QUERY EXECUTION

cho một câu query Q, mục đích là chia nhỏ việc tính toán Q thông qua server và client. Server sẽ được dùng để hiện thực hiện các toạn hạng quan hệ của câu Query Q càng nhiều càng tốt, phần còn lại của việc tính toán câu Query được hiện thực tại client. Mục tiêu là tạo ra một "query plan" tốt nhất cho Q để tối thiểu việc thực hiện ở bên phía client. Trong việc thiết lập này, chi phí của một query bao gồm nhiều thành phần- Chi phí I/0 và CPU tại server, chi phí truyền tải giữa client và server, và chi phí I/O và CPU tại client. Một ví dụ, xem xét một câu query thông qua bảng "emp", lấy tất của các nhân viên có thu nhập lớn hơn thu nhập trung bình của nhân nhiên trong phòng ban có did =1.

SELECT emp.name FROM emp WHERE emp.salaray > (SELECT AVG(salary) FROM emp WHERE did =1 ).

Cây Query ứng với câu query trên và vài đánh giá về các chiến lược được minh họa từ Figure 3 đến 6.

Chiến lược đầu tiên ( hình 4) là đơn giản ( giảm thiểu) việc truyền tải bảng emp đến client.

Chiến lược thứ hai ( hình 5) là tính toán một phần câu query tại phía server, chọn các bộ dữ liệu tương ứng với điều kiện Mapcond(did =1) nhiều nhất có thể, sau đó server gửi tới client một bản mã hóa của table emp kèm theo một bản mã hóa thể hiện các tập dữ liệu mà thõa câu query bên trong ( câu query chọn lương của nhân viên trong phòng ban có did =1). Client sẽ mã hóa các bộ dữ liệu và bảng emp và làm các công việc còn lại ( tính toán giá trị trung bình và so sánh).

Chiến lược thứ ba ( Hình 6) là thực hiện câu query ở bên trong tại server, nghĩa là việc lựa chọn các nhân viên làm việc tại phòng ban có did = 1 được thực thiện tại phía server. Kết quả được truyền cho client, client sẽ giải mã và tính toán giá trị trung bình. Giá trị trung bình được mã hóa tiếp tục và gửi lại cho server, sau đó việc lựa chọn những nhân viên có lương lớn hơn trung bình được thực hiện tại server và trả kết quả đạt được về cho client, cuối cùng client chỉ việc giải mã thông tin được nhận.

HỖ TRỢ TOÁN TỬ TỔNG HỢP ( AGGREGATION) TRONG CÂU QUERY:

Có nhiều công nghệ để dịch một câu query được thảo luận ở trên được thiết kế cho các toán tử quan hệ để thực hiện việc so sánh. Trong khi kỹ thuật " ẩn thông tin " làm việc được với các toán tử quan hệ, nó không làm việc được với các toán tử đại số ví dụ như toán tử tổng hợp (aggregration). Chú ý rằng trong câu query trước, có một phép tổng hợp nhưng phép tổng hợp này được thực hiện ở phía client sau khi đã giải mã dữ liệu. Nếu phép tổng hợp này được thực hiện tại phía server, kỹ thuật " ẩn thông tin " phải được hỗ trợ bởi một phương pháp mã hóa mà hỗ trợ các toán tử đại số trên dữ liệu được mã hóa,[27] minh họa cho việc làm thế nào mà PH có thể kết hợp với cách tiếp cận cơ bản để đáp ứng cho mục đích trên. Ngoài ra độ phức tạp tăng lên khi kỹ thuật "ẩn thông tin" không xác định chính xác nhóm để sử dụng toán tử tổng hợp ( phía server sẽ chứa những kết quả không chính xác). paper [39, 16] phát triển một kỹ thuật thao tác trên các biểu thức đại số là chia một toán tử tổng hợp làm 2 tập con, một tập chứa những dữ liệu chắc chắn thõa mãn điều kiện trong câu query, và một tập chữa dữ liệu không chắc chắn là thõa mãn điều kiện trong câu query. Tập con đầu tiên có thể được thực hiện phép tổng hợp trực tiếp tại server thông qua việc sử dụng PH trong khi tập con thứ 2 cần phải truyền cho client để xác định lại dữ liệu nào là không thõa mãn câu query.

TỐI ƯU CÂU QUERY TRONG DAS:

Như một thực hiện một câu query truyền thống, trong DAS cũng có nhiều cách hiện thực mà tương đương nhau cho cùng một câu query. Điều này nảy sinh vấn đề tối ưu hóa cho câu query. Trong tài liệu [24], việc tối ưu hóa câu query trong DAS được hình thành dựa trên việc tối ưu dựa trên chi phí thực hiện bằng cách đưa ra những hàm mới để xử lý cho câu query và những luật mới để chuyển hóa một câu query.

Với những ràng buộc phần cứng khác nhau và khả năng phần mêm tại phía client và việc đo đạc các chi phí khác nhau được áp dụng tại phía client và phía server. Một "query plan" mới được phát triển để giảm thiểu chi phí thực hiện.

2.4 KEYWORD SEARCH ON ENCRYPTED TEXT DATA

Một kỹ thuật chính được đề xuất trong

Alice có một tập tài liệu dạng văn bản D = {D1, .., Dn}. Một tài liệu Di được hình thành từ tập các từ khóa Di = {W1D, ..., WnD}, mỗi từ w thuộc W, trong đó (W) là tập các từ khóa có thể. Alice lưu trữ tập tài liệu của mình tại một nhà cung cấp dịch v. Khi nhà cung cấp dịch vụ là không tin cậy, các tài liệu sẽ được mã hóa. Mỗi tài liệu được mã hóa tại "mức từ" (word level) như sau: mỗi tài liệu được chia ra thành các từ bằng nhau. Điển hình là các từ tương ứng với ngôn ngữ là tiếng anh sẽ được thêm vào các bit 0 và 1 để cho các từ này có chiều dài bằng nhau. Alice có thể đưa ra một câu truy vấn đến server để lấy một phần dữ liệu cần thiết một cách định kỳ. Câu truy vấn này là một tập các từ khóa và câu trả lời phản hồi lại là một tập các tài liệu có chưa tất cả các từ khóa trong câu truy vấn. Tổng quát hơn, một câu trả lời cho một câu truy vấn được cho bởi:

Ans(q) = {Di ϵ D | với mọi kj ϵ q, kj ϵ D }

Mục đích của kỹ thuật thiết kế này là lấy được câu trả lời mà không tiết lộ bất cứ thông tin về từ khóa ( của câu truy vấn ) trong mỗi tài liệu.

PRIVATE KEY BASE SEARCH SCHEME ON ENCYPTED TEXT DATA

Alice muốn lưu trữ dữ những tài liệu của mình với Bob (Bob là nhà cung cấp dịch vụ). Alice mã hóa mỗi tài liệu D trước khi lưu trữ với Bob. Ngoài ra, alice tạo một chỉ mục bí mật (secure index), I(D), được lưu trữ tại nhà cung cấp dịch vụ để giúp cho việc thực hiện việc tìm kiếm dựa trên các từ khóa. Chỉ mục bí mật này sẽ không tiết lộ bất kỳ thông tin gì về nội dung của tài liệu cho kẻ thù. Tuy nhiên, nó có thể cho phép kẻ thù có kiểm tra sự có mặt hay vắn mặt của từ khóa bằng cách sử dụng "trapdoor" liên quan đến từ khóa đó khi mà một "trapdoor" được tạo ra với một khóa bí mật mà của người sở hữu tài liệu. Một người dùng mong muốn tìm kiếm một tài liệu có chứa từ khóa w, tạo một "trapdoor" cho w mà có thể được dùng bởi kẻ thù để lấy các tài liệu thích hợp.