

Secure Database Service

Sunny Aggarwal
04329011

December 5, 2004

Abstract

This report explores a novel paradigm for data management in which a third party service provider hosts database as a service, providing its customers seamless mechanisms to create, store, and access their databases at the host site. Such a model alleviates the need for organizations to purchase expensive hardware and software, deal with software upgrades, and hire professionals for administrative and maintenance tasks which are taken over by the service provider, thereby freeing them to concentrate on their core businesses. One of the most critical challenges introduced by database as a service is the provision for a secure database service, which is the focus of this report. Various alternative solutions proposed so far, based on data encryption, and the problems relating to them are surveyed. The limited support for database encryption in commercial products like Oracle is investigated. Further, a model for secure database service, based upon the fundamental concepts of Distributed Databases is detailed. This model tries to counter the problems inherent in solutions based upon database encryption.

1 Introduction

The Internet has made it possible for all computers to be connected to one another. The influence of transaction- processing systems and the Internet ushered in the era of e-business. The Internet has also had a profound impact on the software industry. It has facilitated an opportunity to provide software usage over the Internet, and has led to a new category of businesses called “application service providers” or ASPs. ASPs provide worldwide customers the privilege to use software over the Internet. ASPs are staffed by experts in the art of putting together software solutions, using a variety of software products, for familiar business services such as payroll, enterprise resource planning, supply chain management and customer-relationship marketing. ASPs offer their services over the Internet to small and large worldwide organizations. Since fixed costs are amortized over a large number of users, there is the potential to reduce the service cost even after possibly increased telecommunications overhead.

1.1 Database as a Service

It is possible to provide storage and file access as services. The natural question is the feasibility of providing the next value-add layer in data management. From the business perspective, database as a service inherits all the advantages of the ASP model, indeed even more, given that a large number of organizations have their own DBMSs. The model allows organizations to leverage hardware and software solutions provided by the service providers,

without having to develop them on their own. Perhaps more importantly, it provides a way for organizations to share the expertise of database professionals, thereby cutting the people cost of managing a complex information infrastructure, which is important both for industrial and academic organizations. From the technological angle, the model poses many significant challenges foremost of which is the issue of data privacy and security. In the database-service-provider model, user data resides on the premises of the database-service provider. Most corporations view their data as a very valuable asset. The service provider would need to provide sufficient security measures to guard data privacy. At least two data-privacy challenges arise –

- The first challenge is: how do service providers protect themselves from theft of customer data from hackers that break into their site and scan disks? Encryption of stored data is the straightforward solution, but not without challenges. Trade-offs need to be made regarding encryption techniques and the data granularity for encryption.
- The second challenge is that of “total” data privacy, which is more complex since it includes protection from the database provider itself. The requirement is that under any circumstances the database provider must not be able to infer some substantial information from the data stored in the database.

1.2 Traditional Database Systems

Database systems typically offer access control as the means to restrict access to sensitive data. This mechanism protects the privacy of sensitive information provided data is accessed using the intended database system interfaces. However, access control, while important and necessary, is often insufficient. Attacks upon computer systems have shown that information can be compromised if an unauthorized user simply gains access to the raw database files, bypassing the database access control mechanism altogether. For instance, a recent article published in the Toronto Star [8] describes an incident where a disk containing the records of several hundred bank customers was being auctioned on eBay. The bank had inadvertently sold the disk to the eBay re-seller as used equipment without deleting its contents. Drawing upon privacy legislations and guidelines worldwide, Hippocratic databases also identify the protection of personal data from unauthorized acquisition as a vital requirement [2].

1.3 Secure Database Service

There has been considerable interest in the notion of a “secure database service” : A Database Management System that could that could manage a database without knowing the contents [4]. While the business model is compelling, it is important that such a system be provably secure. Existing proposals have problems in this respect; the security provided leaves room for information leaks. Any method for database encryption that does not meet rigorous cryptography - based standards security must be used carefully. For example, methods that quantize or “bin” values [4] reveal data distributions. Methods that hide distribution, but preserve order [3], can also disclose information if used naively. While they may effectively hide values in isolation, using such techniques on multiple attributes in a tuple can pose dangers. Following is an example of how naive use of prior proposals can disclose sensitive information. Suppose a bank is trying to find who is responsible for missing money (e.g., fraud or embezzlement). They have gathered information on suspect employees and customers. Even though much of the information is publicly known (name, size of mortgage, age, postal code, ...), simply revealing who is being investigated is sensitive: The appearance that you are accusing a customer of fraud could well lead to a libel suit. Therefore they have encrypted each of the values using an order-preserving encryption scheme. Are they protected? The

answer is probably not. Assume a newspaper wants to know if an individual “Chris” is being investigated. They obtain the encrypted database. They know that the name “Chris” would rank at about 15% of all names, so if it appears in the encrypted database, it will be roughly in that position (the range for a given sample size and probability can be calculated using order statistics). The newspaper can do the same with size of mortgage, age, and other known data about Chris and with the other employees/customers of the bank. If there is a tuple in the database whose rank on all attributes is close to the corresponding rank of Chris (in the overall dataset), and there is no other tuple among the customers/employees whose ranks are similar, then the newspaper knows that with high probability, Chris is under investigation.

1.4 Real Threat Model

Before further deliberation on a Secure database Service, it would be better to formally specify a realistic threat model, faced by a Secure Database Service. This model comprises the following assumptions :

- *The storage system used by the database software is vulnerable to compromise.* While current database systems typically perform their own storage management, the storage system remains part of the operating system. Attacks against storage could be performed by accessing database files following a path other than through the database software, or in the ex-treme, by physical removal of the storage media.
- *The database software is trusted.* The database software employed may be curious in the sense it may monitor the data stored inside it, but, it is assumed to not being malicious, so that it carries out all its operations correctly and returns the desired results.
- *All the data in the database is obfuscated.* The data inside the database must be stored in such a format which does not reveal any information about the data distribution.

1.5 Further Layout

Starting from the next section, we investigate the various data encryption based alternatives for Secure Database Service, differing on the granularity and the volume of encryption, the level of query optimization that can be achieved and the level of security that can be achieved. Next, we start deliberating upon a new approach to implementing a Secure Database Service which is based upon the idea of distributed databases. We look towards the advantages this architecture provides and move towards a deeper study of this architecture involving a discussion about the specification and fulfilment of privacy requirements, query processing aspects and the problem of optimal decomposition of a relation into two or more relations, each of which satisfies the privacy requirements and entails minimum overhead on query processing.

2 Secure Database Service-Encryption Alternatives

The basic architecture and the control flow of a Secure Database Service modeled using an encrypted database is shown in Figure 2.

A user poses the query to the client. A server is hosted by the service provider who stores the encrypted database. The encrypted database is augmented with additional information (called index) that allows certain amount of query processing to occur at the server without jeopardizing data privacy. A client stores the data at the server. Client also maintains metadata for translating user queries to the appropriate representation on the server, and

performs post-processing on server query results. Based on the auxiliary information stored, an original query over unencrypted relations is split into –

1. A corresponding query over encrypted relations to run on the server.
2. A client query for post-processing results of the Server query.

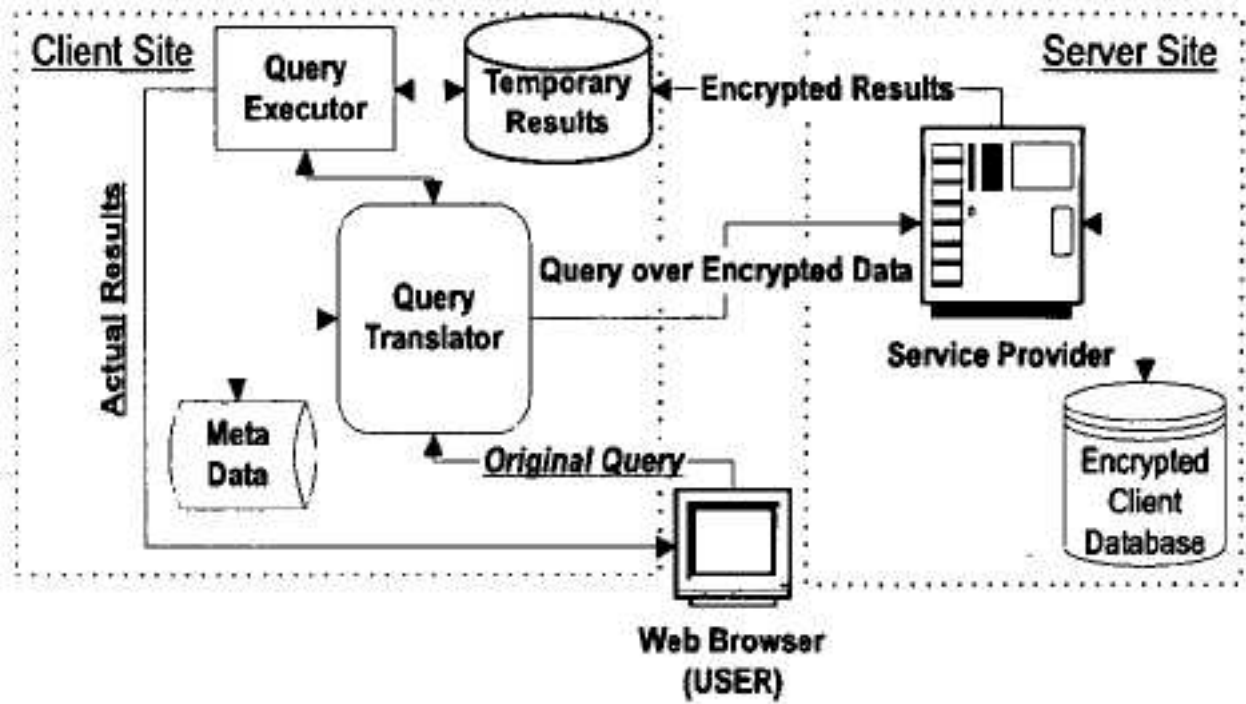


Figure 1: The service-provider architecture

This is done by defining an algebraic framework for query rewriting over encrypted representation. Another interpretation of a database service model [3] using encrypted databases is given in Figure 2. This introduces a trusted database software layer in between the client application and the untrusted storage. This trusted software transparently manages database security, while presenting a simple front-end to the client application.

Apart from this general architecture of a Secure Database Service using encryption, there are various other dimensions to database encryption. These are –

- **Granularity of database encryption** - A relation can be encrypted as a whole, at the field level, row level or the page level. The most naive solution is to encrypt the whole relation as for every access to a tuple, the whole relation must be decrypted. Also the field level encryption, though it seems lucrative, in practice is quite expensive. This is because there is a considerable startup cost associated with each call to an encryption function. With encryption being done at the field level, the no. of items to encrypt increases and so does the start overhead. Also while implementing field level encryption with a block-cipher, the size of the resulting relation may be substantially larger than the original relation since the values in fields being encrypted may be much smaller in size than the block-sizes. So we are left with row-level and page-level encryption alternatives. Out of these page-level is more preferred as it subsumes the row-level encryption, since while reading a row from disk, the whole page is read.

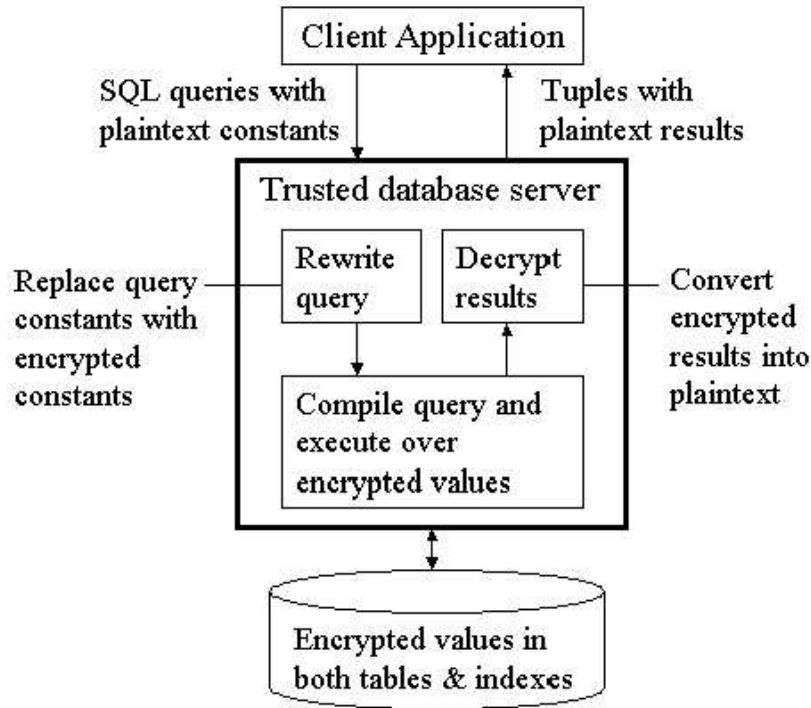


Figure 2: Transparent encryption in the “trusted database software with vulnerable storage” setting

- Query Rewriting to improve Software Encryption** - It is possible to automate the rewrite of a query to use temporary materialized views. A method for common subexpression elimination (CSE) needs to be applied to expensive user defined functions for a query. Common sub-expression detection and elimination are well known in compiler optimization. An occurrence of an expression is a common subexpression (CS) if there is another occurrence of the expression whose evaluation always precedes this one in execution order and if the operands of the expression remain unchanged between the two evaluations.
- Specialized Techniques for executing SQL over Encrypted Data** - The entire domain of an attribute is divided into bins, so that each value of that attribute in the relation can be represented by the identifier of the bin in which the value falls into. This is done for every attribute of interest. This amounts to building sparse indices for each of the attributes. Now this set of index values is associated with the corresponding encrypted tuples. Now, most of the query operations can be carried out on the encrypted database after slight algebraic adaptations. These operations, however produce a superset of the ultimate result. This superset must be decoded and further queried to obtain final answers as is shown in Figure 2.
- Order preserving Encryption for Numeric Data** - The basic idea of OPES is to take as input a user-provided target distribution and transform the plaintext values in such a way that the transformation preserves the order while the transformed values follow the target distribution. Clearly, this scheme does not reveal any information about the original values apart from the order, since the encrypted values were generated solely from the user-specified target distribution, without using any information from the original distribution. Even if an adversary has all of the encrypted values, he cannot infer the original distribution from those values. By appropriately choosing

target distribution, the adversary can be forced to make large estimation errors.

However, the basic assumption here is that, the adversary has no prior knowledge of the domain of the attribute being encoded. This implies that the adversary must not be able to launch known-plaintext attack and may at most launch a ciphertext-only attack.

- **Problems with encrypted databases** - Some of the problems inherent with encrypted databases are -
 1. Extremely difficult to hide data distributions prevalent in unencrypted data.
 2. Privacy-Efficiency Trade-off, i.e., to guarantee stricter levels of privacy, stronger and more comprehensive encryption methodologies have to be applied, which entail extra overhead. If we want efficient database service, then we are forced to define loose levels of privacy. In fact it has been shown [5] impossible to develop a server that meets cryptographic-style definitions of security and is still efficient enough to be practical. However, it is feasible to design a practical system [7] that provides probabilistic security guarantees, as is shown in next few sections.

3 Secure database Service: A Distributed Architecture

The key idea here is to allow the client to partition its data across two, (and more generally, any number of) logically independent database systems that cannot communicate with each other. The partitioning of data is performed in such a fashion as to ensure that the exposure of the contents of any one database does not result in a violation of privacy. The client executes queries by transmitting appropriate sub-queries to each database, and then piecing together the results at the client side.

3.1 Advantages

Among the many advantages offered by the use of a distributed database for implementing a Secure Database Service, some are -

- **Untrusted Service Providers** The client does not have to trust the administrators of either database to guarantee privacy. So long as an adversary does not gain access to both databases, data privacy is fully protected. If the client were to obtain database services from two different vendors, the chances of an adversary breaking into both systems is reduced greatly. Furthermore, “insider” attacks at one of the vendors do not compromise the security of the system as a whole.
- **Provable Privacy** The presence of two databases enables the efficient encoding of sensitive attributes in an information-theoretically secure fashion. To illustrate, consider a sensitive fixed-length numerical attribute, such as a credit-card number. We may represent a credit card number c , by storing c *XOR*ed with a random number r in one database, and storing r in the other database. The set of bits used to represent the credit-card number in either database is completely random, thus providing perfect privacy. However, we may recover the number merely by *XOR*ing the values stored in the two databases, which is more efficient than using expensive encryption and decryption functions.
- **Efficient Queries** The presence of multiple databases enables the storage of many attribute values in unencrypted form. Typically, the exposure of a set of attribute

values corresponding to a tuple may result in a privacy violation, while the exposure of only some subset of it may be harmless. For example, revealing an individual's name and her credit card number may be a serious privacy violation. However, exposing the name alone, or the credit card number alone, may not be a big deal. In such cases, we may place individuals' names in one database, while storing their credit-card number in the other, avoiding having to encrypt either attribute. A consequence is that queries involving both names and credit-card numbers may be executed far more efficiently than if the attributes had been encrypted.

3.2 General Architecture

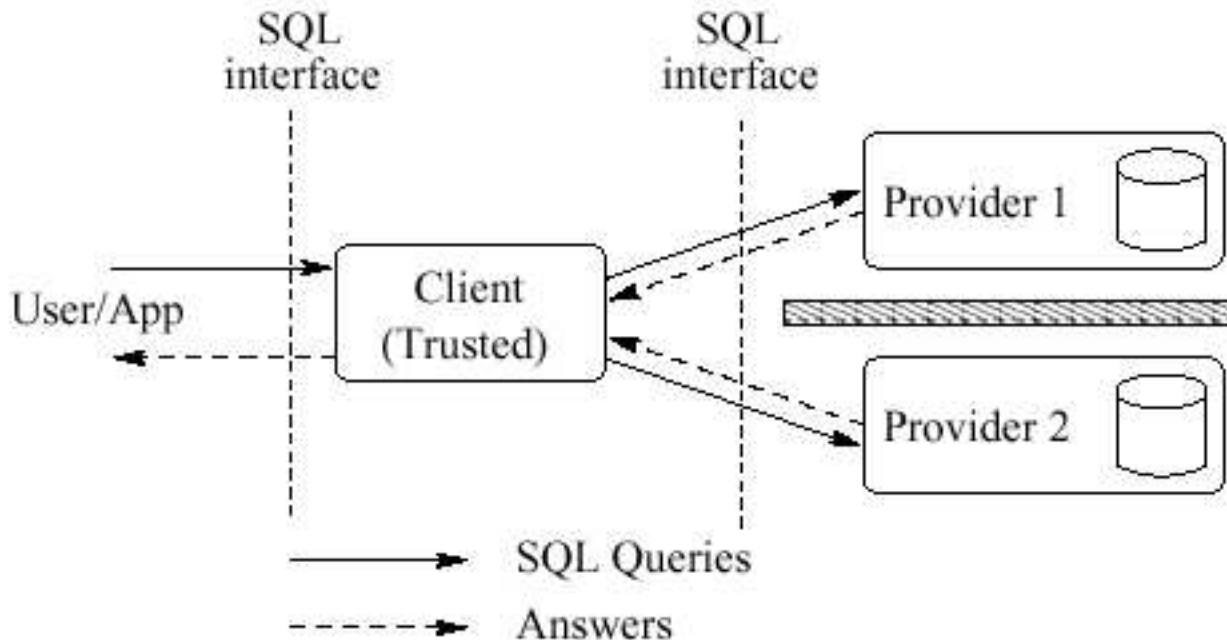


Figure 3: System Architecture

The general architecture of a distributed secure database service, as illustrated in Figure 3.2, consists of a trusted client as well as two or more servers that provide a database service. The servers provide reliable content storage and data management but are not trusted by the client to preserve content privacy. The client wants to out-source the (high) costs of managing permanent storage to the service providers; so the client does not store any persistent data. However, the client has access to cheap hardware - providing processing power as well as temporary storage - which is used to provide -

1. **Offer a DBMS Front-End** The client exports a standard DBMS front-end to client-side applications, supporting standard SQL APIs.
2. **Reformulate and Optimize Queries** The queries received by the client need to be translated into appropriate SQL sub-queries to be sent to the servers; such translation may involve limited forms of query-optimization logic.
3. **Post-process Query Results** The sub-queries are sent to the servers, and the results are gathered and post-processed before being returned to the client-side application.

The basic assumption in this model is that the component servers of the database must not be able to communicate with each other.

3.3 Relation Decomposition

- A universal relation $R(A_1, A_2, \dots, A_1, A_n)$ is to be decomposed and stored at the two servers.
- The decomposition must be :
 - *Lossless* It must be possible to construct R using only the contents in the two servers S_1 and S_2
 - *Privacy Preserving* The contents stored at S_1 and S_2 must not, in themselves, reveal any private information about R .
- Types of Decomposition :
 - *Horizontal Fragmentation* - Server S_1 contains a relation R_1 , and S_2 contains a relation R_2 such that $R = R_1 \cup R_2$.
 - *Vertical Fragmentation* Attributes of R partitioned across S_1 and S_2 , key attributes stored at both sites, some attributes replicated.
- Extensions of Vertical Fragmentation -
 - *Tuple IDs* - A unique ID associated with each tuple of R , is replicated across R_1 and R_2 , so that R_1 and R_2 can be joined in a lossless fashion.
 - *Attribute Encoding* Attributes which must be kept private across both databases need to be encoded. A value 'a' may be encoded into two separate values a_1 and a_2 using various methods, some of which are,
 1. One-time Pad - Provides true information theoretic privacy.
 2. Deterministic Encryption - Key management overhead is reduced. More operations can be pushed down in the query evaluation plan.
 3. Random Addition - Allows efficient execution of aggregate queries.

4 Privacy Requirements Specification and their Fulfilment

- Privacy requirements are specified as a set of privacy constraints P .
- Each privacy constraint is represented by a subset, of the attributes of R and means – Let R be decomposed into R_1 and R_2 , and let an adversary have access to the entire contents of either R_1 or R_2 . For every tuple in R , the value of at least one of the attributes in P must be completely opaque to the adversary, i.e., the adversary should be unable to infer anything about the value of that attribute. It is permissible for the values of some attributes in P to be open, so long as there is at least one attribute completely hidden from the adversary.
- Obtaining Privacy through Decomposition using Vertical Fragmentation – Can be done by ensuring that for every privacy constraint on the relation, all of its elements must not be present together in unencoded form in a single fragment of the relation. The following approach may be adopted –
 1. Ensure that P is not contained in either R_1 or R_2 , using vertical fragmentation. For example, the privacy constraint $\{\text{Name, Salary}\}$ may be obeyed by placing Name in R_1 and Salary in R_2 .
 - 2.

2. Encode at least one of the attributes in P. For example, a different way to obey the privacy constraint $\{\text{Name}, \text{Salary}\}$ would be to use, say, a one-time pad to encode Salary across R1 and R2. Observe that such encoding is the only way to obey the privacy constraint on singleton sets.

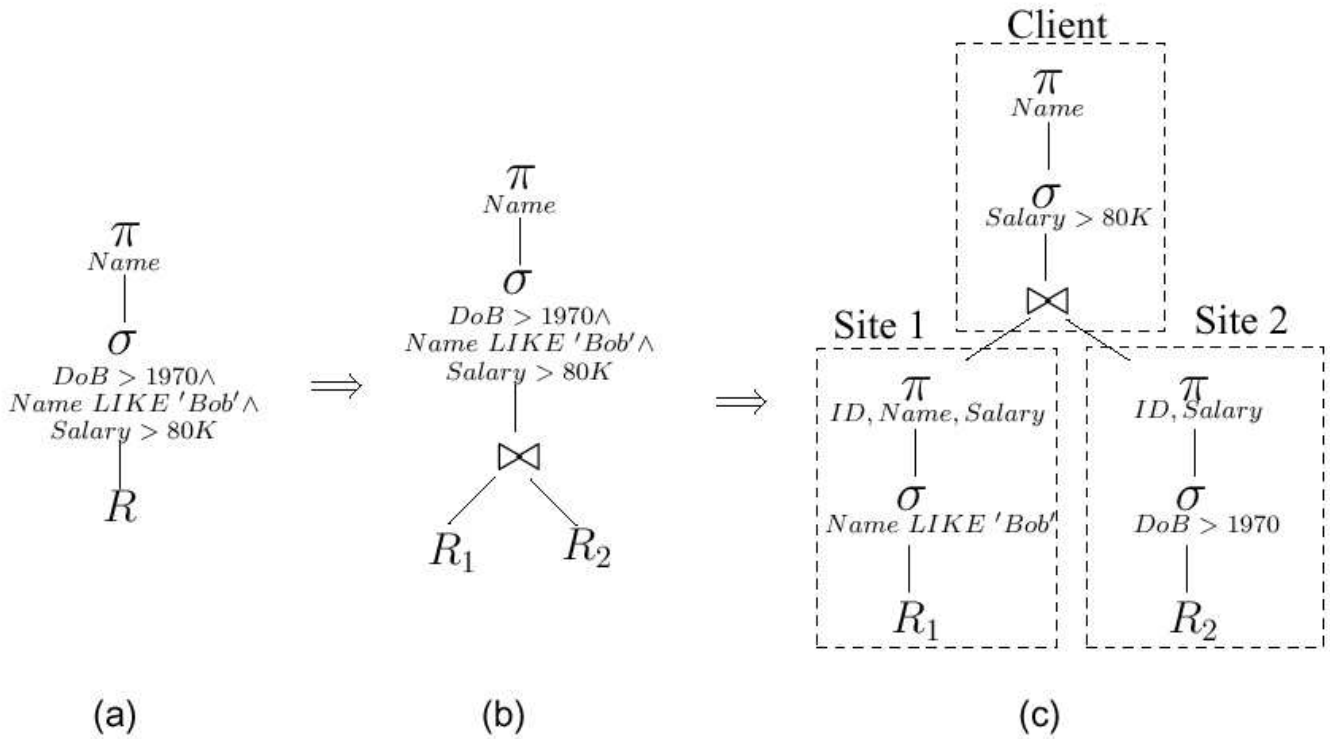


Figure 4: Example of Query Reformulation and Optimization

5 Query Processing

Figure 4 shows a typical query on R reformulated as a join query on R1 and R2..

1. **Query Reformulation** - A logical query plan on R can be translated to a plan on R1 and R2 by replacing every occurrence of R in the plan with $R_1 \bowtie R_2$.
2. **Query Optimization** - If in a condition such as in $\langle Attr \rangle \langle op \rangle \langle value \rangle$ or in $\langle Attr \rangle \langle op \rangle \langle Attr \rangle$, the attributes involved are unfragmented/unencoded and are present in the same fragment of R, the particular selection condition can be pushed down the query evaluation plan. For further optimization the client may obtain statistics from individual fragment sites. When other (deterministic) encoding techniques are used, many more conditions, even on encrypted attributes may be pushed down.

6 Optimal Decomposition of a Relation

- Given a workload of queries and a set of privacy constraints, a near optimal decomposition of R is to be obtained. An optimal Decomposition is one that reduces query

costs.

- Cost of a decomposition of R $D(R)$ is defined as linear combination of ,
 1. Cost of encoding an attribute in R .
 2. Cost of placing unencoded attributes a_1 and a_2 in different fragments.

These cost estimates are calculated based on the overhead they place on computation power and bandwidth of intermediate channel.

- Finding the optimal decomposition can now be formally specified as an optimization problem of finding an optimal two-coloring in a hypergraph.
 - Vertices of the graph are the attributes in R . Encoding an attribute of relation R is equivalent to deletion of a vertex from this hypergraph.
 - Hyper-edges are the policy constraints, with the vertices they connect being the elements of the privacy constraint set. Bicoloring a hyperedge is equivalent to satisfying a policy constraint.
 - However, finding an optimal bicoloring for even restricted instances of a hypergraph is NP-Hard. So, the key is to find a bicoloring of the hypergraph, by representing it as a simple bigraph and then finding a near optimal bicoloring of this graph.
- Heuristics have been defined for finding an approximate solution to the problem, which use the following two solution components
 - Approximate Min-Cuts Used to obtain approx. bi-colorings [6].
 - Approximate Weighted Set Cover To account for vertex deletion.

7 Conclusion

Various factors like definite shift towards outsourcing of data management, escalating concerns about data privacy and recent government legislations have sparked keen interest in Secure database Services.

In, the light of overheads and security issues involved in past solutions proposed for Secure Database Services, most of which have been based upon encrypted databases, the new approach of using a Distributed Database to fulfil privacy constraints shows good promise. Other factors like no requirement for special support from the database software and the foundations of the system being laid upon the tested domain of distributed databases, lend credence to the idea.

However, this model inherits all the disadvantages associated with the Distributed databases paradigm such as complex management, requirement that both the database sites must be up for the functioning of the system, communication overheads etc. Also the model assumes that the underlying network basis is very well managed so that all the sites are totally immune from other sites. The idea that the two sites involved may be different service providers might cause inconsistencies and fluctuations in the overall system.

The optimal decomposition of a relation across two sites involves prior estimation of workload. In a real system, the character of workload might change swiftly, so the performance may go down and a re-decomposition will be required.

However with assurances of robust networks and systems, this model with further refinements can go a long way.

References

- [1] *Proceedings of the 28th VLDB Conference*. ACM SIGMOD, 2004.
- [2] Rakesh Agrawal et. al. Hippocratic databases. [1].
- [3] Rakesh Agrawal et. al. Order preserving of numeric data. ACM SIGMOD ICMD 2004, 2004.
- [4] Sharad Mehrotra et. al. Executng sql over encrypted data in the database-service-provider model. Technical report, University of California, IBM Silicon Valley Lab, 2002.
- [5] Chris Clifton et.al. Security issues in queying encrypted data. Technical report, Purdue University, 2004.
- [6] David Karger et.al. A new approach to the minimum cut problem. *Journal of th ACM*, 1996.
- [7] Rajeev Motwani et.al. Two can keep a secret: A distributed architecture for secure database services. Technical report, Stanford University, 2004.
- [8] T. Hamilton. Error sends bank files to ebay. September 15, 2003.