because the TMSI is changing continuously, there is a possibility that the tracking sessions end prematurely, thereby obscuring the data collected. The system provided by Path Intelligence provides no means of communicating to the tracked devices. Because the system relies on triangulation for pinpointing the position of a device the complexity of the tracking system greatly increases.

2.2 Radio Frequency Identification

The second project of interest to us uses RFID technology for tracking. Lyngsoe Systems A/S [7] is one of the worlds leading suppliers of solutions for complex RFID based logistic systems, e.g. baggage tracking in airports and parcel tracking in postal services. Currently, Lyngsoe is involved in the same project as BLIP, namely SPOPOS [1], in which Lyngsoe supplies the technology for RFID based tracking in CPH. Lyngsoe uses active RFID tags with a much higher signal strength, as opposed to passive tags with a low signal strength that are depending on the RFID readers to power them [13]. The active tags have been chosen because they have a much longer range and thereby covering the areas wanted to be monitored in the airport. The tags used by Lyngsoe are set to have a range of approximately 30 meters. An overview of the pros and cons of using active RFID tags in this approach can be seen in the following:

Pros: An active RFID tag has a unique identifier which can be continuously tracked with approximately 1 update per second as long as its signal reaches an RFID reader. The range of an active RFID tag is up to 200 meters, in Lyngsoes case, the range has been set to 30 meters to better suit the environment [12]. In order to save battery life and reduce the number of updates sent to the system, Lyngsoe has developed tags that only broadcast when moved.

Cons: Because the system uses costly active RFID tags, tags must be handed out and linked to individuals or otherwise be available to the passengers and collected again when leaving the airport. This requires passengers as well as airport staff to take action for the system to work, thereby lowering the penetration of the system. The system provided by Lyngsoe provides no means of communicating with the tracked individuals. To implement this functionality would require the RFID tag to be integrated in a device such as a PDA, mobile

phone or similar. In order to perform analysis of a specific passenger's subsequent visits, information about the passenger has to be collected and the tags have to be registered to the passenger when handed out.

2.3 Bluetooth

BLIP use Bluetooth technology for their tracking system in the SPOPOS project, in the form of access points scanning Bluetooth enabled devices in range for their unique Bluetooth addresses and other metadata. Bluetooth devices are very common in mobile phones, PDAs and laptops. Bluetooth transceivers are available in three classes, defining the maximum power output and thereby range of the signal. A class 1 transceiver has the highest power output and with it the longest range of up to 100 meters. Class 2 and 3 devices have a range of up to 10 meters and 1 meter, respectively [15]. The range of class 2 and 3 devices can be slightly extended due to higher sensitivity by using class 1 access points. An overview of the pros and cons of this approach can be seen in the following:

Pros: Most mobile phones come with Bluetooth modules, meaning that the best case penetration is nearly as high as for TMSI, but because they have to be activated to be seen, there is a moderate penetration rate when using this technology for tracking. BLIPs system queries for unique devices once per second. The Bluetooth system provided by BLIP is ready for communicating location and context specific content on the network.

Cons: According to SPOPOS the range of the Bluetooth tracking system in CPH is only up to 20 meters. The short range requires many access points to be installed in order to cover the area. If the system should be upgraded to support triangulation, many more access points would be required. Due to the fact that most Bluetooth transceivers in mobile devices are class 2 transceivers and the signal output of different vendor Bluetooth modules varies greatly, their range is limited to approximately 10-30 meters with a precision of about 5-10 meters.

2.4 Comparison

The three different approaches to tracking described in the previous sections allows us to compare the strengths and weaknesses of each approach in relation to the airport tracking case. This is shown in Table 1.

	TMSI	**RFID**	**BT**
Penetration	High	Low	Medium
Range	Long	Short	Short
Effort required to implement 2-way communication	N/A	High	Low
Update frequency	Infrequent	Frequent	Frequent
Flexible updates	÷	✓	÷
Unique tracking	÷	✓	✓

Table 1: Technology comparison.

As can be seen in the table, the highest level of penetration is achieved using the TMSI for tracking. However, RFID, as well as Bluetooth, can, in the case of an airport, achieve an equally high level, by providing services that adds positively to a passengers travel experience. In the case of RFID this would have to include a solution to the problem of communicating information to the passengers. With respect to the Bluetooth solution, useful services could motivate many users to turn on the Bluetooth transceiver on their mobile phone [1]. The limited number of nodes and the high precision offered by the TMSI solution would be desirable, but the TMSI solution's update rate is no match to that of the RFID or Bluetooth solutions. In our case, a delay of e.g. 5 minutes in the update of a position, is not suitable for tracking within a small area like a airport. The flexible updates supported by Lyngsoe's RFID tags means that the system needs to process fewer updates per second than BLIPs system does. The short range and low precision provided by RFID and Bluetooth is one of the major drawbacks opposed to TMSI, but this is made up for by the ability to uniquely and continuously track passengers, even on subsequent visits, and the ability to implement communication solutions.

3 The BLIP Systems Data Set

This section describes how BLIPs tracking data set is recorded and stored. It introduces some quantitative measures about the data set and devices tracked, as well as some of the irregularities within the data set. Lastly, it introduces some terms relevant when analyzing the data, which will be used for the remainder of this report.

3.1 Data Collection

BLIP has developed a system that collects data of the movement of passengers in airports. This is done utilizing strategically placed Bluetooth access points, hereafter referred to as access points, that monitors a given area. Figure 2 shows BLIPs system architecture. A central monitor-server polls each of the access points every second. Each access point replies to the poll by returning a list containing all devices in the vicinity including various device information.

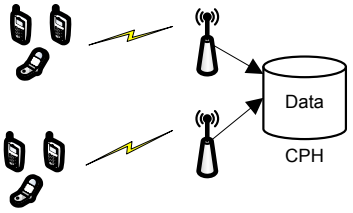


Figure 2: BLIP system architecture.

The Bluetooth device information retrieved is a unique Bluetooth address, signal strength, class of device, device manufacturer and model. Class of device is a 3 byte variable, that indicates which type of device that has been tracked, e.g. camera, mobile phone or smart phone. Signal strength stores the Received

Signal Strength Indicator (RSSI), which is a measurement of the signal strength in a received radio signal. By logging the first and last time each device is detected by an access point, the entry and exit time for the devices are obtained. To know when the device was closest to an access point, BLIP store the peak time[2] of the device, as well as signal strength at the peak time, this also gives an indication of how close a device was to the access point. The signal strength, peak time, entry time, exit time, Bluetooth device information and access point information is continuously logged to a database.

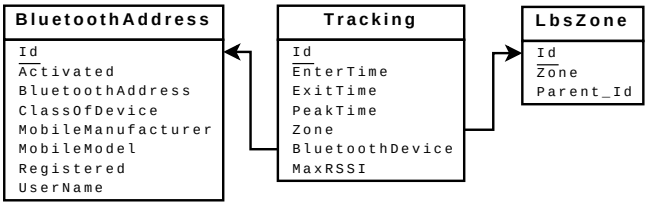


Figure 3: BLIP source schema.

The table configuration of the database is based on the data schema seen in Figure 3. The data schema consists of a main table, `Tracking`, and two supporting tables, namely `LbsZone` and `BluetoothAddress`. The `tracking` table contains `EnterTime`, `ExitTime`, `PeakTime`, and `MaxRSSI` as well as two foreign keys to the supporting tables. `BluetoothAddress` includes more data than the name indicates, as it also contains miscellaneous data retrieved by the access points. BLIP has prepared to include additional personal metadata through the `Registered` and `UserName` attributes. However, BLIP is not using these values in the data set. The name of a zone is stored in the `Zone` attribute in the `LbsZone` table. The attribute `Parent_Id` is used to describe a parent-child relationship between zones, which is used to define zones as a tree-structure with arbitrary height.

3.2 Copenhagen Airport Implementation

BLIP have this tracking solution implemented in CPH where they collect up to 500.000 tracking records per day, distributed over up to 6.500 unique devices. Figure 4 shows a part of the CPH access point setup. The airport is physically divided into three areas, the land area represented with the light grey color on Figure 4, the fly area represented with the dark grey color, and a security area connecting the two areas. E.g. the security area and the immediate area surrounding it is covered by access point 16, 17 and 18. The issue of dividing the airport into different areas will be addressed further in Section 4.1. A map over the entire BLIP access point setup can be seen in Appendix A.

The data set provided by BLIP is sampled over a period from 04/23/2008 to 10/16/2008, registering 21 million tracking records distributed on 310.000 unique devices sampled from 25 access points. However, problems occurred with the system resulting in an incomplete data set. The majority of the 25 access points were running as intended from 04/23/2008 to 05/23/2008. The

[2] Timestamp that indicates when the signal was strongest.

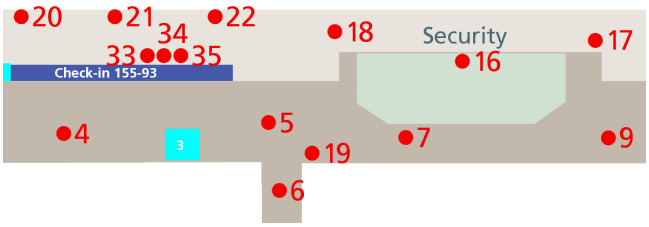


Figure 4: Extract of the access point setup.

remaining period, from 05/23/2008 to 10/26/2008, had 12 operational access points in average.

In the CPH access point setup, the access points do not provide complete coverage of the airport, which is partly to limit the amount of data to process and because only some areas are relevant to monitor. The access points are placed in so called *hot spots*, areas of particular interest. These hot spots are typically chosen to be areas where most passengers must pass by at some time, such as entry points, exit points, bottleneck passages, gates and security checks, and areas where passengers spend time, such as waiting areas, queuing areas, shopping areas and other areas that might have specific interest to the airport administration.

When tracking passengers movement in an airport, we need to define some measurements indicating the amount of time spent in the airport. A simple measure is to look at the passengers *dwell time* in different areas of the airport. BLIP defines *dwell time* as $t_{exit} - t_{enter}$, where t_{enter} is the time a device is first detected near an access point and t_{exit} is the last detection of that device before it moves to another access point.

In order to take the areas that are not covered by access points into account we introduce the notion of *idle time*, i.e. the time a passenger spends in between two access points. We define *idle time* as $t_{enter} - t_{exit}$, where t_{enter} is the enter time at the current access point and t_{exit} is the exit time from the previous access point.

The data schema BLIP is using to store the data set is simple and is not normalized, so it contains a high degree of redundancy. With the right indexes it can be used to answer simple statistical queries within acceptable time, e.g *What is the average dwell time of visitors in year 2008?*. The airport is interested in learning more about the movement of devices within the airport in terms of both *dwell-* and *idle time*, however these are facts that are not stored in the current schema. While the calculation of *dwell-* and *idle time* are trivial for each tracking record, it is not possible to write efficient queries on these measurements and how they change over time.

We propose a multidimensional schema in Section 4 which joined with OLAP (Online Analytic Processing) [6] tools can provide fast results to these more advanced historical analysis queries. An example of an advanced query could be *What is the average dwell time of visitors on weekends compared to weekdays?*.

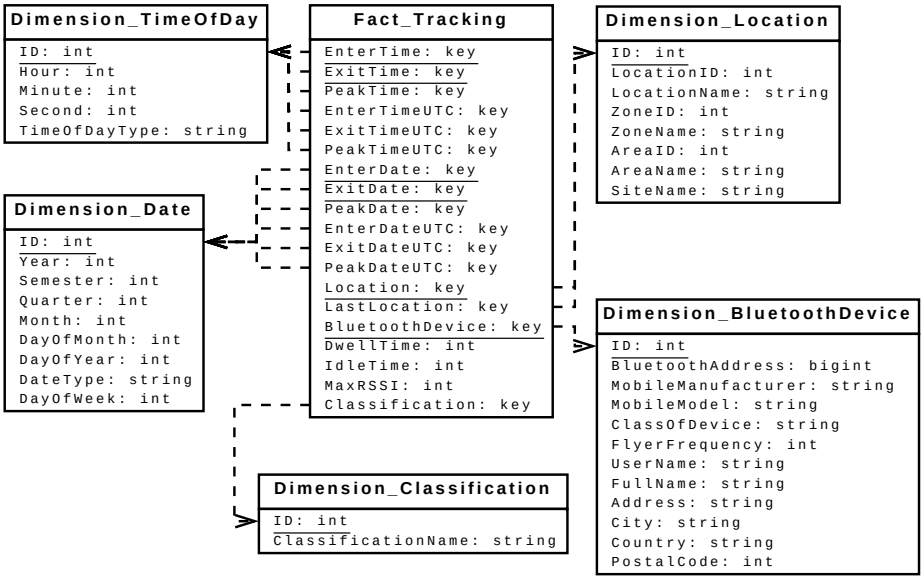


Figure 5: The proposed data warehouse schema.

4 Data Warehouse Design

To support Business Intelligence analysis, and thereby answer the questions introduced in Section 1, we propose the data warehouse design shown in Figure 5. The proposed multi-dimensional database schema is a star schema with one fact table and five dimension tables. The star schema is less complex compared to a snow flake schema where the dimensions are partly normalized. In this case, the storage saved by normalizing the dimensions is not worth the performance penalty obtained from the multiple joins needed to query the dimensions in a snowflake schema. In Section 4.1, the different dimensions are described, after which the fact table is introduced to link all the dimensions to the facts and calculated measures in the fact table. As seen in Figure 5, the date and time has been split up into two dimensions in order to save records compared to a combined dimension table. If date and time were modeled in one table there could be over 86400 records for each day. In a year that would give over 31 million records in the dimension which could result in a slow query performance. Another reason for splitting up the two dimensions is that queries are normally performed on either a date basis or a time of day basis.

As seen in the Figure 5, all the dimension tables have their own surrogate key, `ID`, which is common practice used to give dimensions a unique key, and to prepare the design for slowly changing dimensions [6]. The rest of the attributes, e.g. version timestamps and expiration timestamp, needed to handle slowly changing dimensions, has not been implemented since it is not relevant in the current case.

4.1 Dimension Descriptions

The following dimensions are made to suite a star schema layout and to support the analysis needed to answer the proposed BI related questions.

Date and TimeOfDay Dimensions

The `Date` dimension consists of the three attributes needed to describe a date hierarchy: `Year`, `Month`, and `DayOfMonth`. To make a more detailed gradation the dimension includes the following attributes: `Semester`, `Quarter`, and `DayOfYear`. `DayOfWeek` is included as a one level hierarchy to specify the specific days of the week. The `Semester` attribute is often known as *HalfYear*, and indicates if it the first or last half of the year. The `DayOfYear` attribute enables query on a specific day or range of days. Moreover `DayOfYear` attribute can be used to identify specific weeks of a year, without having the week attribute in the dimension. The `DayOfWeek` makes it possible to query on a specific weekday and the last attribute `DateType` enables classification of days, e.g. *holiday*, *weekend*, *weekday*, etc.

The time dimension called `TimeOfDay` consists of the three attributes needed to specify a time hierarchy down to a specific second: `Hour`, `Minute`, and `Second`. The last attribute, `TimeOfDayType` enables classification of a specific time of day as, e.g. *morning rush hour*, *afternoon rush hour*, etc.

Location Dimension

In order to refer to different points of interest in the airport, a hierarchy of four levels is introduced to handle the various degrees of detail. The lowest level in the hierarchy is denoted as a `location`, which describes the region covered by an access point, which is retrieved from the `Zone` table in the BLIP data set. A number of `locations` form what we denote as a `zone`. An `area` consists of an amount of `zones`. Lastly, the highest level of granularity is denoted as a `site` and covers all `areas` within an airport. The dimension `Location` is used to model the location in which a tracking record was made. This location dimension is a four level hierarchy as shown in Figure 6. The different layers in the hierarchy are represented by an `ID` and a `Name` attribute. The `ID` is included to make it possible to trace back to the source data set. The source data has a recursive parent child structure that gives the possibility for unlimited layers in their hierarchy. We limited our hierarchy to four levels, since the unlimited layers is not needed and it gives less complexity with four levels in the hierarchy. The highest layer in the hierarchy does not contain an `ID`, since this attribute describes the site at which the data is collected from, e.g. CPH, and is not present in the source data.

BluetoothDevice Dimension

The `BluetoothDevice` dimension is used to model the Bluetooth devices tracked by the system. The dimension consists of `BluetoothAddress`, `MobileManufacturer`, `MobileModel`, and `ClassOfDevice` which are used to describe the specific device. These are included in the `BluetoothDevice` hierarchy. The `FlyerFrequency` attribute stores the number of times a specific device has been detected in the airport and is included in the `FlyerFrequency` hierarchy.

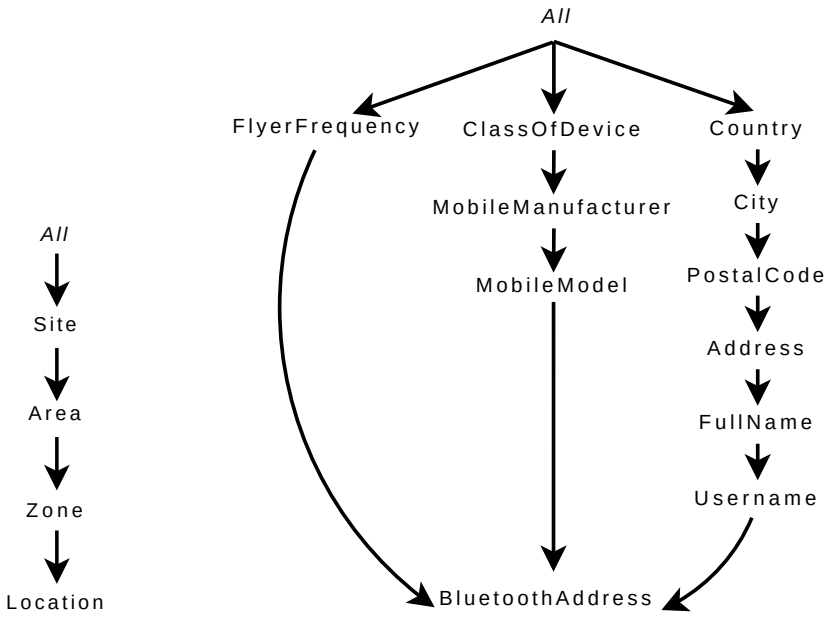


Figure 6: Location hierarchy. Figure 7: Bluetooth device dimension hierarchies.

Flyer frequency calculation is described in Section 5.2. The remaining six user information attributes are included to prepare the layout for future implementation of users into the `BluetoothDevice` dimension. The hierarchies of the `BluetoothDevice` dimension are shown in Figure 7.

Classification Dimension

The `Classification` dimension enables classification of the tracking records in the data warehouse e.g. *ok*, *wrong time*, etc. Classification is utilized in, e.g. the bounce detection algorithm, Section 5.3, to mark tracking records as *ModifiedBounceRecord*. This enables querying on specific parts of the data set, e.g. to only query on data that is not classified as *ModifiedBounceRecord*. The classification string is stored in the attribute called `ClassificationName`.

Fact Tracking

The `Tracking` fact table binds the tracking records to the relevant entries in the dimensions. The relations to the `TimeOfDay` and `Date` dimension are represented twice for each `datetime` attribute from the source data. This is to store both the wall clock time, and the UTC time when considering tracking records from different sites, e.g. CPH. Another solution for the UTC time would be to include extra attributes into the `TimeOfDay` and `Date` dimensions and then create two hierarchies in each dimensions, one representing UTC and one representing wall clock. We chose the current solution where we doubled the amount of time and date keys, to simplify loading the data through the ETL, considering the problems occurring when the UTC time and wall clock time are on different dates. The two attributes `Location` and `LastLocation` link to the locations in

which the specific device was detected and previously detected. The computed *DwellTime* and *IdleTime* of a given device is stored in the `DwellTime` and `IdleTime` attributes and the maximum signal strength is stored in the `MaxRSSI` attribute.

4.2 Cube design

We build a multidimensional cube on top of the the previously described data warehouse design. This will be described in the following section, starting with the different measures and calculations.

Aggregations have been implemented to speedup the browsing process of the cube. The aggregation design is primarily done by utilizing the built-in aggregation optimization algorithms in Microsoft Business Intelligence Studio and will therefore not be discussed further in this report.

Measures

In the fact table we have the following base facts: `DwellTime`, `IdleTime`, and `MaxRSSI`. From these three base facts we introduce the following measures. A measure is a numeric fact combined with a formula that computes the measure from the base fact.

The `DwellTime` base fact is used in the following measures which are computed in the scope given by the selected dimensions:

`Dwell Time Average` is the average `DwellTime` over time, e.g. how much time do devices in average use at a given location.

`Dwell Time Maximum` is the maximum value of the `DwellTime`, e.g. the longest time spent at a location.

`Dwell Time Minimum` is the minimum value of the `DwellTime`, e.g. the least time spent at a location.

The `IdleTime` base fact is similar to the `DwellTime` fact measures, but instead of focusing on time spent at locations, the `IdleTime` focus on the time spent in between locations.

`Idle Time Average` is the average `IdleTime` over time, e.g. how much time do devices in use average between locations.

`Idle Time Maximum` is the maximum value of the `IdleTime`, e.g. the longest time spent between two locations.

`Idle Time Minimum` is the minimum value of the `IdleTime`, e.g. the least time spent between two locations.

The `MaxRSSI` base fact, which stores the maximum signal strength, is used in the following measures:

`RSSI Average` is the average `MaxRSSI` over time, e.g. how strong is the average signal strength given the dimension conditions.

`RSSI Maximum` is the maximum `MaxRSSI`, e.g. the strongest signal strength measured in the specific dimension scope.

`RSSI Minimum` is the minimum `MaxRSSI`. e.g. the weakest signal strength measured in the specific dimension scope.

Besides measures computed from the base facts, we also include the following measures which are computed from the fact table:

`Bluetooth Device Distinct Count`, which is the distinct number of `BluetoothDevice` attribute values.

`Location Distinct Count`, which is the distinct number of `Location` attribute values.

`Fact Tracking Count`, which is the number of tracking records in the given dimension scope.

Fact Dimension

To support the BI related questions about *DwellTime* and *IdleTime* distribution, the *DwellTime* and *IdleTime* needs to be clustered into groups. This is done by creating a fact dimension also called a degenerate dimension [2] on the fact table attribute `DwellTime` and `IdleTime`. This dimension then clusters the `DwellTime` and `IdleTime` values into groups which are then used like a normal dimension.

Calculations

Percentage calculations was made to ease comparison of data when browsing the cube. The calculation simply takes the tracking record count at current hierarchy level and divide it with the tracking record count at the parent hierarchy level. Since this percentage calculation uses the dimension hierarchy when counting the tracking records it needs to be specified for each dimension in which it is to be used. We have made the percentage calculation for the following dimensions:

`Location Fact count` calculation enables comparison on percentage of tracking records per location.

`DwellTime Fact count` calculation enables comparison on percentage of tracking records per dwell time group.

`TimeOfDay Fact count` calculation enables comparison on percentage of tracking records per TimeOfDay entry.

5 Extract Transform Load

In this section we present the Extract-Transform-Load application designed to load BLIP's data set into the data warehouse design presented in Section 4. In the following overview we present the abstract layered architecture of the ETL, followed by the data types used in this section, the selection criteria used in the extraction layer and finally some problems with the source data that will be handled in the transformation layer. In Section 5.2 we present the Frequent Flyer Analysis used in the transformation layer and in Section 5.3 we present an algorithm that solves one of the problems with the source data.

5.1 ETL Overview

The ETL consists of the following three layers:

Extractor The Extractor is responsible for extracting the data from the source database, and structures the data using the data types described later in this section. The data is selected from the source according to the selection criteria presented later in this section where all data matching any one of the criteria is either disregarded or classified.

Transformer The Transformer is responsible for transforming the data extracted by the Extractor into data that match the data warehouse design. This includes making preaggregations like the frequent flyer analysis presented in Section 5.2 and performing data cleansing in the form of bounce detection presented in Section 5.3. The data modified by the cleansing algorithm is classified accordingly.

Loader The Loader is responsible for pre-filling the dimensions and for loading the transformed data into the fact table of the warehouse. The pre-filling of the dimensions is done to speedup the load of tracking records, since it would take to much time to fill the dimensions as needed.

Data Types

The nontrivial data types used by the ETL is presented here to make a common understanding of the data when explaining the different algorithms used.

Tracking record consists of the following attributes, an identifier, timing information in form of *EnterTime*, *ExitTime*, *PeakTime*, *DwellTime*, and *IdleTime*, a *BluetoothAddress*, location information through *Location* and *LastLocation* and lastly, a *Classification*.

Classification options consist of the following classifications: *Ok*, *ModifiedBounceRecord*, and *PeakError*. All tracking records are initially classified as *Ok*, after which they are reclassified if they fit one of selection criteria 3 or 4. The classification *ModifiedBounceRecord* consist of all records transformed by the bounce detection algorithm and therefor records already classified as *PeakError* can be reclassified as *ModifiedBounceRecord*. This means that you do not know if a record classified as *ModifiedBounceRecord* also has errors related to the *PeakError* classification.

Selection Criteria

The following selection criteria are used to remove or classify erroneous tracking records. The first two criteria concerns invalid data that will affect *dwell time* analysis if loaded into the data warehouse. The third and fourth criteria concerns data that has an invalid *PeakTime* attribute or is missing a *MaxRSSI* value. This data can still hold valid information for *dwell time* analysis, but is classified so it can be disregarded at a later point.

1. Records missing *EnterTime*, *ExitTime*, or *PeakTime* attributes are considered incomplete and are discarded.

2. Records with *ExitTime* < *EnterTime* are invalid and are discarded.
3. Records with *EnterTime* > *PeakTime*, or *PeakTime* > *ExitTime* are classified as *PeakError*.
4. Records missing the *MaxRSSI* attribute are also classified as *PeakError*.

Bounce Problem

The data gathering application used to gather the BLIP data set is not designed to handle devices that are tracked in more than one location at a time. This means that when a device is located in an area that is covered by more than one access point the data gathering application creates *bounce records*. An example of this is shown in Figure 8 where the grey areas are overlapping areas. If a device is traversing though `AP1`, `AP2`, `AP3`, and `AP4` as indicated by the red line, there is a large possibility that the system will generate *bounce records* when the device is located in the grey areas.

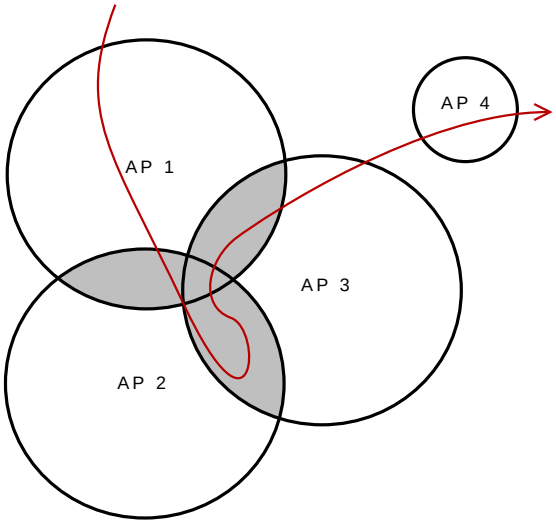


Figure 8: Bounce problem scenario.

The problem with these *bounce records* is that they can have a very negative effect when analyzing the *dwell time* of devices. This is because a device that is located in an area covered by multiple access points can produce many tracking records with a *dwell time* of 0 seconds. Summing up the *dwell time* of these records will produce an inaccurate total time spent in the areas by the device and lower the average *dwell time*.

In order to compensate for these inaccurate results, we propose a *Bounce Detection* algorithm in Section 5.3.

5.2 Frequent Flyer Analysis

As Section 1 states, one of the BI questions that we would like to answer is if there is a difference between how different types of people interact in the airport. A method to distinguish people is how often they appear in the airports, i.e. tourists will travel less frequent than businessmen. A *frequent flyer* is a person, who visits the airport more than once. The notion *frequent flyer* covers seeing a device entering and exiting the airport a number of times with at least a implementation specific *threshold* between each visit. It does not cover any information about if there are certain patterns within each occurrence, but is merely a number, denoted as *flyer frequency*, representing how often the device has been seen in the airport. A visit describes the enter time and leave time for one distinct appearance in the airport from a given device.

The identification of *frequent flyers* is done by examining each tracking record in the entire data set, and keeping track of when a device was last seen. If the time between the last point of time where a device has been spotted and the point of enter time of a new tracking object exceeds a pre-defined threshold, then the *flyer frequency* for the device in question is incremented. When performing these operations on each element in the data set, the output is a set describing the *flyer frequency* for each device.

Example If given two passengers A and B where, A has two visits, namely 8.00-8.30 and 15.00-15.30 and B has two visits, namely 8.00-8.30 and 17.00-17.30 and the flyer frequency *threshold* is set to eight hours. The *flyer frequency* would get incremented once for passenger A and twice for passenger B.

The algorithm for calculating *flyer frequency* can be seen in Algorithm 1. The algorithm uses two data types, a tracking record tr as defined in Section 5.1 and a temporary device td. The td type contains three attributes, $BluetoothAddress$ to identify which device it represents, $FlyerFreqency$ representing the frequency of the device in the current data set, and $LastSeen$ used to keep track of when the device was last seen in any zone.

On Line 2 the algorithm starts to loop through all the tracking records in TRQ. If there all ready exists a temporary device record td for the current tracking record device in the device set DS, the record is extracted in Line 4. The device record is removed from DS in Line 5 so it can be added again with the updated attributes. If the time between the *enter time* of the current tracking record and the *last seen* attribute of the device exceeds the given *threshold*, the *flyer frequency* is incremented and *last seen* is updated. Otherwise only the *last seen* attribute is updated. On Line 12 the updated td is added to DS.

Line 14 to 17 handles the case the device of the tracking record is not previously registered in DS. A new temporary device record is created with the device information, *flyer frequency* is set to 1 and the record is added to DS.

5.3 Bounce Detection

As a solution to the bouncing problem presented in Section 5.1 we propose a bounce detection algorithm.

To describe the algorithm a clear definition of the following terms is needed. *Bounce threshold* is an implementation specific threshold set according to the

Algorithm 1: `FlyerFrequencyCalculation(`TRQ`,` *threshold*`)`, calculate *flyer frequency* per device given a queue of tracking records ordered by ascending enter time and a *threshold*. Returns a set of device records with the calculated *flyer frequency*.

```
1  DS ← ∅;
2  while tr ← TRQ.dequeue() do
3      if tr_BluetoothAddress ∈ DS then
4          td ← The element td from DS where
           td_BluetoothAddress = tr_BluetoothAddress;
5          DS ← DS\{td};
6          if tr_EnterTime − td_LastSeen > threshold then
7              td_FlyerFreqency ← td_FlyerFreqency + 1;
8              td_LastSeen ← tr_ExitTime;
9          else
10             td_LastSeen ← tr_ExitTime;
11         end
12         DS ← DS ∪ {td};
13     else
14         td_BluetoothAddress ← tr_BluetoothAddress;
15         td_FlyerFreqency ← 1;
16         td_LastSeen ← tr_ExitTime;
17         DS ← DS ∪ {td};
18     end
19 end
20 Return DS;
```

constraints of the given data set. We define a *bounce record* as a tracking record with *dwell time* < *bounce threshold* for *bounce threshold* > 0. We define a *bounce region* as a time span in which a device produces only *bounce records* and the *idle time* between the records is less than *bounce threshold*. The terms *bounce record* and *bounce region* are clarified by the following example.

Example Figure 9 presents an example of a device that moves through the area presented in Figure 8 on page 15. At time `t0` the device is detected by `AP1`, and at time `t1` the device starts to bounce between `AP1` and `AP2`, which starts the *bounce region*, until `t2`. As the device moves around the area covered by multiple access points, the *bounce region* continues. The device is next being tracked through a number of *bounce records* between `AP2` and `AP3` in the timespan from `t2` to `t3`. In the timespan from `t3` to `t4`, the device is seen in three different access points, `AP1`, `AP2` and `AP3`. The last part of the *bounce region* is from `t5` to `t6`, where the device is seen at both `AP1` and `AP3`. At time `t5` the *bounce region* ends because the device produces a tracking record with a *dwell time* > *bounce threshold* in `AP3`. At time `t6` the device makes a clean shift from `AP3` to `AP4` with no *bounce records*.

By using this *Bounce Detection* approach, there can be tracking records that are not actually bouncing between two or more locations yet they will be

classified as such. E.g. if you have a device that reappears in a single location with a short and dwell- and idle time each time the device is tracked, then the current device creates a *bounce region* even though it is only been tracked from one location. This produces misleading data classification as the *bounce records* should be classified as *ok*, yet are classified as *ModifiedBounceRecord*.

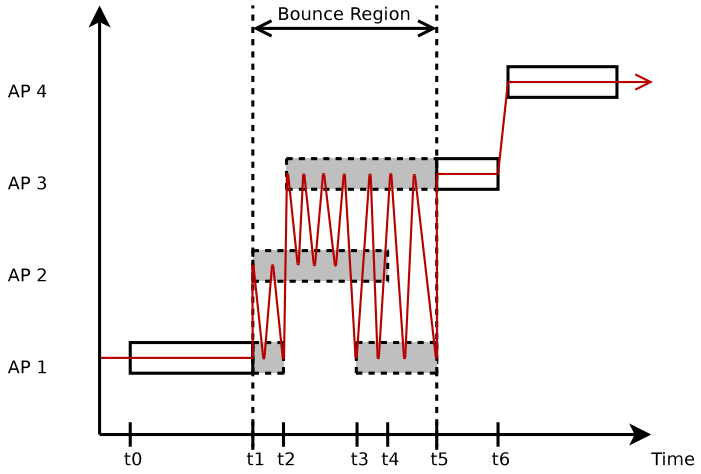


Figure 9: Time line example showing the tracking history of the device movement shown in Figure 8. Each solid white box represents a clean tracking record, and the greyed boxes represent many *bounce records*. The red line presents an example of the tracking record distribution.

The *Bounce Detection* algorithm is shown in Algorithm 2. The algorithm takes an ordered queue of tracking records (TRQ) as input and the previously mentioned *bounce threshold*.

From Line 2 to 17 in the algorithm every tracking record tr is processed to detect *bounce regions* and eliminate the *bounce records* in these.

On Line 3 in the algorithm, a queue (BE) of all previous *bounce records* from BQ with the same *Bluetooth Address* as the current tr, is created. On Line 4 the *dwell time* of tr is compared to $Bounce_{threshold}$. $Bounce_{threshold}$ is an implementation specific variable that needs to be set according to the constraints of the given data set. Empirical tests with this value are presented in Section 6.1. If the *dwell time* of tr is larger than $Bounce_{threshold}$ this ends the current *bounce region* and all *bounce records* in BE are removed from BQ. On Line 6 to 7 the *Bounce Elimination* algorithm presented later in this Section, is run on the records in BE and an *AddToLoadQueue* function is run on the current tr. *AddToLoadQueue* takes a processed tracking record and adds it to a queue of records that are ready to be loaded into the data warehouse. On Line 9 to 14 the algorithm handles the case where *dwell time* of tr is less than or equal to $Bounce_{threshold}$. If the *idle time* of tr is unknown or less than $Bounce_{threshold}$ we add tr to BQ, which continues the current *bounce region* or starts a new one. Otherwise, we end the current *bounce region* by removing the records in BE from BQ, running the *Bounce Elimination* algorithm on BE and add tr to BQ to start a new *bounce region*.

To process all *bounce records* left in BQ a set (BT) of the Bluetooth addresses

of the devices that still have records in BQ is created in Line 18. From Line 19 to 23 all the remaining *bounce records* in BQ are processed.

Algorithm 2: `BounceDetection(`TRQ`,` $Bounce_{threshold}$`)`, detect *bounce regions* within a queue of tracking records ordered by ascending enter time.

```
1  BQ.clear();
2  while tr ← TRQ.dequeue() do
3  |  BE ← Queue of tracking records t from BQ where
   |  t_BluetoothAddress = tr_BluetoothAddress;
4  |  if tr_DwellTime > Bounce_threshold then
5  |  |  Remove all elements in BE from BQ;
6  |  |  BounceElimination(BE) ;
7  |  |  AddToLoadQueue(tr) ;
8  |  else
9  |  |  if tr_IdleTime < Bounce_threshold ∨ tr_IdleTime = Unknown then
10 |  |  |  BQ.enqueue(tr);
11 |  |  else
12 |  |  |  Remove all elements in BE from BQ;
13 |  |  |  BounceElimination(BE);
14 |  |  |  BQ.enqueue(tr);
15 |  |  end
16 |  end
17 end
18 BT ← Set of all unique BluetoothAddress in BQ;
19 foreach bt ∈ BT do
20 |  BE ← Queue of tracking records t from BQ where
   |  t_BluetoothAddress = bt;
21 |  Remove all elements in BE from BQ;
22 |  BounceElimination(BE);
23 end
```

Bounce Elimination

The *Bounce Elimination* algorithm is shown in Algorithm 3. It takes a queue of *bounce records* detected as a *bounce region* and eliminates the *bounce records* by combining them into as few records as possible.

On Line 1 the algorithm handles the case where there is only 1 *bounce record* in BE. The *bounce record* is simply added unmodified to loader since there are no other records to combine it with.

If there is more than one record in BE, the start and end time of the *bounce region* is found in Line 5 and 6. The start time is set to the minimum enter time and the end time is set to the maximal exit time of all the records in the *bounce region*. On Line 7 a set (L) of all locations in the *bounce region* is created. In the case that only 1 location is part of the *bounce region*, a new tracking record is created and added to the load queue in Line 8 to 15.

If more than one location is part of the *bounce region* a new tracking record for each location is created in Line 18 to 28. The value of $region_{total}$ represents the total time span of the *bounce region* in seconds and t_{offset} is the amount

Algorithm 3: `BounceElimination(`BE`)`, given a queue of *bounce records* it creates new records that match all locations and time span of the *bounce region*.

Input: Queue of bounce records BE ordered by enter time ascending

```
1  if BE.length() = 1 then
2      br ← BE.dequeue();
3      AddToLoadQueue(br);
4  else
5      region_Start ← br_EnterTime, where br is the element i BE with the
       lowest enter time;
6      region_End ← br_ExitTime, where br is the element in BE with the
       highest exit time;
7      L ← Queue of all unique Location in BE ordered be enter time;
8      if L.length() = 1 then
9          br ← Any element from BE;
10         br_EnterTime ← region_Start;
11         br_ExitTime ← region_End;
12         br_DwellTime ← br_ExitTime − br_EnterTime;
13         br_Classification ← ModifiedBounceRecord;
14         AddToLoadQueue(br);
15     else
16         region_Total ← region_End − region_Start ;
17         t_Offset ← 0;
18         while loc ← L.dequeue() do
19             t_Count ← The number of elements br in BE where
               br_Location = loc;
20             t_DwellTime ← (t_Count / |BE| * region_Total);
21             br ← Any element br from BE where br_Location = loc;
22             br_EnterTime ← region_Start + t_Offset;
23             br_ExitTime ← br_EnterTime + t_DwellTime;
24             br_DwellTime ← t_DwellTime;
25             br_Classification ← ModifiedBounceRecord;
26             AddToLoadQueue(br);
27             t_Offset ← t_Offset + t_DwellTime;
28         end
29     end
30 end
```

of dwell time already allocated to already processed locations in the current *bounce region*. On Line 19 the total amount of tracking records for the given location is counted in order to make a weighted distribution of the $region_{total}$ on Line 20. This is done to divide the total time of the *bounce region* according to the amount of tracking records produced for a given location. On Line 21 to 25 the new tracking record is created and classified as a *ModifiedBounceRecord*. The record is then added to the load queue and t_{offset} is updated.

6 Results

In this section we present experimental results to support the $Bounce_{threshold}$ setting introduced in Section 5.3 and analysis of the BI related questions introduced in Section 1.

The data warehouse design presented in Section 4 has been implemented in Microsoft SQL Server 2008, the cube design presented in Section 4.2 has been implemented in Microsoft Analysis Services using MOLAP as the storage model, and we have used TARGIT BI Suite [4] to perform the analysis and create the graphical output.

The analysis has been performed on a modern desktop computer, with a dual core 2.66GHz CPU and 8 GB memory, running Windows Server 2008 64bit Edition. On this platform we can transform and load the entire BLIP data set in approximately 30 minutes using the ETL, presented in Section 5, implemented in Microsoft C# .NET. After the data is loaded it takes approximately 30 minutes to build the cube and process the aggregations in Analysis Services. When the data is loaded and the cube is fully processed, TARGIT BI Suite is able to generate each of the analysis results presented later in this section in near-real time, i.e less than 10 seconds per result.

We have used the previously described data set collected at CPH. The data set contains a total of 21,161,406 tracking records collected from 10-25 access points. The number of access points vary, because not all access points were active in the whole period. During the selection phase (extraction of source data) the ETL discarded 92,377 records and the transformation phase resulted in the removal of 2,549,718 tracking records. When all the data has been loaded into the data warehouse there is a total of 18,519,311 tracking records in the fact table.

6.1 Bounce Detection Test

In this experiment we simulate loading the complete data set into the data warehouse using different values (0 - 30 seconds) on the $Bounce_{threshold}$ setting in the ETL. The simulation results are shown in Figure 10 and include the total number of records loaded into the warehouse, the number of records classified as *ModifiedBounceRecord*, and the number of records classified as *Ok*.

The results show that the number of *ModifiedBounceRecord* records increases rapidly in the beginning and flattens out at approximately a 7 second threshold. Figure 11 shows the increment in the number of records classified as *ModifiedBounceRecord* at each $Bounce_{threshold}$ in the interval. The figure shows that the flattening also starts at a 7 second $Bounce_{threshold}$.

Because we do not have the precise location and coverage radius of each access point, we have to rely on intuition and the results presented in Figures 10 and 11 to estimate the best $Bounce_{threshold}$ setting. If $Bounce_{threshold}$ is set too high, records that are not actual bounce records, but merely people crossing in an access points peripheral, will be marked as bounce records. If it is set too low, too few of the bounce records are eliminated. Based on the previous results we set the $Bounce_{threshold}$ to 7 seconds, which eliminates 2.5 million bounce records.

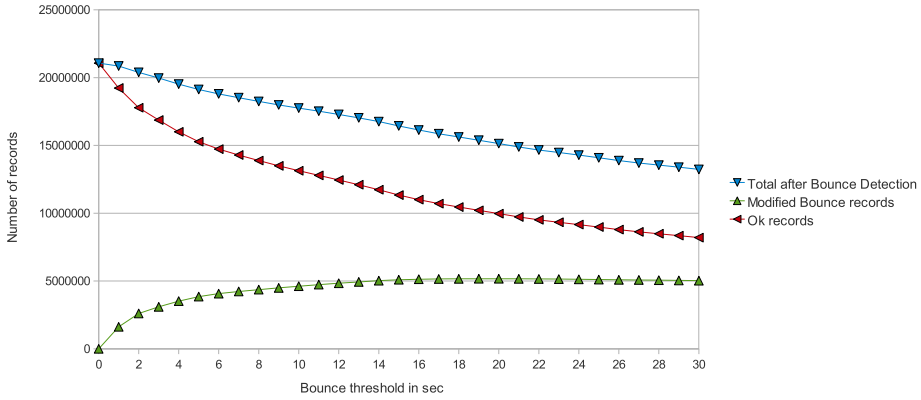


Figure 10: Effect of the $Bounce_{threshold}$ attribute on the number of records.

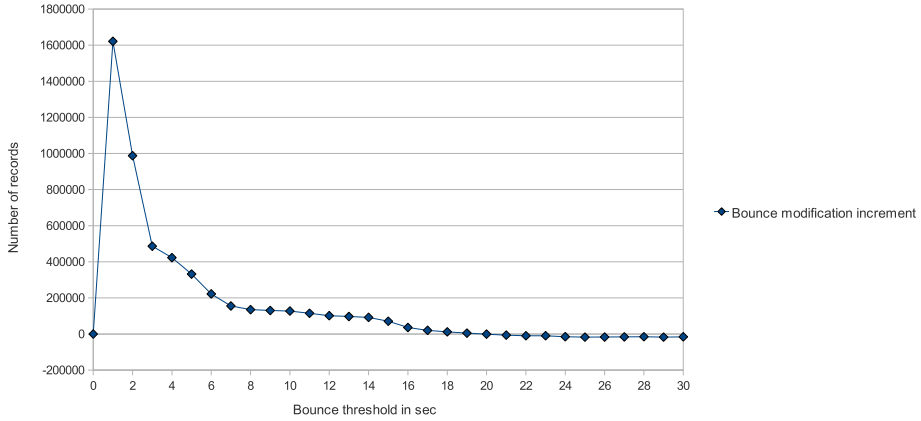


Figure 11: Effect of the $Bounce_{threshold}$ attribute on the increment of records classified as *ModifiedBounceRecord*.

6.2 Analysis of BI Related Questions

In the following we present the results to answer the BI related questions introduced in Section 1. Figure 12 shows the number of active access points during the entire span of the data set we are working on. The graph clearly shows the many fluctuations caused by the offline access points as mentioned in Section 3.2. The results presented in this section are, unless stated otherwise, based on data collected from 04/23/2008 - 05/30/2008, since this is the period in which we have the most complete data set.

How often have visitors been seen in the airport?

To show how often visitors have been seen in the airport we query the cube with the `Bluetooth Device Distinct Count` as the fact measure, grouped by the `FlyerFrequency` hierarchy from the `BluetoothDevice` dimension. Because the flyer frequency is computed on the entire data set, this graph is also computed

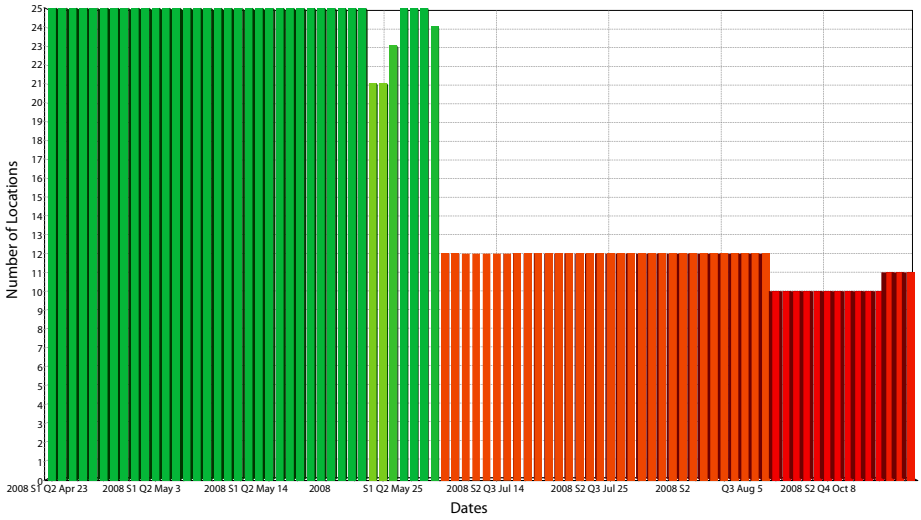


Figure 12: Number of monitored locations over time.

on the complete data set. The result shown in Figure 13 is limited to a maximum flyer frequency of 30 to make the graph more readable.

How is the visitor load distributed over the weekdays?

To show the distribution of visitors in the airport on the different days of the week, we query the cube with `Bluetooth Device Distinct Count` as the fact measure, grouped by the `DayOfWeek` hierarchy from the `Date` dimension. The result is shown in Figure 14. Surprisingly, the airport experiences the largest number of distinct visitors on Thursdays in this period.

How much time do visitors spend at the different locations in the airport?

To show how much time visitors spend in average in the different locations we query the cube with the `Average Dwell Time` as the fact measure, grouped by the `Location` hierarchy. The result is shown in Figure 15. The graph can be used by the airport administration to identify *high congestion areas*. According to BLIP, the location named "ms-spopos1.16", covers part of the security and tax free shopping area. This is a valid explanation for the high *dwell time* in this area.

How is the distribution of time spent at a specific location?

To show the distribution of time spent at a specific location, in this case the "ms-spopos1.16" location, we query the cube with the `Fact Tracking Count` as the fact measure grouped by the `Grouped Dwell Time` hierarchy from the degenerate dimension and the location set as a criteria. The result is shown in Figure 16.

The graph shows a large number of records with a low *DwellTime*, which could be devices tracked in the peripheral of the access point or visitors quickly

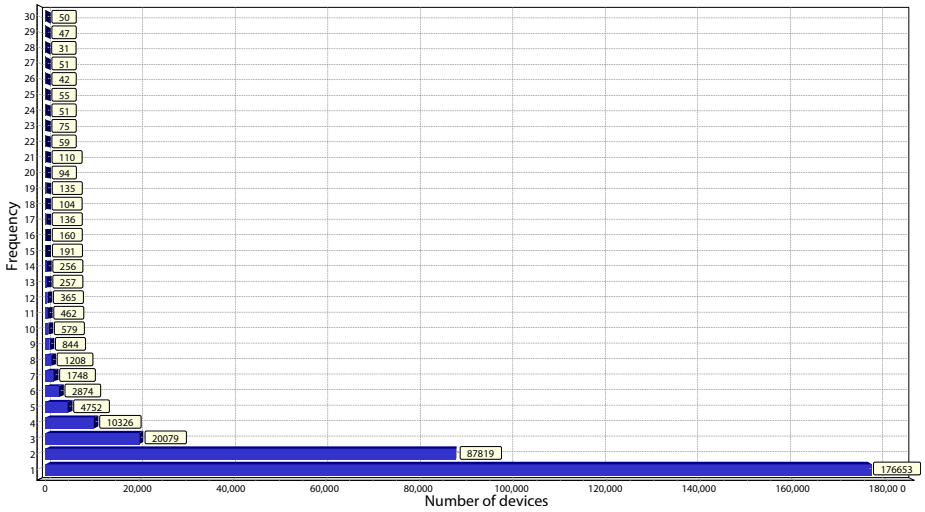


Figure 13: Number of devices distributed over flyer frequency.

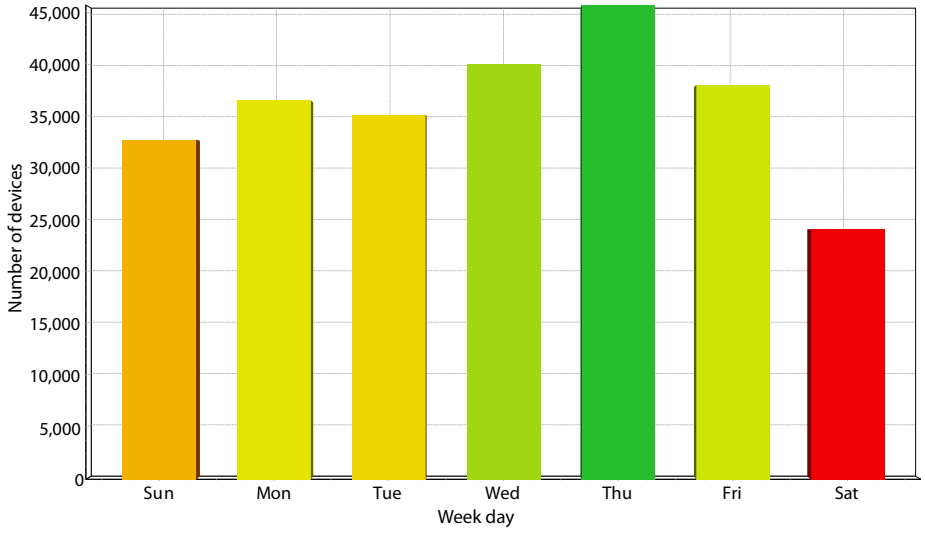


Figure 14: Number of distinct devices distributed over weekdays.

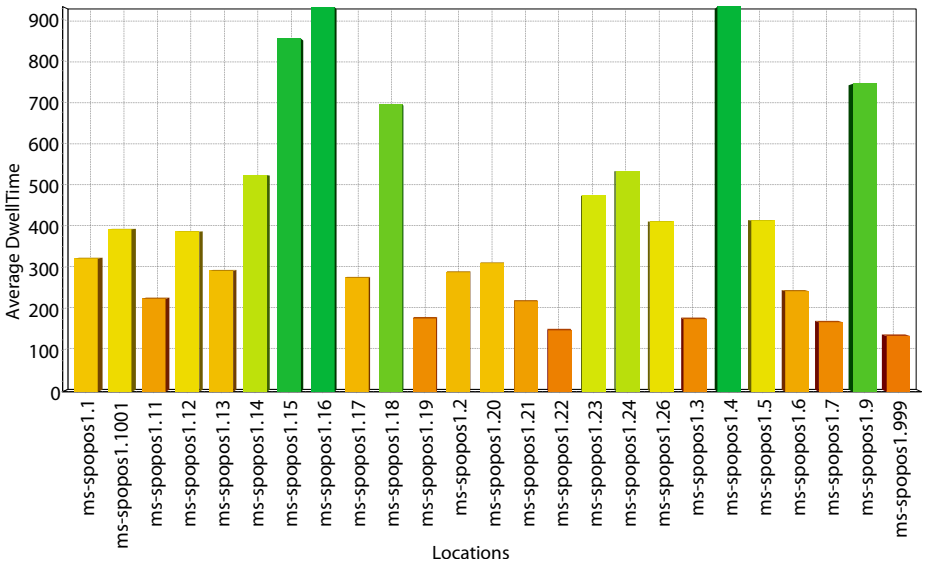


Figure 15: Average *DwellTime* in the different locations.

passing through the location. We notice that the number of records slowly start to increase as *DwellTime* gets larger, this could be explained by people stopping to look at merchandise in the tax free shopping area. We see that a small number of people are staying in the location for a significantly long period of time. We believe these people to be employees in the shop.

How is the distribution of time spent at different times of the day?

To show the distribution of *DwellTime* in the airport over 24 hours, we select `Average Dwell Time` as the fact measure grouped by the `TimeOfDay` hierarchy. The result shown in Figure 17 is a comparison between the average *DwellTime* in the weekend (green area) and the weekdays (red area). To show how this relates to the number of passengers we made a query on the number of devices over 24 hours. This is done using `Bluetooth Device Distinct Count` as the fact measure grouped by the `TimeOfDay` hierarchy and again made as a comparison between weekend and the weekdays. This result is shown in Figure 18.

When examining the figures it can be seen that the number of passengers have a direct impact on the average time spent at locations in the airport, which could be caused by increased queue time caused by the increased density in visitors in the airport. Another thing to notice is the early peak in the number of passengers flying early in the weekdays and the peak in the number of passengers flying late afternoon in the weekends. This information could be very useful for the airport when deciding on the number of staff needed during a week.

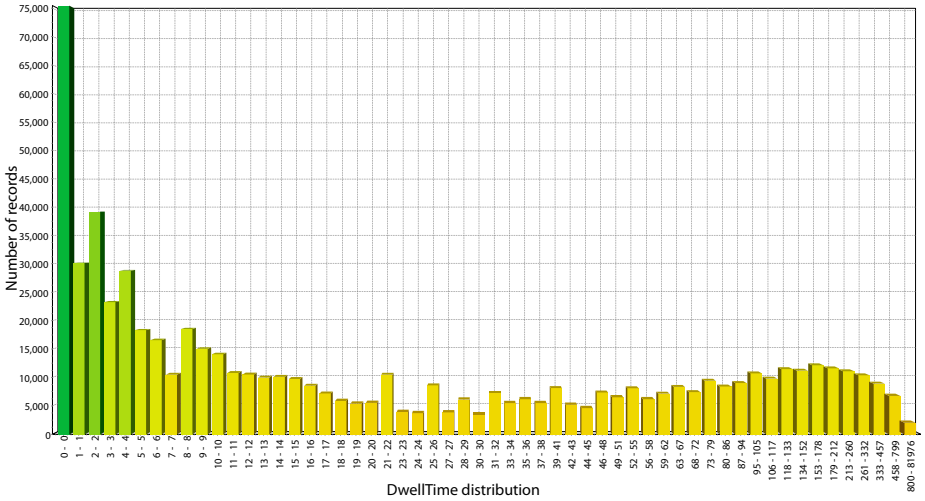


Figure 16: Distribution of *DwellTime* in the tax free area.

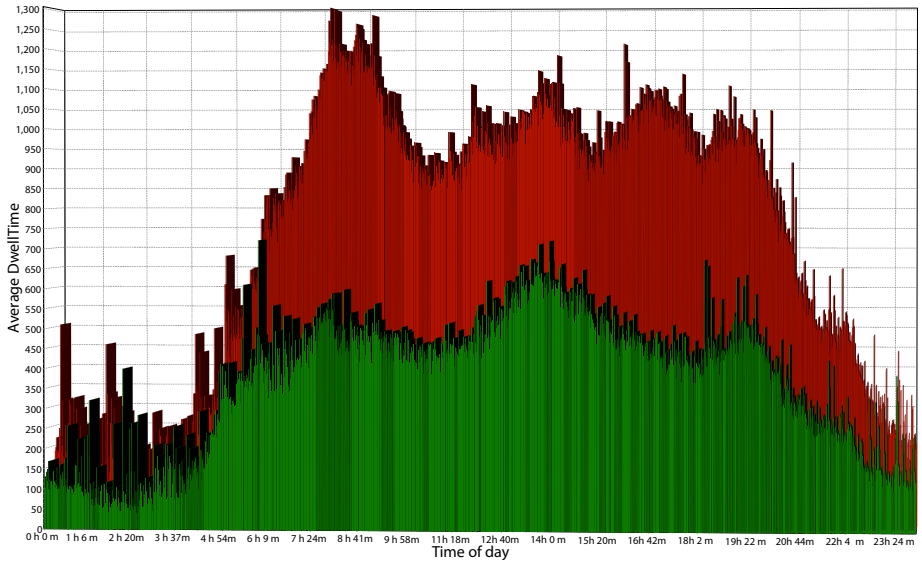


Figure 17: Average *DwellTime* in the weekends (green) and the weekdays (red) over `TimeOfDay` hierarchy.

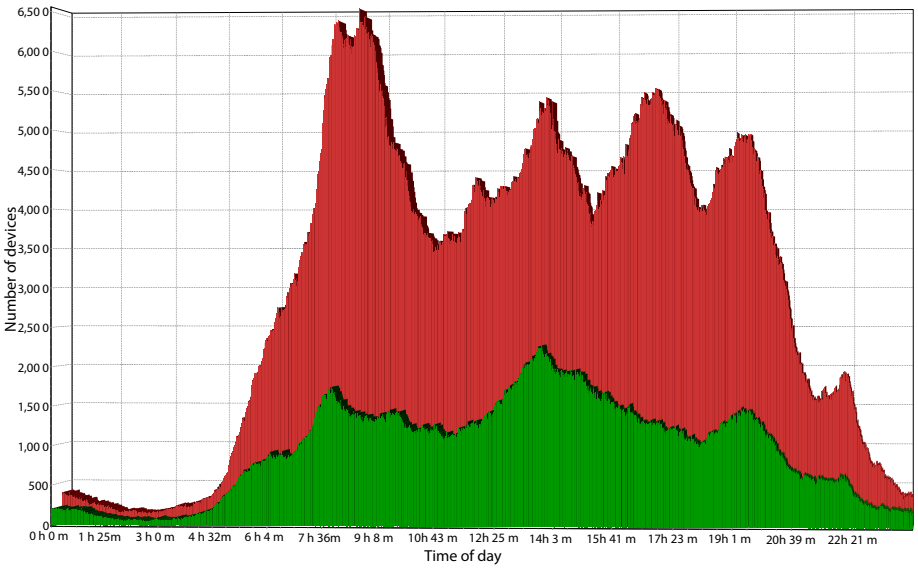


Figure 18: Distinct devices in the weekends (green) and the weekdays (red) over `TimeOfDay` hierarchy.

7 Conclusion

The main focus of this report is to provide an effective way to provide historical analysis of the tracking data collected by BLIP in CPH. To achieve this we design and implement a data warehouse to suit the analysis and an ETL to load the data into the data warehouse. BLIP tracks Bluetooth devices in CPH to help the airport administration answer a series of BI related questions. We propose several BI related questions that can be answered by analyzing the data.

To provide a motivational reason to get involved in this kind of project, we discuss some of the different technologies that can be used for in-door tracking. These provide roughly the same features, but they vary greatly with respect to how well they provide these features. We conclude that the choice of technology depends entirely on the requirement of the project.

We present a data warehouse design based on a star schema and implement this in Microsoft SQL Server 2008. We define the dimensions, dimension hierarchies and measures necessary for analysis in Microsoft Analysis Services 2008.

The ETL application developed in Section 5 extracts data from the BLIP data set, then transforms the data to suit the data warehouse and finally loads the data into the warehouse. The ETL utilizes the proposed algorithms to perform data cleansing and classification and thereby reduces the total number of records from approximately 21 to 18.5 million of which 4.2 million are marked as modified bounce records. The result of the transformation is directly seen when analyzing the data, since the data provides a more realistic measure of the dwell times in the different locations than before the transformation. The classification dimension allows the person performing the analysis to disregard any data modified in the transformation.

We show how the implemented solution can be used to answer the introduced BI related questions by utilizing Targit BI Suite to visualize the results. By studying the results we discover different tendencies like that CPH have the most visitors on Thursdays, and that the amount of visitors peak in the morning during weekdays and at noon in the weekends.

8 Future Work

This section contains a description of topics for future work. We will describe each new topic and the possible knowledge to derive from its analysis. This section is divided into two major areas of interest that we would like to address, namely flow analysis and real-time monitoring.

Flow Analysis

Flow analysis covers the ability to, based on the movements of the passengers in the airport, do further classifications. The concept of flow analysis introduced as *flyer frequency* was found insufficient. One of the BI related questions that we would like to answer is *How do transit passengers deviate from forced routes during rush hour?*. This question introduces two new concepts.

First, it redefines the flyer concept, since our *flyer frequency* cannot distinguish between different types of people seen in the airport. If you have information about where you track a device for the first and last time within a visit, you can classify people in the airport into, e.g., guests, that enter and leave through an entrance, transit passengers, that enter and leave through a gate, departing passengers that enter at an entrance and leave at a gate, and arriving passengers that enter though a gate and leave though an entrance.

Second, it introduces forced routes, where a forced route is a route the person has to follow, typically the shortest route, and a non-forced route is one that deviates from the forced route. A forced route can either be dictated by duration or a pattern, meaning if most people take 20 minutes to get from A to B, or a specific route, e.g., the specific route from security to a gate. It would also be interesting to find general patterns within the devices such as correlation.

The movement of people in an airport can be analyzed on various levels of detail depending on which information the airport wants to derive. We propose two different approaches namely *visit* and *visitor movement* to model movement in the airport into a format we can perform data analysis on.

Visit

A simple option is to consider flow as being an enter time and location and a leave time and location for a given device which can be implemented in various ways. This paragraph will describe a case utilizing a new fact table, and thereby expanding the star schema to a constellation schema, and using this new fact table to store movement information. A concrete fact record will be denoted as a *visit*.

The current data warehouse schema should be expanded with a new fact table `Fact_Visit` to incorporate *visits*. The table `Fact_Visit` should share the dimensions `Dimension_TimeOfDay`, `Dimension_Date`, `Dimension_BluetoothDevice`, and `Dimension_Location` with `Fact_Tracking`. `Fact_Visit` consists of keys to model enter and leave time, keys to the `Dimension_Location` to describe enter- and leave location, and `Dimension_BluetoothDevice` to connect a *visit* to a device. Additionally, it contains measures for *duration* of the visit for the given airport as well as total dwell- and idle time.

Detecting visits would be a part of the ETL, namely in the transformation layer. The frequent flyer algorithm, described in Section 5.2, can be modified to support *visits* by expanding it to include logic for determining type of *visit* and maintaining a data type that maps a device to a number of *visits*.

The fact table `Fact_Visit` can be used for the airport to answer a number of questions that were unanswerable with the current tracking fact table. The airport is interested in having more device classification options than frequent flyer, such as guests, arriving-, departing-, and transit passengers. This would enable answering queries such as *How many visits do transit passengers have over a period of time?*.

Visitor Movement

Rather than seeing flow as an entry point and an exit point, a more complex *visitor movement* model is proposed. *Visitor movement* approaches flow differ-

ently as it considers the movement inside the airport as important, and focuses on movement patterns. A pattern is an abstract representation of movement, that stores timing and locations bounded by a *visit*. Additionally, it would be possible to look at sub-patterns within patterns. By representing movement in such a way, the system would be able to answer more complex queries, e.g., *Do people follow forced routes during rush hours?*.

The *visitor movement* should be created each time the tracked device is considered to be on a new visit to the airport. Related work proposes a number of ways to model movement [17] [5] such as tourists moving at a site or tracking navigation on a website, which has techniques that are applicable for the CPH case. The proposed Bounce Detection algorithms, as seen in Section 5.3, cleanses the tracking records by combining bounce records into longer records. Having fewer, longer records reduces the amount of possible patterns, which makes it easier to find tendencies within movement patterns.

Real-Time Data Warehouse

Another interesting topic for future work would be to make a real-time data warehouse implementation. A real-time data warehouse could give possibilities like prediction of device destinations and the usage of monitoring systems that alerts the airport-, gate-, or security-personnel when congestion in certain areas exceed some threshold.

A real-time data warehouse could also be the underlying database, e.g., the data warehouse presented in Section 4, for LBS services to the mobile device users, e.g., the passengers would be able to receive information from the airport if they somehow identified themselves. The system could also provide assistance with time management within the airport as it would help the users to arrive at the gate in time.

One approach for implementing this could be to develop a multidimensional database schema and implement this in both a MOLAP cube and ROLAP cube. This would give the possibility to query the MOLAP cube for historical data, and the ROLAP for real-time data. Since the ROLAP cube is much easier to update with new data, this could be used to hold the data for, e.g., one week. All data older than one week would then be loaded into the MOLAP cube.

References

[1] Alexandra Instituttet A/S. Spopos project. http://www.spopos.dk, 2007, Accessed: 12/19/2008.

[2] Bob Becker - Kimball Design Tip 46:. Another look at degenerate dimensions. www.rkimball.com/html/designtipsPDF/DesignTips2003/KimballDT46AnotherLook.pdf, 2003, Accessed: 12/19/2008.

[3] Bluetooth security review. http://www.securityfocus.com/infocus/1830, 2008, Accessed: 12/19/2008.

[4] Targit a/s. http://www.targit.com, 2008, Accessed: 12/19/2008.

[5] Jensen, A. H., Larsen, R. S., Pedersen, T. B., Andersen, J., Andersen, J., Giversen, A., Giversen, A., Rune, L., Torben, S., Pedersen, B., Jensen, A., Skyt, J., and Skyt, J. Analyzing clickstreams using subsessions. In *In Proceedings of the 3rd International Workshop on Data Warehousing and OLAP* (2000), ACM Press, pp. 25–32.

[6] Jensen, C. S., and Pedersen, T. B. Multidimensional databases and olap. *Submitted to J. Hammer and M. Schneider (Ed.s): Handbook of Database Technologies, CRC Press, forthcoming* (2007).

[7] Lyngsoe Systems A/S. Lyngsoe systems a/s. http://www.lyngsoesystems.com, 2007, Accessed: 12/19/2008.

[8] Michal Lev-Ram - for Business 2.0 Magazine. Putting mobile services on the map. http://money.cnn.com/2007/01/26/magazines/business2/gpsservices.biz2/index.htm, 2007, Accessed: 12/19/2008.

[9] Microsoft. Business intelligence capabilities. http://www.microsoft.com/bi/bicapabilities/default.aspx, 2008, Accessed: 12/19/2008.

[10] Path Intelligence Ltd. Our footpath technology. http://www.pathintelligence.com/website-prodnserv.htm, 2007, Accessed: 12/19/2008.

[11] Path Intelligence Ltd. Path intelligence ltd. http://www.pathintelligence.com, 2007, Accessed: 12/19/2008.

[12] Purelink Technology Inc. Active rfid and real-time location system. http://www.purelink.ca/Product/Tags.aspx, 2008, Accessed: 12/19/2008.

[13] Q.E.D. Systems, Savi Technologies. Active and passive rfid: Two distinct, but complementary, technologies for real-time supply chain visibility. http://www.autoid.org/2002_Documents/sc31_wg4/docs_501-520/520_18000-7_WhitePaper.pdf, 2002, Accessed: 12/19/2008.

[14] Tutorials Point. Gsm - addresses and identifiers. http://www.tutorialspoint.com/gsm/gsm_addressing.htm, 2007, Accessed: 12/19/2008.

[15] Wikipedia. Bluetooth. http://en.wikipedia.org/wiki/Bluetooth#Uses, 2008, Accessed: 12/19/2008.

[16] Blip systems - bluetooth marketing solutions. http://www.blipsystems.com/, 2008, Accessed: 12/19/2008.

[17] Xia, J., Ciesielski, V., and Arrowsmith, C. Data mining of tourists spatio-temporal movement patterns —a case study on phillip island. *Proceedings of the 8th International Conference on GeoComputation* (2005).

A BLIP systems access point setup

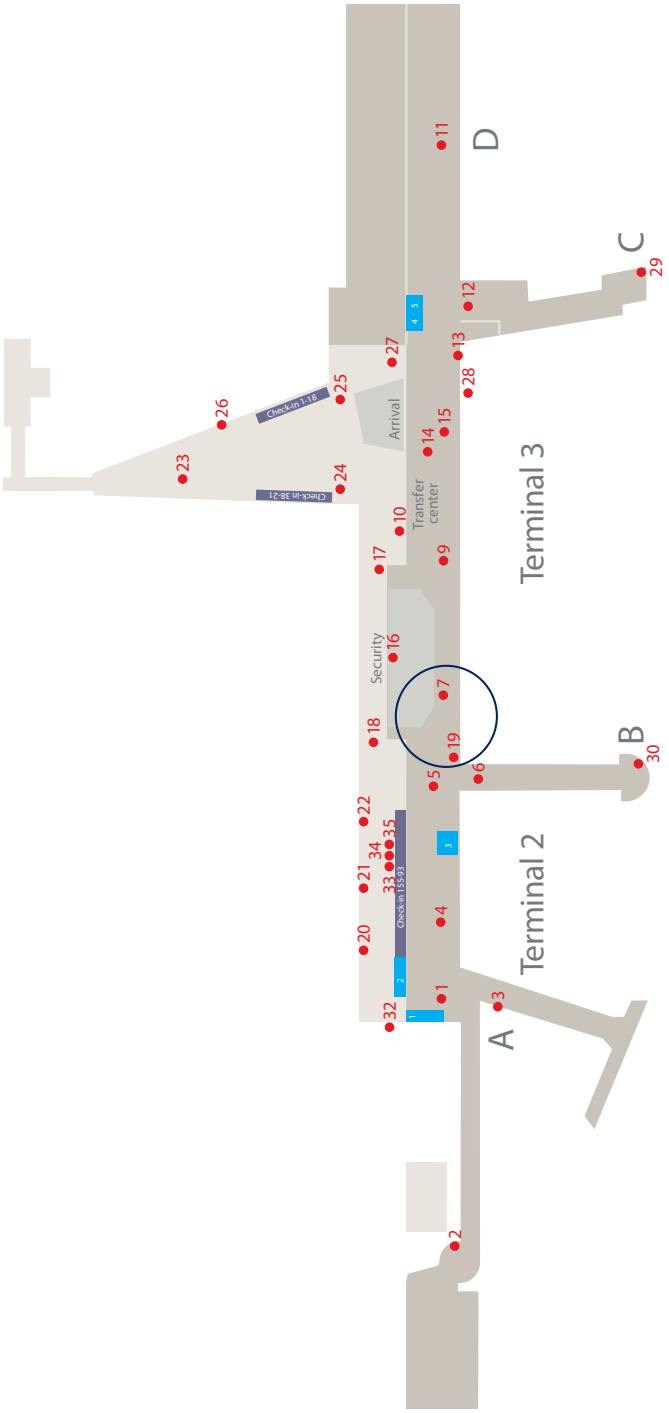


Figure 19: BLIP systems access point setup.