GIẢI PHÁP KIỂM THỬ ĐỘT BIẾN CÁC CÂU LỆNH TRUY VẤN CƠ SỞ DỮ LIỆU

A SOLUTION TO THE MUTATION TESTING OF SQL DATABASE QUERRIES

Nguyễn Thanh Bình
Trường Đại học Bách khoa, Đại học Đà Nẵng

Hồ Văn Phi
Trường Cao đẳng CNTT Hữu nghị Việt-Hàn

TÓM TẮT

Kiểm thử đột biến (mutation testing) là một trong những kỹ thuật kiểm thử cấu trúc được sử dụng phổ biến, nhờ vào khả năng tự động hóa cao làm giảm chi phí của tiến trình kiểm thử. Kiểm thử đột biến đã được áp dụng rộng rãi cho các ngôn ngữ lập trình khác nhau. Trong bài báo này, chúng tôi trình bày giải pháp hỗ trợ kiểm thử đột biến áp dụng cho các câu lệnh truy vấn cơ sở dữ liệu. Một công cụ được xây dựng hỗ trợ việc tự động hóa kiểm thử đột biến các câu lệnh truy vấn. Công cụ được thử nghiệm trên một số lớn các ứng dụng sử dụng các lệnh truy vấn cơ sở dữ liệu cho kết quả khả quan.

ABSTRACT

Due to the automation and cost reduction of testing process, mutation testing is one of the structural testing methods that are popularly used. Mutation testing is applied to many programming languages. In this paper, we present a solution for mutation testing applied to SQL database queries. A tool is constructed for automating the mutation testing process of SQL database queries. The solution is applied to some SQL database queries and the results are very promising.

1. Đặt vấn đề

Kiểm thử là một trong những hoạt động quan trọng trong tiến trình phát triển phần mềm. Nó góp một phần rất lớn trong việc đánh giá chất lượng của một phần mềm và là qui trình bắt buộc trong các dự án phát triển phần mềm trên thế giới cũng như trong nước. Có rất nhiều kỹ thuật kiểm thử phần mềm được phát triển, trong đó, kiểm thử đột biến là kỹ thuật hỗ trợ việc đánh giá chất lượng của bộ dữ liệu kiểm thử.

Hiện nay, nhu cầu sử dụng ngôn ngữ vấn tin có cấu trúc (SQL) trong các đơn vị nghiên cứu và phát triển phần mềm ngày càng cao. Do đó, có được một hệ thống ứng dụng kiểm thử đột biến để đánh giá chất lượng các bộ dữ liệu kiểm thử khi thực hiện kiểm thử các câu lệnh truy vấn SQL thực sự là cần thiết.

Trong bài báo này, chúng tôi tập trung vào nghiên cứu đột biến các toán tử trong câu lệnh truy vấn SQL và xây dựng công cụ kiểm thử đột biến cho các câu lệnh truy vấn SQL.

Nội dung chính của bài báo như sau: phần 2 giới thiệu ngắn gọn kỹ thuật kiểm thử đột biến; phần 3 trình bày kỹ thuật đột biến câu lệnh truy vấn SQL; phần 4 trình bày

các bước xây dựng công cụ kiểm thử đột biến cho các câu lệnh truy vấn SQL; phần 5 trình bày kết quả thử nghiệm và một số thảo luận; cuối cùng là kết luận.

2. Kỹ thuật kiểm thử đột biến

Kỹ thuật kiểm thử đột biến do Demillo đề xuất đầu tiên vào năm 1978 [1]. Kiểm thử đột biến được thiết kế để hỗ trợ các kiểm thử viên tạo ra một bộ dữ liệu kiểm thử có hiệu quả. Kiểm thử đột biến có chức năng quan trọng là làm lộ ra lỗi của chương trình. Kiểm thử đột biến được xây dựng căn cứ vào hai giả thuyết cơ bản [3]: lập trình viên giỏi và hiệu ứng liên kết. Một trình viên giỏi được giả thuyết là chỉ gặp phải những sai sót đơn giản, chứ không gặp phải các sai sót nghiêm trọng, ví dụ sai sót về thuật toán. Hiệu ứng liên kết giả thuyết rằng nếu dữ liệu thử phát hiện các lỗi đơn giản thì cũng phát hiện được các lỗi phức tạp.

Kỹ thuật kiểm thử đột biến câu lệnh truy vấn SQL giúp cho kiểm thử viên tạo dữ liệu kiểm thử và đánh giá chúng đầy đủ bằng cách chèn một cách có hệ thống các lỗi nhân tạo vào trong câu lệnh cho trước và đánh giá tỷ lệ lỗi được phát hiện bởi bộ dữ liệu kiểm thử.

Đột biến được tạo ra bằng cách thay đổi câu lệnh gốc sử dụng một tập hợp các luật (toán tử đột biến) được định nghĩa sẵn để gây ra sự thay đổi cú pháp dựa trên các lỗi mà các lập trình viên thường mắc phải.

Mỗi đột biến được thực thi cùng với một bộ dữ liệu và khi chúng sản sinh ra một kết quả đầu ra không đúng (đầu ra khác với đầu ra của câu lệnh gốc), đột biến đó được gọi là *đột biến bị diệt*. Một trường hợp kiểm thử được gọi là hiệu quả nếu chúng diệt được các đột biến mà chưa bị diệt bởi các trường hợp kiểm thử trước đó. Một số đột biến luôn cho ra cùng một kết quả đầu ra so với câu lệnh gốc, vì vậy không có trường hợp kiểm thử có thể diệt chúng. Các đột biến này được gọi là *đột biến tương đương*. Sau khi thực thi một trường hợp kiểm thử trên một số lượng đột biến, *tỷ lệ đột biến* được định nghĩa bằng tỷ lệ phần trăm của các đột biến bị diệt chia cho số lượng đột biến không tương đương.

Kiểm thử đột biến có thể dễ dàng tích hợp vào trong hệ thống mà đột biến được sinh tự động và thực thi. Sản sinh các trường hợp kiểm thử diệt các đột biến có thể được thực hiện thủ công hoặc tự động bằng cách sử dụng một tập ràng buộc bộ sinh trường hợp kiểm thử.

3. Kỹ thuật đột biến câu lệnh truy vấn SQL

Các toán tử đột biến được tổ chức theo các nhóm [4] được định nghĩa bằng hai ký tự đầu in hoa.

- Đột biến cho các mệnh đề chính (SC - SQL Clause).
- Đột biến cho các toán tử trong các biểu thức và điều kiện (OR – Operator replacement).
- Đột biến quan hệ để xử lý giá trị NULL (NL – NULL).

- Thay thế các định danh: cột tham chiếu, tham số và hằng số (IR – Identifier replacement).

Hầu hết các toán tử có thể áp dụng trong các mệnh đề SQL khác nhau, mỗi nhóm được chia thành các nhóm con nhỏ hơn, mỗi nhóm tham chiếu đến một kiểu đột biến tiêu biểu khi nó áp dụng cho các mệnh đề (select, join, where, group by, having và order by).

3.1. Các toán tử đột biến mệnh đề chính (SC)

Các toán tử SC góp phần phát hiện một số lỗi như liên kết không đúng, sử dụng phép lượng hóa (distinct) không đúng có thể dẫn đến sự có mặt của các hàng trùng nhau không mong muốn, tính toán các hàm gộp nhóm không đúng hay không đúng thứ tự trong tập kết quả.

Mệnh đề lựa chọn (SEL – SELECT): Mỗi sự xuất hiện của một từ khóa SELECT hay SELECT DISTINCT được thay thế bằng một từ khóa khác trong số đó.

Mệnh đề liên kết (JOI –JOIN): Mỗi sự xuất hiện của một từ khóa kiểu liên kết (INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, OUTER JOIN, CROSS JOIN) được thay thế bằng các từ khóa kiểu liên kết khác. Khi một kiểu liên kết được thay thế bằng CROSS JOIN, điều kiện tìm kiếm sau từ khóa ON bị loại bỏ. Khi CROSS JOIN được thay thế bằng kiểu liên kết khác, một mệnh đề ON được thêm vào và điều kiện liên kết tương ứng của nó được tạo ra dựa trên các khóa của các bảng liên kết.

Grouping (GRU): Mỗi biểu thức GROUP BY được loại bỏ. Nếu biểu thức loại bỏ trong mệnh đề GROUP BY có trong danh sách SELECT hoặc trong danh sách ORDER BY, biểu thức này phải được đặt trong hàm gộp nhóm để tránh lỗi cú pháp câu lệnh truy vấn. Trong trường hợp này, hai đột biến được sinh ra cho mỗi biểu thức, một sử dụng hàm MIN và một sử dụng hàm MAX. Nếu chỉ có một biểu thức GROUP BY thì toàn bộ mệnh đề được loại bỏ.

Hàm gộp nhóm (AGR): Mỗi sự xuất hiện của một trong các hàm gộp nhóm (MIN, MAX, AVG, AVG(DISTINCT), SUM, SUM(DISTINCT), COUNT, COUNT(DISTINCT)) trong danh sách SELECT hoặc danh sách HAVING được thay thế bởi một trong các hàm khác. Sự thay thế đó phải đảm bảo kiểu dữ liệu tính toán được của hàm gộp nhóm. Nếu kiểu dữ liệu là kiểu ký tự thì hàm AVG và hàm SUM bị loại ra khỏi sự thay thế. Nếu kiểu dữ liệu là ký tự và hàm gộp nhóm sau mệnh đề HAVING thì hàm COUNT không được thay đổi nếu nó tham gia vào sự so sánh với một biểu thức số học.

Truy vấn nối (UNI): Mỗi sự xuất hiện của một trong những từ khóa (UNION, UNION ALL) được thay thế bởi một từ khóa khác. Mỗi truy vấn trong truy vấn hợp được loại bỏ.

Sắp xếp kết quả (ORD): Mỗi sự xuất hiện của biểu thức ORDER BY được thay thế bởi các từ khóa khác (ASC, DESC). Nếu không có từ khóa nào (mặc định là ASC) thì từ khóa DESC được thêm vào. Loại bỏ mệnh đề ORDER BY nếu chỉ có một biểu thức hoặc thay thế mỗi cặp gần nhau.

3.2. Toán tử thay thế (OR)

Mục tiêu của các đột biến là phát hiện các lỗi lô-gíc trong các mệnh đề WHERE và HAVING.

Thay thế các toán tử quan hệ (ROR - Relational operator replacement): mỗi sự xuất hiện của một trong các toán tử quan hệ {=, <>, <, <=, >, >=} hoặc được thay thế bởi một toán tử khác hoặc được thay thế bởi falseop (luôn luôn trả về giá trị sai) hoặc được thay thế bởi trueop (luôn luôn trả về giá trị đúng).

Toán tử lô-gíc (LCR - Logical connector operator): Mỗi sự xuất hiện của một trong các toán tử logic (AND, OR) được thay thế bởi một trong các toán tử khác, hoặc bởi falseop hoặc trueop hoặc leftop (trả về toán hạng bên trái) hoặc rightop (trả về toán hạng bên phải).

Thay thế toán tử số học (AOR - Arithmatic Operator Replacement): Mỗi toán tử số học {+, -, *, /, %} được thay thế bởi một trong toán tử số học khác hoặc các toán tử leftop và rightop được áp dụng cho biểu thức toán học.

Vị từ Between (BTW - Between): Mỗi điều kiện trong dạng thức BETWEEN x AND y được thay thế bởi a>x AND a<=y và a>=x AND a<y. Nếu điều kiện là NOT BETWEEN thì đột biến được phủ định.

Vị từ Like (LIKE - Like): Khả năng kết hợp các chuỗi điều kiện trong dạng thức a LIKE s là vô hạn, từ đó s là một mẫu tìm kiếm. Do đó các đột biến sẽ được hạn chế để thực thi hành vi của các ký tự đại diện {%, _}. Mỗi sự xuất hiện của ký tự đại diện được biến đổi bằng cách loại bỏ ký tự đại diện, hoặc thay thế ký tự đại diện bằng ký tự khác, hoặc loại bỏ ký tự trước ký tự đại diện nếu nó không phải là ký tự đầu tiên của s, hoặc loại bỏ ký tự phía sau ký tự đại diện nếu không phải là ký tự cuối cùng của s. Nếu không có ký tự đại diện tại vị trí bắt đầu của s, thì thêm mỗi ký tự đại diện tại vị trí bắt đầu và nếu không có ký tự đại diện tại vị trí cuối cùng của s thì thêm mỗi ký tự đại diện tại vị trí cuối cùng.

3.3. Thay thế định danh (IR)

Toán tử IR thay thế các định danh cột, hằng và tham chiếu đến các tham số truy vấn, vì vậy chúng có khả năng phát hiện các lỗi như sử dụng không đúng các trường. Thay thế cột tham chiếu trong các truy vấn có chứa truy vấn con, gộp nhóm hay truy vấn kết hợp chỉ hạn chế thay thế các cột nằm trong phạm vi của mệnh đề chứa cột được thay thế.

Thay thế cột (IRC – Column replacement): Mỗi cột tham chiếu được thay thế bằng một trong các cột tham chiếu khác, hằng số và các tham số trình bày trong truy vấn và kiểu phù hợp.

Thay thế hằng số (IRT – Constant replacement): Mỗi hằng số được thay thế bằng một trong các hằng số khác, cột và các tham số trình bày trong truy vấn và kiểu phù hợp.

Thay thế tham số (IRP – Parameter replacement): Mỗi tham số được thay thế bằng một trong các tham số khác, cột và các hằng số trình bày trong truy vấn và kiểu phù hợp.

Thay thế cột ẩn (IRH – Hidden column replacement): Mục tiêu của toán tử này là phát hiện các lỗi tiềm ẩn khi nhiều cột tương tự nhau xuất hiện trong cùng một bảng và các trường hợp kiểm thử không đủ các loại giá trị khác nhau để phát hiện việc sử dụng sai tên cột. Mỗi thuộc tính cột tham chiếu được thay thế bằng một cột khác được định nghĩa trong bảng cung cấp của nó.

4. Xây dựng công cụ kiểm thử đột biến

Công cụ kiểm thử đột biến cho các câu lệnh truy vấn SQL được viết bằng ngôn ngữ lập trình C#. Trong bài báo này chúng tôi áp dụng kỹ thuật đột biến chọn lựa (selective mutation) vào hệ thống này.

4.1. Giải pháp

Trên cơ sở phân tích các đột biến câu lệnh truy vấn cơ sở dữ liệu SQL ở phần trên, chúng tôi đề xuất thuật toán xây dựng công cụ kiểm thử đột biến cho câu lệnh SQL ở dưới.

Bước 1. Nhận câu lệnh SQL vào, kiểm tra, phân tích cú pháp câu lệnh và lưu dưới dạng cây cú pháp XML.

Bước 2. Thực hiện đột biến cho câu lệnh kết quả tạo ra tập hợp các đột biến của câu lệnh SQL gốc.

Bước 3. Thực hiện vòng lặp gồm các bước sau:

a. Thực thi câu lệnh SQL đột biến.

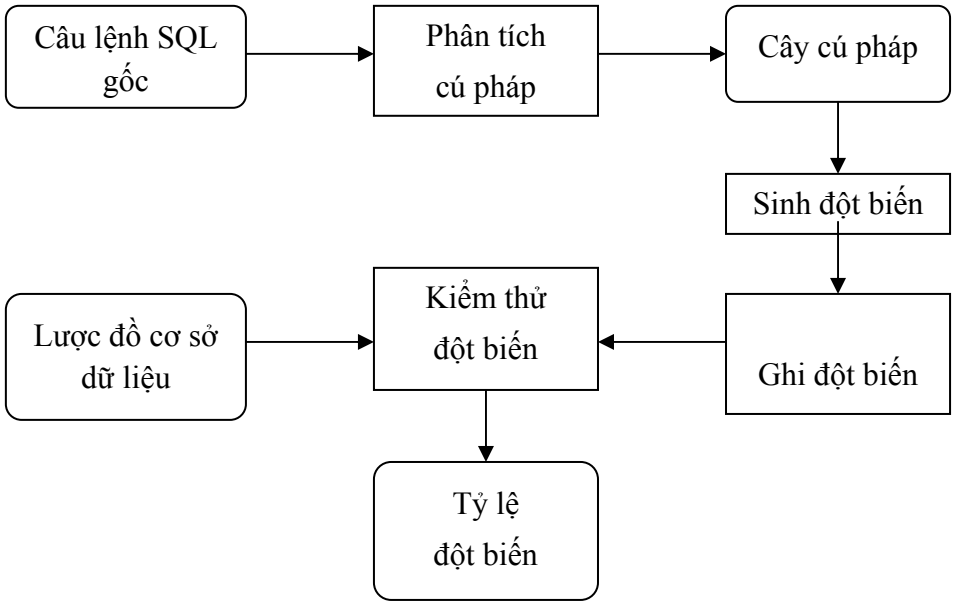
b. So sánh kết quả thực thi câu lệnh đột biến và kết quả thực thi câu lệnh gốc.

c. Nếu kết quả khác nhau thì đột biến bị diệt, nếu ngược lại sẽ đánh dấu câu lệnh đột biến và xem xét tính tương đương câu câu lệnh đột biến với câu lệnh gốc.

Bước 4. Kết thúc quá trình, xuất ra báo cáo sơ bộ gồm các thông tin về số lượng đột biến sinh ra, số lượng đột biến bị diệt và tỷ lệ đột biến.

4.2. Kiến trúc của công cụ

Cốt lõi của hệ thống này là bộ sản sinh đột biến, nghĩa là tạo ra các đột biến cho hệ thống. Bộ sinh đột biến được hỗ trợ bởi lược đồ cung cấp thông tin về các phần tử trong lược đồ cơ sở dữ liệu. Bộ phân tích chuyển câu lệnh truy vấn SQL sang cú pháp phù hợp với bộ sinh đột biến. Bộ ghi đột biến ghi lại các đột biến được sản sinh ra và trả về cho ứng dụng theo đúng định dạng mà ứng dụng xử lý được. Hệ thống có thể được sử dụng tích hợp vào các ứng dụng khác hoặc được sử dụng độc lập để kiểm thử.



***Hình 1**. Kiến trúc của công cụ kiểm thử đột biến cho các câu lệnh truy vấn SQL.*

4.3. Lược đồ cơ sở dữ liệu, bộ phân tích và bộ ghi đột biến

Lược đồ cơ sở dữ liệu được cung cấp dưới dạng các bảng của cơ sở dữ liệu quan hệ. Quá trình thực hiện đột biến yêu cầu người sử dụng phải cung cấp cả lược đồ cơ sở dữ liệu (tên bảng, tên cột, kiểu dữ liệu, khóa chính, khóa ngoại,…).

Bộ phân tích chuyển câu lệnh truy vấn SQL gốc sang định dạng tài liệu XML bằng cách thay thế các từ khóa của câu lệnh bằng các phần tử XML và tổ chức lại tài liệu XML theo thứ tự phù hợp với cấu trúc của câu truy vấn SQL. Các từ khóa được biểu diễn bởi các phần tử và các tên cột, tên bảng, tham số và các hằng số được biểu diễn bởi thuộc tính text của các phần tử XML. Ví dụ, sau đây là một câu lệnh truy vấn được biểu diễn bằng tài liệu XML:

```
<sql><select> firstname</select>
<from>EMP</from>
<where> empid <eq/> ‘101’ </where> </sql>
```

Trong khi bộ sinh đột biến sản sinh ra mỗi đột biến thì nó sẽ gọi thực thi bộ ghi đột biến để lưu lại đột biến đó. Các đột biến được lưu trong một tệp tin tài liệu XML.

4.4. Bộ sinh đột biến

Bộ sinh phân tích nhận đầu vào là lược đồ cơ sở dữ liệu và câu lệnh truy vấn SQL để phân tích câu lệnh truy vấn và chuyển các tài liệu XML của câu lệnh truy vấn vào mô hình DOM. Bộ sinh đột biến sẽ duyệt qua các phần tử trong DOM và khi bắt gặp một phần tử hoặc thuộc tính text của một nút thì thực hiện một trong các thao tác sau:

- *Thiết lập phạm vi:* Các cột tham chiếu phải tương ứng với các cột trong bảng

của lược đồ cơ sở dữ liệu đã được khai báo. Khi gặp mệnh đề SELECT hoặc mệnh đề JOIN, danh sách các bảng trong phạm vi nút đó phải được cập nhật. Hơn nữa danh sách các cột, tham số, hằng số cũng phải được thay thế theo cơ chế đột biến IR.

- *Chọn cột:* Khi bắt gặp thuộc tính text của một nút thì lược đồ phải xác định đó là một tham số, hằng số hay một cột tham chiếu. Nếu là cột tham chiếu thì bảng chứa cột tham chiếu phải có trong danh sách bảng. Nếu một bảng xuất hiện nhiều hơn một lần thì bảng phải được đặt lại tên mới.
- *Đột biến:* Nếu một nút là một phần tử hoặc một thuộc tính text có thể sinh ra một hoặc nhiều đột biến thì bộ sinh đột biến sẽ thực hiện đột biến và gọi thủ tục ghi đột biến để lưu lại đột biến đó. Sau khi quá trình sinh đột biến kết thúc, chúng ta có được tập hợp các đột biến của câu lệnh truy vấn.

5. Kết quả thử nghiệm

Công cụ đã được áp dụng trên một tập các lệnh truy vấn cơ sở dữ liệu. Bảng 1 trình bày một số kết quả thử nghiệm.

Bảng 1. *Một số kết quả thử nghiệm.*

Trường hợp kiểm thử	Tổng số đột biến	Số đột biến không diệt được	Tỷ lệ đột biến
SELECT deptid, count(empid) FROM emp where deptid = '50' group by deptid having count(empid) > 2	75	6	92%
SELECT deptid, count(empid) FROM emp where deptid = '50' group by deptid having max(empid) > 2	75	6	92%
SELECT deptid, count(empid) FROM emp where deptid = '50' OR deptid = '60' group by deptid	77	10	87%
SELECT deptname, count(empid) FROM emp inner join dept on emp.deptid = dept.deptid group by deptname having count(empid) > 2	75	0	100%
SELECT empid, firstname, deptname FROM emp inner join dept ON emp.deptid=dept.deptid WHERE empid > 20	25	0	100%
SELECT empid, firstname, deptname FROM emp, dept WHERE emp.deptid=dept.deptid AND firstname like 'Jonh'	45	5	89%

SELECT empid, firstname, deptname FROM emp, dept WHERE emp.deptid=dept.deptid AND deptname = 'IT'	39	1	97%
SELECT empid, firstname, deptname FROM emp, dept WHERE emp.deptid=dept.deptid AND empid > 30	36	4	89%
SELECT empid, jobtitle FROM emp, job WHERE Jobtitle = 'IT' AND emp.jobid = job.jobid	36	4	89%
SELECT deptname, count(empid) FROM emp, dept where emp.deptid = dept.deptid group by deptname having count(empid) > 5	54	21	61%

Như vậy, trong trong quá trình kiểm thử thực thi, tỷ lệ đột biến khi áp dụng cho mỗi câu lệnh có sự khác nhau, cao nhất 100% và thấp nhất 61%. Tỷ lệ đột biến trung bình xấp xỉ 90%. Qua một số trường hợp kiểm thử chứng tỏ chất lượng bộ dữ liệu kiểm thử là chưa cao, vì còn có rất nhiều trường hợp kiểm thử không diệt được tất cả các đột biến. Như vậy, giải pháp đã đưa ra cảnh báo kịp thời để kiểm thử viên xem xét và xây dựng lại các trường hợp kiểm thử và dữ liệu kiểm thử tốt hơn để đảm bảo chất lượng của phần mềm.

6. Kết luận

Trên cơ sở phân tích các đột biến câu lệnh truy vấn cơ sở dữ liệu SQL, chúng tôi xây dựng được công cụ kiểm thử đột biến có thể áp dụng cho nhiều ứng dụng sử dụng các lệnh truy vấn cơ sở dữ liệu cho kết quả khả quan. Công cụ kiểm thử đột biến có thể cung cấp một giải pháp sơ bộ ứng dụng kiểm thử đột biến để làm cơ sở tham khảo và ứng dụng thực tế cho các đơn vị phát triển phần mềm để nâng cao chất lượng kiểm thử.

TÀI LIỆU THAM KHẢO

[1] A.T. Acree, T.A. Budd, R.A. DeMillo, R.J. Lipton, and F.G. Sayward, ”Mutation Analysis”, Georgia Institute of Technology, *Technical Report*, GIT-ICS-79/08, 1979.

[2] Nguyễn Thanh Bình and Chantal Robach, “Mutation Testing Applied to Hardware: the Mutants Generation”, *Proceedings of the 11th IFIP International Conference on Very Large Scale Integration*,118--123, Montpellier, France, December, 2001.

[3] Jeff Offutt, Gregg Rothermel, Roland H. Untch and Christian Zapf, *An Experimental Evaluation of Selective Mutation,* Baltimore, MD, 1993.

[4] Javier Tuya, Mª José Suárez-Cabal and Claudio de la Riva, *Mutating database queries*, Information and Software Technology**,** ISSN:0950-5849, 2007.

[5] Jeff Offutt and Stephen D. Lee, "An Empirical Evaluation of Weak Mutation", *IEEE Transactions on Software Engineering,* 20(5):337-344, 1994.

[6] Jeff Offutt, "Practical Mutation Testing", *Twelfth International Conference on Testing Computer Software,* pages 99--109, Washington, DC, 1995.

[7] Stefan Brass, Christian Goldberg, *Semantic errors in SQL queries: a quite complete list*, Journal of Systems and Software, v.79 n.5, p.630-644, 2006.

[8] Javier Tuya , Ma Jose Suarez-Cabal, Claudio de la Riva, *SQLMutation: A tool to generate mutants of SQL database queries*, Proceedings of the Second Workshop on Mutation Analysis, 2006.

[9] Roland H. Untch, A. Jefferson Offutt, Mary Jean Harrold, Mutation analysis using mutant schemata, ACM SIGSOFT Software Engineering Notes, 1993.