PicoDBMS: Scaling down Database Techniques for the Smartcard

OBSERVATION ABOUT EXISTING PROBLEM

Smartcards are the most secure portable computing device today. As smartcards get more and more powerful (with 32 bit CPU and more than 1 MB of stable memory storage), versatile and multi-applications, the need for database management arises. However, smartcards have severe hardware limitations (very slow write, very little RAM, constrained stable memory, no autonomy, etc.) which make traditional database technology irrelevant. The major problem is scaling down database techniques so they perform well under these limitations. smartcards have severe hardware limitations which stem from the obvious constraints of small size (to fit on a flexible plastic card and to increase hardware security) and low cost (to be sold in large volumes). Smart Card is used in every sectors these days like health card storing a complete medical folder including the holder's doctors, blood type, allergies, prescriptions, etc. The volume of data can be important and the queries fairly complex (select, join, aggregate).
However, smartcards also have severe hardware limitations which stem from the obvious constraints of small size (to fit on a flexible plastic card and to increase hardware security) and low cost (to be sold in large volumes).
These limitations (tiny RAM, little stable storage, very costly write and lack of autonomy) make traditional database techniques irrelevant. With the extreme constraints of the smartcard, the major problem is scaling down database techniques. While there has been much excellent work on scaling up to deal with very large databases, e.g., using parallelism, scaling down has not received much attention by the database research community. However, scaling down in general is getting very important for commodity computing and is quite difficult.

GOAL AND KEY CONTRIBUTIONS

In this paper, In-depth analysis of the above problem is given and a PicoDBMS solution is explained which is based on highly compact data structures and query execution without RAM. Paper is showing the effectiveness of our techniques through performance evaluation.
In this paper, the problem of scaling down database techniques and propose the design of what we call a PicoDBMS is addressed. The design has been made with smartcard applications in mind but its scope extends as well to any ultra-light computer device based on a secured monolithic chip. This paper makes the following contributions:

- Analyze the requirements for a PicoDBMS based on a typical healthcare application and justify its minimal functionality.
- give an in-depth analysis of the problem by considering the smartcard hardware trends and derive design principles for a PicoDBMS.
- propose a new pointer-based storage model that integrates data and indices in a unique compact data structure.
- propose query execution techniques which handle complex query plans (including joins and aggregates) with no RAM consumption.

- shows the effectiveness of each technique through performance evaluation.

BASIC IDEA OF KEY CONTRIBUTIONS

There are many types of smartcard applications for example in Money and identification (credit cards, e-purse, SIM for GSM, phone cards), Downloadable databases (list of restaurants, hotels and tourist sites), User environment (address book),personal folders (personal folders may be of different nature: scholastic, healthcare, car maintenance history, loyalty. They roughly share the same requirements, which we illustrate next with the healthcare example.) etc.

The idea of PicoDBMS is based on healthcare application:- The health card is very representative of personal folder applications and has strong database requirements. Several countries (France, Germany, USA, Russia, Korea...) are developing healthcare applications on smartcards. The initial idea was to give to each citizen a smartcard containing her identification and insurance data. As smartcard storage capacity increases, the information stored in the card can be extended to the holder's doctors, emergency data (blood type, allergies, vaccination...), surgical operations, prescriptions, insurance data and even links to heavier data (e.g., X-ray examination, scanner images...) stored on hospital servers. Different users may query, modify and create data in the holder's folder: the doctors who consult the patient's past records and prescribe drugs, the surgeons who perform exams and operations, the pharmacists who deliver drugs, the insurance agents who refund the patient, public organizations which maintain statistics or study the impact of drugs correlation in population samples and finally the holder herself.

The benefits of distributing the healthcare database on smartcards than on Centralized database.

- First, health data must be made highly available (anywhere, anytime, on any terminal and without requiring a network connection).
- Second, storing sensitive data on a centralized server may hurt privacy.
- Third, maintaining a centralized database is fairly complex due to the variety of data sources.

Now assuming data is stored in smartcard then the reason why the aforementioned database capabilities need be hosted in the smartcard rather than the terminals because of availability (the data must be exploited on any terminal) and privacy. Regarding privacy, since the data must be confined in the chip, so must be the query engine and the view manager. The smartcard being the unique trusted part of the system, access rights and transaction management cannot be delegated to an untrusted terminal.
The main constraints of current smartcards are therefore:

- the very limited storage capacity;
- the very slow write time in EEPROM (electrically erasable programmable read-only memory);
- the extremely reduced size of the RAM;

- the lack of autonomy and a high security level that must be preserved in all situations.

Why we need DBMS on smart-card

- Multiapplication, versatility
- Improvement of prosessor power
- Volume of data
- Complexity of queries
- Meet Transaction ACID properties

Hardware Limitation of Smart Card:-

- Memory bottleneck (96Kb of ROM, 4Kb of RAM, 128Kb of stable storage like EEPROM)
- EPROM has a very fast read time but very slow write time (10msword)
- Lack of autonomy, no asenchronious and disconnected prosessing.

Solutions to scale-down are:-

- Sysbase Adaptive Server Anywere
- Oracle 8i Lite
- DB2 Anywhere
- ISOL's SQL Java machine DBMS
- SCQL, The standart for smartcard database language

Requirement of Healthcare smart card:-

- Amount of data is significant
- Queries can be rather complex (Ex: doctor asks for the last antibiotic prescribed to the patient)
- Sophisticated access right management (aggregation, views)
- Automicity must be preserved
- Durability is mandatory

Storing database in a smart card rather than centralized database contributes data availability, security and simplicity.

Smart constraints:-

- 32 bir RISC processor about 30 MIPS (monolithic chip).
- Very limited storage capacity
- Very slow write time in EEPROM

- Extremely reduced size of RAM
- Lack of autonomy
- High security level
- Tiny die size (about 25 mm²)
- Low cost (less than 5 dollars)

As the chip size is limited MORE RAM means LESS EEPROM

How Smartcard properties impact on DBMS Architecture

- Highly Secure – access right management
- Highly portable- high availability
- Limited storage resources-memory consumption during execution
- Stable storage is not main main memory
- Non autonomous

Design Components for a PicoDBMS

- Storage manager
- Query manager
- Transaction manager
- Access right manager

Desin criterias for Smartcard DBMS

- RAM Rule
- Write Rule
- Read Rule
- Access Rule
- Security Rule
- Compactness rule (data,index)

PicoDBMS Storage Model

- Existing Storage Models
- Storage Cost Evaluation

Existing Storage Models

- Flat storage
- Domain Storage
- Ring Storage

Flat storage

- Simplest way to organize data

- Tuples are stored sequentially
- Has 2 drawbacks
  - Space consuming
  - Inefficient
- Indexed FS

Domain Storage

- Data Compactness
- The amount of data to be written is much smaller
- The length of the attribute and length of the pointer
- Cardinality of the relation
- All tuples of all relations become fixed size
- Tuple to value pointers

Ring Storage

- Index compactness + data compactness.
- Storing <u>Value to Tuple</u> pointers instead of <u>Tuple to Value</u> Pointers.
- One pointer per domain value whatever the cardinality of the relation is.

Storage cost evaluation

- PicoDBMS storage model combines FS (file system), DS (data structure), and RS.
- Need for indexing attributes - the choice is between RS and Indexed FS.
- No need for indexing attributes- the choice is between FS and DS.

Why PicoDBMS cannot use those state of the art algorithms

- Lifetime of stable memory
- Unable to estimate RAM size
- Those algorithms are complex
- TO SOLVE THIS PROBLEM, WE PROPOSE QUERY PROCESSING TECHNIQUES THAT DO NOT USE ANY WORKING RAM NOR INCUR ANY WRITES IN THE STABLE MEMORY

Query Optimizer can consider different shapes of query execution plan

- Left-deep tree
- Right-deep tree
- Bushy trees
- Extreme right-deep tree (PicoDBMS offered)

PicoDBMS solution

- Aggregate and duplicate removal can be done in pipeline if the incoming tuples are yet grouped by gistinct values
- Pipeline operators are order-preserving since they consume tupls in the arrival order.
- THUS, ENFORCING AN ADEQUATE CONSUMPTION AT THE LEAF OF THE EXECUTİON TREE ALLOWS PIPELIED AGGREGATION AND DUPLICATE REMOVAL

Performance Evaluation

- Reasonable response time is the main concern
- Evaluating response time in samrtcards is complex depending on platform parameters (Processor speed, caching strategy, RAM and EEPROM speed)
- Which acceleration techniques are really mandatory
  - ring indices
  - query optimization

Test Platform

- Intel 486 (with no cache, CISC)
- 3 types of data volumes is considered (Small, medium,large)
- for the platform, the system parameters (clock frequency, cache) are varied and experimentation test is performed) (best/worst configuration)
- Ring indices are tested, how it makes change in the response time

Test Results

- Join indices are mandatory in all classes
- Query optimiztion is useful only with large databases and complex queries

CONCLUSION

As smartcards becomebecome more and more versatile, multiapplications and powerful, the need for database techniques arises. Performance evaluations show that PicoDBMS’s query and execution models meet the smartcard requirments.