Chapter 1

Introduction

This thesis describes a new method of calculating data priority by using adaptive mobile user and device profile which changes with user location, time of the day, available networks and data access history. The profile is used for data prefetching, selection of most suitable network and cache management on the mobile device in order to optimally utilize the device’s storage capacity and bandwidth.

The most commonly used method to cope with unstable connection and limited bandwidth is to cache the required data on the mobile device. Due to the limited storage capacity of mobile devices, all the required information cannot be cached. Moreover, unlike conventional distributed information systems, user’s data requirements, needs and access priorities depend heavily on the user’s current location, context and device capabilities. Thus, the solutions proposed for cache management and data prefetching on wired stationary systems do not work in a Mobile Environment. A solution is required which addresses following problems:

- Limited or no bandwidth
- Change in available networks at different locations
- Unstable Connection
- Limited Storage Space on a Mobile Device
- Limited document support of mobile devices

To date, the work on mobile user profiles, preemptive data caching and cache management on mobile devices [13, 14] has been limited to address the unstable nature of mobile networks only. Mitch proposed the implementation of distributed system consisting of a network of profile managers. The profile managers, located throughout the internet, will then be used for data retrieval from the source to the
delivery on a mobile device including the data packaging. The system proposed by Mitch also addressed the issues of limited local storage and weight allocation to data items based on users’ interests. However, the proposed system did not include the mobile device’s capabilities in terms of data support, impact of user’s movements and history of data access on data utility for the user.

This thesis is devoted to propose and test a method to address following issues which were ignored by previous methods:

1. Change in the priority of subset of data on the basis of device capabilities.
2. Change in data priority used for preemptive perfecting on the basis of user’s current location while considering user’s appointment information.
3. Use of user defined data priorities, device storage capabilities and history of data access by the user in order optimally manage cache on the mobile device.
4. Integration of device profile, user profile, data profile with location and network information.

1.1 Motivation

Mobile devices as tools for providing data access to users on the move is now widely accepted. Most commonly used mobile devices for this purpose are laptops, tablets, personal digital assistants (PDAs) and smart phones. Current technology enables the users to manually maintain and synchronize the data with the fixed devices i.e. data sources. However the currently used synchronization process requires stable and fast connection, thus this process is commonly performed with specific manually configured fixed systems like user’s workstation or home computer. These restrictions currently place severe limitations on the range and sophistication of the applications that can run and perform effectively on mobile devices. It is proposed in this thesis that mobile devices will use prefetching along with intelligent cache management system. This will remove the limitations and allow the system to cache
the required data before it is requested. The system will also use the device storage capacity optimally and select most suitable wireless network automatically at each new location.

**Example:**

Following is an example that will help to illustrate the problems with the current proposed methods. This example is also used later used to explain the advantages and assumptions of the new proposed method.

Consider a company sales representative, Albert, who is equipped with Compaq iPaq with Bluetooth, WLAN and GPRS wireless network capabilities with the storage capacity of 50 Mega Bytes. Let’s assume that Albert is traveling for a series of meetings with the potential customers i.e. Mr. John, Mr. Bill and Miss Kathy. These customers are interested in different products and the data required for each meeting is 35, 30 and 25 Mega Bytes respectively. In the conventional system, Albert would have copied all the data he will need for all the three meetings by anticipating the customer’s needs and requirements, as storage capacity has never been a major issue in standard desktops. However, in the case of mobile device before each appointment Albert will be required to delete the obsolete data which he does not need for the next appointment and copy the required data for next meeting(s) by using wired network. Albert, however, will fail to copy all the required data needed for all the three meetings as the total size of the required data exceeds the mobile device storage capacity.

In situations where the device storage capacity is less than current and future data needs and there is no guarantee of stable wireless connection at the meeting sites, Albert will be forced to either manually use a wireless network, if it is available at all, in order to access and download the required data from the source. The second choice is for Albert to visit his office after each meeting in order to delete the obsolete data and copy the data needed for next meeting(s). For example, after the meeting with John, Albert will need to copy the data required for the meeting with Bill, as the utility of data related to John will be obsolete once the meeting is over. He will be faced with a similar situation before Kathy’s meeting. So in both the cases
Albert will have to manually delete the data which he does not further require and copy the data required for the next meeting. Moreover, if Albert decides to move to the next meeting without the required information at hand, he is unlikely to work efficiently due to the absence of the required data as the current bandwidth of the wireless connection could be less than what is required. Also there might be no supported wireless networks at the meeting location which will lead to embarrassment, frustration and loss of sales. It is not practical for Albert to make a visit to his office after each meeting to copy the data.

Three major facts are highlighted from the above scenario:

1. The utility of data objects that are only related to a specific customer or meeting can be obsolete once the meeting is over unless these are required in the future.

2. The user of the mobile device cannot entirely rely on and assume the availability of a stable supported wireless network at the meeting location as the required wireless network may not be entirely available or may be unstable at certain times or at certain locations.

3. If the user copies all the data related to a customer or meeting then he may be wasting precious storage space for the documents which are not supported by the currently used mobile device e.g. smart mobile phones currently do not support PDF documents.

1.2 Summary and Thesis Overview

The major contribution of this thesis is the development of a new user profiling mechanism which takes into account:

- User preferences in terms of data
- Supported wireless networks by the device
• Device storage capacity
• Types of documents supported by the device
• User appointment information in order to determine the set of data needed to be prefetched
• Available networks at the current location
• Data access history of the user

The capability of our profiling system to take into account all of the above factors in order to calculate the data utility for a set or subset of data, makes our proposed profiling system different and better than the previously proposed methods.

The thesis first evaluates all of the proposed methods currently being used for calculating data priority in a mobile environment. Chapter 3 outlines the domain scenario which is used for the development, implementation and evolution of our system and method. Chapter 4 describes the detailed structure, design and implementation of our system. The complete system was developed by using Java 2 Standard Edition (JDK 1.5 +) and MySQL. The results and evolution of the system along with conclusion is given in Chapter 5. The proposed mechanism has been evaluated with a simulator modeling. Chapter 6 contains the future directions.
Chapter 2

Background

This thesis deals with the profile driven management of mobile users so a lot of disciplines are integrated in this. First a review will be done of the current work on the use of profiles for data prefetching in a mobile environment. In the end a review will be done on the works of cache management in a mobile environment that is relevant to the application of this work and have influenced this project.

2.1 Prefetching Background

Prefetching or preloading followed by Caching have been essential parts of processors and file systems from the very early days of microcomputer. The use of prefetching dates back to mid 1965’s and it started with simple read-ahead in uniprocessors. It is now being used in a variety of fields including multimedia applications, World Wide Web and even in mobile computing.

Multics (Multiplexed Information and Computing Service), a mainframe timesharing operating system began in 1965. Multics begun as a research project heavily influencing operating system development. The system became a commercial product sold by Honeywell to educational institutions, government and industry. Multics supported prefetching, as did UNIX.

Earlier work was focused on the benefit of prefetching, either by allowing applications to give prefetching hints to the operating system or by automatically discovering file access patterns in order to better predict which blocks to prefetch. The Multic’s I/O system used prefetching by reading one step ahead from input devices before the read call was actually issued. Even from early days caching was used with prefetching. Most systems relied on cache buffers, a small fast memory holding the most recently accessed data, for performance gains as the access to this
buffer was faster in relation to the main memory. Prefetching and caching techniques later merged together in order to improve the system performance by caching the information before it was requested. This resulted in major focus on prefetching of elements of caches, file systems and databases. However, till early 1980’s the most common technique for caching was one block look-ahead (OBL) which assumed that the cache can only handle one request at a time. In 1981, the first lockup-free cache was introduced by Kroft called “miss information/status handling registers” (MSHR).

After the discovery of the first dual processor (VAX 11/780) at the Purdue University School of Electrical and Computer Engineering in 1981, the era of multiprocessing began. In 1989 Kotz & Ellis [1] outlined the two essential elements of any file system design intended for highly parallel environments: a) parallel I/O and b) effective caching schemes. They concentrated on the second aspect of file system design and specifically, on the question of whether prefetching blocks of a file into the block cache can effectively reduce the overall execution time of a parallel computation. They also studied [2] the detection of more complex access patterns within a file by using past file accesses to predict future data use. In this way, they provided a non-sequential prefetching mechanism.

Since 1990's many researchers have invested a great deal of effort in the computation of new and more effective prefetching and caching techniques. These techniques are mainly divided into two different types of approaches a) hardware techniques [3 4 5 6] and b) software or compiler-directed techniques [7 8 9].

Very little work was done on combining caching and prefetching, till 1995, as the overall system performance actually degraded due to I/O stall times because of underutilization of disk parallelism and file cache buffers in traditional file systems. Cao [10] was the first one to study caching and prefetching in an integrated fashion. Although many researchers have proposed extended solutions before him, most of the studies on prefetching have been conducted in the absence of caching or are done for a fixed caching strategy. The four rules for optimal prefetching and caching proposed by Cao are outlined below:
1. **Optimal Prefetching**: Every prefetch should bring into the cache the next block in the reference stream that is not in the cache.

2. **Optimal Replacement**: Every prefetch should discard the block whose next reference is furthest in the future.

3. **Do No Harm**: Never discard block A to prefetch block B when A will be referenced before B.

4. **First Opportunity**: Never perform a prefetch-and-replace operation when the same operations (fetching the same block and replacing the same block) could have been performed previously.

The rules outlined by Cao are essential - any algorithm can result in good performance gain, as long as it properly implements the four rules. Patterson [11] on the basis of rules that Cao presented, proposed aggressive and proactive mechanisms that tailored file system resource management to the needs of I/O-intensive applications. They showed how to use application-disclosed access patterns (hints) to expose and exploit I/O parallelism. In that way they sought to eliminate I/O stall time while maximizing buffer availability for caching. As applications needed to process larger objects and faster CPU’s were available, effective caching became more vital to overcome the limitations of traditional I/O. Since most of the modern applications were I/O intensive and the amount of data required for such applications was much more than file cache sizes, I/O stall time consisted of significant fraction of total execution time. The only feasible solution to these new challenges was to achieve a balance between caching and prefetching. Patterson with a powerful resource management scheme (TIP – transparent informed prefetching) proposed the use of informed prefetching and informed caching where the prefetching adapted to available storage bandwidth and an application’s use of buffers, and it optimized disk throughput with scheduling and batching. They recommended using predictability that is disclosure from applications about their future requests when accessing the file system.
Mobile computing along with distributed databases and file systems added new challenges to the problems of prefetching and caching. Limited bandwidth, high latency and unstable connections in mobile environment increased the impact of prefetching and caching on mobile information systems by many fold. The commonly used prefetching and caching techniques used for simple memory access or client-server applications over wired links could not provide desirable results due to diverse environment and configuration of mobile applications.

### 2.2 Profile Driven Prefetching

Adaptive Power – Aware Prefetch in Wireless Networks [12]

Data prefetching is considered to be an effective method to reduce the query latency. However, it consumes power which is a limited and scarce resource in wireless networks. This work addresses the issue of finding a balance between prefetching and power consumption and provides a mechanism, adaptive value-based prefetch scheme (AVP), which identifies the data needed to be prefetched in order to reduce the power consumption while providing better performance. The proposed scheme is superior from the previous methods as it considers the constraints of the mobile clients including the data access rate, the size of the data items and the data update rate. AVP also dynamically adjusts the number of prefetches to get better tradeoff between performance and power. In addition to the above, AVP makes smarter prefetch decisions, as it adjusts the prefetch rate based on the current energy level of the device.

### 2.3 Cache Management and User Profile

Expressing User Profiles for Data Recharging [13]

In this work the authors make an analogy between power recharging and data recharging on a mobile device. The authors argue that while electrons are basically interchanged, but the data needed on a particular mobile device is
highly dependent on the user of that device and the tasks the user will perform when they need the data. Since, the choice of data is dependent on the semantics of the applications thus the system must maintain and exploit information about the data needs of the users. The authors propose the implementation of a distributed system that consists of a network of profile managers located throughout the Internet. The profile managers are responsible for retrieving the needed data from the data sources while its packaging and delivery to specific devices is based on each device’s recharging profiles.

![Figure 1: Architecture of Data Recharging Service](image)

The profile managers use profiles for data recharging which consist of two types of information. First, relevant types of data for the user from within the data domain. Second, specified amount of data needed to be sent to a device during the connection on the basis of bandwidth, device storage capabilities and network limitations.
The set of interesting objects specified in a profile is that profile’s profile domain (D). The system first uses the profile domain information to determine the data of interest to the user. Then the profile manager is used to locate the data objects. The data objects which are successfully located result in the profile instantiation (I). Then a smaller subset of data in I is identified, by the profile manager, to be copied to the mobile device on the basis of P’s utility function specification. The utility function ultimately establishes the relative worth of data objects in the light of resource restrictions (for example, available bandwidth and available storage space on the mobile device) at the time data copy.

Profile – Driven Cache Management [14]

This paper proposes the use of profile – driven cache management system in order to solve problems related to non-persistent wireless connection and limited wireless bandwidth. The proposed system is based on a user profile which acts like a data recharging middleware system. The system gathers data of interest to a user upon connection. The system then delivers the data to the user’s mobile device on the basis of calculated data weight. The profile, while determining the data to be cached, also takes into accounts the user’s context (e.g. location, workflow) and the resources available on the mobile device (e.g. memory, applications). However in data recharging the authors assume that the users connect to well-defined data recharging stations.
This paper proposes a mobility-aware algorithm called MARS+ to detect regular client movement patterns and to provide information in order to improve cache performance. MARS [16] is primarily used for optimal replacement of cache. In MARS when a new data object is needed to be cached, and there is insufficient cache space, the object with the lowest replacement cost is removed repeatedly until there is enough space to cache the new object. However, MARS calculates the spatial score by using the client's current location and velocity. It does not take into account the movement history of mobile clients or repeated patterns in their paths. In relation to MARS, MARS+ records client’s movement history in mobility profiles which are then used to detect regular client movement patterns. MARS+ takes advantage of the fact that users’ movements usually comprise of both regular and random movements. According to the authors, MARS+ is capable of accurately predicting the future location of the mobile clients involved. Using this information MARS+ makes better cache replacement decisions. It is assumed that the server periodically broadcasts invalidation reports to inform clients of updates at the server. In the paper it has also been proved with the help of test results that MARS+ provides efficient cache replacement for mobile clients and is able to achieve 16% improvement in cache hit ratio as compared to existing replacement policies.

In this paper the authors propose a system called "MobiScape", consisting of a fixed host and a mobile one, typically a Notebook or a Personal Digital Assistant. The study assumes that the fixed host will have a permanent Internet connection and will act like a Support Station (SS). MobiScape relies on SS for reliable Internet connection and storage capabilities. The research proposes to use two caching schemes for WWW documents, one on Mobile Host (MH) and another one on the Support Station (SS), where MH will use SS in order to access Web.
In order to guarantee the caching needs of the users MobiScape is modeled to use two policies. First the user’s interaction is recorded which stores every newly received document by means of a cache that is not yet present in the cache. Secondly, the system also enables the user to specify a set of important documents that he or she would like to have permanently in cache, thus determining a profiling policy.

The system uses a storage and network demanding profile on SS, as fixed hosts are likely to have cheap internet connection and higher storage capabilities in relation to mobile hosts. While on MH a subset of SS profiling script is used in order to save storage space and network. When a request is made for a document by the user, MobiScape first searches the MH cache, if the document is not found then the request is forwarded to SS proxy where the same procedure is repeated. If the document is found in the cache then reference count is incremented. Moreover, recycling times are used at SS in order to update the cache on regular basis. This two-tiered proposed profile and cache policy fits well in a mobile environment. The overall system is depicted in figure below:

![Figure 3: The MobiScape Model](image-url)
Limitations of Current Techniques:

There are some inherent characteristics of mobile users which are not addressed by the proposed solutions discussed above, resulting in following limitations:

a. Impact of changes in geographical location of the user on user data priority and requirements.
b. Changes in user data requirements at different times of the day.
c. Frequent changes in quality and type of connection.
d. Data utility in relation to device document processing capabilities.
e. Integration of user meetings information with movements.
Chapter 3

Application Domain

This thesis describes a new method of calculating data priority by using adaptive mobile user and device profile which changes with user location, time of the day, available networks and data access history. The profile is used for data prefetching, selection of most suitable network and cache management on the mobile device in order to optimally utilize the device’s storage capacity and bandwidth. Although the proposed method can be used in any scenario, where user’s movements, device and wireless network characteristics along with user’s appointment information is accessible. However, it was decided that the design, implementation and refinement of simulation environment and system should fit the real estate sales person domain. This chapter provides a comprehensive description of the Real Estate Sales Person domain as it greatly influences this work. The system evaluation also has been described in terms of this domain in Chapter 5.

Mobile Real Estate Sales Person:

In real estate the sale representatives are required to travel on regular basis for meetings with clients. In this context we have assumed that there are product types such as Units, Houses, Land, Shops and Offices which belong to different categories like Residential, Commercial or Investment properties. Each product’s complete type consists of property type and property category and for each type of product random number of products will be instantiated e.g. Unit number 1/5 in Manahan St, Parramatta will be considered as an instance of property type Residential Unit. Moreover, it is assumed that each customer’s interest in properties will include different instances of properties belonging to only one product type, while each employee’s specialist area will belong to only one category of products e.g. employee specialist in Commercial properties will deal with both Shops and Offices.
Clearly the above assumptions complement and match real world’s real estate properties and the involved parties’ characteristics.

Consider a real estate company’s sale representative, Michael, who regularly visits the clients at remote locations and needs to access the information wirelessly. Let us assume that for wireless access and processing of data, Michael is equipped with Compaq iPaq with WLAN and GPRS wireless network capabilities with the storage capacity of 50 Mega Bytes and that Michael is traveling for a series of meetings with the potential customers i.e. Mr. John, Bill and Miss Kathy respectively. These customers are interested in different products and the data required for each meeting is 35, 30 and 25 Mega Bytes respectively.

In the conventional system Michael is likely to copy all the data he will need for all three meetings by anticipating the customer’s needs and requirements, as storage capacity has never been a major issue in standard desktops. However, in the case of mobile device before each appointment Michael will be required to delete the obsolete data which he does not need for next appointment and copy the required data for next meeting(s) by using wired network. This is because of the fact that Michael cannot copy all the required data needed for all three meetings as the total size of the required data exceeds the mobile device storage capacity.

In situations where the device storage capacity is less than the current and future data needs and there is no guarantee of stable wireless connection at the meeting sites, Michael will be forced to either manually use a wireless network, if available at all, in order to access and download the required data from the source or visit his office after each meeting in order to delete the obsolete data and copy the data needed for next meeting(s). For example after the meeting with John, Michael will need to copy the data required for the meeting with Bill, as the utility of data related to John will be obsolete once the meeting is over and likewise the same steps will be required for the meeting with Kathy once Bill’s meeting is over. In both the cases Michael will have to manually delete the data which he does not require anymore and copy the data required for next meeting(s). Moreover if Michael decides to move to the next meeting without the required information at hand, he is unlikely to work efficiently due to the absence of the required data. This can happen if the current bandwidth of
the wireless connection is less than required or if there is no supported wireless networks at the meeting location leading to embarrassment, frustration and loss of sales. Or else he will have to visit office after each meeting only to copy the required data which result in wastage of time, resources and energy due to extra travel. This will essentially lead to dramatic reduction in employee’s performance dropping his capacity to visit and cater the needs of the company’s customers.

The above scenario highlights three facts:

1. The utility of data objects which are only related to a certain customer or meeting is likely to drop to zero once the meeting ends, provided they are not required in the future.

2. The user of the mobile device cannot entirely rely on the availability of a stable supported wireless network at the meeting location as the required wireless network may not be entirely available or may be unstable at certain times or at certain locations.

3. If the user copies all the data related to a customer or meeting, some of it may not be supported through a software installed on the currently used mobile device and he may just end up wasting precious storage space. For example, smart phones currently do not support PDF documents.
Chapter 4

Implementation

4.1 Brief Overview

This thesis describes a new method of calculating data priority by using adaptive mobile user and device profiles which change with user location, time of the day, available networks and data access history. The profile is used for data prefetching, selection of most suitable network and cache management on the mobile device in order to optimally utilize the device’s storage capacity and bandwidth.

The main system is divided into five major components as outlined below:

1. Cache Management System

Cache management system provides enough storage space to prefetching system to enable the system to store data objects with highest utility, while keeping the data objects in the cache with highest utility or the data objects which are most likely to be requested again.

2. Device Profiling System

Device profiling system is used to enable the cache management system and prefetching system to access the device storage capabilities including the supported networks by the device and the MIME and document types supported by the mobile device.
3. Location Profiling System

Location profiling system is an integral part of the device system. Location profile system tracks the available wireless networks at each location and their performance.

4. User Profiling System

User profiling system is accessed by both cache management system and prefetching system. This system enables the prefetching system to access the weights assigned by the user to different data sets or domains. These weights are assigned by the user to data domains or to product types/categories instead of the weights which are assigned to different MIME or document types.

5. Prefetching System

Prefetching system gathers and uses movements, appointment and user’s current location information along with active wireless network properties and current mobile device profile information in order to determine the data objects needed to be pre-cached on the system.

4.2 Broad Objective of the System:

The broad objective of our system is to prefetch and cache the data objects on the mobile device which the user is likely to need in future or may access frequently thus enabling the mobile device user to use it effectively and efficiently. The system is designed to overcome mainly the following problems:

1. Frequent disconnections
2. Low bandwidth availability at certain locations
3. Not covered or only poorly covered areas
4. Limited storage capacity of Mobile Device
5. Burst-like network access
6. Limited or varying MIME type and document support by mobile devices

4.3 Core Objectives of the System:

The core objectives of this project are:

- Propose and develop a caching algorithm which will optimally utilize the device’s limited storage space by
  
  - Removing the data objects which the user might not need anymore
  - Keeping the most frequently used data objects in the cache, resulting in lesser downloads
  - Keeping the data objects cached which the user might need for next meeting
  - Making enough space available on the device in order to enable the prefetching system to save the data items with highest utility/weight

- Propose and develop a prefetching algorithm which will determine the objects to be cached and perfected on the device by:
  
  - Looking at the user’s current and previous locations, time of the day, user’s appointment information, and user’s customers profile and interest
  - Integrating device profile information consisting of available storage space, total storage capacity of the device and the device capacity to handle and process different MIME types
  - Integrating the wireless network properties currently available at user’s current location
  - Assisting the device to select the best suitable supported network at each new location
  - Integrating the priority set by the employee to each data domain and/or data subsets
4.4 Device Profiling System

This system is an integral part of the overall system. It is designed to run on the mobile device. The primary role of the device profiling system is to provide accurate and up-to-date information to the prefetching system. This system is also responsible for managing cache i.e. removal and/or addition of data objects on device cache and selection of appropriate wireless network at each new location. This system also tracks the history of cached data access by the user.

Device profiling system deals with the following aspects of the device:

1. Device Location
2. Device Supported Networks
3. Properties of Supported Networks
4. Available Networks and their Properties at Current Location
5. Available Data in Cache
6. Total Available Size of the Cache
7. Access History of Cached Data
8. MIME/Document Support of the Device
9. Priority Assigned by the User to Each Document/MIME Type
10. Priority Assigned by the User to Each Data Object Domain

The above mentioned information is used by the Device Profiling System for network selection and cache management. Since the device profiling system covers a wide area of device related functions and properties therefore this component is subdivided into three core components, where each component is responsible for a specific function.
4.4.1 Document (MIME) Support System

Document/MIME support information forms a major part of the mobile device profile. This system maintains and configures the types of documents supported by the device. This system provides the document support information to the Prefetching System and Cache Management system so that appropriate weights could be calculated for each data object. This system also enables the end user to assign different weights to each document or MIME type, resulting in fine tuning of the overall system. This information plays very crucial role when:

1) the bandwidth is very low so only a subset of required information can be downloaded from the server.
2) the storage size on the device is limited and only a subset of required information can be stored on the device.

4.4.2 Cache Management System

This system is used to maintain the data to be stored and/or removed from the device cache. The overall system works by calculating the weights for each data object. A data object can be only text from the document, without any formatting or pictures. The weights are calculated on the basis of:

1. Access frequency of cached data objects by the user
2. Priority assigned by the user to each information domain
3. User assigned weights to each document or MIME type

However, this system makes sure that it always provides free space required by the prefetching system as long as the required space is less than the device storage capacity. The prefetching system is an independent system, which first determines the data objects the user currently needs or may need in future. Then the system calculates the weights and the total space the data objects will require for storage on the device. This information is sent to the device. In response to the information received, the cache management system deletes the data objects with lowest weights until there is enough free space.
space, so that prefetching system can cache the data objects on the mobile device. The interesting fact here is that two weight/priority assigning systems are used side by side. One deals with cache management of the device while the other only arranges the data needed to be pushed to the device.

4.4.3 **Automatic Network Selection System**

Estimation and prediction of the performance and availability of wireless networks at all the locations a user may visit leads to very complex and resource hungry calculations. As the availability and condition of wireless networks depend on factors which cannot be estimated accurately. Therefore in order to keep things simple and practical, a system which runs on the mobile devices with a very small footprint is developed. This system scans all the available wireless networks supported by the device at current device’s location. Once it gathers the list of all the supported networks, it requests the Location Network Profiling System for each wireless network’s current bandwidth, condition and service status. Once the network selection system knows the network with best connection, it sends the request to cache management and prefetching system to use most suitable network. The device in response, automatically selects the directed network. On the basis of currently selected network, prefetching system calculates the order in which the data objects will be pushed to the device.

4.5 **User Profiling System**

User profiling system is solely used to maintain information about the user’s information interests, user’s priorities in terms of data domains, user’s assigned weights to each data and MIME or document type. This system stores the information on both the server and the mobile device. The cache management system accesses the user profile in order to decide which data objects should be deleted from the cache, while the prefetching system uses user profile along with device profile in order to decide which objects it should push to the device for caching. Some aspects of the User profiling system are configured by the user, while others are
automatically configured and developed by integrating information from the device and user’s appointment. The main parts of User profiling system are:

1. Weights Assigned by the User to Each Information Domain
2. Weights Assigned by the User to Each Document or MIME Type
3. Data Objects Required by the User at Different Times of the Day

On the first execution, the system requests the user to enter weights for each information domain, followed by requests for weights for each data/ MIME type. This information can be changed by the user at anytime. Meanwhile the needs of the user in terms of data objects are calculated by the system at the runtime. The system estimates the data objects user will need on the basis of user’s current location, movement, time of the day and available appointment information. Upon estimation, the user profiling system then passes a request to the prefetching system, where the weight for each data object is calculated and the data objects to be pushed to the mobile device are sorted in descending order. The User profiling system’s estimation engine produces very accurate results as it uses the appointment location, appointment time, and information requested by the customer and user movements information for estimates. From the above information user’s current and next appointments are determined. User profiling system then uses the appointment information to determine the set of data objects the user will need by getting product and customer information, thus enabling the system to estimate the data objects the user will require.

4.6 Context Aware Prefetching

Context aware prefetching system gets the following information from User Profiling System:

1. The Data Objects User will need for Next or Current Appointment at Predefined Location and Time.
2. User Assigned Weights to each MIME/Document Type.
3. User Assigned Weights to each Information Domain i.e. Data Related to a Particular Product or a Set of Products.

And the following information is collected from Device Profiling System

1. Device’s Total Cache Capacity
2. Available Cache Space on the Current Mobile Device
3. Device Location and Movement
4. List of Supported MIME/Document Types

On the basis of the above information, this system first determines the data objects which must be prefetched on the device. Once the data objects to be cached are determined, the device related information is used by this system in order to discard the data objects which the device does not support. The final subset of data objects is then sorted by using user assigned weights. The sorted final subset is then matched with the data objects currently cached on the device. Already cached data objects are removed from the prefetching list and the device is requested to free enough space so that the first data object on the prefetching list could be stored on the mobile device.

The Cache Management System is designed to entertain all the data storage requests it receives from Prefetching system. After each data object is prefetched on the device and if there is not enough space available to store the next data object then the Device’s Cache Management System frees more space by deleting a data object with lowest calculated weight. While deleting the data objects, the Cache Management System uses Equation 1 in order to calculate weights for each data object currently residing on the device’s cache. The data object with lowest weight is always removed first.

\[
\text{Cached Data Weight} = \text{Frequency} \times \text{Weight} \times P
\]

Equation 1: Cached data weight equation

Where

\text{Frequency} = \text{Number of times a data packet is accessed by the user from the mobile device cache}
**Weight** = Allocated weight for each product type or data object domain by the user

**P** = Priority of data packed based on the MIME/document type priority

However the Context Aware Prefetching System uses Equation 2 to determine the order in which data objects will be prefetched on the device. Refer to Equation 2:

\[
\text{Prefetch Data Object Weight} = (\text{Weight} \times P) \times S
\]

**Equation 2: Prefetch data packet weight equation**

Where

**S** = Device’s support for document or MIME type. ‘S’ can be either 0 or 1. It is 0 if the current mobile device does not support the mime or document type associated with the packets else its value is 1.

**P** = Priority of data packet based on the Mime type priority.

**Weight** = Allocated weight for each product type or data object domain by the user.

**Knapsack**

On the basis of data weights Knapsack [18] is used to determine the selection and sequence of data objects to be prefetched. Need for Knapsack arises whenever there is a resource allocation constraint with different cost and value of objects in question. For example, in the proposed system there is storage constraint i.e. there is fixed storage space on the mobile device and there are data objects with various weights and sizes, also each data object cannot be further subdivided. On the basis of the above conditions Knapsack enables the system to select a set of data objects which must be prefetched on
the mobile device so that we get maximum aggregate weight (utility) while keeping the size of prefetched items less than or equal to available storage space on the mobile device.

4.7 Simulator Architecture

As already stated in Chapter 1, the system was tested with the help of simulator modeling. The Simulator developed for this project, not only simulates the complete environment, but it also implements the complete system for evaluation, testing and refinement. Simulator was developed by using Java 2 Standard Edition and MySQL database. I have divided the simulator into three layers, named below. Each layer’s details and complete in-depth system architecture is covered later in this chapter and in Appendix A.

4.7.1 Static Data Layer
4.7.2 Runtime Information Generator
4.7.3 Runtime Environment
   4.7.3.1 Runtime Conventional System Layer
   4.7.3.2 Runtime Proposed System Layer

**Structure of Layers**

![Layered Structure of Simulator](image)

**Figure 4: Layered Structure of Simulator**
4.7.1 Static Data Layer

This layer stores information which remains constant during the simulator execution cycle. The information is stored in a Relational Database and is used by runtime information generator to generate the runtime data by using random numbers, which changes with each execution cycle. Every effort was made to keep the static data as less as possible. The structure and contents of static data layer are outlined below.
Figure 5: Tables and their Relations in Static Layer
Device

Device table stores information regarding all the Mobile Devices available to employees. The stored information includes each device’s cache capacity and title. This table currently holds only three mobile devices, i.e. Laptop, Mobile phone and PDA with 51200, 30720 and 16384 bytes respectively.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key of each device</td>
</tr>
<tr>
<td>cacheCapacity</td>
<td>The total cache capacity of the mobile device</td>
</tr>
<tr>
<td>title</td>
<td>The String representation of the Mobile device</td>
</tr>
</tbody>
</table>

Employee

Arbitrary employee information is stored in this table. The Employee table only stores the last and first names of the employees. No information related to employee’s appointments, interests and priorities is saved in this layer.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key of each employee</td>
</tr>
<tr>
<td>fName</td>
<td>Employee’s first name</td>
</tr>
<tr>
<td>lName</td>
<td>Employee’s last name</td>
</tr>
</tbody>
</table>

Location

The Location table stores the geographical information of all the suburbs. In this table, like employee’s table, arbitrary suburbs information is stored. Each suburb’s neighboring (adjacent) suburb(s) information is also stored in this table which represents all possible user movements. It was assumed that each suburb will have Octagon shape, so that every suburb will have either eight or three direct links to neighboring suburbs. Here I have assumed that the
maximum number of suburbs is thirty nine, however more suburbs can be added to this table, if desired, without making any changes in the implementation code. Figure 1 gives visual layout of all the suburbs and the arrows represent all possible movements a user can have while traveling from one suburb to another. It is assumed here that the user cannot skip a suburb while traveling from Suburb A to Suburb B. As stated above each suburb has maximum eight neighboring suburbs e.g. Suburb 20 is connected to Suburb 18, 17, 16, 19, 21, 22, 23 and 24, while Suburb 39 shares boundary with three suburbs i.e. Suburb 36, 35 and 38.

Each Suburb's Possible Connecting Points

Figure 6: Suburb Structure an Shape

Figure 7: Inter Suburb Links and User Movements
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key of each Location or Suburb</td>
</tr>
<tr>
<td>postCode</td>
<td>Arbitrary post code assigned to each Suburb</td>
</tr>
<tr>
<td>title</td>
<td>Suburb name e.g. Suburb 1</td>
</tr>
<tr>
<td>x1 *</td>
<td>ID of the base suburb</td>
</tr>
<tr>
<td>x2 *</td>
<td>ID of the upper suburb</td>
</tr>
<tr>
<td>y1 *</td>
<td>ID of the left suburb</td>
</tr>
<tr>
<td>y2 *</td>
<td>ID of the right suburb</td>
</tr>
<tr>
<td>a *</td>
<td>ID of the lower left suburb</td>
</tr>
<tr>
<td>b *</td>
<td>ID of the lower right suburb</td>
</tr>
<tr>
<td>c *</td>
<td>ID of the upper left suburb</td>
</tr>
<tr>
<td>d *</td>
<td>ID of the upper right suburb</td>
</tr>
</tbody>
</table>

* The value of these variables is set to -1 there is no connecting suburb.

**Mime Type**

The MIME type table is used to store available MIME types information including their file extensions and corresponding priorities.

<table>
<thead>
<tr>
<th>Extension</th>
<th>MIME type</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bmp</td>
<td>image/bmp</td>
<td>1</td>
</tr>
<tr>
<td>.gif</td>
<td>image/gif</td>
<td>1</td>
</tr>
<tr>
<td>.jpg</td>
<td>image/jpeg</td>
<td>1</td>
</tr>
<tr>
<td>.swf</td>
<td>application/x-shockwave-flash</td>
<td>1</td>
</tr>
<tr>
<td>.ppt</td>
<td>application/mspowerpoint</td>
<td>2</td>
</tr>
<tr>
<td>.css</td>
<td>text/css</td>
<td>3</td>
</tr>
<tr>
<td>.txt</td>
<td>text/plain</td>
<td>3</td>
</tr>
<tr>
<td>.js</td>
<td>application/x/javascript</td>
<td>3</td>
</tr>
<tr>
<td>.pdf</td>
<td>application/pdf</td>
<td>4</td>
</tr>
<tr>
<td>.doc</td>
<td>application/msword</td>
<td>4</td>
</tr>
<tr>
<td>.html</td>
<td>text/html</td>
<td>5</td>
</tr>
</tbody>
</table>
It was assumed here that MIME type priority value varies from 1 – 5, where 1 represents the lowest priority and value 5 represents the highest priority.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key of each Mime Type</td>
</tr>
<tr>
<td>extension</td>
<td>Extension of each file type</td>
</tr>
<tr>
<td>title</td>
<td>MIME Type</td>
</tr>
<tr>
<td>priority_id</td>
<td>Priority Id (Foreign key)</td>
</tr>
</tbody>
</table>

Priority

This table only stores the ID (values) and string representation of each priority scale. This table only contains 5 rows where the Maximum priority represents an integer value 5 and lowest priority represents value 1.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key and Value of each Priority</td>
</tr>
<tr>
<td>title</td>
<td>String representation of each Priority level</td>
</tr>
</tbody>
</table>

Network Type

This table is used to store the list and properties of all the wireless networks available within the application. Here it is assumed that there are only four networks i.e. GPRS, 802.11a, 802.11b and 802.11g. This table also stores the maximum supported bandwidth by each network. However, the effective bandwidth of each wireless network, at specific location and time is randomly generated which varies from zero to ½ time’s maximum theoretical bandwidth.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key of each Network Type</td>
</tr>
<tr>
<td>Column Name</td>
<td>Usage</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>product_id</td>
<td>Primary key of each Product Type</td>
</tr>
<tr>
<td>prod_desc</td>
<td>String representation of each product type</td>
</tr>
<tr>
<td>type</td>
<td>Category of the Product</td>
</tr>
</tbody>
</table>

**Product**

Product table stores information regarding the type of products available within the application. This table only stores the product type and their corresponding categories rather than actual product’s information. For example, this table only stores that the company deals in products like Residential Units and Commercial Shops. However, the information related to each particular product e.g. Residential Unit 7/58 Rosewood Crest is for sale is stored in a separate table called “Product Profile”. The data specified in this table is used to assign certain product categories to each employee as their specialist area or domain. For example, Michael could be profiled to handle all the requests related to Commercial Properties in Suburbs 20 – 29 etc.

**Product Profile**

Product Profile table is used to store the information regarding the actual product instances available within the system. The information stored in this table provides description regarding each product’s instance containing type/category and location e.g. Shop Number 7 in Opera House Sydney is for Sale which is of type shop, and belongs to the commercial property category.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary key of each Product Instance</td>
</tr>
</tbody>
</table>
Employee Profile

This table is used to assign each product instance to exactly one employee, so that the product category and type belongs to employee’s defined specialist area or domain.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee_id</td>
<td>Employee Id (Foreign Key)</td>
</tr>
<tr>
<td>product_id</td>
<td>Product Id (Foreign Key product Profile)</td>
</tr>
<tr>
<td>location_id</td>
<td>Location Id (Foreign Key)</td>
</tr>
</tbody>
</table>

Device MIME/Document Type Support

This table is used to store the list of document/MIME types supported by each mobile device. Device MIME/Document type supports table is populated on the basis of specifications of currently available mobile devices to the local market, for example a Laptop supports all the documents while a PDA does not support JavaScript, PDF and PPT.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>device_id</td>
<td>Device ID (Foreign Key)</td>
</tr>
<tr>
<td>mimeType_id</td>
<td>MIME Type ID (Foreign Key)</td>
</tr>
</tbody>
</table>

Device Network Support

Device network support table is used to store the list of network types supported by each mobile device. The network type support allocation is
based on the specifications of currently available device in the market, for example a Laptop supports all 4 types of networks i.e. 802.11a, 802.11b, 802.11g and GPRS while a mobile phone only supports GPRS.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>device_id</td>
<td>Device ID (Foreign Key)</td>
</tr>
<tr>
<td>networkType_id</td>
<td>Network ID (Foreign Key)</td>
</tr>
</tbody>
</table>

4.7.2 Runtime Information Generator

Runtime information generator consists of Java Classes which get the data from static data layer and generates new data by using random numbers. The data generated by runtime data generator changes for each execution cycle but the structure and integrity of generated data is maintained as it is generated in accordance to the rules and semantics of the overall system. The randomly generated data is stored in a new Database which is used by Runtime Environment.

Implementation and Outcome

Below is provided a detailed description of tables, their relations, generated data characteristics, assumptions and description of main instructions used for data generation. In this section I have divided the complete architecture into subsections, where each section outlines the structure of the relevant tables and explanation of instructions followed for data generation.
Figure 8: Table and Their Relations in Runtime Layer
**Employee**

The Employee table is generated after reading each employee’s first name, last name and id from static layer employee table. After reading each employee’s data, each employee’s available time and starting location is randomly generated.

1. Since humans do not start and end their work at exactly the same time everyday therefore in order to simulate this behavior for the table random starting time between 8:00 – 10:59 AM and random end time between 14:00 – 18:59 is generated for each employee. These starting and end times represent the time during which the employee will be available for meetings with clients.

2. After allocating random start and end available times to each employee, Suburbs ID’s are read and randomly selected for each employee from the Location table in the static layer. The randomly selected Location ID in the Employee table represents the starting location of the employee, i.e. the point from where we will start tracking the employee movements. Moreover, it is also used to determine employee’s route for the first appointment.

**Table:**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Employee Id Primary Key</td>
</tr>
<tr>
<td>fName</td>
<td>First Name of the Employee</td>
</tr>
<tr>
<td>lName</td>
<td>Last Name of the Employee</td>
</tr>
<tr>
<td>startLocation</td>
<td>Starting Location of Employee</td>
</tr>
<tr>
<td>availableStartTime</td>
<td>Available Start time of the Employee for meetings</td>
</tr>
<tr>
<td>availableEndTime</td>
<td>Available End time of the Employee for meetings</td>
</tr>
</tbody>
</table>
Employee Has Device

This table stores the information regarding the mobile devices which are available to each employee. For the data generation of this table, first ID’s of all the mobile devices are read from the Device table in static layer and randomly assigned to each employee, thus allocating different types of mobile devices to each employee, moreover in order to make the environment more realistic we assign random number of devices i.e. between 1 and 3 mobile devices to each employee.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee_id</td>
<td>Employee Id (Foreign Key)</td>
</tr>
<tr>
<td>device_id</td>
<td>Device Id (Foreign Key)</td>
</tr>
</tbody>
</table>

Customers

Arbitrary customer’s credentials consisting of ID, First Name and Last Name are generated and stored in the Customer table. Currently the application is designed to generate and store sixty customers information. However this can be changed.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Customer Id (Primary Key)</td>
</tr>
<tr>
<td>fName</td>
<td>Customer First Name</td>
</tr>
<tr>
<td>lName</td>
<td>Customer Last Name</td>
</tr>
</tbody>
</table>
Customer Profile

The Customer Profile table stores the information regarding the customer and the products they are interested in. While developing Customer Profile, it was made sure that each customer is interested in exactly one category of product e.g. Commercial or Residential or Investment property. In order to generate the above information, a randomly selected product type is assigned to each customer. Then random number of product instances between 1 and $\frac{1}{2} \times$ total number of products, belonging to customer’s interested product type are selected from the Product Profile table. The Location and product category ID of each randomly selected product instance is then stored in this table, thus enabling us to implement consistent taste and interest for each customer.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Sequence Number Primary Key</td>
</tr>
<tr>
<td>customer_id</td>
<td>Customer ID (Foreign Key)</td>
</tr>
<tr>
<td>product_id</td>
<td>Product Type ID (Foreign Key)</td>
</tr>
<tr>
<td>location_id</td>
<td>Location ID (Foreign Key)</td>
</tr>
</tbody>
</table>

Appointments

On the basis of each employee’s starting time, location, end available time, each customer’s profile and product profile the Appointment Table is populated with the following information:

1. Random number of appointments for each employee with random number of Customers
2. At random locations
3. With random customers
4. For random products
5. Random mobile device is assigned to the employee for each day
6. Random time is allocated to each employee for traveling from location A to B
7. Random appointment duration is assigned for each appointment

While making sure that:

1. The appointment start and end time is between employee’s available times
2. There is no overlapping in terms of appointment times for any employee or customer
3. Product location match the location of the appointment
4. Appointment duration is between 20 minutes and 59 minutes
5. Travel time for each appointment is between 20 and 39 minutes
6. The product belongs to Customer’s and Employee’s Profile i.e. the product instance belongs to employee’s specialization area and is in customer’s profile

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Sequence Number of Appointment</td>
</tr>
<tr>
<td>startTime</td>
<td>Appointment Start Time</td>
</tr>
<tr>
<td>endTime</td>
<td>Appointment End Time</td>
</tr>
<tr>
<td>product_id</td>
<td>ID of the product</td>
</tr>
<tr>
<td>location_id</td>
<td>Appointment Location (Foreign Key)</td>
</tr>
<tr>
<td>customer_id</td>
<td>Customer ID (Foreign Key)</td>
</tr>
<tr>
<td>day</td>
<td>Appointment Day (1 – 5) where 1 represents Monday and 5 represents Friday</td>
</tr>
<tr>
<td>appointmentDuration</td>
<td>The duration of appointment (Start – End Time)</td>
</tr>
<tr>
<td>travelTime</td>
<td>Travel Time available to the employee for</td>
</tr>
</tbody>
</table>
On the basis of the time, day and location of each appointment, the movement of each employee for each appointment to the next is calculated and stored in this table. Since this work does not concentrate on user’s movements estimation therefore of the shortest path between two Suburbs is considered as employee route. After calculating user route, random travel time is assigned to each section of the employee route while making sure that the total travel time is exactly same as the available travel time, assigned to each user while generating appointment information.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>appointment_id</td>
<td>Appointment ID (Foreign Key)</td>
</tr>
<tr>
<td>employee_id</td>
<td>Employee ID (Foreign Key)</td>
</tr>
<tr>
<td>Day</td>
<td>Appointment Day</td>
</tr>
<tr>
<td>location_id</td>
<td>Location ID (Foreign Key)</td>
</tr>
<tr>
<td>TimeSpent</td>
<td>Time Spent while crossing a suburb</td>
</tr>
<tr>
<td>Sequence</td>
<td>Sequence of Movements</td>
</tr>
</tbody>
</table>

* Generated by the runtime layer therefore it is covered in the next section
Location Available Networks

In this table all the available networks within the system random number of networks are made available to each location. No information regarding the performance or status of wireless network is saved in this table, as this information is generated and assigned at the runtime, when the environment simulation executes.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>location_id</td>
<td>Location ID (Foreign Key)</td>
</tr>
<tr>
<td>network_id</td>
<td>Network ID (Foreign Key)</td>
</tr>
</tbody>
</table>

Employee Locations

Employee Location table is populated after matching each appointment’s information with user route and available networks at each suburb. The employee predicted locations table is used by the third layer to calculate user data access needs at each suburb both while traveling and during the appointment. The main purpose of this table is to provide the most commonly used data required by the runtime environment in a concise and organized way resulting in optimal execution of runtime environment.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary Key (Sequential Number)</td>
</tr>
<tr>
<td>appointment_id</td>
<td>Appointment ID (Foreign Key)</td>
</tr>
<tr>
<td>employee_id</td>
<td>Employee ID (Foreign Key)</td>
</tr>
<tr>
<td>product_id</td>
<td>Product ID (Foreign Key)</td>
</tr>
</tbody>
</table>
Product Data

Product data table is used to store detailed information regarding the data of each product including its contents and their sizes. The total data size and the types of documents/MIME in each product’s data are calculated by using random numbers. In order to generate random product data, first all the products ID’s and MIME ID’s are read from the relevant tables in static layer. Then minimum and maximum limits for product’s data size are assigned between 25% - 33% of minimum cache capacity among all the mobile devices. Although the above assumption can be changed, but this is assumed in order to evaluate and test the impact and effectiveness of prefetching and caching system. After determining random total data for each product, random number of MIME/document types with random sizes are set as contents of each product, so that the sum of all the MIME/document types size is same as the products total data size.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Primary Key (Sequential Number)</td>
</tr>
<tr>
<td>p_id</td>
<td>Product ID (Foreign Key)</td>
</tr>
<tr>
<td>mimeId</td>
<td>MIME ID (Foreign Key)</td>
</tr>
<tr>
<td>size</td>
<td>Size in bytes</td>
</tr>
</tbody>
</table>
Assumptions:

1. The employee’s active mobile device remains same for the whole day
2. Each Product’s total data size must be between 25% - 33% of minimum cache capacity among all the mobile devices
3. At each location, at least one wireless network will be made available
4. Each employee will have at least one mobile device
5. The generated travel time for each employee will vary between 20 and 39 minutes
6. The appointment will always be between 20 and 59 minutes
7. Each appointment will be between exactly one customer and employee regarding one product instance
8. Product data and its contents are not changed during the course of the day
9. Employee cannot manually reset the device cache
10. Each mobile device will have build-in support for at least one wireless network

4.7.3 Runtime Environment Layer

Runtime environment layer is used to simulate the environment by reading the randomly generated data and static data from the database. In this layer, following objects are instantiated:

1. Device Objects – representing each type of mobile device
   The Device objects are latter cloned to represent the device’s different instances operated by different employees
   - Cache Capacity
   - Available Cache
   - Title
   - Supported Networks : Network [] (Array of Network Objects)
   - Supported Mime Types : Mime Type [] (Array of Mime Type Objects)
   - Cache Data : CacheData – Vector of Cached Data Objects
2. Device Cache Data (Inner Class of Device Class) – representing each data object in the device cache
   - Data Mime Type ID
   - Size in bytes
   - Product Id
   - Frequency of Access – number of time a data object is accessed by the user from the cache

3. Location Objects – representing each suburb
   - Title
   - Post Code
   - Available Networks : Network [] (Array of Network Objects)

4. Network Objects – representing each type of wireless network
   The Network objects are cloned for each containing suburb (location). As the same wireless network will have different bandwidth and status at different locations
   - Title
   - Max Bandwidth
   - Current Bandwidth

5. Employee Objects – representing each employee of the company
   - Current Device
   - Name
   - Current Location : Location
   - Next Location : Location
   - Available Devices : Device [] (Array of Device Objects)
   - Current Product : Product - The reference of the product required either for the current or next appointment
   - My Products : Product[] - (Array of products which are in Employee Profile)
• Product Weight : Integer [] [] (Weights assigned to each product by the Employee – Integer [Product ID] [Weight] )

6. Product Object – representing each product’s instance object
   • Product Type Id – Id of the Product Type (Category) this product belongs to
   • My Location : Location – Location of the Product
   • Product Data : ProductData – [] (Array of Product Data)

7. Products data – representing each product’s contents
   • Mime Type : MimeType – Reference of Mime Object representing the type of this data object
   • Size in bytes

8. MIME Objects – representing each MIME type
   • Priority – This value is read and assigned from the static layer mime type table. The priority of each MIME type is same for all employees
   • Title
   • Extension – File extension e.g. doc, txt pdf etc

9. Customer Objects – representing each customer
   • My Products : Product [] (Array of products the customer is interested in)
   • Name
   • ID

Cloned Objects:
Device and Network objects are cloned and assigned to the Employee and Location objects respectively, as the using entities require a copy of the actual object.
Please refer to the UML – Object Diagram, representing the Runtime Environment visually and a Class Detailing Diagram outlining each class’s methods and properties.
Figure 9: UML - Object Diagram of Runtime Environment
Once the objects are instantiated and their relations are assigned, two simulation components are executed. The first component, i.e., Runtime Conventional System Simulation, uses the same data to simulate user data usage, access, and movements by integrating our proposed method of prefetching and caching, thus enabling us to compare the outcome and performance of both the systems. In the next
section, first I have outlined the semantics and structure of Conventional System Simulation followed by the detailed architecture of Proposed System Simulation.

Please note that both systems use the same generated data for execution. The idea behind using two separate layers is to make sure that the simulation runs at optimal speed by limiting number of database calls while keeping the simulation separate from the main runtime environment.

**4.7.3.1 Conventional System Simulation**

In this module of simulation no profiling is used for prefetching or caching, however user’s data access is recorded in order to calculate the total utility of accessed data. In this module we first read each appointment’s information in a sequential manner and move the relevant employee to the appointment location. On the basis of the location, the list of available wireless networks is retrieved and random number as current bandwidth is assigned to each wireless network varying between 0 and 50% of maximum possible bandwidth of that network. Then the user’s current mobile device reference is retrieved from the runtime environment in order to determine the network to be used which is available and is supported by the current mobile device with best current bandwidth. Once we have determined the network to be used for data transfer, the contents of the product data are retrieved from the product object and data transfer is simulated until one of the following conditions is true:

1. There is at least one wireless network which is available and supported by the current device
2. The difference between available time and used time at current location is more than zero
3. There is still more product data which is needed to be transferred to the mobile device
Once the data transfer stops due to one of the above the following information is recorded into the database which is later used to compare the performance of our proposed system with the conventional system.

**Information Recorded:**

i. Total size of the transferred data
ii. Details of data objects which could not be transferred
iii. Total Utility of all the transferred data objects is calculated by using Equation 3:

\[ U = (P \times \text{Weight}) \times S \]

*Equation 3: Utility per data object*

Where:

- **Utility** = Utility per data object
- **P** = Product Data Object’s MIME Priority
- **Weight** = Weight Assigned by the User to the Product Type or Category
- **S** = Device Support for the data object i.e. document or MIME type. S can be either 0 or 1. It is one if the device supports the document else it is set to zero, the result is that utility becomes zero as user will not be able to access or use the data object by using the current mobile device

An important point regarding the above equation is that the utility for the data object is set to zero, irrespective of the values of P and Weight if the data object type is not supported by the current mobile device. During the execution of this system the following tables are created and/or modified:
Missed Data without Profile:

This table stores the detailed list of all the data objects including each data’s object size, appointment id, MIME id, device id and product id which the employee needed for the appointment but could not access.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>appoint_id</td>
<td>Appointment ID (Foreign Key)</td>
</tr>
<tr>
<td>mime_id</td>
<td>MIME Type ID (Foreign Key)</td>
</tr>
<tr>
<td>product_id</td>
<td>Product ID (Foreign Key)</td>
</tr>
<tr>
<td>device_id</td>
<td>Device ID (Foreign Key)</td>
</tr>
<tr>
<td>size</td>
<td>Size in bytes</td>
</tr>
</tbody>
</table>

Utility of Product Data Access without Profile:

This table is used to store utility of each data object which was accessed by the employee. The Utility is calculated by using Equation 3. The utility for each data object is stored against each appointment.

Table:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>appointment_id</td>
<td>Appointment ID (Foreign Key)</td>
</tr>
<tr>
<td>utility</td>
<td>Utility of each accessed data object</td>
</tr>
</tbody>
</table>

The information from this table is further used to calculate and store the following information in the Appointment Table:
a. Aggregate Utility of Cached Data Without Using Profile for each Appointment.
b. Aggregate Time used to Download Data Objects without using profile for each Appointment.
c. Aggregate Size of Downloaded Data Without Using Profile for each Appointment.

The above information is used in Chapter 5 to compare and evaluate the proposed system with the conventional system.

4.7.3.2 Proposed System Simulation

Proposed System Simulation executes after the execution of conventional system simulation and uses the same random data which was used by the conventional system simulation. In the proposed system simulation there is an integration of the proposed method of prefetch and cache data objects on the mobile device thus resulting in increased utility of device. For simulation, in this section we first retrieve references of appointment, employee, customer, product, time spent at each location and current mobile device in use by the employee for each record in the employee’s route table. On the basis of this information the references of the currently available wireless networks at employee’s current location along with the references of the networks supported by the currently active mobile device are retrieved. The properties of networks are matched in order to determine best available bandwidth by finding a network which is available and is supported by the current mobile device with highest current data transfer rate or bandwidth. Then from product object total, product data size along with the data contents are retrieved in order to determine the time required to transfer all the data objects. If the required time to prefetch the data objects is less than the time the employee will spend at current location, then the system tries to prefetch all the data objects in sequence where the data objects with highest prefetch weights, calculated by using Equation 2, are transferred first followed by the objects with lesser weights. The above technique of ordering data objects
by using prefetch weight is used, even if the time required to transfer the
data objects is more than the available time as the wireless networks are
unstable and the system is designed to get most out of it by prefetching
the data objects which will yield the highest utility to the user of the
device.

Once the system identifies the data objects to be prefetched, this
information is passed to the Cache Management System. The cache
management system first searches the device cache for the data objects, if
the data object already exists on the mobile device cache, then it is not
downloaded from the server, else the prefetching system is notified which
pushes the data object to the device. The Cache Management System is
designed to make optimal use of device cache therefore it concentrates on
optimal use of the mobile device’s limited storage capacity. The Cache
Management System uses Knapsack, which uses the available storage
size, the size of each data object and their weight to determine the
sequence in which data objects will be cached; however for invalidation
of data objects the system uses the Equation 1.

In this the module employee’s movements, data access patterns and data
transfers to the mobile device are record in a temporary Cache table. The
records of the cache table are then used to populate following tables/columns.

**Missed Data with Profile:**

This table stores the detailed list of all the data objects including each data
object size, appointment id, MIME id, device id, employee’s current
location id and product id of the products which the employee needed for
each appointment but could not access.
Utility Per Product Data Downloaded With Profile:

This table is used to store utility of each data object which was accessed by the employee. The Utility is calculated by using Equation 3. The utility for each data object is stored against each appointment.

The information from this table is further used to calculate and store the following information in the appointment table:

a. Aggregate Utility of Cached Data by Using Profiles for each Appointment
b. Aggregate Time to Download Data Objects by Using Profiles for each Appointment
c. Aggregate Size of Data Downloaded by Using Profiles for each Appointment
The above information is used to compare the proposed system with the conventional system.

### 4.7.4 Simulator Performance

In this section the results from application profiling are provided. This information is used to identify the code segment which can be further optimized in future implementations in order to make the application more scalable. Please note that profiling itself requires a lot of resources, therefore the execution time and resources used by each method, recorded here, are significantly shorter than the profiling information.
<table>
<thead>
<tr>
<th>Method</th>
<th>Time [%]</th>
<th>Time</th>
<th>Time (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>31606 ms</td>
<td>31606 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.productsSetup ()</td>
<td>10030 ms</td>
<td>8939 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>7572 ms</td>
<td>7572 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>2509 ms</td>
<td>2509 ms</td>
<td>91.4 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.customerSetup ()</td>
<td>1509 ms</td>
<td>1509 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.employeeSetup ()</td>
<td>658 ms</td>
<td>658 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.locationsSetup ()</td>
<td>415 ms</td>
<td>415 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.deviceSetup ()</td>
<td>124 ms</td>
<td>124 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.networksSetup ()</td>
<td>107 ms</td>
<td>107 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.mimeTypesSetup ()</td>
<td>62.8 ms</td>
<td>62.8 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ClassLoader.loadClassInternal (String)</td>
<td>19.7 ms</td>
<td>19.7 ms</td>
<td>0.000 ms</td>
</tr>
</tbody>
</table>

**Self time**

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [%]</th>
<th>Time</th>
<th>Time (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.WeakHashMap.get (Object)</td>
<td>0.080 ms</td>
<td>0.080 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ClassLoader.checkPackageAccess (Class, java.security.ProtectionDomain)</td>
<td>0.054 ms</td>
<td>0.054 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.String.startsWith (String)</td>
<td>0.038 ms</td>
<td>0.038 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.Class.getName ()</td>
<td>0.028 ms</td>
<td>0.028 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>sun.management.ThreadImpl.currentThreadUserIDTime ()</td>
<td>0.026 ms</td>
<td>0.026 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.Class.getClassLoader ()</td>
<td>0.012 ms</td>
<td>0.012 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.String.equals (Object)</td>
<td>0.002 ms</td>
<td>0.002 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ref.Reference.get ()</td>
<td>0.002 ms</td>
<td>0.002 ms</td>
<td>0.000 ms</td>
</tr>
</tbody>
</table>

**Self time**

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [%]</th>
<th>Time</th>
<th>Time (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>31606 ms</td>
<td>31606 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.productsSetup ()</td>
<td>10030 ms</td>
<td>8939 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>7572 ms</td>
<td>7572 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>2509 ms</td>
<td>2509 ms</td>
<td>91.4 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.customerSetup ()</td>
<td>1509 ms</td>
<td>1509 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.employeeSetup ()</td>
<td>658 ms</td>
<td>658 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.locationsSetup ()</td>
<td>415 ms</td>
<td>415 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.deviceSetup ()</td>
<td>124 ms</td>
<td>124 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.networksSetup ()</td>
<td>107 ms</td>
<td>107 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.mimeTypesSetup ()</td>
<td>62.8 ms</td>
<td>62.8 ms</td>
<td>0.000 ms</td>
</tr>
</tbody>
</table>

**Self time**

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [%]</th>
<th>Time</th>
<th>Time (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.WeakHashMap.get (Object)</td>
<td>0.080 ms</td>
<td>0.080 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ClassLoader.checkPackageAccess (Class, java.security.ProtectionDomain)</td>
<td>0.054 ms</td>
<td>0.054 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.String.startsWith (String)</td>
<td>0.038 ms</td>
<td>0.038 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.Class.getName ()</td>
<td>0.028 ms</td>
<td>0.028 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>sun.management.ThreadImpl.currentThreadUserIDTime ()</td>
<td>0.026 ms</td>
<td>0.026 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.Class.getClassLoader ()</td>
<td>0.012 ms</td>
<td>0.012 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.String.equals (Object)</td>
<td>0.002 ms</td>
<td>0.002 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ref.Reference.get ()</td>
<td>0.002 ms</td>
<td>0.002 ms</td>
<td>0.000 ms</td>
</tr>
</tbody>
</table>

**Self time**

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [%]</th>
<th>Time</th>
<th>Time (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>31606 ms</td>
<td>31606 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.productsSetup ()</td>
<td>10030 ms</td>
<td>8939 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>7572 ms</td>
<td>7572 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.&lt;init&gt; ()</td>
<td>2509 ms</td>
<td>2509 ms</td>
<td>91.4 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.customerSetup ()</td>
<td>1509 ms</td>
<td>1509 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.employeeSetup ()</td>
<td>658 ms</td>
<td>658 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.locationsSetup ()</td>
<td>415 ms</td>
<td>415 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.deviceSetup ()</td>
<td>124 ms</td>
<td>124 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.networksSetup ()</td>
<td>107 ms</td>
<td>107 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>RunTimeEnv.MainRunTime.mimeTypesSetup ()</td>
<td>62.8 ms</td>
<td>62.8 ms</td>
<td>0.000 ms</td>
</tr>
</tbody>
</table>

**Self time**

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [%]</th>
<th>Time</th>
<th>Time (CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.WeakHashMap.get (Object)</td>
<td>0.080 ms</td>
<td>0.080 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ClassLoader.checkPackageAccess (Class, java.security.ProtectionDomain)</td>
<td>0.054 ms</td>
<td>0.054 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.String.startsWith (String)</td>
<td>0.038 ms</td>
<td>0.038 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.Class.getName ()</td>
<td>0.028 ms</td>
<td>0.028 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>sun.management.ThreadImpl.currentThreadUserIDTime ()</td>
<td>0.026 ms</td>
<td>0.026 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.Class.getClassLoader ()</td>
<td>0.012 ms</td>
<td>0.012 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.String.equals (Object)</td>
<td>0.002 ms</td>
<td>0.002 ms</td>
<td>0.000 ms</td>
</tr>
<tr>
<td>java.lang.ref.Reference.get ()</td>
<td>0.002 ms</td>
<td>0.002 ms</td>
<td>0.000 ms</td>
</tr>
</tbody>
</table>

**Self time**
From the profiling information it is clear that most of the execution time is used by the following three methods:

1. **Products Setup**: This method is used to instantiate and setup all the products and each product’s data objects. On further analysis it was identified that most of the execution time is used for database connections setup and for executing queries, as shown in Figure 11.

2. **Without Profile**: This method is used to simulate user movements, data access and usage without using any profiling. During the execution of this method most of the time is used to add records in the Appointment Table regarding data access, data utility and time used for data access per appointment. Please see figure 12 for reference.
3. **With Profile**: This method is used to simulate user movements, data access and usage by using profiling in order to prefetch and cache data objects on the mobile device. During the execution of this method most of the time is used by the Java Profiler and StringBuffer manipulations required to record cache contents of the mobile device. Please see figure 13 for reference.

After analyzing the profiling results I realized that the application execution can be improved dramatically by caching database connections, using prepared statements and triggers. However, profile results proved that the instructions related to environment simulation are optimized.
Chapter 5

Evaluation and Conclusion

This thesis describes a new method of calculating data priority by using adaptive mobile user and device profile which changes with user location, time of the day, available networks and data access history. The profile is used for data prefetching, selection of most suitable network and cache management on the mobile device in order to optimally utilize the device’s storage capacity and bandwidth. In the previous chapter we looked at the structure, semantics and implementation of the system. In this chapter I have evaluated the performance of the proposed system followed by its performance comparison with the conventional system.

5.1 Evaluation

The data generated by Conventional System and Proposed System Simulation enables us to analyze, evaluate and compare the results of our proposed system with the conventional system. The data stored during the execution of both Runtime Environments enables us to calculate following:

1. Total Utility from network usage Per Second by using Profile for all the appointments

2. Total Utility from network usage Per Second without using Profile for all the appointments

3. Total Utility per byte prefetched without using Profile for all the appointments

4. Total Utility per byte prefetched by using Profile for all the appointments
5. Percentage Improvement in Time Utilization

6. Percentage Improvement in Data Utility

Since the data generated for each test is different, therefore I have used one test result for analysis. The test results are given below:

Aggregate utility of accessed data without using profile for all appointments is sum of utility derived from the data downloaded on the mobile device is 31426.

Total time taken to download the required data without using profile for each appointment by all the employee’s is 106443 seconds.

Total data downloaded in the conventional system by all the users during the life span of simulation is 1524776 bytes.

From the above figures we first calculate utility derived from access time (per second) followed by the utility derived from each byte downloaded in the conventional system

\[ UT_1 = \frac{D}{T} \]

Equation 4: Utility from Access Time (Per Second) without using Profiles

\[ UT_1 = \frac{31426}{106443} \]

\[ UT_1 = 0.2952 \]

Where:

\[ UT_1 = \text{Utility from Access Time (Per Second) without using Profiles} \]

\[ D = \text{Aggregate Utility of Downloaded Data} \]

\[ T = \text{Total Time Taken to Access Data in Seconds} \]
\[ UD_1 = D / TD \]

Equation 5: Utility from Downloaded Data (Per byte) without using Profiles

\[ UD_1 = \frac{31426}{1524776} \]
\[ UD_1 = 0.0206 \]

Where:
\[ UD = \text{Utility from Downloaded Data (per byte) without using Profiles} \]
\[ D = \text{Aggregate Utility of Downloaded Data} \]
\[ TD = \text{Total data downloaded by all the users during the life span of simulation in bytes} \]

Same calculations as above were performed on the generated data by using Proposed System Simulation which used our proposed method of prefetching and caching data objects on the mobile device by using Device and User Profiles. The compiled results are given below:

Aggregate utility of accessed data by using profile for all appointments and sum of utility derived from the data downloaded on the mobile device is 32794.

Total time taken to download the required data by using profile for each appointment by all the employee’s is 61901 seconds.

Total data downloaded in the proposed system by all the users during the life span of simulation is 613200 bytes

With the above figures we first calculate utility derived from access time (per second) followed by the utility derived from each byte downloaded in the proposed system.

\[ UT_2 = D / T \]

Equation 6: Utility from Access Time (Per Second) by using Profiles
\[ UT_2 = \frac{32794}{61901} \]
\[ UT_2 = 0.5298 \]

Where:

\[ UT_2 = \text{Utility from Access Time (Per Second) by using Profiles} \]

\[ D = \text{Aggregate Utility of Downloaded Data} \]

\[ T = \text{Total Time Taken to Access Data in Seconds} \]

\[ UD_2 = \frac{D}{TD} \]

\[ \text{Equation 7: Utility from Downloaded Data (per byte) by using Profiles} \]

\[ UD_2 = \frac{32794}{1014347} \]
\[ UD_2 = 0.0323 \]

Where:

\[ UD_2 = \text{Utility from Downloaded Data per byte by using Profiles} \]

\[ D = \text{Aggregate Utility of Downloaded Data} \]

\[ TD = \text{Total data downloaded by all the users during the life span of simulation in bytes} \]

5.2 Conclusion

In this section the results calculated and compiled in the previous section by using conventional and proposed system are used to calculate percentage change in utilities.

Percentage change in utilities derived from both systems is:

\[ \text{Change in } UT = \left( UT_2 / UT_1 \right) \times 100 \]

\[ \text{Equation 8: Percentage Change in Utility Derived from Access Time per Second} \]
Change in UT = \( \frac{0.5298}{0.2952} \) * 100

Change in UT = 179% increase in Derived Utility

Where:

\[
\text{Change in UT} = \text{Percentage Change in Utility Derived from Access Time per Second}
\]

\[
\text{Change in } UD = \left( \frac{UD_2}{UD_1} \right) \times 100
\]

*Equation 9: Percentage Change in Utility Derived from Downloaded Data per Byte*

Change in UD = \( \frac{0.0323}{0.0206} \) * 100

Change in UD = 156% increase in Derived Utility

Where:

\[
\text{Change in UD} = \text{Percentage Change in Utility Derived from Downloaded Data per Byte}
\]

From the above compiled results, it is clear that the proposed system works and performs much better than the conventional system by increasing overall system utility for the user. The major contributor for the increase in utility derived from access time per second is due to the Cache Management System which optimally utilizes the device storage space by caching the most frequently used and highly weighted data objects. However, the major contributor for the increase in utility derived from downloaded data per byte is because of the Context-Aware Prefetching System which prefetches the data objects by automatically selecting the network with best performance and by pushing the data objects on the mobile device cache before they are requested.
Chapter 6

Future Directions

The results from application profiling and evaluation of system output along with system limitations provide some basis for future improvements. In the future the caching of database connections, use of triggers and prepared statements will increase the performance of the overall system. It will help if the system is ported to Java 2 Micro Edition. In this project it was assumed that once the runtime environment starts, the data at the source is not changed, therefore data is not validated for possible changes, but I recommend to integrate a data validation component in future implementations. Moreover, it was assumed that each meeting will require data related to only one product, which fits our application scenario. However, this assumption can be included in the future implementations by enabling the employee and the customer to use more than one product’s information during the course of the meeting.
Appendix A: Execution Flow Diagrams

Runtime Information Generator

Inter Class Method Calls
Sequence Diagram Part A

Default actor

GenerateRunTime()

copy : CopyData

copyData() void

setUp() void

populateEmployee() void

populateEmployeeHasDevice() void

generateCustProfile() void

generateAppointments() void

Generates Appointments for Each Employee
With Random Traveltime Between Appointments,
Appointment Duration, Devices and Location
Please See Sequence Diagram of this method
Generates Appointments for Each Employee with Random Traveltine Between Appointments, Appointment Duration, Devices and Location.
Please See Sequence Diagram of this method.

Assigns Random Networks at Each Location.

Gathers and Stores each appointment's information with user route available networks at each suburb, device in use by the employee and time spent at each suburb.

Generates Product Data. Each Product's Total Data Size is Randomly Generated and Random Number of Data Elements are Assigned with Random Sizes and MIME Types.

Close All Database Connections.
Generate Run Time Class :: Generate Appointments Sequence Diagram

GenerateRunTime

RandomGenerator

Populate Employee Array

getInt()

assignDevices()

Assign Random Devices to Employees

more

Assign Random Start & End Available Times to Employees

addAppointmentStartT

Add Appointment Information to the Database
Recursive Method

Calls

please see comments within the code for details.
Populate Route Class :: Populate Table Method Sequence Diagram

- Populate Suburb Data
- Calculate Shortest Route Between two Suburbs
- Add Random Travel Times to the Route
- Store Route Data in the Database
Find Route :: find Route Method Sequence Diagram

Find Route Sequence Diagram

Find Method Collaboration Diagram

1. find(Suburb, int)
1.1: findMin([int]): int
1.2: get(int): Object
1.3: add(Object): boolean
1.4: find(Suburb, int): void
The find route method is used to determine the route the employee will take while traveling from Suburb A to Suburb B.
The Product Data Information generated in this method is stored in productData table. In the table Product ID, MIME ID and Size of each data object is stored, product priority weights are not stored in the database as they are assigned at the runtime.
Sequence Diagram Representing Initialization of Employee Object

- Employee
  - getStatement()
  - getLocation()
  - getDevice()
  - getProduct()
  - getRank()

- Device
  - getDevice()
  - getRank()

- Location
  - getLocation()
  - getRank()

- MainRunTime
  - getRank()

- Product
  - getProduct()
  - getRank()

- RandomGenerator
  - getRank()
This method is used to assign random current bandwidth to each network object before it is cloned.
Sequence Diagram Representing Initialization of Product Object

Each product object contains the reference of the Suburb, representing the location of the product, product Id, product type and array of product data objects.
Conventional System Simulation
The next sequence diagram represents second half of add to device method which calculates and stores the utility of accessed data.
Without Profile:: Add to Device Sequence Diagram Part B

- `Device`:
  - `getDevice()`
  - `getSupportedMimeTypes()`
  - `getDeviceInfo()`
- `DeviceCacheData`:
  - `getOptionData()`
- `MainRuntime`:
  - `getDevice()`
- `MimeType`:
  - `getPriority()`

From All the Product Data Objects, get the List of Data Objects Which Are Supported by the Device.

Retrieve the Weights Assigned to Each MIME Type by the Employee.

Calculate and Save the Utility of Data Objects Accessible By Using Device Support, MIME Priority and Weights Assigned by the Employee.
Flow Diagram of calculateAndStoreProductData Without Using Profiles

1. Get the Weight Assigned By Employee to Product Types
   - deviceSupport = new b...
   - MainRuntime.filterDat...
   - weights = MainRuntime...
   - selectProductTypeId =...
   - rs = MainRuntime.getSt...
   - rs.nextInt()
   - productTypeId = rs.get...
   - rs.close()

2. For i = 0; i < inCache;
   - if (inCache[i] == false... For Each Data Object Check Whether It Was Accessed Or NOT

3. InsertIntoMissedDataWith...
   - t = MainRuntime.getSt...

4. Identify Data Objects which Could Not be Accessed By the User and Store the Details into Missed Data Without Profile Table
   - Calculate Utility of Each Accessed Data Object
     - if (deviceSupport[i] == true... else
     - utility = pData[i].get...

5. Store The Utility Derived Details Into Utility Per Product Data Downloaded Without Profile Table
   - InsertIntoUtilityPerPr... t = MainRuntime.getSt...

Return data AnyTime
With Profile :: Sequence Diagram of Run Method PART A

getLocation() -> getDevice()

getNetworks()

getEmployee()

getDevice()

getNetworks()

access()

getWeights()

equals()

dataAccessed()

getDevice()

getNetworks()

getEmployee()

getDevice()

getNetworks()

getEmployee()

getDevice()
With Profile :: Sequence Diagram of Run Method PART B

```
Define and Store Product
- getProfile()
- addProfile()

Device
- getBandwidth()
- getCurrentBandwidth()

Main Run Time
- getID()
- getSize()
- getCacheCapacity()

Network
- determine Best Performing Wireless Network

Product
- determine Total Data Size

This Method is Used to Determine the Maximum Sum of Data Which can be Prefetched and Maximum Connection Time At Current Location With the Mobile Devices By Using Device Cache Capacity, Employee Stay At Current Location and Best Available Bandwidth.
```

Past the Time and Network Constraints along with Product Information to Device for Prefetching.
With Profile::Calculate Time and Data Method Flow Diagram

This method is used to calculate Maximum Data which can be downloaded and maximum connection time available at employee’s current location. For the above calculations, device cache capacity, available bandwidth, product data size and employee stay at current location are used.

```
1. double[] calculateTimeAndData(
   Device device, int bestBandwidth, int
   productDataSize, int timeAtLocation)

   double[] maxTimeAndData = new double[2]

   // [(0) = maximumAvailableTime [1] = maximumPossibleDataTransfer

   maxTimeAndData[0] = 0.0
   maxTimeAndData[1] = 0.0

   download = 0
   productDataSize >= device.getCacheCapacity()

   download = device.getCacheCapacity()
   else
         download = productDataSize

   time = ((double)download)/((double)bestBandwidth)

   if(time <= timeAtLocation)
       if(productDataSize is more than device cache capacity then set
          MAXIMUM data to be downloaded to
          device storage capacity, also set it to
          product data size)

   else
       maxTimeAndData[0] = Double.parseDouble(time)
       maxTimeAndData[1] = download

   return maxTimeAndData
```
Appendix B: Source Code of the System

Random Information Generator

CopyData Class

package RunTimeDb;
/*
 * CopyData.java
 *
 * Created on 8 October 2004, 14:43
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
/**
 * @author omahmood
 */
public class CopyData {
    private final String DB_URL_STATIC = "jdbc:mysql:///application_static_data?user=root";
    private final String DB_URL_RUNTIME = "jdbc:mysql:///application_runtime_data?user=root";
    private Connection connStatic = null;
    private Statement stmtStatic = null;
    private Connection connRun = null;
    private Statement stmtRun = null;

    public void copyData(){
        setUp();
        deleteAndSetupTables();
        copyDevice();
        copyPriority();
        copyNetworkType();
        copyMimeType();
        copyDeviceMimeTypes();
        copyDeviceNetwrokTypes();
        populateLocation();
        populateProduct();
        populateEmployeeProfile();
        populateProductProfile();
    }
close();
}

private void populateProductProfile(){
    try{
        String selectFromLocation = "Select product_id,location_id from product_profile";
        ResultSet rsLocationSelect = stmtStatic.executeQuery(selectFromLocation);
        while(rsLocationSelect.next()){
            String insertIntoLocation = "INSERT INTO product_profile (product_id,location_id) VALUES (" +
            rsLocationSelect.getInt(1) + "," + rsLocationSelect.getInt(2)+");";
            int t = stmtRun.executeUpdate(insertIntoLocation);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

private void populateEmployeeProfile(){
    try{
        String selectFromLocation = "Select employee_id,product_id,location_id from employee_profile";
        ResultSet rsLocationSelect = stmtStatic.executeQuery(selectFromLocation);
        while(rsLocationSelect.next()){
            String insertIntoLocation = "INSERT INTO employee_profile (employee_id,product_id,location_id) VALUES (" +
            rsLocationSelect.getInt(1) + "," + rsLocationSelect.getInt(2) + "," + rsLocationSelect.getInt(3) + ");";
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```java
    int t = stmtRun.executeUpdate(insertIntoLocation);
    }
}catch(Exception e){
    e.printStackTrace();
}
}
private void populateProduct(){
    try{
        String selectFromLocation = "Select product_id,prod_desc.type from product";
        ResultSet rsLocationSelect = stmtStatic.executeQuery(selectFromLocation);
        while(rsLocationSelect.next()){
            String insertIntoLocation = "INSERT INTO product (product_id,prod_desc,type) VALUES (" + rsLocationSelect.getInt(1) + "," + rsLocationSelect.getString(2) + "," + rsLocationSelect.getString(3) + ");";
            int t = stmtRun.executeUpdate(insertIntoLocation);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
private void deleteAndSetupTables(){
    try{
        
    }
```
int t = stmtRun.executeUpdate("CREATE DATABASE /*!32312 IF NOT EXISTS*/ application_runtime_data; ");
t = stmtRun.executeUpdate("USE application_runtime_data; ");

t = stmtRun.executeUpdate("DROP TABLE IF EXISTS networktype; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS customer_profile; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS customer; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS route; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS priority; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS mimetype ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS device_supports_mimetype ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS device_supports_networktype ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS device ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS location_available_network ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS location ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS employee ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS employee_has_device ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS employee_predictedlocations ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS product ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS appointment ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS employee_profile ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS product_profile ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS productdata ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS cache; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS misseddatawithoutprofile; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS misseddatawithprofile; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS UtilityPerProductDataDownloadedWithoutProfile; ");
t = stmtRun.executeUpdate("DROP TABLE IF EXISTS UtilityPerProductDataDownloadedWithProfile; ");
t = stmtRun.executeUpdate("CREATE TABLE employee_predictedlocations ( id bigint(20) NOT NULL auto_increment, appointment_id bigint(20) NOT NULL default '0', employee_id bigint(20) NOT NULL default '0', product_id bigint(20) NOT NULL default '0', day bigint(20) NOT NULL default '0', location_id bigint(20) NOT NULL default '0', cache_id bigint(20) NOT NULL default '0', device_id bigint(20) NOT NULL default '0', PRIMARY KEY (id)) TYPE=MyISAM; ");
t = stmtRun.executeUpdate("CREATE TABLE appointment ( id bigint(20) NOT NULL auto_increment, startTime varchar(255) NOT NULL default '', endTime varchar(255) NOT NULL default '', product_id bigint(20) NOT NULL default '0', location_id bigint(20) NOT NULL default '0', custo mer_id bigint(20) NOT NULL default '0', day bigint(20) NOT NULL default '0', appointmentDuration varchar(255) NOT NULL default ' ', travelTime varchar(255) NOT NULL default '', employee_id bigint(20) NOT NULL default '0', device_id bigint(20) NOT NULL default '0', utilityOfCachedDataWithoutProfile bigint(20) NOT NULL default '0', timeToDownloadWithoutProfile double(20,2) NOT NULL default '0.00', utilityOfCachedDataWithProfile bigint(20) NOT NULL default '0', timeToDownloadWithProfile double(20,2) NOT NULL default '0.00', dataDownloadedWithoutProfile bigint(20) NOT NULL default '0', dataDownloadedWithProfile bigint(20) NOT NULL default '0', PRIMARY KEY (id)) TYPE=MyISAM; ");
t = stmtRun.executeUpdate("CREATE TABLE product_profile ( id int(11) NOT NULL auto_increment, product_id int(11) NOT NULL default '0', location_id int(11) NOT NULL default '0', PRIMARY KEY (id), KEY ProductProduct_Profile (product_id), KEY LocationProduct_Profile (location_id)) TYPE=MyISAM; ");
CREATE TABLE location_available_network (location_id int(11) NOT NULL default '0', network_id int(11) NOT NULL default '0', PRIMARY KEY (location_id,network_id), KEY network_id (network_id), KEY LocationLocation_Available_Network (location_id), KEY network_id (network_id), KEY NetworkTypeLocation_Available_Network (network_id)) TYPE=MyISAM;"

CREATE TABLE customer (id int(11) NOT NULL auto_increment, fName varchar(50) default NULL, lName varchar(50) default NULL, PRIMARY KEY (id), KEY Customerid (id)) TYPE=MyISAM;"

CREATE TABLE route (appointment_id int(11) default '0', employee_id int(11) default '0', day int(11) default '0', location_id int(11) default '0', timeSpent varchar(50) default NULL, sequence int(11) NOT NULL auto_increment, PRIMARY KEY (sequence), KEY location_id (location_id), KEY EmployeeRoute (employee_id), KEY Routeday (day), KEY AppointmentRoute (appointment_id), KEY LocationRoute (location_id), KEY employeeId (employee_id)) TYPE=MyISAM;"

CREATE TABLE employee (id int(11) NOT NULL auto_increment, fName varchar(50) default NULL, lName varchar(50) default NULL, PRIMARY KEY (id)) TYPE=MyISAM;"

CREATE TABLE networktype (id int(11) NOT NULL auto_increment, maxBandwidth double default '0', title varchar(50) default NULL, PRIMARY KEY (id)) TYPE=MyISAM;"

CREATE TABLE priority (id smallint(6) NOT NULL default 0, title varchar(5) default NULL, PRIMARY KEY (id)) TYPE=MyISAM;"

CREATE TABLE mimetype (id int(11) NOT NULL auto_increment, extension varchar(255) default NULL, title varchar(255) default NULL, priority_id smallint(6) default '0', PRIMARY KEY (id), KEY id (id), KEY PriorityMimeType (priority_id)) TYPE=MyISAM;"

CREATE TABLE device_supports_mimetype (device_id int(11) NOT NULL default '0', mimeType_id int(11) NOT NULL default '0', PRIMARY KEY (device_id,mimeType_id), KEY device_id (device_id), KEY mimeType_id (mimeType_id)) TYPE=MyISAM;"
DeviceDevice_MimeType (device_id), KEY mimeType_id (mimeType_id), KEY MimeTypeDevice_MimeType (mimeType_id))
TYPE=MyISAM;);

t = stmtRun.executeUpdate("CREATE TABLE device_supports_networktype ( device_id int(11) NOT NULL default '0',
networkType_id int(11) NOT NULL default '0', PRIMARY KEY (device_id,networkType_id), KEY
Device_NetworkTypeDeviceHASNetworkType (networkType_id), KEY DeviceDeviceHASNetworkType (device_id))
TYPE=MyISAM;);

t = stmtRun.executeUpdate("CREATE TABLE device ( id int(11) NOT NULL auto_increment, cacheCapacity double default '0',
title varchar(50) default NULL, PRIMARY KEY (id)) TYPE=MyISAM;";);

t = stmtRun.executeUpdate("CREATE TABLE location ( id int(11) NOT NULL auto_increment, title varchar(50) default NULL, postCode varchar(6) default NULL, PRIMARY KEY (id), KEY postcode (postCode)) TYPE=MyISAM;";);

t = stmtRun.executeUpdate("CREATE TABLE customer_profile ( id int(11) NOT NULL auto_increment, customer_id int(11) NOT NULL default '0', product_id int(11) NOT NULL default '0', location_id int(11) NOT NULL default '0', PRIMARY KEY (id,customer_id,product_id,location_id), KEY CustomerCustomer_Profile (customer_id), KEY LocationCustomer_Profile (location_id), KEY prod_id (customer_id), KEY ProductCustomer_Profile (product_id)) TYPE=MyISAM;";);

t = stmtRun.executeUpdate("CREATE TABLE employee_has_device ( employee_id int(11) NOT NULL default '0', device_id int(11) NOT NULL default '0', PRIMARY KEY (employee_id,device_id), KEY device_id (device_id), KEY DeviceEmployee_Has_Device (device_id), KEY employee_id (employee_id), KEY EmployeeEmployee_Has_Device (employee_id))
TYPE=MyISAM;";);

t = stmtRun.executeUpdate("CREATE TABLE product ( product_id int(11) NOT NULL auto_increment, prod_desc varchar(255) default NULL, type varchar(50) default NULL, PRIMARY KEY (product_id)) TYPE=MyISAM;";);

t = stmtRun.executeUpdate("CREATE TABLE employee_profile ( employee_id int(11) NOT NULL default '0', product_id int(11) NOT NULL default '0', location_id int(11) NOT NULL default '0', PRIMARY KEY (employee_id,product_id,location_id),

96
CREATE TABLE productdata (id bigint(20) NOT NULL auto_increment, p_id int(11) NOT NULL default '0', mimeId int(11) NOT NULL default '0', size int(11) default '0', PRIMARY KEY (id), KEY p_id (p_id), KEY MimeTypeProductData (mimeId), KEY ProductProductData (p_id) TYPE=MyISAM;);

CREATE TABLE cache (id bigint(20) NOT NULL auto_increment, cache longtext NOT NULL, employee_id bigint(20) NOT NULL default '0', time double(20,1) NOT NULL default '0.0', predictedlocation_id bigint(20) NOT NULL default '0', bandwidth bigint(20) NOT NULL default '0', newDataSize bigint(20) NOT NULL default '0', availableCache bigint(20) NOT NULL default '0', appId bigint(20) NOT NULL default '0', cacheSize bigint(20) NOT NULL default '0', PRIMARY KEY (id)) TYPE=MyISAM;);

CREATE TABLE misseddatawithoutprofile (id int(11) NOT NULL auto_increment, appoint_id int(11) default '0', mime_id int(11) default '0', product_id int(11) default '0', device_id int(11) default '0', size int(11) default '0', reason varchar(255) default NULL, PRIMARY KEY (id), KEY appoint_id (appoint_id), KEY device_id (device_id), KEY id (id), KEY product_id (product_id) TYPE=MyISAM;);

CREATE TABLE misseddatawithprofile (id int(11) NOT NULL auto_increment, appoint_id int(11) default '0', mime_id int(11) default '0', product_id int(11) default '0', device_id int(11) default '0', size int(11) default '0', predictedLocation_id int(11) default NULL, PRIMARY KEY (id), KEY appoint_id (appoint_id), KEY device_id (device_id), KEY id (id), KEY predictedLocation_id (predictedLocation_id) TYPE=MyISAM;);

CREATE TABLE UtilityPerProductDataDownloadedWithoutProfile (id int(11) NOT NULL auto_increment, appointment_id int(11) default '0', utility int(11) default '0', PRIMARY KEY (id), KEY appointment_id (appointment_id), KEY id (id)) TYPE=MyISAM;);
t = stmtRun.executeUpdate("CREATE TABLE UtilityPerProductDataDownloadedWithProfile (  id int(11) NOT NULL auto_increment,  appointment_id int(11) default '0',  utility int(11) default '0', PRIMARY KEY (id), KEY appointment_id (appointment_id), KEY id (id)) TYPE=MyISAM;")

//

//
}

private void populateLocation(){
    try{
        String selectFromLocation = "Select id,PostCode,title from location";
        ResultSet rsLocationSelect = stmtStatic.executeQuery(selectFromLocation);
        while(rsLocationSelect.next()){
            String insertIntoLocation = "INSERT INTO location (id,PostCode,title) VALUES (" + rsLocationSelect.getInt(1) + "," + rsLocationSelect.getInt(2) + "," + rsLocationSelect.getString(3) + ");";
            int t = stmtRun.executeUpdate(insertIntoLocation);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
private void copyPriority(){
    try{
        String select = "Select id,title from priority";
        ResultSet rsSelect = stmtStatic.executeQuery(select);
        while(rsSelect.next()){  
            String insert = "INSERT INTO priority (id,title) VALUES ("+ rsSelect.getInt(1) +"," + rsSelect.getString(2) + ");";
            int t = stmtRun.executeUpdate(insert);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

private void copyDevice(){
    try{
        String select = "Select id,cacheCapacity,title from device";
        ResultSet rsSelect = stmtStatic.executeQuery(select);
        while(rsSelect.next()){
            String insert = "INSERT INTO device (id,cacheCapacity,title) VALUES ("+ rsSelect.getInt(1) +"," + rsSelect.getInt(2) +"," + rsSelect.getString(3) + ");";
            int t = stmtRun.executeUpdate(insert);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
private void copyMimeType()
{
    try{
        String select = "Select id,extension,title,priority_id from mimetype";
        ResultSet rsSelect = stmtStatic.executeQuery(select);
        while(rsSelect.next()){
            String insert = "INSERT INTO mimetype (id,extension,title,priority_id) VALUES (" + rsSelect.getInt(1) + "," +
            rsSelect.getString(2) + "," + rsSelect.getString(3) + "," + rsSelect.getInt(4) + ");";
            int t = stmtRun.executeUpdate(insert);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

private void copyNetworkType()
{
    try{
        String select = "Select id,maxBandwidth,title,minimumBandwidth from networktype";
        // More code here...
    }catch(Exception e){
        e.printStackTrace();
    }
}
ResultSet rsSelect = stmtStatic.executeQuery(select);
while(rsSelect.next()){
    String insert = "INSERT INTO networktype (id,maxBandwidth,title) VALUES (" + rsSelect.getInt(1) + "," + rsSelect.getDouble(2) + "," + rsSelect.getString(3) + "');";
    int t = stmtRun.executeUpdate(insert);
}
}catch(Exception e){
e.printStackTrace();
}

private void copyDeviceMimeTypes(){
    try{
        String select = "Select device_id,mimeType_id from device_supports_mimetype";
        ResultSet rsSelect = stmtStatic.executeQuery(select);
        while(rsSelect.next()){
            String insert = "INSERT INTO device_supports_mimetype (device_id,mimeType_id) VALUES (" + rsSelect.getInt(1) + "," + rsSelect.getInt(2) + ");";
            int t = stmtRun.executeUpdate(insert);
        }
    }catch(Exception e){
e.printStackTrace();
}
private void copyDeviceNetworkTypes()
{
    try{
        String select = "Select device_id, networkType_id from device_supports_networktype";
        ResultSet rsSelect = stmtStatic.executeQuery(select);
        while(rsSelect.next()){
            String insert = "INSERT INTO device_supports_networktype (device_id, networkType_id) VALUES (" + rsSelect.getInt(1) + "," + rsSelect.getInt(2) + ");";
            int t = stmtRun.executeUpdate(insert);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

private void setUp()
{
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        connStatic = DriverManager.getConnection(DB_URL_STATIC);
        stmtStatic = connStatic.createStatement();
    }catch(Exception e){
        e.printStackTrace();
    }
}
try{
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    connRun = DriverManager.getConnection(DB_URL_RUNTIME);
    stmtRun = connRun.createStatement();
}catch(Exception e){
    e.printStackTrace();
}

private void close(){
    try{
        stmtRun.close();
        connRun.close();
        stmtStatic.close();
        connStatic.close();
    }catch(Exception e){
        e.printStackTrace();
    }
}
}
package RunTimeDb;

/**< *
 * does not insert anything into database but only calculates shortest path
 * between 2 suburbs
 *
 *
 * Created on 28 September 2004, 00:17
 */

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

/**< *
 *
 * @author omahmood
 */

public class FindRoute {

    private final String DB_URL = "jdbc:mysql://application_static_data?user=root";
    private Connection conn = null;
}
private Statement stmt = null;

private java.util.ArrayList <Suburb> sub = null;
private java.util.ArrayList <Suburb> route = null;

//private java.util.ArrayList sub = null;
//private java.util.ArrayList route = null;

private final int LENGTH = Suburb.COLUMNS;

public java.util.ArrayList findRoute(int originId, int destinationId){
    route = new java.util.ArrayList <Suburb> (1);
    sub = new java.util.ArrayList <Suburb> (6);
    java.util.ArrayList <Suburb> routeReturn = null;
    setUp();

    populateSuburbs();

    Suburb origin = sub.get(originId - 1);
    Suburb destination = sub.get(destinationId - 1);

    if(origin.id <= destination.id){
route.add(origin);
find(origin,destination.id);
return route;
}
else{
    route.add(destination);
    find(destination,origin.id);
    int size = route.size();
    routeReturn = new java.util.ArrayList<Suburb>(size);
    for(int i = size-1;i>=0;i--)
        routeReturn.add(route.get(i));
}
close();
return routeReturn;
}
private void find(Suburb origin,int destinationId){
    int direction[] = new int[LENGTH];
    for(int i = 0;i<LENGTH;i++)
        direction[i] = destinationId - origin.joins[i];
    if(origin.id == destinationId)
        return;
    int move = findMin(direction);
Suburb temp = (Suburb) sub.get(origin.joins[move]-1);
route.add(temp);
find(temp, destinationId);
}

private int findMin(int arr []) {
    int location = 0;
    int min = 1000;
    for (int i = 0; i < arr.length; i++)
        if (arr [ i ] < min && arr[i] >= 0) {
            min = arr [ i ];
            location = i;
        }
    return location;
}

private void populateSuburbs(){
    try{
        //ResultSet location = stmt.executeQuery("Select * from Location where id >=" + originId + " and id <=" + destinationId + " ORDER BY id");
        ResultSet location = stmt.executeQuery("Select * from Location ORDER BY id");
        int joins[] = null;
        while(location.next()){
            joins = new int[LENGTH];
for(int i = 0;i<LENGTH;i++)
    joins[i] = location.getInt(i+4);

    sub.add(new Suburb(location.getInt(1),joins));
}
stmt.close();
conn.close();
}catch(Exception e){
e.printStackTrace();
}
private void setUp(){
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        conn = DriverManager.getConnection(DB_URL);
        stmt = conn.createStatement();
    }catch(Exception e){
e.printStackTrace();
    }
}
private void close(){
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        conn = DriverManager.getConnection(DB_URL);
        stmt = conn.createStatement();
    }catch(Exception e){
e.printStackTrace();
    }
}
stmt.close();
conn.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
package RunTimeDb;

/*
 * GenerateCustomerProfile.java
 *
 * Created on 21 February 2005, 12:58
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
/**
 *
 * @author omahmood
 */
public class GenerateCustomerProfile {
    private final String DB_URL_STATIC = "jdbc:mysql://application_static_data?user=root";

    private Connection connStatic = null;
    private Statement stmtStatic = null;
public void generateCustProfile(){
    setUp();
    int custLength = 0;
    int prodLength = 0;
    int prodLocationLength = 0;
    String custSelectIdSql = "Select id from customer;";
    String productSelectIdSql = "Select product_id from product;";
    int locationIds[] = null;
    int prodIds[] = null;
    int custIds[] = null;
    int prodIdCheck = 0;

    String recordCustNumbers = "SELECT COUNT(id) AS rowcount FROM customer;";
    String recordProdNumbers = "SELECT COUNT(product_id) AS rowcount FROM product;";

    try{//Populate the array with customer IDs
        ResultSet rsCount = GenerateRunTime.getStatement ().executeQuery(recordCustNumbers);
        rsCount.next();
        custLength = rsCount.getInt("rowcount");
        ResultSet custIdRs = GenerateRunTime.getStatement ().executeQuery(custSelectIdSql);
        custIds = new int[custLength];
    }
for(int r = 0; custIdRs.next() && r < custLength; r++) {
    custIds[r] = custIdRs.getInt(1);
}
}
catch(Exception e) {
    e.printStackTrace();
}
}
try {
    // Populate the array with product IDs
    ResultSet rsCount = GenerateRunTime.getStatement().executeQuery(recordProdNumbers);
    rsCount.next();
    prodLength = rsCount.getInt("rowcount");
    ResultSet prodIdRs = GenerateRunTime.getStatement().executeQuery(productSelectIdSql);
    prodIds = new int[prodLength];

    for(int r = 0; prodIdRs.next() && r < prodLength; r++) {
        prodIds[r] = prodIdRs.getInt(1);
    }
}
catch(Exception e) {
    e.printStackTrace();
}
for(int i = 0; i < custIds.length; i++) {
    if(prodIdCheck == prodLength)
prodIdCheck = 0;

try{//get location ids of each product and store it in an array
    String recordLocationNumbers = "SELECT COUNT(location_id) AS rowcount FROM product_profile where product_id=":"+prodIds[prodIdCheck];
    String productLocationsSelectSql = "Select location_id from product_profile where product_id=":"+prodIds[prodIdCheck];

    ResultSet rsCount = GenerateRunTime.getStatement().executeQuery(recordLocationNumbers);
    rsCount.next();
    prodLocationLength = rsCount.getInt("rowcount");

    ResultSet locationIdRs = GenerateRunTime.getStatement().executeQuery(productLocationsSelectSql);
    locationIds = new int[prodLocationLength];

    for(int r = 0;locationIdRs.next() && r<prodLocationLength;r++){
        locationIds[r] = locationIdRs.getInt(1);
    }
}catch(Exception e){
    e.printStackTrace();
}
int interestedNumberOfLocations = RandomGenerator.getInt(1, prodLocationLength/2);
for(int j = 0; j < interestedNumberOfLocations; j++) {
    String insertIntoCustomerProfile = "INSERT INTO `customer_profile` (`customer_id`, `product_id`, `location_id`) VALUES (" + custIds[i] + "," + prodIds[prodIdCheck] + "," + locationIds[RandomGenerator.getInt(0, prodLocationLength-1)] + ");";
    try{
        int t = GenerateRunTime.getStatement().executeUpdate(insertIntoCustomerProfile);
    } catch (Exception e) {
        e.printStackTrace();
    }
    prodIdCheck++;
}
close();

//public static void main(String args[]){
//    new GenerateCustomerProfile().generateCustProfile();
//}

private void setUp(){
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        connStatic = DriverManager.getConnection(DB_URL_STATIC);
        stmtStatic = connStatic.createStatement();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
catch(Exception e){
    e.printStackTrace();
}

private void close(){
    try{
        stmtStatic.close();
        connStatic.close();
    }catch(Exception e){
        e.printStackTrace();
    }
}
package RunTimeDb;
/
*This program generate Customers, dummy customer profiles and appointments
*/
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class GenerateRunTime {
    private final String DB_URL_STATIC = "jdbc:mysql:///application_static_data?user=root";
    private final String DB_URL_RUNTIME = "jdbc:mysql:///application_runtime_data?user=root";
    private static Connection connStatic = null;
    private Statement stmtStatic = null;
    private static Connection connRun = null;
    private Statement stmtRun = null;
    private CopyData copy = new CopyData();
    private int devices[];
    private PopulateRoute popRoute = new PopulateRoute();
    public GenerateRunTime() {
        System.out.println("START @@:"+new java.util.Date() );
    }
}
copy.copyData();
setUp();
populateEmployee();
populateEmployeeHasDevice();
generateCustomer(60);
generateCustProfile();
populateDevices();
generateAppointments();
popRoute.populateRouteTable();
populateAvailableLocationNetwork();
generateCustPredictedLocations();
generateProductData();
close();
System.out.println("COMPLETE @@: "+new java.util.Date());
}

private void populateDevices(){
    try{
        String deviceIds = "SELECT id from device;"
        String totalNumberOfDevices = "SELECT COUNT(id) AS rowcount FROM device;";

        ResultSet rsCount = stmtStatic.executeQuery(totalNumberOfDevices);
        rsCount.next();
    }
devices = new int[rsCount.getInt("rowcount")];

ResultSet deviceIdsRs = stmtRun.executeQuery(deviceIds);
for(int i = 0;i<devices.length && deviceIdsRs.next();i++){
    devices[i] = deviceIdsRs.getInt(1);
}
}catch(Exception e){
    e.printStackTrace();
}
}

private void generateAppointments(){
    int empLength = 0;
    int employee[][][] = null; // employee[][][5] ==> device of the day
    int appointmentTime = -100;
    int currentEmployeeId =0;
    int currentEmployeeIndex = 0;
    int gap = 0;

    try{
        //POPULATE EMPLOYEE ARRAY
        String startAndEndTimeEmployee = "SELECT id,availableStartTime,availableEndTime from employee;"; //, customer where employee.id = customer.empId;"
        String recordEmployeeNumbers = "SELECT COUNT(id) AS rowcount FROM employee;";
    
    }
ResultSet rsCount = stmtStatic.executeQuery(recordEmployeeNumbers);
rsCount.next();
empLength = rsCount.getInt("rowcount");
ResultSet startAndEndTimeEmployeeRs = stmtRun.executeQuery(startAndEndTimeEmployee);
employee = new int[empLength][2][6];
populateEmployeeArray(startAndEndTimeEmployeeRs, employee);
}
}catch(Exception e){
e.printStackTrace();
}

try{
    String custProfileInfoSelectSql = "SELECT Customer_Profile.customer_id, Customer_Profile.product_id, 
                     Employee_Profile.location_id, Employee_Profile.employee_id FROM Customer_Profile INNER JOIN Employee_Profile ON 
                     (Customer_Profile.product_id = Employee_Profile.product_id) AND (Customer_Profile.location_id = Employee_Profile.location_id) 
                     ORDER BY Customer_Profile.id;";
    ResultSet custEmployeeInfo = stmtRun.executeQuery(custProfileInfoSelectSql);
    for(int r = 0;custEmployeeInfo.next();r++){
        currentEmployeeId = custEmployeeInfo.getInt(4);
        boolean flag = true;
        for(currentEmployeeIndex = 0;flag && currentEmployeeIndex<empLength;currentEmployeeIndex++)
            if(employee[currentEmployeeIndex][0][0] == currentEmployeeId){
appointmentTime = addAppointmentStartTime(employee, currentEmployeeIndex, 60, currentEmployeeId);
flag = false;
}
}
currentEmployeeIndex--;
String temp[] = new String[4];
for(int s = 0; s < temp.length; s++) {
    temp[s] = Integer.toString(employee[currentEmployeeIndex][0][s+1]);
    if(temp[s].length() == 1) {
        temp[s] = "0" + temp[s];
    }
}
gap = addAppointmentStartTime(employee, currentEmployeeIndex, 40, currentEmployeeId);
int location = custEmployeeInfo.getInt(3);
int prod_id = custEmployeeInfo.getInt(2);
int customer_id = custEmployeeInfo.getInt(1);
int employee_id = custEmployeeInfo.getInt(4);
String productId = "Select id from product_profile where product_id=" + prod_id + " AND location_id=" + location + ";";
ResultSet prodIdRs = GenerateRuntime.getStatement().executeQuery(productId);
prodIdRs.next();
String insertAppointment = "INSERT INTO appointment (startTime, endTime, location_id, customer_id, product_id, day, appointmentDuration, travelTime, employee_id, device_id) VALUES (" +

120
int t = stmtRun.executeUpdate(insertAppointment);
}
}catch(Exception e){
e.printStackTrace();
}
}

private int assignDevices(int employeeId){
    int id = 0;
    try{
        String deviceIdsSelect = "Select device_id from employee_has_device where employee_id=" + employeeId+ ";
        ResultSet rs = GenerateRunTime.getStatement().executeQuery(deviceIdsSelect);

        String recordNumbers = "SELECT COUNT(device_id) AS rowcount FROM employee_has_device where employee_id=" + employeeId+ ";
        ResultSet rsCount = GenerateRunTime.getStatement().executeQuery(recordNumbers);
        rsCount.next();
        int length = rsCount.getInt("rowcount");

        int temp[] = new int[length];
for(int i = 0;i<length && rs.next();i++){
    temp[i] = rs.getInt(1);
}

id = RandomGenerator.getInt(0,length-1);
return temp[id];
}
catch (Exception e){
    e.printStackTrace();
}
return id;
employee[i][0][2] = Integer.parseInt(tempStart[1])+RandomGenerator.getInt(15,20)+10;//Available Start Time Min + Meeting Starttime

if(employee[i][0][2]>59){
    employee[i][0][2] = employee[i][0][2] - 59;
    employee[i][0][1] ++;
}

employee[i][0][3] = employee[i][0][1]; //Available Start Time Hour
employee[i][0][4] = employee[i][0][2]; //Available Start Time Min for the first meeting

employee[i][1][0] = employee[i][0][0]; //Emp ID
employee[i][1][1] = employee[i][0][1]; //Available Start Time Hour
employee[i][1][2] = employee[i][0][2]; //Available Start Time Min for the first meeting
employee[i][1][3] = Integer.parseInt(tempEnd[0]); //Available Last Meeting Time Hour
employee[i][1][4] = Integer.parseInt(tempEnd[1]); //Available Last Meeting Time Min
employee[i][0][5] = 1; //Day
employee[i][1][5] = assignDevices(employee[i][0][0]);

} catch(java.sql.SQLException e){
    e.printStackTrace();
}
}
private int addAppointmentStartTime(int[][][] employee, int x, int max_duration, int currentEmployeeId){
    int duration = RandomGenerator.getInt(20, max_duration-1);
    if(employee[x][0][1]+1 < employee[x][1][3]) { // If more than one hour is available for meeting
        employee[x][0][1]=employee[x][0][3]; // Prev available time is set to new available time (hour)
        employee[x][0][2]=employee[x][0][4]; // Prev available time is set to new available time (min)
        employee[x][0][4] += duration; // Add meeting duration
        if (employee[x][0][4] >= 60) { // Set proper end meeting time
            employee[x][0][3]++;
            employee[x][0][4] -= 60;
        }
    }
    return duration;
} else if((employee[x][0][4]+15) <= 59 && (employee[x][0][3]+1) == employee[x][1][3]) {
    employee[x][0][1]=employee[x][0][3];
    employee[x][0][2]=employee[x][0][4];
    employee[x][0][4] += duration;
    if (employee[x][0][4] > 59) {
        employee[x][0][4] -= duration;
        employee[x][0][4] += 15;
    }
} else if(max_duration != 40) {
    employee[x][0][5]++;
}
//assign new device
//
employee[x][1][5] = assignDevices(currentEmployeeId);

employee[x][0][1] = employee[x][1][1]; // reset times
employee[x][0][2] = employee[x][1][2];
employee[x][0][3] = employee[x][1][1];
employee[x][0][4] = employee[x][1][2];
return addAppointmentStartTime(employee,x,max_duration,currentEmployeeId);
}

return duration;

private void generateCustomer(int no){
    for(int i = 0;i<no;i++){
        String fName = "Cust_"+(i+1) + "_FN";
        String lName = "Cust_"+(i+1) + "_LN";

        String sqlCustomer = "INSERT INTO customer (fName, lName) VALUES (""+fName+"",""+lName+"";"");
        String sqlGetId = "Select id from customer WHERE customer.fName = ""+fName+"" AND customer.lName = ""+lName+"";"");
        try{
            int t = stmtRun.executeUpdate(sqlCustomer);
            ResultSet rs = stmtRun.executeQuery(sqlGetId);
            rs.next();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
private void linkEmployeesWithCustomer(){
    String empSelectIdSql = "Select id from employee;";
    String custSelectIdSql = "Select id from customer;";
    int empIds[];
    String recordNumbers = "SELECT COUNT(id) AS rowcount FROM employee;";
    try{
        ResultSet rsCount = stmtStatic.executeQuery(recordNumbers);
        rsCount.next();
        int length = rsCount.getInt("rowcount");
        empIds = new int[length];
        ResultSet empIdRs = stmtStatic.executeQuery(empSelectIdSql);

        for(int i = 0;empIdRs.next() && i<length;i++){
            empIds[i] = empIdRs.getInt(1);
        }
        empIdRs = null;
    }
}
rsCount = null;

ResultSet custIdRs = stmtRun.executeQuery(custSelectIdSql);
int i = 0;
while(custIdRs.next()){
    if(i == length-1)
        i = 0;

    String insertSql = "INSERT INTO employee_customer (employee_id,customer_id) VALUES 
("+empIds[i]+","+custIdRs.getInt(1)+");";
    int t  = stmtRun.executeUpdate(insertSql);
    i++;
}
}catch(Exception e){
    e.printStackTrace();
}

private void populateAvailableLocationNetwork(){
    String networkSelectIdSql = "Select id from networktype;";
    String locationSelectIdSql = "Select id from location;";
    int networkIds[]=null;
}
int length=0;
ResultSet networkIdRs = null;
ResultSet rsCount = null;
ResultSet locationIdRs = null;
String recordNumbers = "SELECT COUNT(id) AS rowcount FROM networktype;";
try{
    rsCount = stmtRun.executeQuery(recordNumbers);
    rsCount.next();
    length = rsCount.getInt("rowcount");
    networkIds = new int[length];
    networkIdRs = stmtRun.executeQuery(networkSelectIdSql);
    for(int i = 0;networkIdRs.next() && i<length;i++) {
        networkIds[i] = networkIdRs.getInt(1);
    }
    networkIdRs = null;
    rsCount = null;
    locationIdRs = stmtRun.executeQuery(locationSelectIdSql);
}
int loop = RandomGenerator.getInt(1,length);
while(locationIdRs.next()){
    for(int x = 0;x<loop;x++) {
        int i = RandomGenerator.getInt(0,length-1);
String insertSql = "INSERT INTO location_available_network(network_id,location_id) VALUES ("+networkIds[i]+","+locationIdRs.getInt(1)+");";
    try{
        int t = stmtRun.executeUpdate(insertSql);
    }catch(Exception e){
    }
}
}

public static void main(String[] args) {
    new GenerateRunTime();
}

private void setUp(){
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        connStatic = DriverManager.getConnection(DB_URL_STATIC);
        stmtStatic = connStatic.createStatement();
    }catch(Exception e){
        e.printStackTrace();
    }
}
try{
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    connRun = DriverManager.getConnection(DB_URL_RUNTIME);
    stmtRun = connRun.createStatement();
}catch(Exception e){
    e.printStackTrace();
}

private void populateEmployeeHasDevice(){
    String deviceSelectIdSql = "Select id from device;";
    String employeeSelectIdSql = "Select id from employee;";
    int deviceIds[] = null;
    int length=0;
    ResultSet employeeIdRs = null;
    ResultSet rsCount = null;
    ResultSet deviceIdRs = null;
    String recordNumbers = "SELECT COUNT(id) AS rowcount FROM device;";
    try{
        rsCount = stmtRun.executeQuery(recordNumbers);
        rsCount.next();
        length = rsCount.getInt("rowcount");
    }
}
deviceIds = new int[length];
deviceIdRs = stmtRun.executeQuery(deviceSelectIdSql);
for(int i = 0;deviceIdRs.next() && i<length;i++){
    deviceIds[i] = deviceIdRs.getInt(1);
}
deviceIdRs = null;
rsCount = null;

employeeIdRs = stmtRun.executeQuery(employeeSelectIdSql);
while(employeeIdRs.next()){
    int loop = RandomGenerator.getInt(2,length);
    for(int x = 0;x<loop;x++){
        int i = RandomGenerator.getInt(0,length-1);
        String insertSqlOne = "INSERT INTO employee_has_device(employee_id,device_id) VALUES (
"+employeeIdRs.getInt(1)+","+deviceIds[i]+")";
        try{
            int t = stmtRun.executeUpdate(insertSqlOne);
        }catch(Exception e){
            }
    }
}
}
private void populateEmployee()
{
    try{

        String recordLocationNumbers = "SELECT COUNT(id) AS rowcount FROM location;"
;

        ResultSet rsCount = stmtRun.executeQuery(recordLocationNumbers);
        rsCount.next();
        int locationLength = rsCount.getInt("rowcount");

        int location[] = new int[locationLength];
        String selectIdLocation = "Select id from location"
;
        ResultSet rsLocationSelect = stmtRun.executeQuery(selectIdLocation);
        for(int i = 0;rsLocationSelect.next() & & locationLength<i;i++)
            location[i] = rsLocationSelect.getInt(1);

        String selectFromEmployee = "Select id,fName,lName from employee"
;
        ResultSet rsEmployeeSelect = stmtStatic.executeQuery(selectFromEmployee);
        int time[] = new int[4];
String temp[] = new String[4];
while(rsEmployeeSelect.next()){
    int locationRandom = RandomGenerator.getInt(1,locationLength-1);
    time[0] = RandomGenerator.getInt(8,10);
    time[1] = RandomGenerator.getInt(0,59);
    time[2] = RandomGenerator.getInt(14,18);
    time[3] = RandomGenerator.getInt(0,59);

    for(int s = 0;s<temp.length;s++){
        temp[s] = Integer.toString(time[s]);
        if(temp[s].length()==1){
            temp[s] = "0"+temp[s];
        }
    }

    temp[0] = temp[0] + ":" + temp[1];

    String insertIntoEmployee = "INSERT INTO employee (id,fName,lName,startLocation,availableStartTime,availableEndTime) VALUES (" + rsEmployeeSelect.getInt(1) + "," + "," + rsEmployeeSelect.getString(2) + "," + rsEmployeeSelect.getString(3) + "," + locationRandom + "," + temp[0] + "," + temp[1] + ");";
    int t = stmtRun.executeUpdate(insertIntoEmployee);
}
catch(Exception e){
    e.printStackTrace();
}

private void generateCustProfile(){
    int custLength = 0;
    int prodLength = 0;
    int prodLocationLength = 0;
    String custSelectIdSql = "Select id from customer;";
    String productSelectIdSql = "Select product_id from product;";
    int locationIds[] = null;
    int prodIds[] = null;
    int custIds[] = null;
    int prodIdCheck = 0;

    String recordCustNumbers = "SELECT COUNT(id) AS rowcount FROM customer;";
    String recordProdNumbers = "SELECT COUNT(product_id) AS rowcount FROM product;";

    try{//Populate the array with customer IDs
        ResultSet rsCount = stmtRun.executeQuery(recordCustNumbers);
        rsCount.next();
        custLength = rsCount.getInt("rowcount");
        custLength = rsCount.getInt("rowcount");
    }catch(Exception e){
        e.printStackTrace();
    }
ResultSet custIdRs = stmtRun.executeQuery(custSelectIdSql);
custIds = new int[custLength];

for(int r = 0; custIdRs.next() && r < custLength; r++) {
    custIds[r] = custIdRs.getInt(1);
}
}

} catch (Exception e) {
    e.printStackTrace();
}

try {// Populate the array with product IDs
    ResultSet rsCount = stmtRun.executeQuery(recordProdNumbers);
    rsCount.next();
    prodLength = rsCount.getInt("rowcount");
    ResultSet prodIdRs = stmtRun.executeQuery(productSelectIdSql);
    prodIds = new int[prodLength];

    for(int r = 0; prodIdRs.next() && r < prodLength; r++) {
        prodIds[r] = prodIdRs.getInt(1);
    }
}
}

} catch (Exception e) {
    e.printStackTrace();
}
for(int i = 0; i < custIds.length; i++) {
    if (prodIdCheck == prodLength)
        prodIdCheck = 0;

try{//get location ids of each product and store it in an array
    String recordLocationNumbers = "SELECT COUNT(location_id) AS rowcount FROM product_profile where product_id=" + prodIds[prodIdCheck];
    String productLocationsSelectSql = "Select location_id from product_profile where product_id=" + prodIds[prodIdCheck];

    ResultSet rsCount = stmtRun.executeQuery(recordLocationNumbers);
    rsCount.next();
    prodLocationLength = rsCount.getInt("rowcount");

    ResultSet locationIdRs = stmtRun.executeQuery(productLocationsSelectSql);
    locationIds = new int[prodLocationLength];

    for(int r = 0; locationIdRs.next() && r < prodLocationLength; r++) {
        locationIds[r] = locationIdRs.getInt(1);
    }
} catch(Exception e) {

}
e.printStackTrace();
}

int interestedNumberOfLocations = RandomGenerator.getInt(0, prodLocationLength/2);

int randomCustomer = RandomGenerator.getInt(0, custIds.length-1);

for(int j = 0;j<interestedNumberOfLocations;j++){
    String insertIntoCustomerProfile = "INSERT INTO `customer_profile` (`customer_id`, `product_id`, `location_id`) VALUES (" + custIds[randomCustomer] + "," + prodIds[prodIdCheck] + "," + locationIds[RandomGenerator.getInt(0, prodLocationLength-1)] + ");";
    try{
        int t = stmtRun.executeUpdate(insertIntoCustomerProfile);
    }catch (java.sql.SQLException e){
        //e.printStackTrace();
    }catch (Exception e){
        e.printStackTrace();
    }
}

prodIdCheck++;
}
private void generateCustPredictedLocations(){
    String sqlRoute = "SELECT Route.employee_id, Route.day, Route.location_id, Route.timeSpent,
    Appointment.device_id, Route.appointment_id, Appointment.product_id FROM Route INNER JOIN Appointment ON
    Route.appointment_id = Appointment.id ORDER BY Route.sequence;";
    String empstartLocation = "SELECT Employee.id, Employee.startLocation FROM Employee ORDER BY Employee.id;";

    int emp[][] = null;

    try{
        String recordEmployeeumbers = "SELECT COUNT(id) AS rowcount FROM employee;";
        ResultSet rsCount = stmtRun.executeQuery(recordEmployeeumbers);
        rsCount.next();
        int empLength = rsCount.getInt("rowcount");
        emp = new int[empLength][2];
    }catch (Exception e){
        e.printStackTrace();
    }catch (Exception e){
        e.printStackTrace();
    }

    try{
        ResultSet rs = stmtRun.executeQuery(empstartLocation);
        for(int i = 0;rs.next() && i < emp.length ;i++){
            emp[i][0] = rs.getInt(1);
        }
    }
}
emp[i][1] = rs.getInt(2);
}
rs.close();
}catch(Exception ex){
ex.printStackTrace();
}

try{
    int emp_id,day,location_id,timeSpent,device_id,appointment_id,prod_id;
    int count = 0;
    ResultSet rs = stmtRun.executeQuery(sqlRoute);

    while (rs.next()){
        emp_id = rs.getInt(1);
        day = rs.getInt(2);
        location_id = rs.getInt(3);
        timeSpent = rs.getInt(4);
        device_id = rs.getInt(5);
        appointment_id = rs.getInt(6);
        prod_id = rs.getInt(7);
String insertIntoPredictedLocations = "INSERT INTO employee_predictedlocations (appointment_id, employee_id, location_id, product_id, day, timeSpent, device_id) VALUES (" + appointment_id + "," + emp_id + "," + location_id + "," + prod_id + "," + day + "," + timeSpent + "," + device_id+ ");");

//boolean flag = true;
//for(int count = 0;count<emp.length;count++){
   
   if(emp_id == emp[count][0] && day == 1 && location_id == emp[count][1] && timeSpent == 0){
      //flag = false;
      int t  = stmtRun.executeUpdate(insertIntoPredictedLocations);
     }
    
    //}
}
else if(timeSpent != 0){// && flag}{
      int t  = stmtRun.executeUpdate(insertIntoPredictedLocations);
   }
else if(timeSpent == 0){// && flag}{
   rs.previous();
   int p_appointment_id = rs.getInt(6);
   if(emp_id == rs.getInt(1) && day == rs.getInt(2)){
      String timeSelect = "Select appointmentDuration from appointment where id=" + p_appointment_id + ";";
      ResultSet timeSelectRs = stmtRun.executeQuery(timeSelect);
      timeSelectRs.next();
      
      //get appointment time by appointment id

insertIntoPredictedLocations = "INSERT INTO employee_predictedlocations
(appointment_id,employee_id,location_id,product_id,day, timeSpent,device_id) VALUES (" + appointment_id + "," + emp_id + "," + location_id + "," + prod_id + "," + day + "," + timeSelectRs.getInt(1) + "," + device_id+ ");";

    int t = stmtRun.executeUpdate(insertIntoPredictedLocations);
    
} else{
    insertIntoPredictedLocations = "INSERT INTO employee_predictedlocations
(appointment_id,employee_id,location_id,product_id,day, timeSpent,device_id) VALUES (" + appointment_id + "," + emp_id + "," + location_id + "," + prod_id + "," + day + "," + timeSpent + "," + device_id+ ");";

    int t = stmtRun.executeUpdate(insertIntoPredictedLocations);
}

rs.next();

}

}catch(Exception ex){
    ex.printStackTrace();
}

}

private void close(){
try{
    stmtRun.close();
    connRun.close();
    
}
stmtStatic.close();
connStatic.close();
}catch(Exception e){
e.printStackTrace();
}
}
public static Statement getStatement(){
try{
    return connRun.createStatement();
}catch(Exception e){
e.printStackTrace();
    return null;
}
}
private void generateProductData(){
    RandomArray random = new RandomArray();
    try{
        String selectProductId = "Select id from product_profile ORDER BY id;";
        String selectMimeInfo = "Select id from mimetype order by id";
        String selectMaxCache = "select min(cacheCapacity) from device;";
        ResultSet rsMinCache = stmtRun.executeQuery(selectMaxCache);
        rsMinCache.next();
final int MIN_PROD_DATA = (int)rsMinCache.getInt(1)/4;
final int MAX_PROD_DATA = (int)rsMinCache.getInt(1)/3;

rsMinCache.close();

String selectMimeCount = "SELECT COUNT(id) AS rowcount FROM mimetype;";
ResultSet rsCount = stmtRun.executeQuery(selectMimeCount);
rsCount.next();
int length = rsCount.getInt("rowcount");
int mimeData[][] = new int[length][2];
rsCount.close();

ResultSet mimeInfo = stmtRun.executeQuery(selectMimeInfo);

for(int i = 0;i<length && mimeInfo.next();i++){
    mimeData[i][0] = mimeInfo.getInt(1);
}

ResultSet rsProducts = stmtRun.executeQuery(selectProductId);

while(rsProducts.next()){

int size = RandomGenerator.getInt(MIN_PROD_DATA, MAX_PROD_DATA);
int temp[] = random.randomArray(length, size, 0);
for (int i = 0; i < length && i < temp.length; i++) {
    mimeData[i][1] = temp[i];
}
for (int i = 0; i < length && i < temp.length; i++) {
    String insertIntoProductData = "Insert into productdata (p_id, mimeId, size) VALUES (" + rsProducts.getInt(1) + "," + mimeData[i][0] + "," + mimeData[i][1] + ");";
    int t = stmtRun.executeUpdate(insertIntoProductData);
}
} catch (Exception e) {
    e.printStackTrace();
}
PopulateRoute Class

/*
 * CalculateRoute.java
 *
 * Created on 6 October 2004, 16:11
 */

package RunTimeDb;

import java.sql.ResultSet;
import java.util.ArrayList;

/** *
 * @author omahmood
 */

public class PopulateRoute {
    private FindRoute routeCal = new FindRoute();
    private RandomArray random = new RandomArray();
    public void populateRouteTable(){
        populateTable();
    }
    public void populateTable(){
        populateTable();
    }
}
int employee[][] = null; //[0] = ID, [1] = startLocation
int maxDay = -1;

employee = populateEmployee();
for(int i = 0; i < employee.length; i++) {
    String selectMaxDay = "SELECT MAX(day) FROM appointment where employee_id="+(i+1)+";";
    try{
        ResultSet rsGetMaxday = GenerateRunTime.getStatement ().executeQuery(selectMaxDay);
        rsGetMaxday.next();
        maxDay  = rsGetMaxday.getInt(1);
    }catch(Exception e){
        e.printStackTrace();
    }
    for(int currentDay = 1; currentDay <= maxDay; currentDay++) { // MAX DAYS
        String employeeAvailableStartTime[] = null;
        String appointmentStartTime[] = null;
        try{
            String appointmentSelectStatement = "SELECT appointment.id, appointment.location_id, appointment.travelTime, employee.availableStartTime, appointment.startTime FROM employee

            appointment_id, appointment.location_id, appointment.travelTime, employee.availableStartTime, appointment.startTime FROM employee

            appointment_id, appointment.location_id, appointment.travelTime, employee.availableStartTime, appointment.startTime FROM employee
INNER JOIN appointment ON employee.id = appointment.employee_id WHERE ((appointment.employee_id="+ (i+1) + ") AND (appointment.day="+ currentDay +")) ORDER BY appointment.id;"

ResultSet rsAppointment = GenerateRunTime.getStatement ().executeQuery(appointmentSelectStatement);
String gap = null;
int totalTime = -10;
boolean newDayFlag = true;// for each day begin
while(rsAppointment.next()){                         if(newDayFlag){                             employeeAvailableStartTime = rsAppointment.getString(4).trim().split(":"); //employee available start time
appointmentStartTime = rsAppointment.getString(5).trim().split(":"); // appointment start time
 totalTime = getGap(employeeAvailableStartTime,appointmentStartTime);
 employee[i][1] = employee[i][2];//reset day
 }                         ArrayList route = routeCal.findRoute(employee[i][1],rsAppointment.getInt(2)); //ORIGIN, DESTINATION
 employee[i][1] = rsAppointment.getInt(2);
 int size = route.size();
 int timePerSub[] = new int[size+1];
 timePerSub[0] = 0;
 int appId = rsAppointment.getInt(1);
if(size > 2){
    int temp[] = random.randomArray(size-1, totalTime, 1);

    for(int k = 1;k<size;k++)
        timePerSub[k] = temp[k-1];
} else
    timePerSub[1] = totalTime;

if(size == 1){
    timePerSub[0] = totalTime;
}

for(int l = 0;l<size;l++) { // array list for location ids
    String insertRoute = "INSERT INTO ROUTE (appointment_id,employee_id,day,location_id,timeSpent) VALUES (" + rsAppointment.getInt(1) + "," + employee[i][0] + "," + currentDay + "," + route.get(l).toString() + "," + timePerSub[l] + ");";
    if(size == 1){
        timePerSub[0] = totalTime;
    }
    int rsInsertRoute = GenerateRunTime.getStatement().executeUpdate(insertRoute);
} // for arraylist
newDayFlag = false;
totalTime = Integer.parseInt(rsAppointment.getString(3));
}  // for (while) each appointment
}  
} catch(Exception e) {
    e.printStackTrace();
}
}  // FOR EACH DAY
}  //for each employee
}
private int getGap(String employeeAvailableStartTime[], String appointmentStartTime[]) {
    int gap = 0;

    int startHour = Integer.parseInt(employeeAvailableStartTime[0]);
    int endHour = Integer.parseInt(appointmentStartTime[0]);

    int startMin = Integer.parseInt(employeeAvailableStartTime[1]);
    int endMin = Integer.parseInt(appointmentStartTime[1]);

    if (startHour == endHour) {
        return endMin - startMin;
    } else {
        return endMin - startMin + 60;
    }
}
private int[][] populateEmployee()
{
    int employee[][] = null;
    String employeeCurrentStartLocation = "Select id,startLocation from employee";
    String empCount = "SELECT COUNT(id) AS rowcount FROM employee;";

    try{
        ResultSet rsCountEmployee = GenerateRunTime.getStatement().executeQuery(empCount);
        rsCountEmployee.next();
        int length = rsCountEmployee.getInt("rowcount");
        employee = new int[length][3];
        ResultSet rsEmployeeCurrentStartLocation = GenerateRunTime.getStatement().executeQuery(employeeCurrentStartLocation);

        for(int i = 0;i<length && rsEmployeeCurrentStartLocation.next();i++){
            employee[i][0] = rsEmployeeCurrentStartLocation.getInt(1);
            employee[i][1] = rsEmployeeCurrentStartLocation.getInt(2);
            employee[i][2] = rsEmployeeCurrentStartLocation.getInt(2);//used to reset start location after each day
        }
    }
    }catch(Exception e){
        e.printStackTrace();
    }
    return employee;
RandomArray Class

/*
 * RandomArray.java
 *
 * Created on 10 October 2004, 22:46
 */
package RunTimeDb;

import java.util.Random;

/**
 * @author omahmood
 */

public class RandomArray {
    private Random rand = new Random();
    private int numbers[];

    /** Creates a new instance of RandomArray
     */
* Randomly generates an array of values totalling 'total'
*/

public int[] randomArray(int elements, int total, int minimum) {
    numbers = new int[elements];
    int delta = total - (minimum * elements);

    //Create a sorted array
    for (int i = 1; i < elements; i++)
        numbers[i] = rand.nextInt(delta);
    quickSort(1, elements - 1);
    numbers[0] = 0;

    //Replace values with the difference between each one + minimum
    for (int i = 0; i < elements - 1; i++)
        numbers[i] = numbers[i+1] - numbers[i] + minimum;
    numbers[elements - 1] = delta - numbers[elements - 1] + minimum;

    return numbers;
}

private void quickSort(int start, int end){


int pivot;
if (end-start > 1) { // more than 2 elements left to check
    pivot = numbers[end];
    int left = start-1;
    int right = end;
    while(true){
        while(numbers[++left] < pivot);
        while(pivot < numbers[--right])
            if(right == start) break; // reached the front
        if(left >= right) break; // started overlapping
        swap(left, right); // exchange a smaller value on right side for a larger one on left (trust me!)
    }
    swap(left, end); // put pivot in place
    if (start < left-1) quickSort(start, left-1);
    if (end > left+1) quickSort(left+1, end);
}
else if (end != start) {
    organise(start, end);
}
}

private void organise(int n1, int n2) {
if (numbers[n2] < numbers[n1])
    swap(n1, n2);
}

private void swap(int n1, int n2){
    int temp = numbers[n1];
    numbers[n1] = numbers[n2];
    numbers[n1] = temp;
    numbers[n2] = temp;
}
RandomGenerator Class

/*
 * RandomGenerator.java
 *
 * Created on 24 September 2004, 14:38
 */

package RunTimeDb;

import java.util.Random;
/**
 * @author omahmood
 */
public class RandomGenerator {
    private static Random random = new Random();
    public static int getInt(int upperLimit){
        return random.nextInt((upperLimit+1));
    }
    public static long getLong(long upperLimit){
        return (long)(random.nextDouble() * (upperLimit+1));
    }
}
public static double getDouble(double upperLimit) {
    return (random.nextDouble() * (upperLimit+1));
}

public static int getInt(int lowerLimit, int upperLimit) {
    return lowerLimit + random.nextInt((upperLimit-lowerLimit+1));
}

public static long getLong(long lowerLimit, long upperLimit) {
    return lowerLimit + (long)(random.nextDouble() * (upperLimit-lowerLimit+1));
}

public static double getDouble(double lowerLimit, double upperLimit) {
    return lowerLimit + (random.nextDouble() * (upperLimit-lowerLimit+1));
}
package RunTimeDb;

/**
 * Suburb.java
 *
 * Created on 30 September 2004, 15:21
 */

/**
 * @author omahmood
 */

public class Suburb {
    public int joins[] = null;
    public int id;
    public static final int COLUMNS = 8;
    public Suburb(int id,int[] joins) {
        this.id = id;
        this.joins = joins;
    }
}
public String toString()
{
    return id+"";
}
}
Customer Class

/*
 * Customer.java
 *
 * Created on 3 March 2005, 14:41
 */

package RunTimeEnv;
import java.sql.ResultSet;
/**
 * @author omahmood
 */
public class Customer {
    private int id;
    private String name;
    private Product myProducts[];
}
/** Creates a new instance of Customer */

public Customer(int id) {
    this.id = id;
    try{
        String select = "Select fName, lName from customer where id=" +id+";";
        ResultSet rsSelect = MainRunTime.getStatement().executeQuery(select);
        rsSelect.next();
        name = rsSelect.getString (1) + " " + rsSelect.getString (2);
        rsSelect.close ();
    }catch (Exception e){
        e.printStackTrace ();
    }
    try{
        String selectCount = "Select COUNT(product_id) AS rowcount from customer_profile where customer_id=" +id+";";
        ResultSet rsCount = MainRunTime.getStatement().executeQuery(selectCount);
        rsCount.next();
        myProducts = new Product[rsCount.getInt ("rowcount")];
        rsCount.close();
        String selectProduct = "Select product_id, location_id from customer_profile where customer_id=" +id+";";
        ResultSet rsSelectProduct = MainRunTime.getStatement().executeQuery(selectProduct);
    }
}
for(int i = 0;rsSelectProduct.next ();i++){  

    String productId = "SELECT Product_Profile.id FROM Product_Profile where product_id=" + rsSelectProduct.getInt(1) +" 
AND location_id=" + rsSelectProduct.getInt(2) + ";";

    ResultSet rsSelectProductId = MainRunTime.getStatement().executeQuery(productId);
    rsSelectProductId.next();
    myProducts[i] = MainRunTime.getProduct(rsSelectProductId.getInt (1));
    rsSelectProductId.close();
}
rsSelectProduct.close ();
}catch (Exception e){  
e.printStackTrace ();
}

private void populateCustomerProducts(){
    try{

        String totalNumberOfproducts = "SELECT COUNT(product_id) AS rowcount from customer_profile where customer_id=" + id + " order by customer_id;";

        ResultSet rsCount = MainRunTime.getStatement ().executeQuery (totalNumberOfproducts);
        rsCount.next ();
        myProducts = new Product[rsCount.getInt ("rowcount")];
rsCount.close();

String custProductsSelect = "Select product_id, location_id from customer_profile where customer_id=" + id + " order by customer_id;";
ResultSet rs = MainRunTime.getStatement().executeQuery(custProductsSelect);
int i = 0;
while(rs.next()){
    String myProduct = "Select id from product_profile where product_id=" + rs.getInt(1) + " AND location_id=" + rs.getInt(2) + ",";
    ResultSet rsMyProducts = MainRunTime.getStatement().executeQuery(myProduct);
    rsMyProducts.next();
    myProducts[i] = MainRunTime.getProduct(rsMyProducts.getInt(1));
    i++;
}
} catch (Exception e){
e.printStackTrace();
}
Device Class

/*
 * Device.java
 *
 * Created on 2 March 2005, 13:01
 */
package RunTimeEnv;
import java.sql.ResultSet;
import java.util.Vector;
/**
 *
 * @author omahmood
 */
public class Device implements Cloneable{

    private Vector <CacheData> cache = null;

    private int id = 0;
    private int cacheCapacity = 0;
    private int availableCache = 0;
    private String title;
private Network supportedNetworks[] = null;
private MimeTpye supportedMimeTypes[] = null;

/**< * Creates a new instance of Device */
public Device(int id){
    try{
        String devices = "Select id,cacheCapacity,title from device where id = " + id + ";";
        ResultSet rsSelectDevice = MainRunTime.getStatement().executeQuery(devices);
        rsSelectDevice.next();
        this.id = id;
        cacheCapacity = rsSelectDevice.getInt(2);
        availableCache = cacheCapacity;
        title = rsSelectDevice.getString(3);
        rsSelectDevice.close();
        String networks = "Select networkType_id from device_supports_networktype where device_id= " + id + ";";
        ResultSet rsNetworks = MainRunTime.getStatement().executeQuery(networks);
        int i = 0;
        while(rsNetworks.next()){
            supportedNetworks = new Network[i];
            i++;
        }
    }
    rsNetworks.first();
    supportedNetworks = new Network[i];
for(int x = 0;x<i;x++){
    supportednetworks[x] = MainRunTime.getNetwork(rsNetworks.getInt(1));
    rsNetworks.next();
}
rsNetworks.close();

String mimeTypes = "Select mimeType_id from device_supports_mimetype where device_id=" + id + ";";
ResultSet rsMimeTypes = MainRunTime.getStatement().executeQuery(mimeTypes);

i = 0;
while(rsMimeTypes.next()){
    i++;
}
rsMimeTypes.first();

supportedMimeTypes = new MimeType[i];
for(int x = 0;x<i;x++){
    supportedMimeTypes[x] = new MimeType(rsMimeTypes.getInt(1));
    rsMimeTypes.next();
}
}catch (Exception e){
    e.printStackTrace();
}
public Device getDevice(){
    try{
        cache = new Vector<CacheData>();
        return (Device)this.clone();
    }catch (Exception e){
        e.printStackTrace();
    }
    return null;
}

public int getId(){
    return id;
}

public Network[] getNetworks(){
    return this.supportednetworks;
}

public int getCache(){
    return this.availableCache;
}

dобавить в кеш(int mimeId, int size, int prod_id){
    if((availableCache - size) < 0)
        return false;

this.cache.add(new CacheData(mimeId, size, prod_id));
availableCache = availableCache - size;
return true;
}

public int getCacheCapacity(){
    return this.cacheCapacity;
}

public void addToCacheWithoutKnapsack(Product product, double availableTime, double maxData, int product_id, int predictedLocationId, int bestBandwidth, int employee_id, int appId){
    ProductData dataObjects[] = product.getData();
double time = 0.0;
int newDataSize = 0;

for(int i = 0; i < dataObjects.length && maxData > 0 && availableCache > 0; i++){
    int data[] = dataObjects[i].getInfo();
    for(int x = 0; x < this.supportedMimeTypes.length; x++){
        if(data[2] == this.supportedMimeTypes[x].getId()){
            // check if the data is not already cached in device
            if(!checkDataInCache(product_id, data[2], data[1])){
                maxData = (maxData - (double)data[1]);
                availableCache = (availableCache - data[1]);
                if(availableCache < 0){
                    return false;
                }
                availableCache = availableCache - data[1];
                maxData = (maxData - (double)data[1]);
            }
        }
    }
}
return true;
}
removeLeastUsed(product_id);

if(maxData >= 0 && availableCache > 0 && availableTime >= time){
    this.cache.add(new CacheData(data[2],data[1],product_id));
    newDataSize += data[1];
    time = time + ((double)data[1]/(double)bestBandwidth);
} // else
    // break;

} //if the mimeType is supported by the Device

} //for each supported mimeType
} //final for loop

if(availableTime > time || (availableCache < 0)){
    if(availableCache < 0){
        removeLeastUsed(product_id);
    }
}

if(time > 0.0)
    writeToPredictedLocation(predictedLocationId,time,bestBandwidth,newDataSize,employee_id,appId);

public void addToCacheWithKnapsack(Product product,double availableTime,double maxData,int product_id,int
predictedLocationId,int bestBandwidth,int employee_id,int appId){
ProductData dataObjects[] = product.getData();
boolean allowed[] = new boolean[dataObjects.length];

for(int i = 0; i < dataObjects.length; i++){
    allowed[i] = false;
    int data[] = dataObjects[i].getInfo();
    for(int x = 0; x < this.supportedMimeTypes.length; x++){
        if(data[2] == this.supportedMimeTypes[x].getId()){
            allowed[i] = true;
        }
    }
}

int size = 0;
if(availableCache < product.getSize())
    for(int i = 0; i < dataObjects.length; i++){
        if(allowed[i])
            size += dataObjects[i].getInfo()[1];
        if(availableCache < size){
            removeLeastUsed(product_id, size);
        }
    }
boolean bestSolution[] = Knapsack.runTest(dataObjects, allowed, availableCache);
double time = 0.0;
int newDataSize = 0;

for(int i = 0;i<dataObjects.length && maxData > 0 && availableCache > 0; i++){
    if(bestSolution[i]){
        int data[] = dataObjects[i].getInfo();
        for(int x = 0;x<this.supportedMimeTypes.length;x++){
            //check if the data is not already cached in device
            if(!checkDataInCache(product_id, data[2], data[1])){
                maxData = (maxData - (double)data[1]);
                availableCache = (availableCache - data[1]);
                if(availableCache < 0){//NOT REQUIRED THANKS TO KNAPSACK
                    removeLeastUsed(product_id);
                }
                if(maxData > 0 && availableCache > 0 && availableTime > time){
                    this.cache.add(new CacheData(data[2], data[1], product_id));
                    newDataSize += data[1];
                    time = time + ((double)data[1]/(double)bestBandwidth);
                } //else
            } //else
            // break;
        }
    }
}
//for each supported mimeType
} //Knopsack allowed
} //final for loop

if(availableTime > time || (availableCache < 0)){
    if(availableCache < 0){
        removeLeastUsed(product_id);
    }
}
if(time >= 0.0)
    writeToPredictedLocation(predictedLocationId,time,bestBandwidth,newDataSize,employee_id,appId);
} //writeToPredictedLocation
private void addToCache(){
}
private void addToCache(){
}

private void writeToPredictedLocation(int predictedLocationId,double time,int bestBandwidth,int newDataSize,int employee_id,int appId){
    //write the cache to the table
    String data = "";
    int cacheSize = 0;
    int cacheSize = 0;
    for(int index=0;index<cache.size();index++){
        data += cache.get(index);
CacheData temp = (CacheData)cache.get(index);
        cacheSize += temp.getSize();
    }
    this.availableCache = cacheCapacity - cacheSize;

    String insertIntoPredictedLocations = "INSERT INTO CACHE (cache,employee_id, time,
    predictedlocation_id,bandwidth,newDataSize,availableCache,cacheSize,appId) VALUES ('" + data + "," + employee_id + "," + time + "," + predictedLocationId + "," + bestBandwidth+","+newDataSize+","+this.availableCache+"," + cacheSize+","+appId+"');"
    try{
        int st = MainRunTime.getStatement().executeUpdate(insertIntoPredictedLocations);
    }catch (Exception e){
        e.printStackTrace();
    }
}

private void removeFromCache(int mimeId,int prod_id){
    for(int index=0;index<cache.size();index++) {
        CacheData temp = (CacheData)this.cache.get(index);
        boolean flag = temp.equals(prod_id,mimeId);
        if(flag){
            CacheData temp2 = (CacheData)this.cache.get(index);
            this.availableCache = this.availableCache + temp2.getSize();
            this.cache.remove(index);            }
    }
}
private void sortCacheByAccess() {
    for (int pass = 1; pass < cache.size(); pass++)
        for (int i = 0; i < cache.size() - 1; i++) {
            CacheData temp = (CacheData) cache.get(i);
            CacheData temp2 = (CacheData) cache.get(i + 1);
            if (temp.getFreqOfAccess() < temp2.getFreqOfAccess())
                swap(i, i + 1);
        }
}

private void swap(int first, int second) {
    CacheData hold;
    hold = (CacheData) cache.get(first);
    cache.add(first, cache.get(second));
    cache.add(second, hold);
}

private int getLowestAccessedProductId(int product_id) {
    int freq = 99999;
    int productToBeDeleted = -1;
    for (int i = 0; i < cache.size(); i++) {
        CacheData temp = (CacheData) cache.get(i);
if(temp.getFreqOfAccess() <= freq && temp.equals(product_id) == false){
    freq = temp.getFreqOfAccess();
    productToBeDeleted = temp.getId();
}

if(productToBeDeleted != -1)
    return productToBeDeleted;
else
    return -1;

public boolean checkDataInCache(int product_id, int mimeId, int size){
    for(int index=0;index<cache.size();index++) {
        CacheData temp = (CacheData) this.cache.get(index);
        boolean flag = temp.equals(product_id, mimeId, size);
        if(flag)
            return true;
    }
    return false;
}

public void resetCache(){
    this.availableCache = this.cacheCapacity;
    cache.clear();
}
cache.setSize(0);
}

private void removeFromCache(int productId){
    for(int index=cache.size()-1;index>=0;index--){
        CacheData temp = (CacheData)this.cache.get(index);
        if(temp.equals(productId)){
            this.availableCache = this.availableCache + temp.getSize();
            this.cache.remove(index);
        }
    }
}

private void removeLeastUsed(int product_id,int size){
    for(int i = 0;i<cache.size() && (this.availableCache-size) < 0 ;i++ ){
        int id = getLowestAccessedProductId(product_id);
        if(id != -1)
            removeFromCache(id);
    }
}

private void removeLeastUsed(int product_id){
    for(int i = 0;i<cache.size() && this.availableCache < 0 ;i++ ){
        int id = getLowestAccessedProductId(product_id);
        if(id != -1)
            removeFromCache(id);
    }
}
public void access(int productId, int employee_id) {
    int productTypeId = -1;
    try{
        String selectProductTypeId = "Select product_id from product_profile where id="+productId+";";
        ResultSet rs = MainRunTime.getStatement().executeQuery(selectProductTypeId);
        rs.next();
        productTypeId = rs.getInt(1);
    }catch (Exception e){
        e.printStackTrace();
    }

    int weights[][] = MainRunTime.getEmployee(employee_id).getWeights();
    for(int index=cache.size()-1;index>=0;index--){
        CacheData temp = (CacheData)this.cache.get(index);
        if(temp.equals(productId)){
            for(int i = 0;i<weights.length;i++){
                if(weights[i][0] == productTypeId){
                    temp.dataAccessed(weights[i][1]);
                }
            }
        }
    }
}
public MimeType[] getSupportedMimeTypes(){
    return supportedMimeTypes;
}

public class CacheData{
    private int dataMimeType = 0;
    private int size = 0;
    private int productId = 0;
    private int freqOfAccess = 0;
    //private int weight = 0;

    public CacheData(int mimeId,int size,int prod_id){
        this.dataMimeType = mimeId;
        this.size = size;
        this.productId = prod_id;
    }

    public boolean equals(int prod_id,int mimeId){
        if(this.productId == prod_id && this.dataMimeType == mimeId)
            return true;
        return false;
    }
}
public boolean equals(int prod_id, int mimeId, int size) {
    if (this.productId == prod_id && this.dataMimeType == mimeId && this.size == size)
        return true;
    return false;
}

public boolean equals(int prod_id) {
    if (this.productId == prod_id)
        return true;
    return false;
}

public int getId() {
    return this.productId;
}

public int getSize() {
    return this.size;
}

public String toString() {
    return size + "$$" + productId + "$$" + freqOfAccess + "$$" + dataMimeType + "^%^";f / " w=" + weight;
}

public void dataAccessed(int weight) {
this.freqOfAccess++;
this.freqOfAccess = (freqOfAccess * weight);
}
public int getFreqOfAccess(){
    return this.freqOfAccess;
}
}
Employee Class

-packages RunTimeEnv;

/**
 * Employee.java
 *
 * Created on 2 March 2005, 13:01
 */

import java.sql.ResultSet;

public class Employee {
    private Device currentDevice;
    private int id;
}
private String name;
private Location currentLocation;
private Location nextLocation;
private Device availableDevices[];
private Product currentProduct = null;
private Product myProducts[];
//productWeight[][][0] = product id//NOT THIS REPRESENTS THE PRODUCT TYPES NOT THE PHYSICAL PRODUCTS
//productWeight[][][1] = product weight
private int[] productWeight[];

/** Creates a new instance of Employee */
public Employee(int id) {
    this.id = id;
    try{
        String selectEmployeeInfo = "Select fName, lName, startLocation from employee where employee.id=" + id + ";";
        ResultSet rsSelect = MainRunTime.getStatement().executeQuery(selectEmployeeInfo);

        rsSelect.next();
        name = rsSelect.getString(1) + " " + rsSelect.getString(2);
        int currentLocationId = rsSelect.getInt(3);
        currentLocation = MainRunTime.getLocation(currentLocationId);
selectEmployeeInfo = "Select product_id, device_id from employee_predictedlocations where employee_id="+id+" AND location_id=" + currentLocationId + " ORDER BY id;"

rsSelect = MainRunTime.getStatement().executeQuery(selectEmployeeInfo);

rsSelect.next();
currentDevice = MainRunTime.getDevice(rsSelect.getInt(2));

String empCurrentProd= "Select id from product_profile where location_id= "+currentLocationId+" AND product_id=" + rsSelect.getInt(1) + " ORDER BY id;"

rsSelect = MainRunTime.getStatement().executeQuery(selectEmployeeInfo);

rsSelect.next();
currentProduct = MainRunTime.getProduct(rsSelect.getInt(1));

selectEmployeeInfo = "Select device_id from employee_has_device where employee_id="+id+" ORDER BY employee_id;"

rsSelect = MainRunTime.getStatement().executeQuery(selectEmployeeInfo);

int i = 0;

while(rsSelect.next()){
  i++;
}

availableDevices = new Device[i];
rsSelect.first();

for(int x = 0;x<i;x++){
    availableDevices[x] = MainRunTime.getDevice(rsSelect.getInt(1));
    rsSelect.next();
}

String employeeProducts = "SELECT Product_Profile.id FROM Employee_Profile, Product_Profile WHERE
((Employee_Profile.product_id)=Product_Profile.product_id) AND ((Employee_Profile.location_id)=Product_Profile.location_id) AND ((Employee_Profile.employee_id)=" + id + ");";
rsSelect = MainRunTime.getStatement().executeQuery(employeeProducts);
i = 0;
while(rsSelect.next()){  
i++;
}

myProducts = new Product[i];
rsSelect.first();

for(int x = 0;x<i;x++){  
    myProducts[x] = MainRunTime.getProduct(rsSelect.getInt(1));
}
rsSelect.next();
}
catch(Exception e){
e.printStackTrace();
}
allocateProductWeight();
}
public int getId(){
    return id;
}
public Device getDevice(int id){
    for(int i = 0;i<availableDevices.length;i++){
        if(availableDevices[i].getId() == id)
            return availableDevices[i];
    }
    return null;
}
public int[][] getWeights(){
    return this.productWeight;
}
private void allocateProductWeight(){
    try{

String selectProductData = "SELECT COUNT(product_id) AS rowcount FROM product;";
ResultSet rs = MainRunTime.getStatement().executeQuery(selectProductData);
rs.next();
int size = rs.getInt(1);
this.productWeight = new int[size][2];
for(int i = 0;i < this.productWeight.length;i++){
    this.productWeight[i][0] = -1;
    this.productWeight[i][1] = -1;
}
String selectProductId = "SELECT product_id FROM product;";
ResultSet rsProductId = MainRunTime.getStatement().executeQuery(selectProductId);
int i = 0;
while(rsProductId.next()){
    String selectProductIdFromEmployeeProfile  = "SELECT product_id FROM employee_profile where employee_id=" + id + " AND product_id=" + rsProductId.getInt(1) + ";";
    ResultSet rsEmployeeProduct = MainRunTime.getStatement().executeQuery(selectProductIdFromEmployeeProfile);
    if(rsEmployeeProduct.next()){
        productWeight[i][0] = rsEmployeeProduct.getInt(1);
        productWeight[i][1] = RunTimeDb.RandomGenerator.getInt(1, 5);
i++;

} catch (Exception e) {
    e.printStackTrace();
    
}
Knapsack Class

/*
 * Knapsack.java
 *
 * Created on March 23, 2005, 12:31 AM
 */

package RunTimeEnv;

/**
 * @author Omer Mahmood
 * @version
 */

public class Knapsack {

    private int bestPriority;
    private int bestSize;
    private boolean [] bestSolution;
    private boolean [] currentSolution;
    private int [] size;
    private int [] priority;

private int availableCache;
private int numberOfItems;
private int calls;

/** Creates a new instance of Knapsack */
public Knapsack (int availableCache, int [] size, int [] priority){
    this.availableCache = availableCache;
    this.size = size;
    this.priority = priority;
    bestPriority = 0;
    numberOfItems = priority.length;
    bestSolution = new boolean[numberOfItems];
    currentSolution = new boolean[numberOfItems];
    for (int i=0; i<numberOfItems; i++){
        bestSolution[i]=false;
        currentSolution[i]=false;
    }
    calls=0;
    this.search (0, 0, 0);
}

public int getBestPriority (){  
    return bestPriority;
public boolean [] getBestSolution (){
    return bestSolution;
}

public int [] getSize (){  
    return size;  
}

public int [] getPriority (){  
    return priority;  
}

private void storeNewBestSolution (){  
    for (int i=0; i<numberOfItems; i++)  
        bestSolution[i]=currentSolution[i];
}

public String toString (){  
    String s = "";

    s = s + "sizes: ";
    for (int i=0; i<numberOfItems; i++)
        s = s + size[i] + " ";

    s = s + "priorities: ";
    for (int i=0; i<numberOfItems; i++)
        s = s + priority[i] + " ";
s = s + "\nTake: ";
for (int i=0; i<numberOfItems; i++)
    s = s + bestSolution[i] + " ";
s = s + "availableCache: " + availableCache;
s = s + " size: " + bestSize;
s = s + " priority: " + bestPriority;
s = s + " Calls: " + calls;
return s;
}

private String showCurrent (int depth, int weight, int value){
    String s = "";
    for (int i=0;i<depth;i++)
        s = s + currentSolution[i];
s = s + " size: " + weight + " priority: " + value;
    if (value>bestPriority && weight<=availableCache)
        s = s + " *** new best ***",
    return s;
}

public void search (int depth, int currentSize, int currentPriority){
calls++;
}
/*System.out.println (showCurrent (depth,currentSize, currentPriority));*/
if (currentSize<=availableCache && currentPriority>bestPriority){
    bestPriority=currentPriority;
    bestSize=currentSize;
    storeNewBestSolution ();
}
if (depth<numberOfItems){
    currentSolution[depth]=true; // take, go left
    search (depth+1, currentSize+size[depth],currentPriority+priority[depth]);
    currentSolution[depth]=false; // don't take, go right
    search (depth+1, currentSize, currentPriority);
}
}
public static boolean[] runTest (ProductData dataObjects[], boolean allowed[], int availableCache){
    //int [] size = {3,8,6,4,2};
    //int priority [] = {2,12,9,3,5};
    //int cache = 19;

    //Knapsack KS = new Knapsack (cache, size, priority);
    //System.out.println (KS);

    int size[] = new int[dataObjects.length];
int priority[] = new int[dataObjects.length];
for(int i = 0;i<dataObjects.length;i++){
    size[i] = dataObjects[i].getSize();
    if(allowed[i])
        priority[i] = dataObjects[i].getMimePriority();
    else
        priority[i] = -1;
}
Knapsack ks = new Knapsack(availableCache, size, priority);
//System.out.println(ks);
return ks.getBestSolution();
}
Location Class

/*
 * Location.java
 * *
 * Created on 2 March 2005, 13:11
 */

package RunTimeEnv;
import java.sql.ResultSet;
/**
 ** Creates a new instance of Location */
    public Location (int id){
        this.id = id;
        try{
            try{
String select = "Select title,postCode from location where id=" + id + ";";
ResultSet rs = MainRunTime.getStatement().executeQuery(select);
rs.next();
title = rs.getString(1);
postCode = rs.getInt(2);
rs.close();

String selectNetwork = "SELECT COUNT(network_id) AS rowcount FROM location_available_network where location_id=" + id+ ";";
ResultSet rsCount = MainRunTime.getStatement().executeQuery(selectNetwork);
rsCount.next();
availableNetworks = new Network[rsCount.getInt("rowcount")];
rsCount.close();

String networks = "Select network_id from location_available_network where location_id=" + id+ ";";
ResultSet rsNetwork = MainRunTime.getStatement().executeQuery(networks);
for(int i = 0;rsNetwork.next();i++){
    availableNetworks[i] = MainRunTime.getNetworkClone(rsNetwork.getInt(1));
}
rsNetwork.close();

}catch(Exception e){
e.printStackTrace();
}
}

public int getId(){
    return id;
}

public Network[] getNetworks(){
    return availableNetworks;
}

}
MainRunTime Class

/*
 * MainRunTime.java
 *
 * Created on 2 March 2005, 13:22
 */

package RunTimeEnv;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.StringTokenizer;
/**
 *
 * @author omahmood
 */

public class MainRunTime {
    private static Employee employees[];

    private final String DB_URL_RUNTIME = "jdbc:mysql:///application_runtime_data?user=root";
    private static Connection connRun = null;
    private static Statement stmtRun = null;
}
private static Network networks[];
private static Location locations[];
private static Product products[];
private static Mime.Type mimeTypes[];
private static Device devices[];
private static Customer customers[];

public static void main(String args[]){
    new MainRunTime();
}

/** Creates a new instance of MainRunTime */
public MainRunTime() {
    setUp();
    try{
        networksSetup();
        mimeTypesSetup();
        locationsSetup();
        productsSetup();
        deviceSetup();
        employeeSetup();
        customerSetup();
    }
}
}catch (Exception e){
    e.printStackTrace();
}
new WithoutProfile();
new WithProfile();
checkForCacheAtAppointmentTime();
close();
}

private void calculateTotalUtilityPerAppointment(){
    try{
        String appointmentSelect = "Select id from appointment order by id;";
        ResultSet stAppointment = MainRunTime.getStatement().executeQuery(appointmentSelect);
        while(stAppointment.next()){                 int appId = stAppointment.getInt(1);

            String sumOfUtility = "SELECT SUM(u tility) as totalUtility FROM UtilityPerProductDataD ownloadedWithProfile WHERE 
appointment_id = " + appId + ";";
            ResultSet stUtility = MainRunTime.getStatement().executeQuery(sumOfUtility);
            stUtility.next();                 int sumUtility = stUtility.getInt("totalUtility");
            stUtility.close();
    }
}
String sumOfTime = "SELECT SUM(time) as totalTime FROM cache WHERE appId = " + appId + ";";
ResultSet stTime = MainRunTime.getStatement().executeQuery(sumOfTime);
stTime.next();
int sumTime = stTime.getInt("totalTime");
stTime.close();

String sumOfData = "SELECT SUM(newDataSize) as totalData FROM cache WHERE appId = " + appId + ";";
ResultSet stData = MainRunTime.getStatement().executeQuery(sumOfData);
stData.next();
int sumData = stData.getInt("totalData");
stData.close();

String insertIntoAppointment = "UPDATE appointment SET appointment.timeToDownloadWithProfile=" + sumTime + ",
appointment.utilityOfCachedDataWithProfile=" + sumUtility + ", appointment.dataDownloadedWithProfile=" + sumData + " where
appointment.id=" + appId + ";";
    int temp = MainRunTime.getStatement().executeUpdate(insertIntoAppointment);
    
}  
}catch (Exception e){
    e.printStackTrace();
}

private void checkForCacheAtAppointmentTime(){


try{
    String selectappointment = "Select id,product_id,device_id,employee_id from appointment order by employee_id;";
    ResultSet rsAppointment = MainRunTime.getStatement().executeQuery(selectappointment);
    while(rsAppointment.next()){
        int appId = rsAppointment.getInt(1);
        int prodId = rsAppointment.getInt(2);
        int deviceId = rsAppointment.getInt(3);
        int employee_id = rsAppointment.getInt(4);
        int weights[][] = MainRunTime.getEmployee(employee_id).getWeights();
        ProductData pData[] = MainRunTime.getProduct(prodId).getData();
        String selectProductTypeId = "Select product_id from product_profile where id=\"+prodId+\";";
        ResultSet rs = MainRunTime.getStatement().executeQuery(selectProductTypeId);
        rs.next();
        int productTypeId = rs.getInt(1);
        rs.close();
        boolean deviceSupport[] = new boolean[pData.length];
        boolean inCache[] = new boolean[pData.length];
        for(int i = 0;i<deviceSupport.length;i++){
            deviceSupport[i] = false;
            inCache[i] = false;
        }
    }
}
filterDataForDevice(pData, deviceId, deviceSupport);

String selectCacheId = "Select MAX(predictedlocation_id) from cache where appId=" + appId + ";";
ResultSet rsCacheId = MainRunTime.getStatement().executeQuery(selectCacheId);
rsCacheId.next();
int predictedLocationId = rsCacheId.getInt(1);
rsCacheId.close();

String selectproductAndCache = "Select employee_predictedlocations.product_id from employee_predictedlocations where employee_predictedlocations.id=" + predictedLocationId + ";";
ResultSet rsProductIdAndCache = MainRunTime.getStatement().executeQuery(selectproductAndCache);
if(rsProductIdAndCache.next()){
    String selectCache = "Select cache from cache where predictedlocation_id=" + predictedLocationId + ";";
    ResultSet rsCache = MainRunTime.getStatement().executeQuery(selectCache);
    rsCache.next();
    String cache = rsCache.getString(1);
    rsCache.close();

    StringTokenizer st = new StringTokenizer(cache,"^%^",false);
    while(st.hasMoreTokens()){
        ...
int[] data = new int[3];
String subSt = st.nextToken();
StringTokenizer subTokens = new StringTokenizer(subSt, "$$", false);
data[1] = Integer.parseInt(subTokens.nextToken()); // SIZE
data[0] = Integer.parseInt(subTokens.nextToken()); // productId
subTokens.nextToken(); // SKIP FREQ OF USE
data[2] = Integer.parseInt(subTokens.nextToken()); // MIME
for (int i = 0; i < pData.length; i++) {
    int[] info = pData[i].getInfo();
        inCache[i] = true;
}
int utility = 0;
for (int i = 0; i < pData.length; i++) {
    if (deviceSupport[i] == true && inCache[i] == false) { // MISSING INFORMATION FROM CACHE
        int size = 0;
        try {
            String getSize = "Select size FROM productdata where p_id=" + prodId + " AND mimeId=" + pData[i].getInfo()[2] + "; ";
            ResultSet stSize = MainRunTime.getStatement().executeQuery(getSize);
            // Process the ResultSet...
        } catch (Exception e) {
            // Handle exception...
        }
    }
}
if(stSize.next())
    size = stSize.getInt(1);

String insertIntoMissedDataWithProfile = "Insert into misseddatawithprofile (appoint_id, mime_id, product_id,
device_id, size,predictedLocation_id) VALUES (" + appId + "," + pData[i].getInfo()[2]+ "," + prodId + "," + deviceId + "," + size + "," + predictedLocationId + ");";

    int t = MainRunTime.getStatement().executeUpdate(insertIntoMissedDataWithProfile);
}catch (Exception e){
    e.printStackTrace();
}

} else if(deviceSupport[i] == true && inCache[i] == true){//ADDED DATA TO CACHE
for(int x = 0;x<weights.length;x++){
    if(weights[x][0] == productTypeId){
        utility = pData[i].getMimePriority()*weights[x][1];
    }
}

String inertIntoUtilityPerProductDataDownloadedWithProfile = "Insert INTO UtilityPerProductDataDownloadedWithProfile (appointment_id, utility) VALUES (" + appId + "," + utility + ");";
    int t = MainRunTime.getStatement().executeUpdate(inertIntoUtilityPerProductDataDownloadedWithProfile);
}
}catch (Exception e){
    e.printStackTrace();
}
calculateTotalUtilityPerAppointment();
}

public static void filterDataForDevice(ProductData data[], int deviceId, boolean[] deviceSupport){
    MimeTypes mimeTypes[] = MainRunTime.getDevice(deviceId).getSupportedMimeTypes();
    for(int i = 0; i < data.length; i++){
        int info[] = data[i].getInfo();
        for(int x = 0; x < mimeTypes.length; x++){
            if(info[2] == mimeTypes[x].getId()){
                deviceSupport[i] = true;
            }
        }
    }
}

public static Network getNetworkClone(int id){
    for(int i = 0; i < networks.length; i++){
        if(networks[i].getId() == id)
            return networks[i].getNetwork();
    }
    return null;
public static Network getNetwork(int id) {
  for (int i = 0; i < networks.length; i++) {
    if (networks[i].getId() == id) {
      return networks[i];
    }
  }
  return null;
}

public static Product getProduct(int id) {
  for (int i = 0; i < products.length; i++) {
    if (products[i].getId() == id) {
      return products[i];
    }
  }
  return null;
}

public static Device getDevice(int id) {
  for (int i = 0; i < devices.length; i++) {
    if (devices[i].getId() == id) {
      return devices[i].getDevice();
    }
  }
  return null;
}
public static Location getLocation(int id){
    for(int i = 0; i < locations.length; i++){
        if(locations[i].getId() == id)
            return locations[i];
    }
    return null;
}

public static MimeTpe getMime(int id){
    for(int i = 0; i < mimeTypes.length; i++){
        if(mimeTypes[i].getId() == id)
            return mimeTypes[i];
    }
    return null;
}

private void deviceSetup() throws Exception{
    String select = "SELECT COUNT(id) AS rowcount FROM device;"
    ResultSet rsCount = stmtRun.executeQuery(select);
    rsCount.next();
    devices = new Device[rsCount.getInt("rowcount")];

    select = "SELECT id FROM device;"
    rsCount = stmtRun.executeQuery(select);
for(int i = 0;rsCount.next();i++){             devices[i] = new Device(rsCount.getInt(1));
}

private void customerSetup(){
    try{
        String selectEmployeeId = "Select id from customer ORDER BY id;";
        ResultSet rsSelect = stmtRun.executeQuery(selectEmployeeId);

        int i = 0;

        while(rsSelect.next()){                 i++;
        }
        customers = new Customer[i];
        rsSelect.first();

        for(int x = 0;x<i;x++){                 customers[x] = new Customer(rsSelect.getInt(1));
                rsSelect.next();
        }
        rsSelect.close();
    }
private void employeeSetup(){
    try{
        String selectEmployeeId = "Select id from employee ORDER BY id;";
        ResultSet rsSelect = stmtRun.executeQuery(selectEmployeeId);

        int i = 0;

        while(rsSelect.next()){
            i++;
        }
        employees = new Employee[i];
        rsSelect.first();

        for(int x = 0;x<i;x++){
            employees[x] = new Employee(rsSelect.getInt(1));
            rsSelect.next();
        }
    }
    rsSelect.close();
}
try {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

private void networksSetup() throws Exception{
    String selectNetwork = "SELECT COUNT(id) AS rowcount FROM networktype;";
    ResultSet rsCount = stmtRun.executeQuery(selectNetwork);
    rsCount.next();
    networks = new Network[rsCount.getInt("rowcount")];

    selectNetwork = "SELECT id FROM networktype;";
    rsCount = stmtRun.executeQuery(selectNetwork);
    for(int i = 0;rsCount.next();i++){
        networks[i] = new Network(rsCount.getInt(1));
    }
}

private void locationsSetup() throws Exception{
    String select = "SELECT COUNT(id) AS rowcount FROM location;";
    ResultSet rsCount = stmtRun.executeQuery(select);
    rsCount.next();
    locations = new Location[rsCount.getInt("rowcount")];
select = "SELECT id FROM location;";
rsCount = stmtRun.executeQuery(select);
for(int i = 0;rsCount.next();i++){
    locations[i] = new Location(rsCount.getInt(1));
}

private void productsSetup() throws Exception{
    String select = "SELECT COUNT(id) AS rowcount FROM product_profile;";
    ResultSet rsCount = stmtRun.executeQuery(select);
    rsCount.next();
    products = new Product[rsCount.getInt("rowcount")];

    select = "SELECT id FROM product_profile;";
    rsCount = stmtRun.executeQuery(select);
    for(int i = 0;rsCount.next();i++){
        products[i] = new Product(rsCount.getInt(1));
    }
}

private void mimeTypesSetup() throws Exception{
    String select = "SELECT COUNT(id) AS rowcount FROM mimetype;";
    ResultSet rsCount = stmtRun.executeQuery(select);
    rsCount.next();

mimeTypes = new MimeTypeDefs(rsCount.getInt("rowcount"));

select = "SELECT id FROM mimetype;"
rsCount = stmtRun.executeQuery(select);
for(int i = 0;rsCount.next();i++){
    mimeTypes[i] = new MimeTypeDefs(rsCount.getInt(1));
}

public static Employee getEmployee(int id){
    for(int i = 0;i<employees.length;i++){
        if(employees[i].getId() == id)
            return employees[i];
    }
    return null;
}

private void setUp(){
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        connRun = DriverManager.getConnection(DB_URL_RUNTIME);
        stmtRun = connRun.createStatement();
    }catch(Exception e){
        e.printStackTrace();
    }
}
try{
    int t = MainRunTime.getStatement().executeUpdate("DROP TABLE IF EXISTS misseddatawithoutprofile;" );
    t = MainRunTime.getStatement().executeUpdate("DROP TABLE IF EXISTS misseddatawithprofile; ");
    t = MainRunTime.getStatement().executeUpdate("DROP TABLE IF EXISTS UtilityPerProductDataDownloadedWithoutProfile; ");
    t = MainRunTime.getStatement().executeUpdate("DROP TABLE IF EXISTS UtilityPerProductDataDownloadedWithProfile; ");
    t = MainRunTime.getStatement().executeUpdate("CREATE TABLE misseddatawithoutprofile ( id int(11) NOT NULL auto_increment, appoint_id int(11) default '0', mime_id int(11) default '0', product_id int(11) default '0', device_id int(11) default '0', size int(11) default '0', reason varchar(255) default NULL, PRIMARY KEY (id), KEY appoint_id (appoint_id), KEY device_id (device_id), KEY id (id), KEY product_id (product_id)) TYPE=MyISAM; ");
    t = MainRunTime.getStatement().executeUpdate("CREATE TABLE misseddatawithprofile ( id int(11) NOT NULL auto_increment, appoint_id int(11) default '0', mime_id int(11) default '0', product_id int(11) default '0', device_id int(11) default '0', size int(11) default '0', predictedLocation_id int(11) default NULL, PRIMARY KEY (id), KEY appoint_id (appoint_id), KEY device_id (device_id), KEY id (id), KEY product_id (product_id)) TYPE=MyISAM; ");
    t = stmtRun.executeUpdate("CREATE TABLE UtilityPerProductDataDownloadedWithoutProfile ( id int(11) NOT NULL auto_increment, appointment_id int(11) default '0', utility int(11) default '0', PRIMARY KEY (id), KEY appointment_id (appointment_id), KEY id (id)) TYPE=MyISAM; ");
    t = stmtRun.executeUpdate("CREATE TABLE UtilityPerProductDataDownloadedWithProfile ( id int(11) NOT NULL auto_increment, appointment_id int(11) default '0', utility int(11) default '0', PRIMARY KEY (id), KEY appointment_id (appointment_id), KEY id (id)) TYPE=MyISAM; ");
}
catch (Exception e) {
    e.printStackTrace();
}

public static Statement getStatement() {
    try {
        return connRun.createStatement();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

private void close() {
    try {
        stmtRun.close();
        connRun.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
package RunTimeEnv;
import java.sql.ResultSet;

/**
  * @author Omer Mahmood
  * @version
  */

public class MimeType {
    private int id;
    private int priority;
    private String title;
    private String extension;

    /** Creates a new instance of MimeType */
    public MimeType (int id) {
this.id = id;

try{
    String selectMimeInfo = "Select extension, title, priority_id from mimetype where id=" +id+ ";";
    ResultSet rsSelect = MainRunTime.getStatement().executeQuery(selectMimeInfo);
    rsSelect.next();
    title = rsSelect.getString (2);
    extension = rsSelect.getString (1);
    priority = rsSelect.getInt (3);
    rsSelect.close ();
}catch (Exception e){
    e.printStackTrace ();
}

public int getId(){
    return id;
}

public int getPriority(){
    return this.priority;
}
}
Network Class

/*
 * Network.java
 *
 * Created on 2 March 2005, 14:17
 */

package RunTimeEnv;
import java.sql.ResultSet;
/**
 * @author omahmood
 */
public class Network implements Cloneable{
    private int id;
    private String title;
    private int maxBandwidth;
    private int currentBandwidth = 0;
    /**
     * Creates a new instance of Network *
     */

    public Network(int id){
        id = id;
this.id = id;

try{
    String select = "Select title,maxBandwidth from networktype where id=" + id + ";";
    ResultSet rs = MainRunTime.getStatement().executeQuery(select);
    rs.next();
    title = rs.getString(1);
    maxBandwidth = rs.getInt(2);
    rs.close();
} catch(Exception e){
    e.printStackTrace();
}

public int getId(){
    return id;
}

public int getCurrentBandwidth(){
    return currentBandwidth;
}

public void setCurrentBandwidth(int i){
    currentBandwidth = i;
}

public Network getNetwork();

try{
    Network temp = (Network)this.clone();
    temp.setCurrentBandwidth();
    return temp;
}

private void setCurrentBandwidth(){
    currentBandwidth = RunTimeDb.RandomGenerator.getInt(10,(int)maxBandwidth/2);
}

}
Product Class

/*
 * Poduct.java
 *
 * Created on 2 March 2005, 13:12
 */

package RunTimeEnv;
import java.sql.ResultSet;
/**
 * @author omahmood
 */
public class Product{
    private int id;
    private int productTypeId = 0;
    private String prod_desc, type;
    private Location myLocation;
    private ProductData myData[];
    private int size = 0;
    /** Creates a new instance of Poduct */
    public Product(int id){

try{
    this.id = id;
    String select = "SELECT Product_Profile.location_id,Product_Profile.product_id FROM Product_Profile WHERE Product_Profile.id=" + id + ";"
    ResultSet rs = MainRunTime.getStatement().executeQuery(select);
    rs.next();
    productTypeId = rs.getInt(2);
    myLocation = MainRunTime.getLocation(rs.getInt(1));

    select = "SELECT Product.prod_desc, Product.type FROM Product WHERE Product.product_id=" + rs.getInt(2) + ";"
    rs = MainRunTime.getStatement().executeQuery(select);
    rs.next();
    prod_desc = rs.getString(1);
    type = rs.getString(2);
    rs.close();

    populateProductData();
}
catch (Exception e){
    e.printStackTrace();
}
private void populateProductData(){
    try{
        String select = "SELECT COUNT(p_id) AS rowcount FROM productData where p_id=\"+ this.id +\";";
        ResultSet rsCount = MainRunTime.getStatement().executeQuery(select);
        rsCount.next();
        myData = new ProductData[rsCount.getInt("rowcount")];

        String selectProductdataId = "SELECT id FROM productData where p_id=\"+ this.id +\";";
        ResultSet rsProductData = MainRunTime.getStatement().executeQuery(selectProductdataId);

        for(int i = 0;i<myData.length && rsProductData.next();i++){
            myData[i] = new ProductData(rsProductData.getInt(1));
            this.size += myData[i].getSize();
        }
    }catch (Exception e){
        e.printStackTrace();
    }
}

public int getId(){
    return id;
}
public ProductData[] getData(){
    this.sortByMimePriority();
    return this.myData;
}

public ProductData[] getUnsortedData(){
    this.sortById();
    return this.myData;
}

private void sortById() {
    for (int pass = 1; pass < myData.length; pass++)
        for (int i = 0; i < myData.length - 1; i++)
            if (myData[i].getId() > myData[i + 1].getId())
                swap(myData, i, i + 1);
}

private void sortByMimePriority() {
    for (int pass = 1; pass < myData.length; pass++)
        for (int i = 0; i < myData.length - 1; i++)
            if (myData[i].getMimePriority() < myData[i + 1].getMimePriority())
                swap(myData, i, i + 1);
}

private void swap( ProductData c[], int first, int second ) {
    ProductData hold;
hold = c[first];
c[first] = c[second];
c[second] = hold;
}

public int getSize(){
    return this.size;
}

public int getType(){
    return productTypeId;
}
}
ProductData Class

/*
 * ProductData.java
 *
 * Created on 3 March 2005, 14:21
 */

package RunTimeEnv;
import java.sql.ResultSet;
/**
 * @author omahmood
 */
public class ProductData {
    private int id;
    private int productId;
    private MimeMimeType mime;
    private int size;
    /** Creates a new instance of ProductData */
    public ProductData(int id) {
        this.id = id;
        try{
String selectInfo = "Select mimeId, size, p_id from productData where id=" + id + ";";
ResultSet rsSelect = MainRunTime.getStatement().executeQuery(selectInfo);
rsSelect.next();
mime = MainRunTime.getMime(rsSelect.getInt(1));
size = rsSelect.getInt(2);
productId = rsSelect.getInt(3);
rsSelect.close();
}
}catch (Exception e){
e.printStackTrace();
}
}
public int getId(){
return id;
}
public int[] getInfo(){
int info[] = {productId, size, mime.getId()};
return info;
}
public int getSize(){
return size;
}
public int getMimePriority(){
return this.mime.getPriority();
}
public int getMimeId(){
    return this.mime.getId();
}
}
WithProfile Class

/*
 * WithProfile.java
 *
 * Created on 5 March 2005, 17:34
 */

package RunTimeEnv;
import java.sql.ResultSet;
import java.text.DecimalFormat;
import java.text.NumberFormat;

/**
 * @author omahmood
 */

public class WithProfile {
    double time = -1.0;
    int usedNetwork=-1;
    /** Creates a new instance of WithProfile */
    public WithProfile() {
        try{
            int t = MainRunTime.getStatement().executeUpdate("DROP TABLE IF EXISTS cache;");
            t = MainRunTime.getStatement().executeUpdate("CREATE TABLE cache (  id bigint(20) NOT NULL auto_increment,  cache longtext NOT NULL,  employee_id bigint(20) NOT NULL default '0',  time double(20,1) NOT NULL default '0.0',

228
predictedlocation_id bigint(20) NOT NULL default '0', bandwidth bigint(20) NOT NULL default '0', newDataSize bigint(20) NOT NULL default '0', availableCache bigint(20) NOT NULL default '0', appId bigint(20) NOT NULL default '0', cacheSize bigint(20) NOT NULL default '0', PRIMARY KEY (id)) TYPE=MyISAM;"

```java
try{
    String employeeLocationsSelect = "Select appointment_id, employee_id, product_id, day, location_id, timeSpent, device_id, id from employee_predictedlocations order by id;";
    ResultSet stPredictedLocations = MainRunTime.getStatement().executeQuery(employeeLocationsSelect);
    int previousAppointmentId = 0;
    int previousProductId = 0;
    int previousEmployeeId = 0;
    boolean flag = false;
    while(stPredictedLocations.next()){
        int id = stPredictedLocations.getInt(1);
        int product_id = stPredictedLocations.getInt(3);
        int day = stPredictedLocations.getInt(4);
        int location_id = stPredictedLocations.getInt(5);
```
int timeAtLocation= stPredictedLocations.getInt(6)*60;//CONVERT MINUTES TO SECONDS
int device_id = stPredictedLocations.getInt(7);
int employee_id = stPredictedLocations.getInt(2);
int predictedLocationId = stPredictedLocations.getInt(8);
String customer = "Select customer_id from appointment where id = " + id + ";";
ResultSet custAppointment = MainRunTime.getStatement().executeQuery(customer);
custAppointment.next();
int customerId = custAppointment.getInt(1);
custAppointment.close();

Network currentNetworks[] = MainRunTime.getLocation(location_id).getNetworks();
Device device = MainRunTime.getEmployee(employee_id).getDevice(device_id);
Network deviceNetworks[] = device.getNetworks();
if(previousAppointmentId != id && flag){
    stPredictedLocations.previous();
    stPredictedLocations.previous();
    previousProductId = stPredictedLocations.getInt(3);
    if(previousProductId != productId){
        //device.removeFromCache(previousProductId);
        device.access(previousProductId,previousEmployeeId);
    }
    stPredictedLocations.next();
}
stPredictedLocations.next();
int emp = stPredictedLocations.getInt(8);
}

int bestBandwidth = getBestBandwidth(currentNetworks,deviceNetworks);
calculateAndStoreProductData(device,product_id,timeAtLocation,bestBandwidth,predictedLocationId,employee_id,id);
previousAppointmentId = id;
previousProductId = product_id;
previousEmployeeId = employee_id;
flag = true;
}
}
}

private double[] calculateTimeAndData(Device device,int bestBandwidth,int productDataSize,int timeAtLocation){
    //maxTimeAndData[0] = maximimAvailableTime
    //maxTimeAndData[1] = maximumPossibleDataTransfer
    double maxTimeAndData[] = new double[2];
    maxTimeAndData[0] = 0.0;
    maxTimeAndData[1] = 0.0;
int download = 0;
if(productDataSize >= device.getCacheCapacity()) {
    download = device.getCacheCapacity();
} else {
    download = productDataSize;
}

double t = (double) download/(double)bestBandwidth;
if(t <= timeAtLocation){
    NumberFormat formatter = new DecimalFormat("0.00");
    String time = formatter.format(t);
    maxTimeAndData[0] = Double.parseDouble(time);
    maxTimeAndData[1] = download;
} else {
    NumberFormat formatter = new DecimalFormat("0.00");
    String time = formatter.format((double) timeAtLocation);
    maxTimeAndData[0] = Double.parseDouble(time);
    maxTimeAndData[1] = timeAtLocation * bestBandwidth;
}
return maxTimeAndData;

private int getBestBandwidth(Network available[], Network supported[]){
```java
int max = 0;
for (int a = 0; a < available.length; a++) {
    for (int s = 0; s < supported.length; s++) {
        if (available[a].getId() == supported[s].getId()) {
            if (available[a].getCurrentBandwidth() > max) {
                max = available[a].getCurrentBandwidth();
                this.usedNetwork = available[a].getId();
            }
        }
    }
}
return (int)max/8;  // bit to byte conversion
}

private void calculateAndStoreProductData(Device device, int product_id, int timeAtLocation, int bestBandwidth, int predictedLocationId, int employee_id, int id) {
    if (bestBandwidth > 0 && timeAtLocation > 0) {
        int dataSize = this.calculateProductDataSize(MainRunTime.getProduct(product_id));
        if (dataSize > 0) {
            double[] maxTimeAndData = calculateTimeAndData(device, bestBandwidth, dataSize, timeAtLocation);
            Product product = MainRunTime.getProduct(product_id);
```
private int calculateProductDataSize(Product product) {
    return product.getSize();
}

WithoutProfile Class

/*
 * WithoutProfile.java
 *
 * Created on 5 March 2005, 17:34
 */

package RuntimeEnv;
import java.sql.ResultSet;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.text.NumberFormat; /*
package WithoutProfile;

import java.sql.ResultSet;
import java.sql.Statement;

public class WithoutProfile {
    double time = -1.0;
    int usedNetwork = -1;
    /**
     * Creates a new instance of WithoutProfile
     */
    public WithoutProfile() {
        runMe();
    }
    private void runMe() {
        try {
            String appointmentSelect = "Select id, product_id, location_id, appointmentDuration, device_id, employee_id from appointment order by id;"
            ResultSet stAppointment = MainRunTime.getStatement().executeQuery(appointmentSelect);
            while (stAppointment.next()) {
                int id = stAppointment.getInt(1);
                int product_id = stAppointment.getInt(2);
                int location_id = stAppointment.getInt(3);
                int appointmentDuration = stAppointment.getInt(4) * 60; // converting minutes to seconds
                int device_id = stAppointment.getInt(5);
                int employee_id = stAppointment.getInt(6);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void run() {
        runMe();
    }
}

public class MainRunTime {
    public static Statement getStatement() {
        return null;
    }
}

public class Appointment {
    double time = -1.0;
    int usedNetwork = -1;
    /**
     * Creates a new instance of Appointment
     */
    public Appointment() {
        runMe();
    }
    private void runMe() {
        try {
            String appointmentSelect = "Select id, product_id, location_id, appointmentDuration, device_id, employee_id from appointment order by id;"
            ResultSet stAppointment = MainRunTime.getStatement().executeQuery(appointmentSelect);
            while (stAppointment.next()) {
                int id = stAppointment.getInt(1);
                int product_id = stAppointment.getInt(2);
                int location_id = stAppointment.getInt(3);
                int appointmentDuration = stAppointment.getInt(4) * 60; // converting minutes to seconds
                int device_id = stAppointment.getInt(5);
                int employee_id = stAppointment.getInt(6);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void run() {
        runMe();
    }
}

public class Employee {
    double time = -1.0;
    int usedNetwork = -1;
    /**
     * Creates a new instance of Employee
     */
    public Employee() {
        runMe();
    }
    private void runMe() {
        try {
            String appointmentSelect = "Select id, product_id, location_id, appointmentDuration, device_id, employee_id from appointment order by id;"
            ResultSet stAppointment = MainRunTime.getStatement().executeQuery(appointmentSelect);
            while (stAppointment.next()) {
                int id = stAppointment.getInt(1);
                int product_id = stAppointment.getInt(2);
                int location_id = stAppointment.getInt(3);
                int appointmentDuration = stAppointment.getInt(4) * 60; // converting minutes to seconds
                int device_id = stAppointment.getInt(5);
                int employee_id = stAppointment.getInt(6);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void run() {
        runMe();
    }

Network currentNetworks[] = MainRunTime.getLocation(location_id).getNetworks();
Network deviceNetworks[] = MainRunTime.getEmployee(employee_id).getDevice(device_id).getNetworks();
Device device = MainRunTime.getEmployee(employee_id).getDevice(device_id);
int bestBandwidth = getBestBandwidth(currentNetworks, deviceNetworks);
String dataAndTime[] =
calculateAndStoreProductData(device, product_id, appointmentDuration, bestBandwidth, id, employee_id);

String sumOfUtility = "SELECT SUM(utility) as totalUtility FROM UtilityPerProductDataDownloadedWithoutProfile
WHERE appointment_id = " + id + ";";

ResultSet stUtility = MainRunTime.getStatement().executeQuery(sumOfUtility);
stUtility.next();
int sumUtility = 0;
sumUtility = stUtility.getInt("totalUtility");

String insertIntoAppointment = "UPDATE appointment SET appointment.timeToDownloadWithoutProfile=" +
dataAndTime[1] + ", appointment.utilityOfCachedDataWithoutProfile=" + sumUtility + ", where appointment.id=" + id + ";";
int temp = MainRunTime.getStatement().executeUpdate(insertIntoAppointment);
device.resetCache();
}
} catch (Exception e) {
e.printStackTrace();
}
private int getBestBandwidth(Network available[], Network supported[]) {
    int max = 0;
    for (int a = 0; a < available.length; a++) {
        for (int s = 0; s < supported.length; s++) {
            if (available[a].getId() == supported[s].getId()) {
                if (available[a].getCurrentBandwidth() > max) {
                    max = available[a].getCurrentBandwidth();
                    this.usedNetwork = available[a].getId();
                }
            }
        }
    }
    return (int) max / 8; // bit to byte conversion
}

private String[] calculateAndStoreProductData(Device device, int product_id, int appointmentDuration, int bestBandwidth, int id, int employee_id) {
    String dataAndTime[] = new String[2];
    dataAndTime[0] = "-1":// data size
    dataAndTime[1] = "-1.0":// time
int dataSize = 0;
double time = 0.0;
ProductData pData[] = MainRunTime.getProduct(product_id).getUnsortedData();
boolean inCache[] = new boolean[pData.length];
for(int i = 0;i<inCache.length;i++)
inCache[i] = false;
if(bestBandwidth > 0){
    for(int i = 0;i<pData.length;i++){
        time = time + ((double)pData[i].getSize()/(double)bestBandwidth);
        if(time <= appointmentDuration){
            boolean stored = device.addToCache( pData[i].getMimeId(), pData[i].getSize(), product_id);
            if(stored){
                dataSize += pData[i].getSize();
                inCache[i] = true;
            }
        }else
            time = time - ((double)pData[i].getSize())/(double)bestBandwidth;
    } //for more data
    NumberFormat formatter = new DecimalFormat("0.00");
dataAndTime[1] = formatter.format(time);
dataAndTime[0] = dataSize+"";
}
try{
String insertTotalDataDownloadedIntoAppointment = "UPDATE appointment SET appointment.dataDownloadedWithoutProfile=+ dataSize +" where id= + id + ";
    int t = MainRunTime.getStatement().executeUpdate(insertTotalDataDownloadedIntoAppointment);
} catch (Exception e) {
    e.printStackTrace();
}
else
    dataAndTime[1] = "-100";

int utility = 0;
String reason;
if(dataAndTime[1].equalsIgnoreCase("-100"))
    reason = "No Compatible Network";
else
    reason = "Shortage of Time OR lack of available cache";

try{
    boolean deviceSupport[] = new boolean[pData.length];
    MainRunTime.filterDataForDevice(pData,device.getId(),deviceSupport);
    int weights[][] = MainRunTime.getEmployee(employee_id).getWeights();
    String selectProductTypeId = "Select product_id from product_profile where id=+product_id+";
    ResultSet rs = MainRunTime.getStatement().executeQuery(selectProductTypeId);
rs.next();

int productTypeId = rs.getInt(1);
rs.close();
for(int i = 0;i<inCache.length;i++){
    if(inCache[i] == false){
        String insertIntoMissedDataWithProfile = "Insert into misseddatawithoutprofile (appid, mimeid, productid,
        deviceid, size, reason) VALUES (" + id + "," + pData[i].getMimeId()+ "," + product_id + "," + device.getId() + "," + pData[i].getSize() + "," + reason + ");"
        int t = MainRunTime.getStatement().executeUpdate(insertIntoMissedDataWithProfile);
    }

    if(weights[x][0] == productTypeid){
        if(deviceSupport[i] == true){
            utility = pData[i].getMimePriority()*weights[x][1];
        }else{
            utility = 0;
        }
    }
}
}
String inertIntoUtilityPerProductDataDownloadedWithoutProfile = "Insert INTO UtilityPerProductDataDownloadedWithoutProfile (appointment_id, utility) VALUES (" + id + "," + utility + ");";
int t = MainRunTime.getStatement().executeUpdate(inertIntoUtilityPerProductDataDownloadedWithoutProfile);
} //ADDED TO CACHE END IF
} //END FOR
}
catch (Exception e){
    e.printStackTrace();
    return dataAndTime;
}
Appendix C: Database Scripts

Runtime Layer Database

CREATE DATABASE `application_runtime_data`;
USE `application_runtime_data`;

CREATE TABLE `appointment`
(`id` bigint(20) NOT NULL auto_increment,
`startTime` varchar(255) NOT NULL default ",
`endTime` varchar(255) NOT NULL default ",
`product_id` bigint(20) NOT NULL default '0',
`location_id` bigint(20) NOT NULL default '0',
`customer_id` bigint(20) NOT NULL default '0',
`day` bigint(20) NOT NULL default '0',
`appointmentDuration` varchar(255) NOT NULL default ",
`travelTime` varchar(255) NOT NULL default ",
`employee_id` bigint(20) NOT NULL default '0',
`device_id` bigint(20) NOT NULL default '0',
`utilityOfCachedDataWithoutProfile` bigint(20) NOT NULL default '0',
`timeToDownloadWithoutProfile` double(20,2) NOT NULL default '0.00',
`utilityOfCachedDataWithProfile` bigint(20) NOT NULL default '0',
`timeToDownloadWithProfile` double(20,2) NOT NULL default '0.00',
`dataDownloadedWithoutProfile` bigint(20) NOT NULL default '0',
`dataDownloadedWithProfile` bigint(20) NOT NULL default '0',
PRIMARY KEY (`id`)
) TYPE=MyISAM;
CREATE TABLE `cache` (  
`id` bigint(20) NOT NULL auto_increment,  
`cache` longtext NOT NULL,  
`employee_id` bigint(20) NOT NULL default '0',  
`time` double(20,1) NOT NULL default '0.0',  
`predictedlocation_id` bigint(20) NOT NULL default '0',  
`bandwidth` bigint(20) NOT NULL default '0',  
`newDataSize` bigint(20) NOT NULL default '0',  
`availableCache` bigint(20) NOT NULL default '0',  
`appId` bigint(20) NOT NULL default '0',  
`cacheSize` bigint(20) NOT NULL default '0',  
PRIMARY KEY (`id`)  
) TYPE=MyISAM;

CREATE TABLE `customer` (  
`id` int(11) NOT NULL auto_increment,  
`fName` varchar(50) default NULL,  
`lName` varchar(50) default NULL,  
PRIMARY KEY (`id`),  
KEY `Customerid` (`id`)  
) TYPE=MyISAM;

CREATE TABLE `customer_profile` (  
`id` int(11) NOT NULL auto_increment,  
`customer_id` int(11) NOT NULL default '0',  
`product_id` int(11) NOT NULL default '0',  
`location_id` int(11) NOT NULL default '0',  
PRIMARY KEY (`id`, `customer_id`, `product_id`, `location_id`),  
KEY `CustomerCustomer_Profile` (`customer_id`),  
KEY `LocationCustomer_Profile` (`location_id`),  
KEY `prod_id` (`customer_id`),  
KEY `ProductCustomer_Profile` (`product_id`)
)
CREATE TABLE `device` (  `id` int(11) NOT NULL auto_increment,  `cacheCapacity` double default '0',  `title` varchar(50) default NULL,  PRIMARY KEY (`id`)
) TYPE=MyISAM;

CREATE TABLE `device_supports_mimetype` (  `device_id` int(11) NOT NULL default '0',  `mimeType_id` int(11) NOT NULL default '0',  PRIMARY KEY (`device_id`,`mimeType_id`),  KEY `device_id` (`device_id`),  KEY `DeviceDevice_MimeType` (`device_id`),  KEY `mimeType_id` (`mimeType_id`),  KEY `MimeTypeDevice_MimeType` (`mimeType_id`)  ) TYPE=MyISAM;

CREATE TABLE `device_supports_networktype` (  `device_id` int(11) NOT NULL default '0',  `networkType_id` int(11) NOT NULL default '0',  PRIMARY KEY (`device_id`,`networkType_id`),  KEY `Device_NetworkTypeDeviceHASNetworkType` (`networkType_id`),  KEY `DeviceDeviceHASNetworkType` (`device_id`)  ) TYPE=MyISAM;

CREATE TABLE `employee` (  `id` int(11) NOT NULL auto_increment,  `fName` varchar(50) default NULL,
`lName` varchar(50) default NULL,
`startLocation` int(11) NOT NULL default '0',
`availableStartTime` varchar(50) default NULL,
`availableEndTime` varchar(50) default NULL,
PRIMARY KEY (`id`)  
) TYPE=MyISAM;

CREATE TABLE `employee_has_device` (  
`employee_id` int(11) NOT NULL default '0',  
`device_id` int(11) NOT NULL default '0',  
PRIMARY KEY (`employee_id`,`device_id`),  
KEY `device_id` (`device_id`),  
KEY `DeviceEmployee_Has_Device` (`device_id`),  
KEY `employee_id` (`employee_id`),  
KEY `EmployeeEmployee_Has_Device` (`employee_id`)  
) TYPE=MyISAM;

CREATE TABLE `employee_predictedlocations` (  
`id` bigint(20) NOT NULL auto_increment,  
`appointment_id` bigint(20) NOT NULL default '0',  
`employee_id` bigint(20) NOT NULL default '0',  
`product_id` bigint(20) NOT NULL default '0',  
`day` bigint(20) NOT NULL default '0',  
`location_id` bigint(20) NOT NULL default '0',  
`timeSpent` bigint(20) NOT NULL default '0',  
`cache_id` bigint(20) NOT NULL default '0',  
`device_id` bigint(20) NOT NULL default '0',  
PRIMARY KEY (`id`)  
) TYPE=MyISAM;

CREATE TABLE `employee_profile` (  
...
`employee_id` int(11) NOT NULL default '0',
`product_id` int(11) NOT NULL default '0',
`location_id` int(11) NOT NULL default '0',
PRIMARY KEY (`employee_id`,`product_id`,`location_id`),
KEY `EmployeeEmployee_Profile` (`employee_id`),
KEY `LocationEmployee_Profile` (`location_id`),
KEY `prod_id` (`employee_id`),
KEY `ProductEmployee_Profile` (`product_id`)  
) TYPE=MyISAM;

CREATE TABLE `location` (  
`id` int(11) NOT NULL auto_increment,
`title` varchar(50) default NULL,
`postCode` varchar(6) default NULL,
PRIMARY KEY (`id`),
KEY `postcode` (`postCode`)  
) TYPE=MyISAM;

CREATE TABLE `location_available_network` (  
`location_id` int(11) NOT NULL default '0',
`network_id` int(11) NOT NULL default '0',
PRIMARY KEY (`location_id`,`network_id`),
KEY `location_id` (`location_id`),
KEY `LocationLocation_Available_Network` (`location_id`),
KEY `network_id` (`network_id`),
KEY `NetworkTypeLocation_Available_Network` (`network_id`)  
) TYPE=MyISAM;

CREATE TABLE `mimetype` (  
`id` int(11) NOT NULL auto_increment,
`extension` varchar(255) default NULL,
CREATE TABLE `misseddatawithoutprofile` (  `id` int(11) NOT NULL auto_increment,  `appoint_id` int(11) default '0',  `mime_id` int(11) default '0',  `product_id` int(11) default '0',  `device_id` int(11) default '0',  `size` int(11) default '0',  `reason` varchar(255) default NULL,  PRIMARY KEY (`id`),  KEY `appoint_id` (`appoint_id`),  KEY `device_id` (`device_id`),  KEY `id` (`id`),  KEY `product_id` (`product_id`)) TYPE=MyISAM;

CREATE TABLE `misseddatawithprofile` (  `id` int(11) NOT NULL auto_increment,  `appoint_id` int(11) default '0',  `mime_id` int(11) default '0',  `product_id` int(11) default '0',  `device_id` int(11) default '0',  `size` int(11) default '0',  `predictedLocation_id` int(11) default NULL,  PRIMARY KEY (`id`),  KEY `appoint_id` (`appoint_id`),  KEY `device_id` (`device_id`),  KEY `id` (`id`),  KEY `product_id` (`product_id`)) TYPE=MyISAM;
CREATE TABLE `networktype` (
`id` int(11) NOT NULL auto_increment,
`maxBandwidth` double default '0',
`title` varchar(50) default NULL,
PRIMARY KEY (`id`)
) TYPE=MyISAM;

CREATE TABLE `priority` (
`id` smallint(6) NOT NULL default '0',
`title` varchar(5) default NULL,
PRIMARY KEY (`id`)
) TYPE=MyISAM;

CREATE TABLE `product` (
`product_id` int(11) NOT NULL auto_increment,
`prod_desc` varchar(255) default NULL,
`type` varchar(50) default NULL,
PRIMARY KEY (`product_id`)
) TYPE=MyISAM;

CREATE TABLE `product_profile` (
`id` int(11) NOT NULL auto_increment,
`product_id` int(11) NOT NULL default '0',
`location_id` int(11) NOT NULL default '0',
PRIMARY KEY (`id`),
CREATE TABLE `productdata`
(`id` bigint(20) NOT NULL auto_increment,
`p_id` int(11) NOT NULL default '0',
`mimeId` int(11) NOT NULL default '0',
`size` int(11) default '0',
PRIMARY KEY (`id`),
KEY `p_id` (`p_id`),
KEY `MimeTypeProductData` (`mimeId`),
KEY `ProductProductData` (`p_id`)
) TYPE=MyISAM;

CREATE TABLE `route`
(`appointment_id` int(11) default '0',
`employee_id` int(11) default '0',
`day` int(11) default '0',
`location_id` int(11) default '0',
`timeSpent` varchar(50) default NULL,
`sequence` int(11) NOT NULL auto_increment,
PRIMARY KEY (`sequence`),
KEY `location_id` (`location_id`),
KEY `EmployeeRoute` (`employee_id`),
KEY `Routeday` (`day`),
KEY `AppointmentRoute` (`appointment_id`),
KEY `LocationRoute` (`location_id`),
KEY `employeeId` (`employee_id`)
) TYPE=MyISAM;
CREATE TABLE `utilityperproductdatadownloadedwithoutprofile` (  `id` int(11) NOT NULL auto_increment,  `appointment_id` int(11) default '0',  `utility` int(11) default '0',  PRIMARY KEY  (`id`),  KEY `appointment_id` (`appointment_id`),  KEY `id` (`id`) ) TYPE=MyISAM;

CREATE TABLE `utilityperproductdatadownloadedwithprofile` (  `id` int(11) NOT NULL auto_increment,  `appointment_id` int(11) default '0',  `utility` int(11) default '0',  PRIMARY KEY  (`id`),  KEY `appointment_id` (`appointment_id`),  KEY `id` (`id`) ) TYPE=MyISAM;
CREATE DATABASE `application_static_data`;
USE `application_static_data`;

CREATE TABLE `device` (  `id` bigint(20) NOT NULL default '0',  `cacheCapacity` bigint(20) NOT NULL default '0',  `title` varchar(255) NOT NULL default '' ) TYPE=MyISAM;

CREATE TABLE `device_supports_mimetype` (  `device_id` bigint(20) NOT NULL default '0',  `mimeType_id` bigint(20) NOT NULL default '0' ) TYPE=MyISAM;

CREATE TABLE `device_supports_networktype` (  `device_id` bigint(20) NOT NULL default '0',  `networkType_id` bigint(20) NOT NULL default '0' ) TYPE=MyISAM;

CREATE TABLE `employee` (  `id` int(11) NOT NULL auto_increment,  `fName` varchar(50) default NULL,  `lName` varchar(50) default NULL,  PRIMARY KEY (`id`),  KEY `id2` (`id`) )
CREATE TABLE `employee_profile` (  `employee_id` int(11) NOT NULL default '0',  `product_id` int(11) NOT NULL default '0',  `location_id` int(11) NOT NULL default '0',  PRIMARY KEY (`employee_id`, `product_id`, `location_id`),  KEY `EmployeeEmployee_Profile` (`employee_id`),  KEY `LocationEmployee_Profile` (`location_id`),  KEY `prod_id` (`employee_id`),  KEY `ProductEmployee_Profile` (`product_id`) ) TYPE=MyISAM;

CREATE TABLE `location` (  `id` int(11) NOT NULL auto_increment,  `postCode` smallint(6) default '0',  `title` varchar(50) default NULL,  `x1` int(11) default '-1',  `x2` int(11) default '-1',  `y1` int(11) default '-1',  `y2` int(11) default '-1',  `a` smallint(6) default '-1',  `b` smallint(6) default '-1',  `c` smallint(6) default '-1',  `d` smallint(6) default '-1',  PRIMARY KEY (`id`),  KEY `id` (`id`),  KEY `PostCode` (`postCode`) ) TYPE=MyISAM;

CREATE TABLE `mimetype` (  

252
`id` int(11) NOT NULL auto_increment,
`extension` varchar(255) default NULL,
`title` varchar(255) default NULL,
`priority_id` smallint(6) default '0',
PRIMARY KEY (`id`),
KEY `id` (`id`),
KEY `PriorityMimeType` (`priority_id`)
) TYPE=MyISAM;

CREATE TABLE `networktype` (  
`id` bigint(20) NOT NULL default '0',
`maxBandwidth` double NOT NULL default '0',
`title` varchar(255) NOT NULL default '',
`minimumBandwidth` double NOT NULL default '0'
) TYPE=MyISAM;

CREATE TABLE `priority` (  
`id` smallint(6) NOT NULL default '0',
`title` varchar(5) default NULL,
PRIMARY KEY (`id`)
) TYPE=MyISAM;

CREATE TABLE `product` (  
`product_id` int(11) NOT NULL auto_increment,
`prod_desc` varchar(255) default NULL,
`type` varchar(50) default NULL,
PRIMARY KEY (`product_id`)
) TYPE=MyISAM;
Appendix D: Installation And Execution

---

**Installation & Initial Execution Steps:**

---

1. Double Click on USyd.msi file to Install

2. Select the Installation folder

**NOTE:**

On some computers the application fails to execute if the installation path has folder with a space in name for example C:\Program Files. Therefore it is recommended to install the Simulator at C:\Simulator.

3. Upon Installation Two Short Cuts will be added to the user desktop:
   a. RunSimulator
   b. RunSimulator.bat

   In most cases the simulator will execute if the user clicks on RunSimulator

4. A new folder calls “Omer's Simulator” is also created in Start Menu Program containing 3 shortcuts:
   a. RunSimulator
   b. RunSimulator.bat
   c. Start.bat (contains the list and series of commands which can be used to execute the Simulator from command prompt)
Hardware and Software Requirements:

1. Java Runtime Environment 1.5+
2. PORT 3306 Must be Free
3. There should not any other instance of MySQL running on the machine
4. Windows 2000 or Windows XP Professional
5. 99 Mega Byte of Free Space

Running the Simulator:

When the user double clicks on RunSimulator the following Window is displayed:

1. Start MySql Database Server
2. Stop MySql Database Server
3. Generate Runtime Information – Recreates all the runtime information and new tables. All previously generated data is deleted
4. Start MySql Query, so that the user can see the data in the database
5. Execute Runtime Environment – Both Conventional and Proposed System Simulations are executed and database is populated with newly generated data
6. Check Whether the MySql Server is Running or Not
7. Results Summary – Displays the Summary of Results
8. Exit the Simulator – MySql Database will automatically shutdown upon exit

**Start MySql Database Server** – Start MySql Server by using PORT 3306 and login the user as root. When MySql Server starts the main window is changed to:

![Simulator Interface](image)

**Stop MySql Database Server** – Starts MySql Admin as a separate process and sends shutdown command. This process may take few seconds and the MySql Status may not change to Shutdown therefore the user may have to click Update MySql Status Button again

**Generate Runtime Information** – The following dialog box is displayed:

![Generating Runtime Information](image)
Once all the data is generated the dialog box and the message changes as shown below:

### Execute Runtime Environment
Both Conventional and Proposed System Simulations are executed and database is populated with newly generated data. The user can now see the summary of results by clicking on Results Summary Button.

### Start MySql Query
Start MySql Query. A Dialog box is displayed outlining all the information user has to enter in order to use MySql Query
Results Summary – Following dialog box is displayed with the final results:
The user can re-execute the runtime environments and get new results while using previously generated runtime information. However if the user clicks Results Summary button before executing runtime environment then following alert box is displayed:

Installation Folder Structure:

Root installation folder:

2. MySql
   Contains complete mysql database server
3. MyQuery
MySql Query Browser 1.1.5

4. **Src**
   
   Contains Java Source Code for the following Java Packages
   
   a. RunTimeDb
   b. RunTimeEnv

   The installation folder will also have following files:

   **JAR Files:**

   a. **RunSimulator.jar** – The main jar file which displays the main window by using which the user can start/stop MySql Server, MySql Query and run different sections of the simulator
   b. **xp.jar** – A Free third party Jar files which makes Java GUI look and feel as Windows XP OS
   c. **USyd.jar** – This Jar file contains all the files related to Simulator backend processing, the execution is controlled by RunSimulator.jar but the business logic is implemented in USyd.jar
   d. **driver.jar** – “Mysql Connector Java Version 3.0.15” JDBC Type 4 driver

   **Batch Files:**

   a. **start.bat** – Batch file containing all the batch command used to start MySql and execute java packages
   b. **RunSimulator.bat** – Sets the class path and executes RunSimulator.jar file. This batch file can be used if Jar files are not associated with JRE
Config.ini File:

config.ini – This file is not part of Installation package, but it is created when the user runs the simulator first time. Upon first execution the user is first asked to select the installation folder of the Simulator. When the user selects the installation folder, then an empty file “config.ini” file is created and following information is added:

a. Installation Folder Path:
   C:\Program Files\Simulator

b. MySql Server Bin Folder Path:
   C:\Program Files\Simulator\mysql\bin

c. MySql Query Root Folder Path:
   C:\Program Files\Simulator\MyQuery

d. C:\Program Files\Simulator\driver.jar
   MySql JDBC Driver JAR File Path

Uninstalling Simulator:

In order to uninstall Simulator, go to Control Panel, Add Remove Programs, select Simulator and click on Remove. Upon removal all the files and shortcuts will be removed however config.ini file in installation folder will not be deleted. The installation folder and config.ini file can be deleted safely once the uninstalltion wizard ends.
System Admin and Installer Source Code:

MainRunner Class

```
import java.io.*;

public class MainRunner{
    public static void main(String args[]){
        new MainRunner();
    }

    public MainRunner(){
        try{
            File f = new File("temp");
            f.createNewFile();

            Runtime rt = Runtime.getRuntime();

            String path = ".\" + f.getAbsolutePath();
            path = path.substring(0,path.length()-5);
            path += ".\driver.jar;"+(path+".\USyd.jar;")+(path+".\RunSimulator.jar;")+(path+".\xp.jar;");
        }
    }
}
```
System.out.println("CLASSPATH="+path);

/*
* System.out.println(files);
for(int i = 0;i<files.length;i++){
    String temp = files[i].getName();
    if(temp.endsWith(".jar")){
        path += temp;
    }
}
*/

Process proc = rt.exec("java -cp " + path + " Runner");
f.delete();
StreamGobbler errorGobbler = new StreamGobbler(proc.getErrorStream(), "->");
StreamGobbler outputGobbler = new StreamGobbler(proc.getInputStream(), "OUTPUT");

errorGobbler.start();
outputGobbler.start();

int exitVal = proc.waitFor();
System.out.println("ExitValue: " + exitVal);
} catch(Exception e) {
    e.printStackTrace();
    e.printStackTrace();
}

class StreamGobbler extends Thread {
    InputStream is;
    String type;

    StreamGobbler(InputStream is, String type) {
        this.is = is;
        this.type = type;
    }

    public void run() {
        try {
            InputStreamReader isr = new InputStreamReader(is);
            InputStreamReader isr = new InputStreamReader(is);
        }
    }
}
BufferedReader br = new BufferedReader(isr);
String line=null;
while ( (line = br.readLine()) != null)
    System.out.println(type + " > " + line);
} catch (IOException ioe) {
    ioe.printStackTrace();
}

Runner Class

/*
 * Runner.java
 *
 * Created on 18 May 2005, 14:12
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.io.*;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.lang.reflect.Method;
import com.stefankrause.xplookandfeel.XPLookAndFeel;
import javax.swing.UIManager;
import javax.swing.*;
import java.awt.*;
/**  
 *  
 * @author omahmood
 */
public class Runner extends javax.swing.JFrame {
    private final String DB_URL_RUNTIME = "jdbc:mysql:///application_runtime_data?user=root";
    private Connection connRun = null;
    private Process mySqlProcess = null;
    private File mainFolder = null;
    private File mySqlBinFolder = null;
    private File mySqlQueryExe = null;
    private File mySqlDriver = null;

    private File mainFolder = null;
    private File mySqlBinFolder = null;
    private File mySqlQueryExe = null;
    private File mySqlDriver = null;
/** Creates new form Runner */

public Runner() {
    setLook();
    initComponents();
    setIcon();
    checkInstallation();
    writeInfo();
    checkForDb();
}

private void setIcon(){
    Image img = Toolkit.getDefaultToolkit().getImage(java.net.URLClassLoader.getSystemResource("icon.jpg"));
    this.setIconImage(img);
}

private void setLook(){
    try{
        System.setProperty("sun.java2d.ddscale", "true");
        UIManager.setLookAndFeel("com.stefankrause.xplookandfeel.XPLookAndFeel");
        UIManager.addPropertyChangeListener(new PropertyChangeListener() {
            public void propertyChange(PropertyChangeEvent event) {
            }
        });
    }
    UIManager.addPropertyChangeListener(
        new PropertyChangeListener() {
            public void propertyChange(PropertyChangeEvent event) {
            }
        });
    }
}
Object oldLF = event.getOldValue();
Object newLF = event.getNewValue();

if ((newLF instanceof XPLookAndFeel) == false) {
    try {
        UIManager.setLookAndFeel(new XPLookAndFeel());
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void setSystem(){
    try{
        System.setProperty("CLASSPATH",mySqlDriver.getPath()+";");
        System.out.println("  *****  --> " + System.getProperty("CLASSPATH"));
        }catch (Exception e){
            e.printStackTrace();
        }
}
try {
    System.setProperty("java.library.path", mySqlDriver.getPath()+");
    System.out.println("***** --> " + System.getProperty("CLASSPATH");
} catch (Throwable t) {
    t.printStackTrace();
}
/*
 */
try{

    //Runtime rt = Runtime.getRuntime();

    //Process proc = rt.exec("java -cp " + mySqlDriver.getPath()+";" + " Abc");

    StreamGobbler errorGobbler = new StreamGobbler(proc.getErrorStream(), "ERROR");

    //// any output?
    StreamGobbler outputGobbler = new StreamGobbler(proc.getInputStream(), "OUTPUT");
        System.out.println("**********************");
/// kick them off
errorGobbler.start();
outputGobbler.start();

// any error???
int exitVal = proc.waitFor();
System.out.println("ExitValue: " + exitVal);
}
catch( Exception e ){
    e.printStackTrace();
}
*/
private void checkInstallation(){
    try {  
        //String osName = System.getProperty( "os.name" );
        String[] cmd = new String[3];
        cmd[0] = "cmd.exe";
        cmd[1] = "/C";
        cmd[2] = "cd";
        Runtime rt = Runtime.getRuntime();
        Process proc = rt.exec(cmd);
        BufferedReader read = new BufferedReader(new InputStreamReader(proc.getInputStream()));
mainFolder = new File(read.readLine());
int exitVal = proc.waitFor();
} catch (Throwable t) {
    t.printStackTrace();
}

private void writeInfo(){
    try{
        File info = new File(mainFolder.getPath() + "\config.ini");
        if(info.exists()){
            DataInputStream in = new DataInputStream(new FileInputStream(info));
            mainFolder = new File(in.readUTF());
            mySqlBinFolder = new File(in.readUTF());
            mySqlQueryExe = new File(in.readUTF());
            mySqlDriver = new File(in.readUTF());
            in.close();
        }else{
            selectInstallationFolder();
            DataOutputStream out = new DataOutputStream(new FileOutputStream(info));
            info.createNewFile();
            out.writeUTF(mainFolder.getPath());
            out.writeUTF(mySqlBinFolder.getPath());
        }
    } catch (IOException e) {
    e.printStackTrace();
    }
}
out.writeUTF(mySqlQueryExe.getPath());
out.writeUTF(mySqlDriver.getPath());
out.flush();
out.close();

private void selectInstallationFolder(){
    if(this.jFileChooser1.showOpenDialog(this) == this.jFileChooser1.APPROVE_OPTION){
        mainFolder = this.jFileChooser1.getSelectedFile();
        if(!mainFolder.isDirectory()){ selectInstallationFolder();
    }else{
        File files[] = mainFolder.listFiles();
        for(int i = 0;i<files.length;i++){
            if(files[i].getName().equalsIgnoreCase("MyQuery")){
                mySqlQueryExe = new File(files[i].getPath());// +"\g.exe ");
            }else if(files[i].getName().equalsIgnoreCase("mysql")){
                mySqlBinFolder = new File(files[i].getPath() +"\bin ");
            }else if(files[i].getName().equalsIgnoreCase("driver.jar")){

mySqlDriver = files[i];
}

return;

} else{
    selectInstallationFile();
}

private void selectInstallationFile(){
    if(this.jFileChooser1.showOpenDialog(this) == this.jFileChooser1.APPROVE_OPTION){
        jFileChooser1.setDialogTitle("Select Installation FILE");
        jFileChooser1.setFileSelectionMode(java.awt.FileDialog.FILES_ONLY);
        File info = jFileChooser1.getSelectedFile();
        if(info.exists()){
            try{
                DataInputStream in = new DataInputStream(new FileInputStream(info));
                mainFolder = new File(in.readUTF());
                mySqlBinFolder = new File(in.readUTF());
                mySqlQueryExe = new File(in.readUTF());
                mySqlDriver = new File(in.readUTF());
            }catch (Exception e){
            }
        }
    }
}
e.printStackTrace();
}
}
else
    System.exit(1);
}
private void checkForDb(){
    boolean status = false;
    try{
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        connRun = DriverManager.getConnection(DB_URL_RUNTIME);
        jLabel1.setForeground(new java.awt.Color(0, 102, 102));
        jLabel1.setText("RUNNING");
        startMysql.setEnabled(false);
        stopMySql.setEnabled(true);
        genRuntimeInfoBtn.setEnabled(true);
        runtimeEnvBtn.setEnabled(true);
        resultBtn.setEnabled(true);
    }catch(Exception e){
        jLabel1.setForeground(new java.awt.Color(204, 0, 51));
        jLabel1.setText("STOPPED");
        startMysql.setEnabled(true);
    }
stopMySql.setEnabled(false);
genRuntimeInfoBtn.setEnabled(false);
runtimeEnvBtn.setEnabled(false);
resultBtn.setEnabled(false);
//System.out.println(e.getMessage());
}
}
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
private void initComponents() {
    jFileChooser1 = new javax.swing.JFileChooser();
jDialog1 = new javax.swing.JDialog();
jLabel3 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
    hideDialogBox = new javax.swing.JButton();
    mySqlQueryHelp = new javax.swing.JDialog();
    jTable1 = new javax.swing.JTable();
    hideHelpBtn = new javax.swing.JButton();
}
resultsDialog = new javax.swing.JDialog();
closeResults = new javax.swing.JButton();
jLabel4 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jTable2 = new javax.swing.JTable();
jOptionPane1 = new javax.swing.JOptionPane();
startMysql = new javax.swing.JButton();
stopMySql = new javax.swing.JButton();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
checkMySqlStatus = new javax.swing.JButton();
exitBtn = new javax.swing.JButton();
genRuntimeInfoBtn = new javax.swing.JButton();
runtimeEnvBtn = new javax.swing.JButton();
mysqlQueryBtn = new javax.swing.JButton();
resultBtn = new javax.swing.JButton();

jFileChooser1.setCurrentDirectory(new java.io.File("c:\");
jFileChooser1.setDialogTitle("Select Installation Folder");
jFileChooser1.setFileSelectionMode(java.swing.JFileChooser.DIRECTORIES_ONLY);
jDialog1.getContentPane().setLayout(new AbsoluteLayout());

jDialog1.setTitle("Please Wait");
jDialog1.setAlwaysOnTop(true);
jLabel3.setFont(new java.awt.Font("Microsoft Sans Serif", 1, 11));
jLabel3.setText("Generating Runtime Information");
jDialog1.getContentPane().add(jLabel3, new AbsoluteConstraints(90, 0, 210, 40));

jScrollPane1.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
jScrollPane1.setViewportView(jTextArea1);
jDialog1.getContentPane().add(jScrollPane1, new AbsoluteConstraints(2, 52, 390, 160));

hideDialogBox.setText("Close");
hideDialogBox.setEnabled(false);
hideDialogBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        hideDialogBoxActionPerformed(evt);
    }
});

jDialog1.getContentPane().add(hideDialogBox, new AbsoluteConstraints(0, 220, 390, -1));
mySqlQueryHelp.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
mySqlQueryHelp.setTitle("Enter Following Information");
mySqlQueryHelp.setSize(300,200);
 jTable1.setRowHeight(28);
 jTable1.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
   {"Server Host", "localhost"},  
   {"Port", "3306"},  
   {"Username", "root"},  
   {"Password", "LEAVE BLANK"},  
   {"Default Schema", "LEAVE BLANK"}  
  },  
  new String [] {  
   "Title 1", "Title 2"  
  }  
));
 mySqlQueryHelp.getContentPane().add(jTable1, java.awt.BorderLayout.CENTER);

hideHelpBtn.setText("Close");
hideHelpBtn.addActionListener(new java.awt.event.ActionListener() {  
   public void actionPerformed(java.awt.event.ActionEvent evt) {

278
hideHelpBtnActionPerformed(evt);

mySqlQueryHelp getContentPane().add(hideHelpBtn, java.awt.BorderLayout.NORTH);

resultsDialog.getContentPane().setLayout(new AbsoluteLayout());

resultsDialog.setSize(630,370);
closeResults.setText("Close");
closeResults.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        closeResultsActionPerformed(evt);
    }
});

resultsDialog.getContentPane().add(closeResults, new AbsoluteConstraints(0, 300, 618, -1));

resultsDialog.getContentPane().add(closeResults, new AbsoluteConstraints(0, 300, 618, -1));

jLabel4.setText("Gain in Utility Per Byte Accessed:");
jLabel4.setText("Gain in Utility Per Byte Accessed:");
resultsDialog.getContentPane().add(jLabel4, new AbsoluteConstraints(10, 200, 260, 30));
resultsDialog.getContentPane().add(jLabel4, new AbsoluteConstraints(10, 200, 260, 30));
resultsDialog.getContentPane().add(jLabel6, new AbsoluteConstraints(270, 200, 130, 30));
```
jLabel5.setText("Gain in Utility Per Seconds Connection Time:");
resultsDialog.getContentPane().add(jLabel5, new AbsoluteConstraints(10, 240, 260, 30));

resultsDialog.getContentPane().add(jLabel7, new AbsoluteConstraints(270, 240, 130, 30));

jTable2.setModel(new javax.swing.table.DefaultTableModel(
                      new Object[][] {
                          {"", "Without Profile", "With Profile"},
                          {"Total Utility Of Cached Data", null, null},
                          {"Total Time To Download", null, null},
                          {"Total Data Downloaded", null, null},
                          {"Utility Per Second", null, null},
                          {"Utility Per Byte", null, null}
                      },
                      new String [] {
                          "Title 1", "Title 2", "Title 3"
                      }
                  ));
jTable2.setRowHeight(28);
resultsDialog.getContentPane().add(jTable2, new AbsoluteConstraints(10, 10, 600, 160));
```
getContentPane().setLayout(new AbsoluteLayout());

setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
setTitle("Simulator");
startMysql.setText("Start MySQL Server");
startMysql.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        startMysqlActionPerformed(evt);
    }
});
getContentPane().add(startMysql, new AbsoluteConstraints(40, 30, -1, -1));

stopMySql.setText("Stop MySQL Server");
stopMySql.setEnabled(false);
stopMySql.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        stopMySqlActionPerformed(evt);
    }
});
getContentPane().add(stopMySql, new AbsoluteConstraints(40, 60, -1, -1));
jLabel1.setBackground(new java.awt.Color(255, 255, 255));
jLabel1.setFont(new java.awt.Font("Microsoft Sans Serif", 1, 10));
jLabel1.setForeground(new java.awt.Color(255, 255, 255));
jLabel1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 51, 51), 2));
jLabel1.setFocusable(false);
getContentPane().add(jLabel1, new AbsoluteConstraints(190, 30, 110, 50));

jLabel2.setFont(new java.awt.Font("Microsoft Sans Serif", 1, 11));
jLabel2.setText("MySql Status: ");
jLabel2.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 51, 51), 2));
getContentPane().add(jLabel2, new AbsoluteConstraints(190, 10, 90, 20));

checkMySqlStatus.setText("Update MySql Status ");
checkMySqlStatus.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        checkMySqlStatusActionPerformed(evt);
    }
});

getContentPane().add(checkMySqlStatus, new AbsoluteConstraints(310, 30, 150, 50));
exitBtn.setText("Exit");
exitBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        exitBtnActionPerformed(evt);
    }
});
getContentPane().add(exitBtn, new AbsoluteConstraints(509, 173, 70, 30));

genRuntimeInfoBtn.setText("Generate Runtime Information");
genRuntimeInfoBtn.setEnabled(false);
genRuntimeInfoBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        genRuntimeInfoBtnActionPerformed(evt);
    }
});
getContentPane().add(genRuntimeInfoBtn, new AbsoluteConstraints(40, 100, 210, 60));

runtimeEnvBtn.setText("Execute Runtime Environment");
runtimeEnvBtn.setEnabled(false);
runtimeEnvBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        runtimeEnvBtnActionPerformed(evt);
    }
});
getContentPane().add(runtimeEnvBtn, new AbsoluteConstraints(40, 100, 210, 60));
public void actionPerformed(java.awt.event.ActionEvent evt) {
    runtimeEnvBtnActionPerformed(evt);
}
}

getContentPane().add(runtimeEnvBtn, new AbsoluteConstraints(270, 100, 190, 60));

mysqlQueryBtn.setText("Start MySql Query");
mysqlQueryBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mysqlQueryBtnActionPerformed(evt);
    }
});

g getContentPane().add(mysqlQueryBtn, new AbsoluteConstraints(40, 170, 420, -1));

resultBtn.setText("Results Summary");
resultBtn.setEnabled(false);
resultBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        resultBtnActionPerformed(evt);
    }
});

g getContentPane().add(resultBtn, new AbsoluteConstraints(40, 170, 420, -1));
private void closeResultsActionPerformed(java.awt.event.ActionEvent evt) {
    this.resultsDialog.setVisible(false);
}

private void resultBtnActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        String totalUtilityOfCachedDataWithoutProfile = "select SUM(utilityOfCachedDataWithoutProfile) as total from application_runtime_data.appointment;";

        String totalUtilityOfCachedDataWithProfile = "select SUM(utilityOfCachedDataWithProfile) as total from application_runtime_data.appointment;";

        String totalTimeToDownloadWithoutProfile = "select SUM(timeToDownloadWithoutProfile) as total from application_runtime_data.appointment where timeToDownloadWithoutProfile>0;";
    }
}
String totalTimeToDownloadWithProfile = "select SUM(timeToDownloadWithProfile) as total from application_runtime_data.appointment;";

String totalDataDownloadedWithoutProfile = "select SUM(dataDownloadedWithoutProfile) as total from application_runtime_data.appointment;";
String totalDataDownloadedWithProfile = "select SUM(dataDownloadedWithProfile) as total from application_runtime_data.appointment;";

Statement stmt = connRun.createStatement();
ResultSet rsCount = null;
rsCount = stmt.executeQuery(totalUtilityOfCachedDataWithoutProfile);
rsCount.next();
long uWithoutProfile = rsCount.getLong("total");
rsCount = stmt.executeQuery(totalUtilityOfCachedDataWithProfile);
rsCount.next();
long uWithProfile = rsCount.getLong("total");

rsCount = stmt.executeQuery(totalTimeToDownloadWithoutProfile);
rsCount.next();
long timeWithoutProfile = rsCount.getLong("total");
rsCount = stmt.executeQuery(totalTimeToDownloadWithProfile);
rsCount.next();
long timeWithProfile = rsCount.getLong("total");

rsCount = stmt.executeQuery(totalDataDownloadedWithoutProfile);
rsCount.next();
long dataWithoutProfile = rsCount.getLong("total");
rsCount = stmt.executeQuery(totalDataDownloadedWithProfile);
rsCount.next();
long dataWithProfile = rsCount.getLong("total");

this.jTable2.setValueAt(uWithoutProfile, 1,1);
this.jTable2.setValueAt(uWithProfile, 1,2);
this.jTable2.setValueAt(timeWithoutProfile, 2,1);
this.jTable2.setValueAt(timeWithProfile, 2,2);
this.jTable2.setValueAt(dataWithoutProfile, 3,1);
this.jTable2.setValueAt(dataWithProfile, 3,2);
java.text.DecimalFormat d = new java.text.DecimalFormat("0.0000");

double r1 = (double)uWithoutProfile/(double)timeWithoutProfile;
this.jTable2.setValueAt(d.format(r1), 4, 1);

double r2 = (double)uWithProfile/(double)timeWithProfile;
this.jTable2.setValueAt(d.format(r2), 4, 2);

double r3 = (double)uWithoutProfile/(double)dataWithoutProfile;
this.jTable2.setValueAt(d.format(r3), 5, 1);

double r4 = (double)uWithProfile/(double)dataWithProfile;
this.jTable2.setValueAt(d.format(r4), 5, 2);

jLabel7.setText(d.format((r2/r1)*100) + "%");
jLabel6.setText(d.format((r4/r3)*100) + "%");

if(Double.isNaN(r1) && Double.isNaN(r2) && Double.isNaN(r3) && Double.isNaN(r4)){
    JOptionPane1.showMessageDialog(this, "Please Execute Runtime Environment", "Alert", JOptionPane.ERROR_MESSAGE);
} else{
    if(!resultsDialog.isVisible())
        resultsDialog.setVisible(true);
    resultsDialog.setFocusableWindowState(true);
    resultsDialog.requestFocus();
}
private void hideHelpBtnActionPerformed(java.awt.event.ActionEvent evt) {
    this.mysqlQueryHelp.setVisible(false);
}

private void runtimeEnvBtnActionPerformed(java.awt.event.ActionEvent evt) {
    this.jDialog1.setSize(400,300);
    this.jTextArea1.setText("GENERATING RUNTIME ENVIRONMENT PLEASE WAIT....");
    this.jDialog1.setVisible(true);
    try{
        MyThreadRunner run = new MyThreadRunner(2);
    }catch (Exception e){
        this.jTextArea1.append("ERROR" + e.getMessage() + "\n");
        this.jTextArea1.append("****************************************");
        StackTraceElement el[] = e.getStackTrace();
        for(int i = 0;i<el.length;i++){
            this.jTextArea1.append(el[i].toString() + "\n");
        }
    }
}
private void hideDialogBoxActionPerformed(java.awt.event.ActionEvent evt) {
    this.jDialog1.setVisible(false);
    hideDialogBox.setEnabled(false);
}

private void mysqlQueryBtnActionPerformed(java.awt.event.ActionEvent evt) {
    this.mySqlQueryHelp.setVisible(true);
    try {
        Runtime run = Runtime.getRuntime();
        Process mySqlDownProcess = run.exec(mySqlQueryExe.getPath() + "\q.exe");
    } catch (Exception e) {
        System.out.println(e.getMessage());
        System.out.println("*******************************");
        e.printStackTrace();
        System.out.println("*******************************");
        System.out.println("*******************************");
        e.printStackTrace();
        System.out.println("*******************************");
    }
}
private void genRuntimeInfoBtnActionPerformed(java.awt.event.ActionEvent evt) {
    this.jDialog1.setSize(400, 300);
    this.jTextArea1.setText("GENERATING INFORMATION PLEASE WAIT....");
    this.jDialog1.setVisible(true);

    try{
        MyThreadRunner run = new MyThreadRunner(1);
    }catch (Exception e){
        this.jTextArea1.append("ERROR" + e.getMessage() + 
        "\n");
        this.jTextArea1.append("**********************\n");
        StackTraceElement el[] = e.getStackTrace();
        for(int i = 0;i<el.length;i++){
            this.jTextArea1.append(el[i].toString() + 
            "\n");
        }
        e.printStackTrace();
    }
}

private void exitBtnActionPerformed(java.awt.event.ActionEvent evt) {
    stopMySql();
    System.exit(0);
private void checkMySqlStatusActionPerformed(java.awt.event.ActionEvent evt) {
    checkForDb();
}

private void stopMySql(){
    if(mysqlProcess != null){
        System.out.println("Shutting Down MySQL....");
        mysqlProcess.destroy();
        checkForDb();
    }else{
        try{
            Runtime run = Runtime.getRuntime();
            Process mysqlDownProcess = run.exec(mySqlBinFolder.getPath()+"\mysqladmin -u root shutdown");
            System.out.println(mysqlDownProcess.waitFor());
            checkForDb();
        } catch (Exception e) {
            System.out.println(e.getMessage());
            System.out.println("**********************");
            e.printStackTrace();
            System.out.println("**********************");
        }
    }
}
private void stopMySqlActionPerfomed(java.awt.event.ActionEvent evt) {
    stopMySql();
}

protected void finalize(){
    stopMySql();
}

private void startMysqlActionPerfomed(java.awt.event.ActionEvent evt) {
    Runtime run = Runtime.getRuntime();
    try {
        System.out.println("My SQL Server Starting,...");
        mySqlProcess = run.exec("mySqlBinFolder.getPath()","mysqld /h");
        System.out.println("My SQL Server Started");

        checkForDb();
    }
    catch (Exception e) {
        System.out.println(e.getMessage());
        System.out.println("**************");
        e.printStackTrace();
        System.out.println("**************");
}
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Runner().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton checkMySqlStatus;
private javax.swing.JButton closeResults;
private javax.swing.JButton exitBtn;
private javax.swing.JButton genRuntimeInfoBtn;
private javax.swing.JButton hideDialogBox;
private javax.swing.JButton hideHelpBtn;
private javax.swing.JDialog jDialog1;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JButton runtimeEnvBtn;
private javax.swing.JButton startMysql;
private javax.swing.JButton stopMySql;
// End of variables declaration

class StreamGobblerInt extends Thread {
    InputStream is;
    String type;

    StreamGobblerInt(InputStream is, String type) {
        this.is = is;
        this.type = type;
    }

    public void run() {
        try {
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            String line = null;
            while ((line = br.readLine()) != null)
                System.out.println(type + " > " + line);
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
}
class MyThreadRunner extends Thread{
    private int choice;
    public MyThreadRunner(int choice){
        this.choice = choice;
        this.setPriority(9);
        this.start();
    }
    public void run(){
        if(choice == 1){
            try{
                RunTimeDb.GenerateRunTime g = new RunTimeDb.GenerateRunTime();
                JTextArea1.setText("INFORMATION GENERATED");
                System.out.println("***********************");
            }catch (Exception e){
                JTextArea1.setText("ERROR" + e.getMessage() + "\n");
                JTextArea1.append("***********************\n");
                StackTraceElement el[] = e.getStackTrace();
                for(int i = 0;i<el.length;i++){
                    JTextArea1.append(el[i].toString() + "\n");
                }
            }
        }
    }
}
    e.printStackTrace();
} finally{
    hideDialogBox.setEnabled(true);
}
}
}
}
}
}
}
Bibliography


13 Mitch Cherniack , Michael J. Franklin, Stan Zdonik, “Expressing User Profiles for Data Recharging”. IEEE Personal Communications, August 2001

14 Cherniack, M., Galvez, E.F., Franklin, M.J. and Zdonik, S., “Profile-Driven Cache Management”. In International Conference on Data Engineering (ICDE), (Bangalore, India, 2003)

16 K. Lai, Z. Tari, and P. Bertok, "Mobility aware cache replacement for location
dependent information services”. In Technical Report TR-04-04 (RMIT School of CS &
IT), 2004

17 C. Baquero and V. Fonte and F. Moura and R. Oliveira, "MobiScape: WWW
Browsing under Disconnected and Semi-Connected Operation". First Portuguese WWW
National Conference, Braga, Portugal, July 1995