MultiCraft

International Journal of Engineering, Science and Technology
Vol. 3, No. 1, 2011, pp. 191-205

INTERNATIONAL JOURNAL OF ENGINEERING, SCIENCE AND TECHNOLOGY
www.ijest-ng.com

Performance analysis of online health care system

Narendra Kohli[*] and Nishchal K. Verma

Department of Electrical Engineering, Indian Institute of Technology Kanpur, INDIA
[*]Corresponding Author: e-mail: nkohli@iitk.ac.in, Tel +91-512-2582426, Fax. +91-512-2533412

Abstract

This paper deals with selection of appropriate indexing techniques applied on MySQL database for a health care system and its related performance issues. The proposed Smart Card based Online Health Care System deals with frequent data storage, exchange and retrieval of data from the database servers. Speed and accuracy is of primary concern of the system. An approach towards the actualization of this proposal requires through analysis of performance related issues. B-Tree indexing technique is applied in most of the cases but other indexing techniques such as Hash and Bitmap are also used as per the requirement for fast retrieval of patient information from health care database. Suitable memory management and indexing techniques are analyzed and proposed for fast and accurate data retrieval from the system. Application of the proposed system has been developed as front end in .net whereas, back end as database in MySQL.

Keywords: B-Tree, Bitmap, Hash indexing, Smartcard, MySQL

1. Introduction

Online health care system delivers health care services to the society. It connects people with the hospitals, doctors and other related information of the patients. Health care system increases the accuracy and speed of patient information circulation which results in better services for patients. It also enhances the working efficiency of the hospital system (Hassol, 2004; Liu et al. 2006). The main idea behind developing such an application is to obtain, store, analyze, process and usage of patient information that concerns with the doctors, hospitals, laboratory tests etc. In this proposed system, we have generated 10 digit unique registration number /patient-ID for registration purpose in a hospital. The smart card is to be issued to the patient containing his patient-ID and his basic information like name, address, phone number etc. Patient information like doctor diagnosis, test reports, MRI, CT-scan images etc. will be stored in the databases of the hospital server as per the Patient-ID (Hood and Scott, 2006). For future usage, it helps in quick diagnosing the illness of the concerned patient. While visiting to the hospital for treatment, patient will need to carry only smart card and the administrator of the hospital or the doctor will use this smart card through card reader to extract the patient related needed information. The collected information has been stored and retrieved through MySQL (Sun Microsystems, 2009), a database management tool. MySQL is reliable, robust and widely used open source relational database management tool. For fast retrieval of health care related data, the selection of appropriate indexing techniques has been proposed in MySQL.

This paper is categorized into 7 sections. Literature review has been discussed in section 2. Section 3 discusses with the problem formulation of the health care system. Section 4 explains the methodology used for health care system. Various finding observed are organized in various cases and mentioned in section 5. In section 6, results have been proposed on the basis of various case studies. Section 7 proposes conclusion.

2. Literature Review

Government of India is planning to implement the Multipurpose National Identity Card or Unique Identification Card (UID) project. This card will contain details like the name, sex, address, marital status, photo, identification mark and finger biometrics and will link a person's passport number, driving license, pan card, bank accounts, address and voter-id etc. This information will be checked through an existing database. So, if someone has different addresses on PAN and driving license, is liable to get caught. Those who will opt out of this program will have much inconvenience in doing business, operating bank accounts and other offices which will require a UID (Hickok, 2010). Germany and United States are using health cards and maintaining the patient records electronically to enhance the working efficiency of the hospital system (Marschollek and Demirbilek, 2006; Jha et al., 2009).

The requirement for automation systems in hospitals and health centres are gaining more and more importance in the present scenario. Web based communication between patients and hospital systems will potentially improve the quality of health care (Hassol, 2004). Many Electronic Health Record Systems fail to provide user friendly interface due to the lack of systematic consideration of human centred computing issues. Such interfaces can be improved to provide easy to use, easy to learn and error-resistant Electronic Health Record System to the users. To evaluate the usability of an Electronic Health Record System and suggest area of improvement in the user interface was discussed in paper (Saitwal et al., 2010; Huser et al., 2010). The usage of smart card for storing the important information of the patient has been discussed in papers (Lockett et al., 2003; Liu et al., 2006). Nowadays varieties of smart cards are available in market. These cards have proven to be quite convenient tokens in terms of identification and authentication in day to day activities (Kardas and Tunali, 2006; Charegaonkar et al., 2008). A well protected of the confidentiality and integrity of patient information, a computerized health-care information system has been proposed in paper (Smith and Eloff, 1999). At the same time, the patient information also needs to be readily available to all authorised health-care providers, in order to ensure the proper treatment of the patient (Smith and Eloff, 1999). Further, proper selection of indexing techniques, storage mechanism and methods to design the application for achieving the better performance has been discussed in this paper.

3. Problem Formulation

In the designing of “Smart Card based Online Health Care System”, the following issues have been considered:

3.1: If a patient is registered in one hospital and integration and retrieval of patient information is not possible in the hospital system then while taking the consultation with doctor, patient may easily forget to inform about his/her allergy to the medicine and thus, may not be able to explain the previous treatments properly which might result in incorrect prescriptions to the patient. Keeping this in mind, a Smart Card based online Health Care System is being proposed. The proposed system is shown in Figure 1. The servers for the hospitals with suitable configuration are required. All the local servers of the individual hospitals will be connected with the centralized main server of this system through internet. A smart card reader / writer unit needs to be attached to each computer of the hospital system. The proposed health care application will be installed to all the local servers of the hospitals. These hospitals will be connected via intranet and internet. The patient smart card stores some important information like unique patient Id, name, sex, date of birth, blood group etc. As per the registration number/ patient-ID, patient details like treatment prescriptions, test reports, images like MRI, CT-scan etc. will be stored in the database of the hospital local server. On the basis of stored details of the patient, doctor can prescribe the proper medicine. For fast retrieval of patient data, different indexing techniques have been proposed in MySQL. For improving the performance of our health care system, we have used various tuning techniques available in MySQL.

3.2: An application, Smart Card based online Health Care System has been developed that can serve large number of hospitals and provide an interface for their interaction. As per the application, the system will generate unique registration number/ patient Id for every patient. To increase the performance of the application, we have considered the following database and application design aspects.

- Normalization of tables.
- Maximum utilization of memory.
- Imposing necessary constraints on tables.
- Selecting best possible structure for queries.
- Tuning and optimizing queries.
- Optimal selection of join order and join algorithm.
- Managing insert operation to reduce load from server.
- Front end should be strong enough to reduce basic user faults and provide easy interaction.
- Speed up the DML (insert, update, delete) operations.

4. Methodology

The proposed application is implemented as follows (Kohli and Verma, 2010c).

The patient will provide basic information through smart card (having 10 digit patient-ID and personal information) at registration counter where it will be uploaded to hospital server. The smart card will store registration number/ Patient-ID and personal information like Name, Address and Date of birth etc. of a patient. Patient's files have been stored in two forms i.e. image and PDF. Images are stored separately in image tables and PDFs in PDF table. All the patient related information will be on a centralized database server, so the movement of files is not required. The required files for a particular patient and their lab test report & images, prescriptions, scan document and diagnosis reports can be retrieved from centralized database server. Accessing the information related to patients will be controlled by various levels of access control and it will prevent the unauthorized access from the data base. By normalizing the database all possible redundancies will be removed. Optimization is covered at application design level, database design level. We have done proper memory utilization and optimization of queries for accessing the data base.

Few points about the optimization are as follows:

- Application is studied well and requirements are identified. These requirements are categorized on object level and respective tables are created to store that information. After that database is normalized up to BCNF (Boyce Codd Normal Form) level and 14 tables are created.
- Size of attributes (their respective data types) is further minimized in order to compact record size, so that maximum number of records can be stored in a single data page. Selected queries are designed to handle all possible search criteria.
- Indexes are created to speed up above read access. First database IDs are minutely studied to find out what possible values different attributes can take and what could be the size of tables. Considering both the factors Hash, B-Tree (clustered and non-clustered) and Bitmap indexing techniques are proposed.
- Order of attributes in composite indexes are studied and justified.
- Every hospital should have day to day patient information locally to improve the performance particularly in cases of frequent insert and update operations.
- Due to monotonically increasing nature of patient Id, there is always a problem of unbalancing of B-Tree indexes. Index portioning & reverse key indexing is proposed to get rid-off this problem.
- In a multi-table join queries, we have studied the nature of the table & depending on the characteristics of the table, their order will be pre-estimated.
- Properly joined algorithms are justified in respective cases and hints are given to optimizer.
- Selecting the most optimal execution path by the optimizer is the time consuming process. By giving the hints we have reduced the time taken by the optimizer up to some level.
- To improve the DML operation by the hospital, we have tried to keep the data initially at the hospital local server. This scheme reduces load from main centralized server and problem of frequent unbalancing of index structure is also removed.
- Every day a particular time is selected to automatically update centralized server by collecting information from each hospital servers of this system. The operation to refresh the centralized server may be made more frequent if required.
- Transaction log is permanently off at centralized server as we don't need any log for recovery because all the information are at each hospital server and can be reproduced easily.
- All primary key, foreign key constraints are removed from centralized server as those constraints are already checked at local server of the hospitals. So there is no need to revalidate the already validated information. It results in improved data loading performance at the centralized server.
- For updating centralized server, we have recommended to use bulk copy and bulk insert schemes.

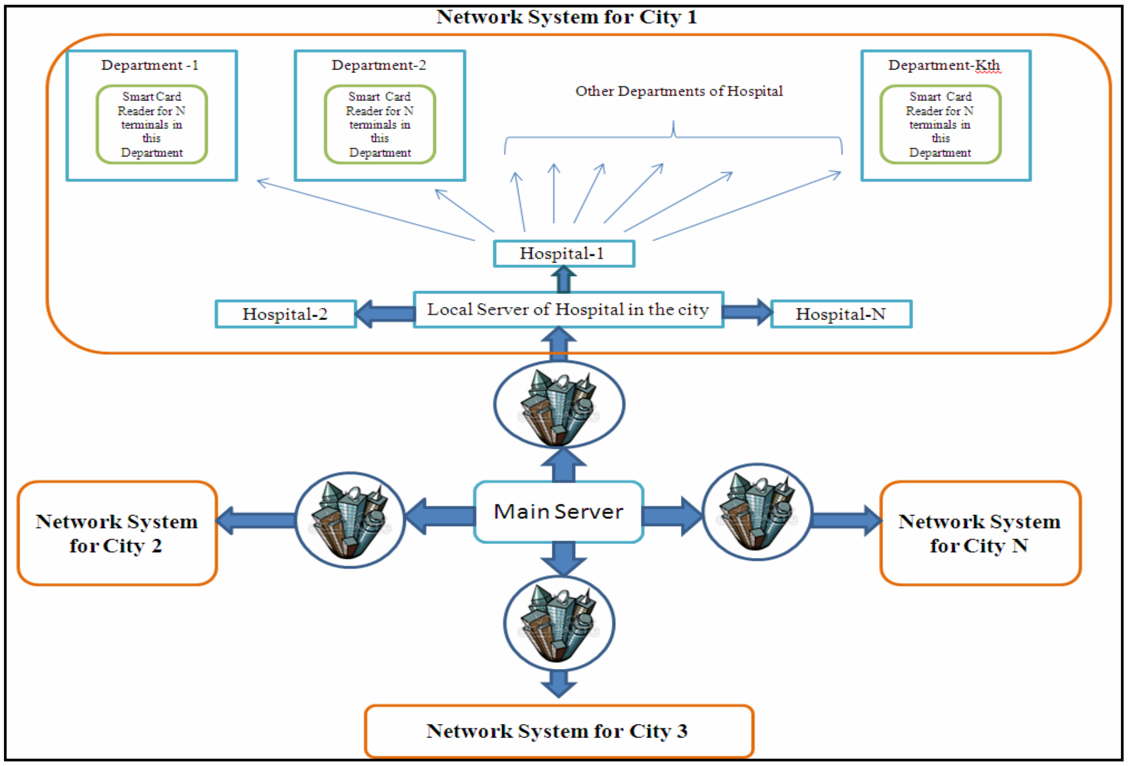

Network System for City 1
Department -1
Smart Card Reader for N terminals in this Department
Department-2
Smart Card Reader for N terminals in this Department
Other Departments of Hospital
Department-Kth
Smart Card Reader for N terminals in this Department
Hospital-1
Hospital-2
Local Server of Hospital in the city
Hospital-N
Network System for City 2
Main Server
Network System for City N
Network System for City 3

Figure 1. Smart card based on line health care system

5. Findings

In this paper, we have discussed few cases on the database where out of 14 tables, 4 tables store patient related information, which are,

- Patient stores registration information of patient.
- PatientDoctor stores doctor related information like diseases, etc.
- PatientLab stores test report of patients.
- PatientRoom stores Room Ids, daily charges and other attributes of room where patient is admitted.

5.1 Case 1: Strategy for retrieving patient information based on Patient ID attribute*:* A table can contain only one clustered and any number of non-clustered B-Tree indexes. Clustered B-Tree index is better for columns with fixed size and unique values. Patient-ID attribute satisfies these criteria but Patient-ID attribute in PatientDoctor, PatientLab and PatientRoom tables is foreign key. So these tables may contain more than one record for one Patient-ID. For these tables non-clustered B-Tree index is preferred and gives better results than other indexing techniques. It gives optimal results for insert, update and read operations.

5.2 Case 2: Strategy for retrieving patient information in original tables and day tables: We do not insert new records directly in original table (say) at centralized server but store it in database of hospital local server (day table). When a patient is registered at some hospital, same day patient information can be retrieved from the local server of the hospital otherwise from main centralized server. Late night at a particular time local hospital records will be merged in the databases of centralized server. The local server of the hospital will be ready to hold new records from the next day morning. Read access is much more at centralized server and **Hash indexing technique** gives better results in read access than other indexing techniques. So while retrieval of patient data from centralized server, we have used **Hash indexing technique**.

5.3 Case 3: Strategy for retrieving patient information based on disease attribute: To find the number of patients of a particular disease, we have used Bitmap index. Disease information is stored with patient registration details in patient table and this attribute will hold a particular value from a list of disease that may contain few disease names. **Bitmap indexing technique**

has a performance advantage for such data.So use of **Bitmap indexing technique** will give the better result than other indexing techniques.

5.4 Case 4: Strategy for retrieving patient information based on Test ID attribute: To find the number of enrolment for a particular test, we have used **Bitmap indexing technique**. This information can be found by the Test ID attribute of TestBooking table. This attribute will have less ID (few data) from a fixed set of Test ID's. So **Bitmap indexing technique** has given better results than other indexes.

5.5 Case 5: Strategy for retrieving patient information based on Doctor ID attribute: To find out the number of patient under a particular doctor, we have used **Bitmap indexing technique**. As this information can be found by Doctor ID attribute of PatientDoctor table. This case is similar to test and disease forecast. So **Bitmap indexing technique** must be useful.

5.6 Case 6: Strategy for retrieving patient information based on Hospital ID, Doctor ID, Lab ID and Test ID attributes: Index for other select queries

- Retrieve hospital details.
- Retrieve doctor details.
- Retrieve lab details.
- Retrieve test details.

Above information's will be retrieved from respective Hospital, Doctor, LabDetails and TestDetails tables. All the above tables have large number of records and new records will be updated time to time. Here **Non-clustered B-Tree indexing technique** has given the most optimal result for select, insert and update operations.

5.7 Proposed indexing scheme*:* As per case1 to case 6, the proposed indexing scheme for health care system will be as per table 1 (Kohli and Verma, 2010c).

Table 1. Proposed indexing scheme

	Table name	**Attribute**	**Indexing scheme**	Index Architecture
Original Tables (At Central Server)	Patient	PatID	Hash	nil
		Disease	Bitmap	nil
	PatientDoctor	PatID	Hash	nil
		Doctor_ID	Bitmap	nil
	PatientLab	PatID	Hash	nil
	PatientRoom	PatID	Hash	nil
	TestInfo	Test ID	Bitmap	nil
	Hospital	Hospital_ID	BTree	Clustered
	Doctor	Doctor_ID	BTree	Clustered
	LabDetails	Lab_ID	BTree	Clustered
	TestDetails	Test_ID	BTree	Clustered
Day Tables (At Local Server)	Patient	PatID	BTree	Clustered
	PatientDoctor	PatID	BTree	Nonclustered
	PatientLab	PatID	BTree	Nonclustered
	PatientRoom	PatID	BTree	Nonclustered

5.8 Database blocks size optimization:

For efficient utilization of memory, database size has been optimized i.e. use of smaller data types in database tables. Database software defines some fixed sized storage structure to store table records and other information. These structures hold some bits for system information and rest for data. The ratio between system information and table records in a storage structure plays an important role as it defines the memory organization of data base. If portion of these structures that holds table records is utilized efficiently then memory will be utilized efficiently and definitely this will result in some performance gain (Kohli and Verma, 2010a,b).

MySQL stores data in a special structure called index pages that are 16Kb (16384 bytes) in size. Some space i.e.152 bits on the data pages are used to store system information, which leaves 16232 bits to store user's data. So, if the table's row size is 8126 bits, then only one row will be placed on each data page and 8106 bits memory unused. If the row size is reduced to 8116 bits, two rows can be stored within a single page. The less space used, the smaller the table and less the time to disk access in MySQL has to perform when reading data pages from disk. We have designed our tables in such a way as to fit maximum number of rows into one data page. To fit maximum number of rows into one data page, we have specified the narrowest columns. The narrower the columns, the less data is stored, and the faster MySQL are able to read and write data. For the same, we have proposed the row size of a table should be some devisor of 16232 when the page size is 16k, 8040 when the size is 8k and 3944 when the page size is 4k etc. for maximum utilization of memory and speed up read/write operations from database. Figure 2, 3, 4

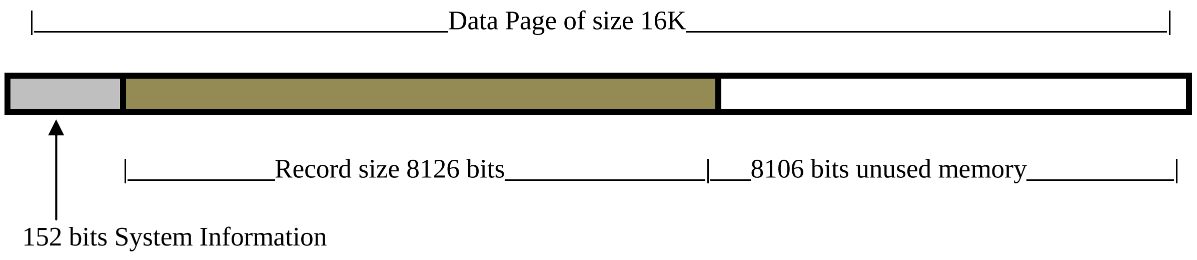


Figure 2: Wastage of Memory

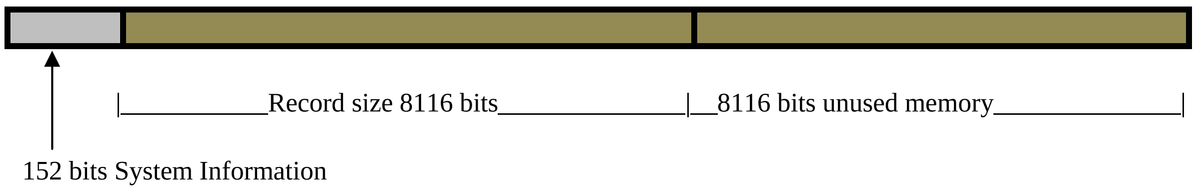


Figure 3(a): full memory utilization

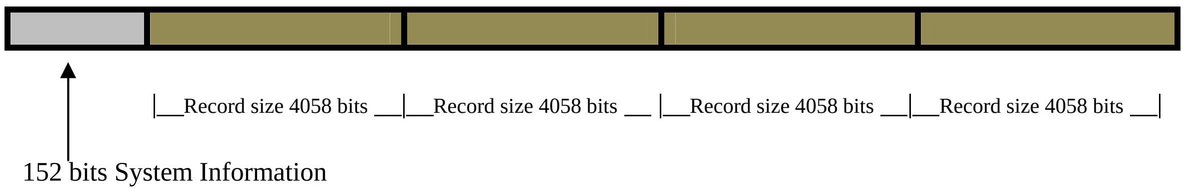


Figure 3(b): full memory utilization

6. Case Study:

To identify the bottleneck and performance related issues we have done the following case studies using different methods and analyze the query and their execution path. We have taken approximate 7 lac records in every table of our database for simulation.

The following query selects patient details from four different tables by applying inner joins.

Query: *SELECT Patient1.PatName, Patient1.Address, Patient1.ContactNo, Patient1.Sex, Patient1.Occupation, Patient1.Guardian, Patient1.RegNo,Patient1.ReferredBy, Patient1.BloodGroup, Patient1.PatientHistory, Patient1.Disease, Patient1.DiseaseCatageory, Patient1.EntryDate,PatientRoomIPD.Hospital_ID, PatientRoomIPD.Room_ID, PatientRoomIPD.StartDate, PatientRoomIPD.EndDate,PatientRoomIPD.Status, PatientRoomIPD.TotalCharge, PatientDoctor.Doctor_ID, PatientDoctor.DoctorName, PatientDoctor.StartDate AS TreatmentStartDate,PatientDoctor.EndDate AS TreatmentEndDate, PatientDoctor.No_Of_Visits, PatientDoctor.Charges AS TreatmentCharge, PatientLab.LabNo,PatientLab.TestNo, PatientLab.TestDate,PatientLab.Unit*

FROM Patient1
INNER JOIN
PatientRoomIPD ON Patient1.PatID = PatientRoomIPD.PatID
INNER JOIN
PatientDoctor ON Patient1.PatID = PatientDoctor.PatID
INNER JOIN
PatientLab ON Patient1.PatID = PatientLab.PatID
where Patient1.PatID=ID;

We have executed above query by putting it into stored procedure using front end and query analyser. We have taken random Patient ID several times. We have got the results with Hardware configuration: Intel® Duel Core 2.8GHz 2GB RAM as per table 2

Case 1: Retrieval time with traditional MySQL database:

After running the query with traditional MySQL database, retrieval time in stored procedure using front end is less than query analyzer. Stored procedure using front end has better understanding about the MySQL statement because MySQL statement is exactly the same as the previous executed statement with different patient ID value only.

Table 2: Retrieval time with traditional MySQL data base

	On Asp.net Search Page (Front-End)		**On Query Analyser**	
Iteration	**Execution time in sec. using Stored Procedure**	**Execution time in sec. using Query**	**Execution time in sec. using Stored Procedure**	**Execut ion time in sec. using query**
1	.0491	.0543	.0564	.0645
2	.0473	.0537	.0554	.0589
3	.0489	.0631	.0572	.0675
4	.0491	.0571	.0602	.0614
5	.0478	.0544	.0546	.0598
6	.0483	.0597	.0617	.0678
7	.0462	.0549	.0545	.0559
8	.0497	.0572	.0621	.0656
Mean value In seconds	**.0483**	**.0568**	**.0577**	**.0626**

Retrieval time of patient-ID with traditional MySQL database in stored procedure:

1. **On Front –end is 0.0483 second**
2. **On Query Analyzer is 0.0577 second**

Probability density function and cumulative density function of the above results (case 1) on front end are as per figure 4a**:**

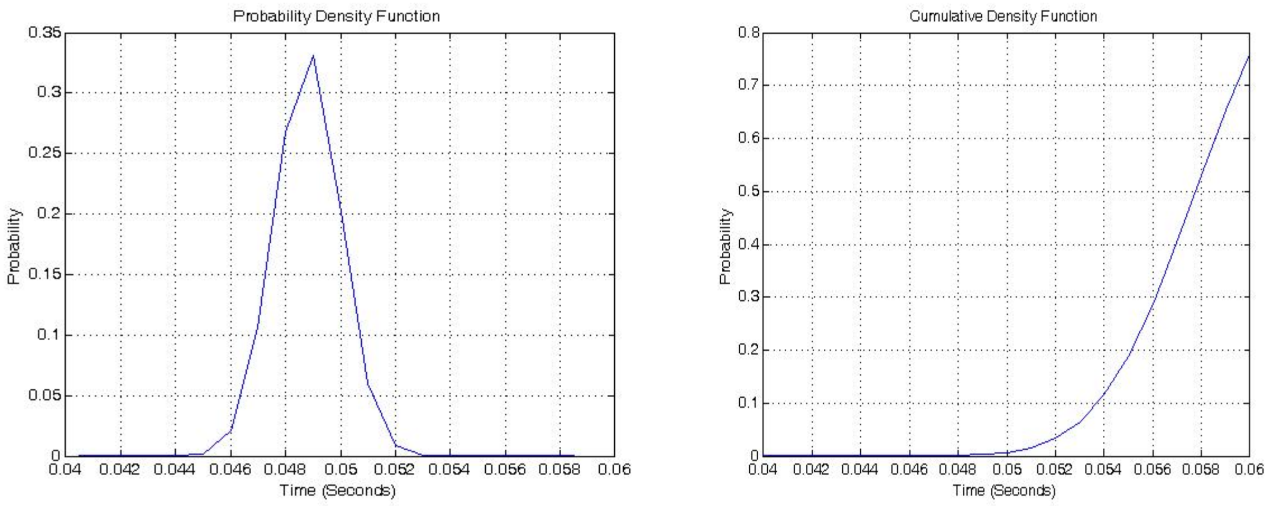


Figure 4a: Probability density function and cumulative density function on front end

Probability density function and cumulative density function of the above results (case 1) on query analyzer are as per figure 4b:

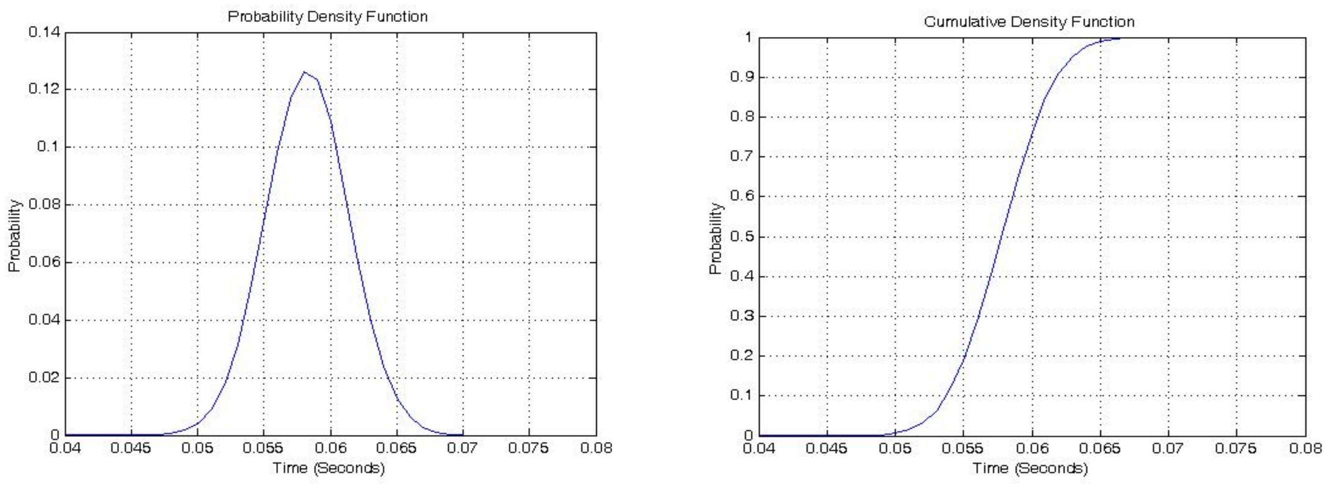


Figure 4b: Probability density function and cumulative density function on query analyzer

Case 2: Retrieval time from new database after block size optimization using B-Tree Indexing technique (hospital local server):

We have executed above mentioned query in new databases getting after block size optimization by putting it into stored procedure using front end and query analyser by taking random Patient ID several times. Execution time of a query on a proper optimized MySQL database using B-Tree indexing technique is less than the traditional MySQL database. The results with same hardware configuration is as given in table 3

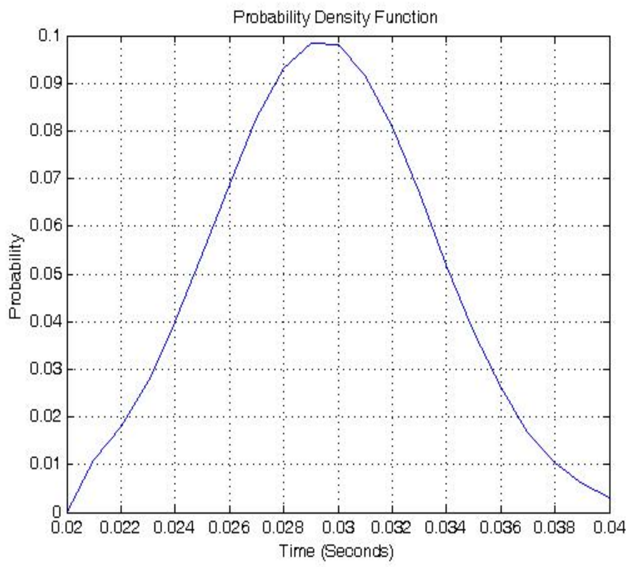
Table 3: Retrieval time from new data base after block size optimization and using B-Tree indexing technique

	On Asp.net Search Page (Front-End)		**On Query Analyser**	
Iteration	**Execution time in sec. using Stored Procedure**	**Execution time in sec. using Query**	**Execution time in sec. using Stored Procedure**	**Execut ion time in sec. using query**
1	.0272	.0311	.0327	.0436
2	.0250	.0277	.0336	.0338
3	.0278	.0317	.0282	.0340
4	.0260	.0345	.0461	.0360
5	.0325	.0332	.0335	.0323
6	.0256	.0321	.0328	.0358
7	.0308	.0300	.0461	.0459
8	.0365	.0326	.0357	.0341
Mean value In seconds	**.0289**	**.0316**	**.0361**	**.0369**

Retrieval time of patient-ID using traditional MySQL database and optimized MySQL database in stored procedure:

1. **On Front end and using traditional database is 0.0483 second and optimized database is 0.0289 second**
2. **On Query analyzer using traditional database is 0.0577 second and optimized database is 0.0361 second**

Probability density function and cumulative density function of the above results (case 2) on front end are as per figure 5a**:**



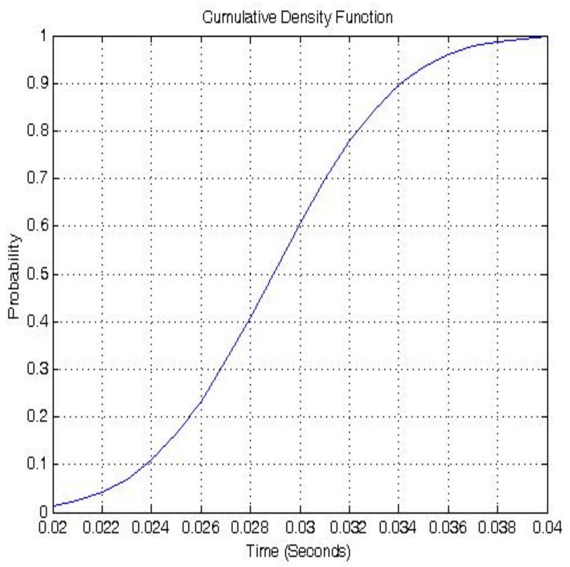


Figure 5a: Probability density function and cumulative density function on front end

Probability density function and cumulative density function of the above results (case 2) on query analyzer are as per figure 5b**:**

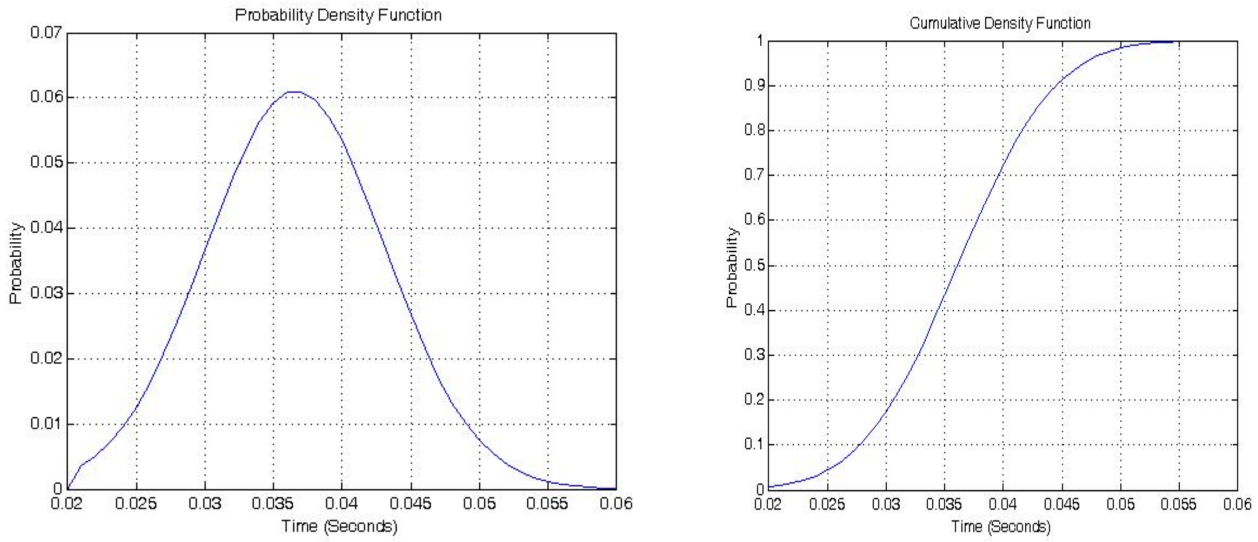


Figure 5b: Probability density function and cumulative density function on query analyzer

Case 3: Retrieval time using Hash indexing technique from optimized MySQL databases of centralized server:

We have executed same query using new databases and Hash indexing technique by putting it into stored procedure on front end and query analyser and taking random Patient ID several times on the database of centralized server. Retrieval time to execute the query using hash indexes from optimized MySQL database of centralized server is less than from the traditional database. The results with same hardware configuration is as given in table 4

Table 4: Retrieval time from optimized MySQL data base after applying Hash index

	On Asp.net Search Page (Front-End)		**On Query Analyzer**	
Iteration	**Execution time in sec. using Stored Procedure**	**Execution time in sec. using Query**	**Execution time in sec. using Stored Procedure**	**Execution time in sec. using query**
1	.0133	.0131	.0289	.0384
2	.0122	.0159	.0246	.0378
3	.0152	.0142	.0290	.0291
4	.0131	.0148	.0394	.0291
5	.0155	.0136	.0379	.0377
6	.0135	.0142	.0290	.0251
7	.0136	.0162	.0287	.0385
8	.0163	.0130	.0325	.0293
Mean value In seconds	**.0140**	**.0143**	**.0312**	**.0331**

Retrieval time of patient-ID using stored procedure on MySQL database of centralized server:

1. **On query analyzer using Hash indexing technique is 0.0312 second and using B-Tree indexing technique is 0.0361 second.**
2. **On front end using Hash indexing technique is 0.0140 second and using B-Tree indexing technique is 0.0289 second.**

Probability density function and cumulative density function of the above results (case 3) on front end are as per figure 6a**:**

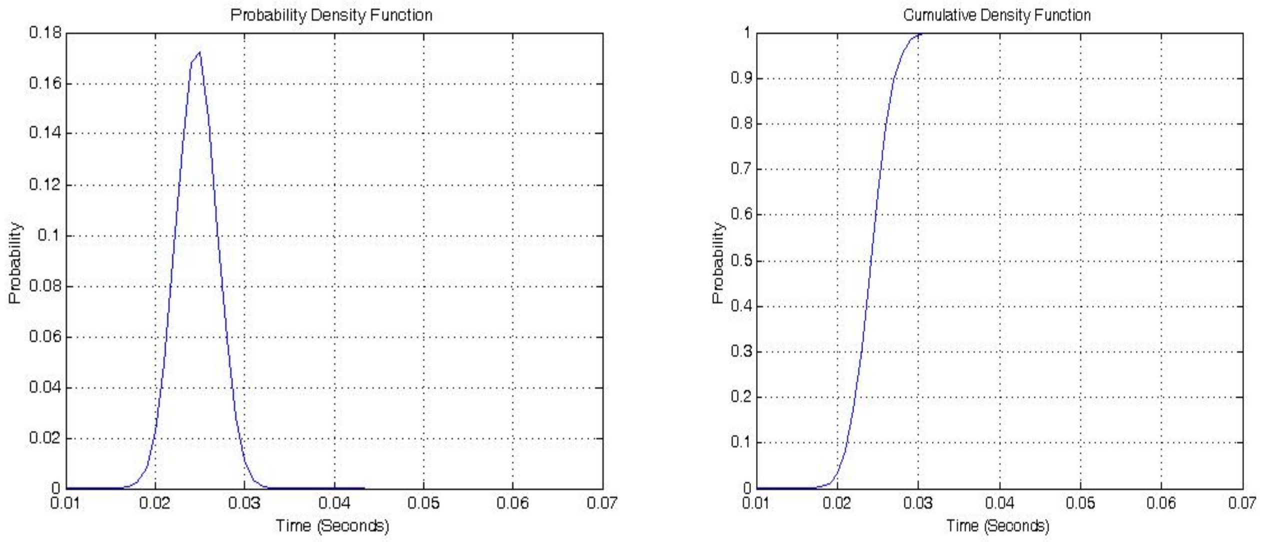


Figure 6a: Probability density function and cumulative density function on front end

Probability density function and cumulative density function of the above results (case 3) on query analyzer are as per figure 6b**:**

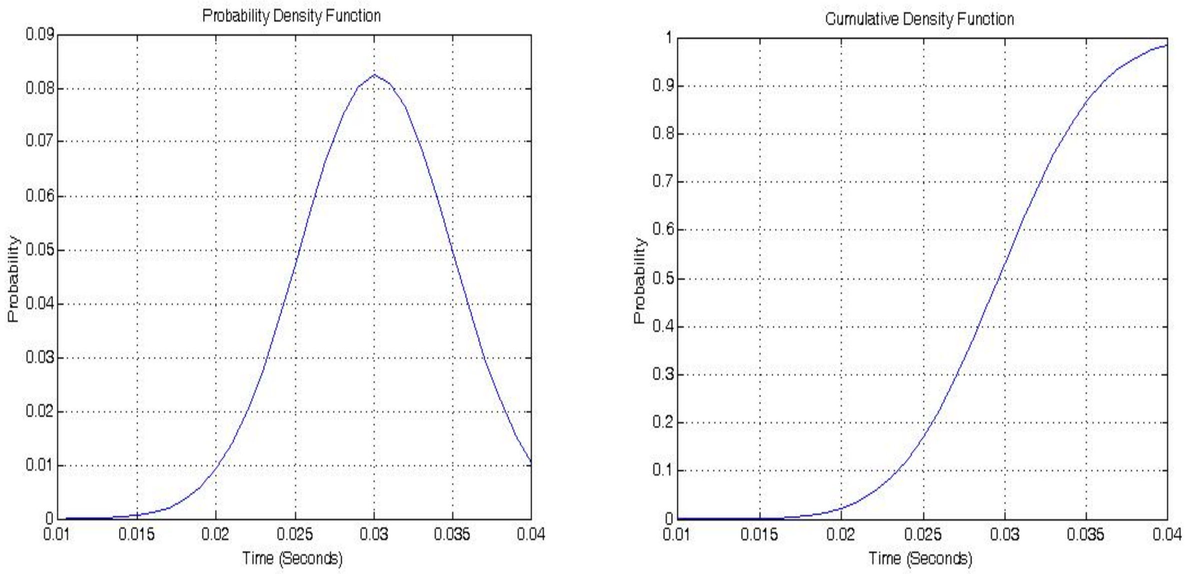


Figure 6b: Probability density function and cumulative density function on query analyzer

Case 4: Retrieval time of patient test information after applying B-Tree index:

We have executed same query on optimized MySQL databases and applied B-Tree indexing technique and put it into stored procedure using query analyser for retrieval of test information by taking random Patient ID several times. The results with same hardware configuration is as given in table 5

Table5: Retrieval time from Testinfo after applying B-Tree index

	On Query Analyzer	
Iteration	**Execution time in sec. using Stored Procedure**	**Execution time in sec. using query**
1	.023	.027
2	.026	.028
3	.028	.031
4	.020	.007
5	.028	.009
6	.003	.030
7	.002	.008
8	.002	.009
Mean value In seconds	**.016**	**.019**

Probability density function and cumulative density function of the above results (case 4) on query analyzer as per figure 7**:**

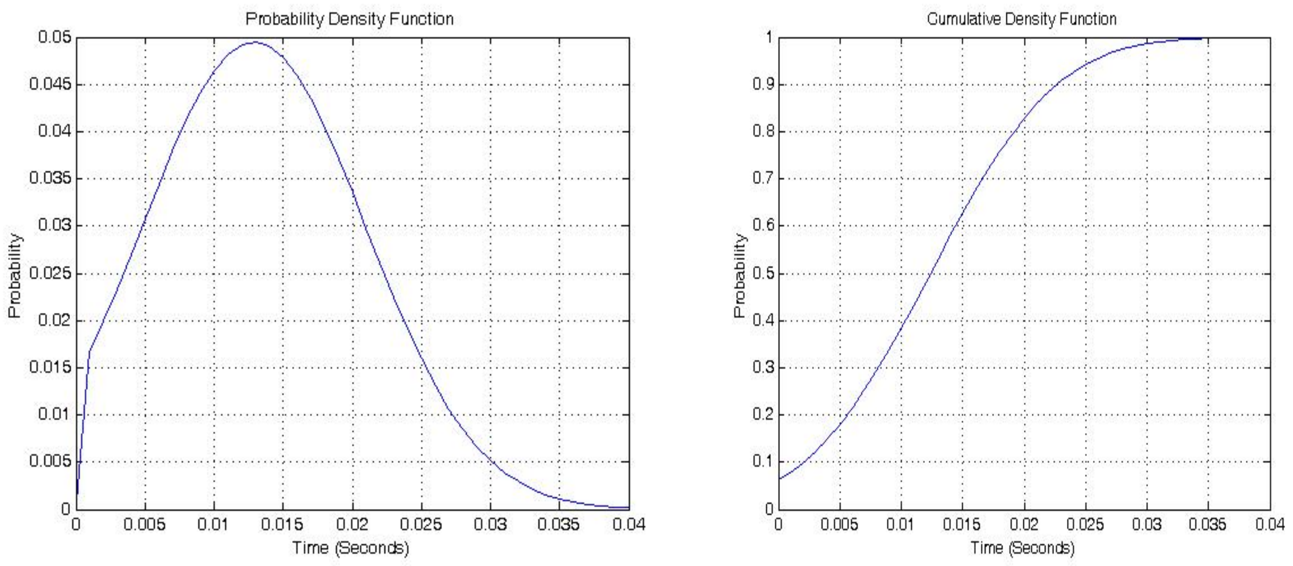


Figure 7: Probability density function and cumulative density function on query analyzer

Case 5: Retrieval time of patient test information of applying Bitmap indexing technique:

Bitmap indexes are applied in that column which has low cardinality but MySQL does not support Bitmap indexing technique till now. So for testing we have prepared our database in oracle 10g which support both B-Tree and Bitmap. We have applied B-Tree indexing technique and Bitmap indexing technique on TestID of TestInfo table. Using bitmap index, retrieval time of patient test information is less than use of B-Tree indexing technique. The results of the same are as per table 6:

Query: *Select * from TestInfo where TestID= ID;*

Table 6**:** Retrieval time from Testinfo after applying Bitmap indexing technique

	On Query Analyser	
Iteration	**Execution time in sec. using Stored Procedure**	**Execution time in sec. using query**
1	.020	.024
2	.019	.022
3	.021	.030
4	.018	.024
5	.008	.008
6	.009	.010
7	.002	.002
8	.002	.005
Mean value In seconds	**.013**	**.016**

Retrieval time on query analyzer:

1. **On stored procedure using B-Tree indexing technique is 0.016 second and Bitmap indexing technique is 0.013 second.**
2. **On query using B-Tree indexing technique is 0.019 second and Bitmap indexing technique is 0.016 second.**

So for retrieval of test related information, use of Bitmap indexing technique always gives the better results than B-Tree indexing technique.
Probability density function and cumulative density function of the above results (case 5) on query analyzer as per figure 8

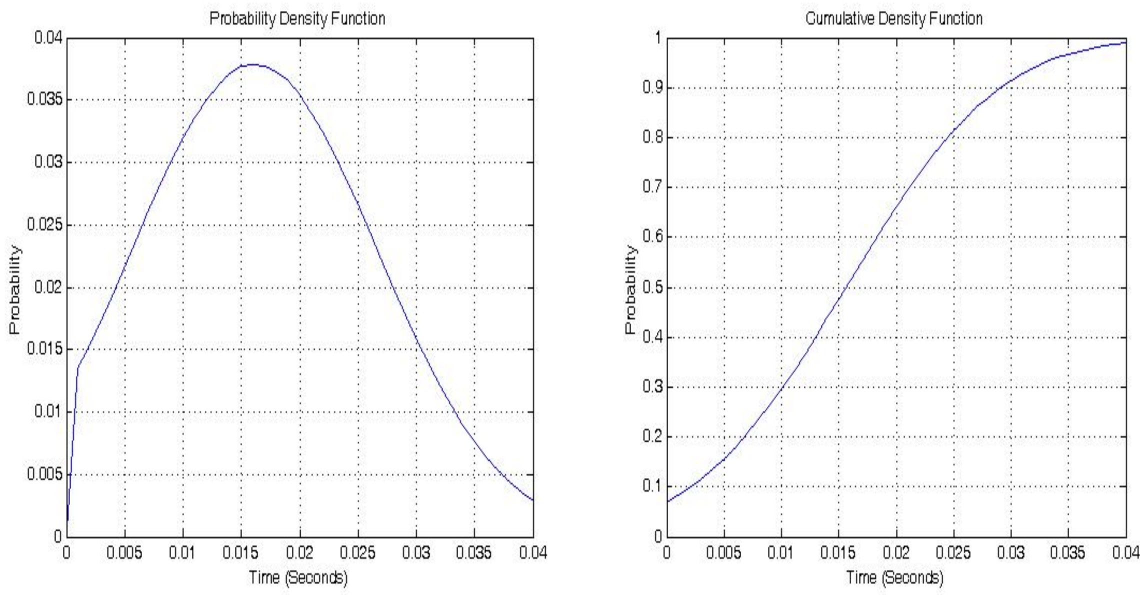


Figure 8: Probability density function and cumulative density function on query analyzer

7. Conclusions

1. When the search criteria is based on the column where cardinality is very high then better to use B-Tree indexing technique otherwise Bitmap indexing technique. In read intensive application, Hash indexing technique gives the better results. So as per our application for fast retrieval of patient related data, we are recommending to use B-Tree, Hash and Bitmap indexing techniques in MySQL database engine instead of using only B-Tree indexing technique.
2. Efficiently utilization of table records in MySQL database improves the performance and memory utilization of the system.

Acknowledgement

We acknowledge Dr. Arnab Bhatacharya and Mr. Kalik Ahmad, IIT Kanpur for their help in the preparation of paper.

References

Liu C.-T., Pei-Tun Yang, Yu-Ting Yeh and Bin-Long Wang, 2006. The impacts of smart cards on hospital information systems-An investigation of the first phase of the national health insurance smart card project in Taiwan, *International Journal of Medical Informatics,* Vol. 75, No. 2, pp. 173-181.

Hassol A., Walker J.M., Kidder D., Rokita K., Young D., Pierdon S., Deitz D., Kuck S. and Ortiz E., 2004. Patient Experiences and Attitudes about Access to a Patient Electronic Health Care Record and Linked Web Messaging, *Journal of Informatics in Health and Biomedicine*, Vol. 11, No. 6, pp. 505-513.

Hickok E. 2010. Presentation of the UID project by Ashok Dalwai- a report in Internet Governance Blog-Sep 08, 06:50 P.M.

Kohli N., Verma N.K. 2010a. Performance Issues of Hospital System using MySQL,"in the *Proceedings of 3rd IEEE International Conference on Computer Science and Information Technology*, China, Vol. 6, pp. 497-501.

Marschollek M., and Demirbilek E., 2006. Providing longitudinal health care information with the new German Health Card-a pilot system to track patient pathways, *Computer Methods and Programs in Biomedicine*, Vol. 81, No. 3, pp. 266-271, March.

Jha A.K., DesRoches C.M., Campbell E.G., Donelan K., Rao S.R., Ferris T.G., Shields A., Rosenbaum S., and Blumenthal D., 2009., Use of Electronic Health Records in U.S. Hospitals, *The New England Journal of Medicine*, Vol. 360, pp. 1628-1638.

Lise Poissant, Jennifer Pereira, Robyn Tamblyn and Yuko Kawasumi, "The Impact of Electronic Health Records on Time Efficiency of Physicians and Nurses: A Systematic Review,"Journal of Informatics in Health and Biomedicine, Vol. 12, Issue 5, pp. 505-516, September-October 2005.

Lockett E., Park S.K., Jiang G.C., Riddle M., 2003. Security Aspects of Smart cards, term project CS 574 Fall 2003, San Diego State University submitted on November, 3rd.

Kardas G., Tunali E.T., 2006. Design and implementation of a smart card based healthcare information system," *Computer Methods and Programs in Biomedicine,* Vol. 81, No. 1, pp. 66-78, January.

Saitwal H., Feng X., Walji M., Patel V. and Zhang J., 2010. Assessing performance of an Electronic Health Record (EHR) using Cognitive Task Analysis, *International Journal of Medical Informatics*, Vol. 79, No. 7, pp. 501-506.

Huser V., Narus S.P. and Rocha R.A. 2010. Evaluation of a flowchart-based EHR query system: A case study of RetroGuide, *Journal of Biomedical Informatics*, Vol. 43, No. 1, pp. 41-50.

Hood M.N., and Scott H., 2006. Introduction to pictural archive and communication systems, *Journal of Rodiology Nursing*, Vol. 25, No. 3, September.

Charegaonkar V., Nair K.H., and Gautam G., 2008. Smart card transaction processing-case study: Applying best practices to improve performance, IBM, web address: www.ibm.com/websphere/developer/zones/hipods Date: 1 February.

Kohli N., Verma N.K., 2010b. Performance issues of health care system with audio and video facilities, in the *Proceedings of the 3rd International Conference on Data Management,* India, March.

Smith E. and Eloff J.H.P. 1999. Security in health-care information systems—current trends, *International Journal of Medical Informatics,* Vol. 54, pp. 39-54.

Sun Microsystems, 2009. Inc. Developer zone. <http://dev.mysql.com/> Last accessed 10.04.09.

Kohli N., Verma N.K. 2010c. MySQL based selection of appropriate indexing technique in hospital system using multiclass SVM, *International Journal of Engineering, Science and Technology*, Vol. 2, No. 6, pp.119-130.

Biographical notes

Narendra Kohli was born in India on April 13, 1963. He received M. Sc. Engg. from Kiev university Kiev in 1988 and perusing Ph.D. from Indian Institute of Technology Kanpur, India since 2004. He is currently an Associate Professor and Head with the Department of Computer Science & Engineering, Harcourt Butler Technological Institute Kanpur, India. He is a Member of IE (India), Fellow of IETE (India), and senior member of ISE.

Dr. Nishchal K.Verma was born in India on September 9, 1973. He received the B.E. degree from the Faculty of Engineering, Dayalbagh Educational Institute, Agra, India, in 1996, the M.Tech. degree from the Indian Institute of Technology (IIT) Roorkee, Roorkee, India, in 2003, and the Ph.D. degree from IIT Delhi, New Delhi, India, in 2007, all in electrical engineering. He is currently an Assistant Professor with the Department of Electrical Engineering, IIT Kanpur, India. His research interests include fuzzy systems, neural networks, data mining, fault diagnosis, bioinformatics, color segmentation, video clip or image sequence modeling, machine learning, and computational intelligence. Dr. Verma is a reviewer for several reputed national and international journals and conferences, including the IEEE TRANSACTIONS ON FUZZY SYSTEMS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, and CYBERNETICS: PARTS A, B, AND C, the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and *Pattern Recognition*.

Received January 2011
Accepted April 2011
Final acceptance in revised form May 2011