

Un experimento controlado sobre pruebas de consultas SQL

Javier Tuya¹, Javier Dolado², M^a José Suárez-Cabal¹, Claudio de la Riva¹

(1) Departamento de Informática
Universidad de Oviedo
Campus Universitario de Gijón
33204 Gijón
[tuya | cabal | claudio]@uniovi.es

(2) Departamento de Lenguajes y Sistemas Informáticos
Universidad del País Vasco
P.M. Lardizabal, 1
20.009 San Sebastián
dolado@si.ehu.es

Resumen

Este artículo presenta los resultados de la validación experimental de una técnica de diseño de casos de prueba basada en un criterio de cobertura estructural para consultas escritas en el lenguaje SQL. Se describe un experimento controlado realizado con el objeto de comparar dicho criterio con criterios convencionales basados en partición de clases de equivalencia y análisis de valores límite. Los resultados de un análisis preliminar muestran que la utilización del criterio de cobertura estructural permite que los usuarios realicen casos de prueba más efectivos y además se obtienen recomendaciones adicionales para mejorar el propio criterio.

1. Introducción

Desde la aparición de los primeros lenguajes de consulta a bases de datos, se han realizado un gran número de estudios empíricos para analizar la efectividad y eficiencia en la construcción de sentencias escritas en estos lenguajes y en especial en SQL. Estudios iniciales (como los recopilados por Reisner en [6]) han permitido comprender los problemas que el usuario se encuentra al escribir consultas SQL y mejorar el propio lenguaje. Otros estudios recientes analizan aspectos relacionados con la normalización [1], la complejidad de la consulta [7], o los defectos típicos que introducen los usuarios al realizar consultas [2].

Por otra parte, en la comunidad de las pruebas del software se han realizado multitud de estudios consistentes en la validación empírica de diferentes técnicas de prueba desde diversos puntos de vista [3] y para diferentes tipos de

lenguajes. Sin embargo, no se han realizado estudios empíricos que puedan contribuir a entender y mejorar las técnicas y procesos de prueba para las consultas SQL que acceden a la información almacenada en bases de datos. Esto contrasta con el inmenso número de aplicaciones que utilizan bases de datos, y manejan la información utilizando SQL.

El lenguaje SQL fue desarrollado y comenzó a ser utilizado de forma industrial a partir de finales de la década de 1970, y continúa en evolución y uso en la actualidad. A pesar de ello, la mayor parte de la investigación en pruebas relacionadas con este tipo de lenguaje pertenece a la década actual y la validación de resultados se suele realizar mediante casos de estudio de pequeño tamaño [5]. En estos casos la validación de resultados no suele estar basada en experimentos controlados que permitan analizar y comprender de una forma rigurosa las ventajas e inconvenientes de unas técnicas frente a otras. Este es el ámbito en el que se enmarca este artículo.

El objetivo del presente artículo es comparar mediante un experimento controlado la efectividad de diferentes técnicas de prueba para sentencias SQL cuando éstas son utilizadas por personas que utilizan dichos criterios para guiar el diseño de los casos de prueba. En concreto se comparará un criterio sistemático y particular para SQL que está basado en cobertura estructural con la utilización de criterios convencionales como la partición en clases de equivalencia y análisis de valores límite.

En la Sección 2 se describe el criterio de cobertura estructural a utilizar. En la Sección 3 se describe el diseño del experimento cuyos resultados serán analizados en la Sección 4.

Finalmente en la Sección 5 se presentan brevemente las principales conclusiones.

2. Criterio de cobertura estructural para consultas SQL

El criterio de cobertura estructural para sentencias SQL se basa en la consideración de que una consulta realiza comparaciones sobre valores que se encuentran en diversas filas de una o varias tablas y por tanto, las comparaciones en general se realizan entre valores no escalares. Se establece una diferenciación entre las comparaciones de izquierda a derecha y de derecha a izquierda, así como la consideración de la existencia de valores desconocidos (valores nulos).

El criterio se describe con más detalle en [8]. Se basa en organizar las condiciones presentes en las cláusulas JOIN y WHERE en un árbol de cobertura que se representa gráficamente en la Figura 1.

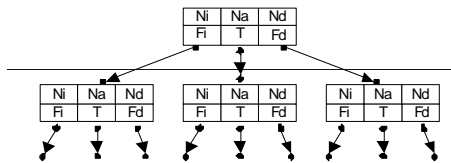


Figura 1. Representación gráfica de un árbol de cobertura

Cada condición se sitúa en un nivel del árbol y su evaluación se representa mediante uno o más "nodos de cobertura". La evaluación de la condición que se representa dependerá de la evaluación de los nodos superiores y se realizará respecto de los valores nulos (Ni, Na, Nd) y respecto de los valores no nulos (Fi, T, Fd), teniendo en cuenta el sentido en el que se realiza: de izquierda a derecha (i) y viceversa (d).

Cuando la consulta se ejecuta con un caso de prueba se evalúan cuáles de las diferentes situaciones representadas en los nodos de cobertura se cumplen y cuales no (seis en cada nodo). El porcentaje de cobertura será entonces el porcentaje de situaciones cubiertas respecto del total de situaciones posibles representadas en el árbol.

En el caso de la cláusula HAVING se realiza un árbol independiente para las condiciones incluidas en dicha cláusula y se evalúa a partir del

resultado obtenido tras la ejecución de las cláusulas FROM, WHERE y GROUP BY de la consulta.

La representación gráfica es muy visual pero difícil de incorporar en una herramienta, por lo que para la realización del experimento se ha optado por representar cada una de las situaciones a cubrir mediante una consulta SQL denominada regla de cobertura. Cada una se ejecuta contra la base de datos de prueba, detectando que la situación representada está cubierta cuando la regla devuelve alguna fila. Un comentario textual descriptivo acompaña a cada una de estas reglas informando al usuario de la situación que se debe tener en los datos de prueba para que se cumpla la regla.

A continuación se muestra un ejemplo de una consulta realizada sobre tres tablas (proyecto, empleado y trabajo):

```
SELECT e.nombre , t.mes , t.horas , p.nombre
FROM empleado e
LEFT JOIN trabajo t ON t.idempleado=e.idempleado
LEFT JOIN proyecto p ON p.idproyecto=t.idproyecto
```

El árbol de cobertura tiene dos niveles (para las condiciones de las cláusulas JOIN). Una regla de cobertura para las situaciones T (en el primer nivel) y FI (en el segundo) es la siguiente:

```
SELECT * FROM empleado e
INNER JOIN trabajo t on e.idempleado=t.idempleado
LEFT JOIN proyecto p on t.idproyecto=p.idproyecto
WHERE t.idproyecto is not null and p.idproyecto is null
```

La primera cláusula JOIN obliga a obtener registros de tabla trabajo con empleado relacionados, mientras que la segunda permite que un trabajo no tenga proyecto seleccionado. La cláusula WHERE impone una restricción adicional para que solamente se extraigan aquellas filas donde no hay un registro correspondiente a la tabla proyecto. Esta regla se presenta textualmente como "Debe existir un empleado que tenga relacionado un trabajo pero que no tiene relacionado un proyecto".

En [9] se enumeran algunas mejoras del criterio encaminadas a mejorar la eficiencia y la eficacia. Respecto de la eficiencia, para evitar la expansión de los árboles se recomienda podar las ramas, por lo en las reglas utilizadas en el experimento se utilizará un subconjunto de todas las posibles: Se genera un primer conjunto de

reglas conteniendo exclusivamente los niveles correspondientes a las cláusulas JOIN, otro conjunto de reglas donde las condiciones de los JOIN solamente continúan hacia el WHERE en la rama para la que éstas están evaluadas a cierto y reglas adicionales en las que alguna de las situaciones correspondientes a las cláusulas JOIN son Fl o Fr. En relación a la eficacia, se ha mostrado que ésta puede aumentar cuando se contemplan los valores límite de las condiciones. Esto se ha incorporado añadiendo reglas que se cumplen cuando los valores en una condición están en los límites que hacen que esta cambie de cierto a falso y viceversa.

3. Diseño del experimento

El objetivo del experimento es determinar si la técnica utilizada en el diseño de casos de prueba para sentencias de SQL influye en la eficacia de los casos para detectar defectos en las consultas. En concreto se compararán dos técnicas:

- Basada en el criterio de cobertura estructural descrito en la sección 2 (denominada en lo sucesivo CC).
- Basada en otras técnicas de prueba convencionales: partición en clases de equivalencia y análisis de valores límite (denominada en lo sucesivo OT).

Así, se establecerá la hipótesis nula H_0 como “La utilización de una técnica para la elaboración de casos prueba de sentencias SQL (CC frente a OT) no influye en la capacidad de detección de defectos de los casos de prueba elaborados”.

La variable independiente del experimento es, por tanto, la técnica utilizada (CC u OT), y la variable dependiente la capacidad de detección de defectos (Score), denominado en adelante, M (mutation score). Este se obtiene al ejecutar los casos de prueba respecto de un conjunto de consultas SQL con fallos (mutantes) obtenidos con la herramienta SQLMutation¹ [10]. Los operadores de mutación para consultas SQL son descritos con detalle en [11] y se agrupan en cuatro categorías:

- *SC - SQL clause mutation operators*: Realizan mutaciones en las cláusulas principales: SELECT, JOIN, GROUP BY, UNION, ORDER BY, subconsultas y funciones agregado.
- *NL - NULL mutation operators*: Relativas al manejo de los valores nulos, tanto en las condiciones como en las salidas de la consulta.
- *OR - Operator replacement mutation operators*: Similares a los operadores de modificación de expresiones descritos en [4] más operadores adicionales para los predicados BETWEEN y LIKE.
- *IR - Identifier replacement mutation operators*: Reemplazo de columnas, constantes y parámetros presentes en la consulta o en las tablas utilizadas por ésta.

El experimento se realizó con un conjunto de 20 alumnos de un curso de extensión universitaria sobre pruebas del software. Todos ellos recibieron formación básica en técnicas de pruebas, así como formación específica en la particularización de éstas técnicas a las pruebas de consultas SQL.

Los alumnos se organizaron en dos grupos (A y B) con 10 alumnos cada uno. Cada grupo realizó dos sesiones de pruebas de 45 minutos cada una. En cada una de las sesiones se realizaron las pruebas de dos consultas SQL diferentes, utilizando para el diseño de los casos de prueba una técnica diferente en cada sesión. La Tabla 1 resume este diseño

Grupo.	Sesión	Consultas	Técnica
A	1	Q1, Q3	CC
B	1	Q1, Q3	OT
A	2	Q2, Q4	OT
B	2	Q2, Q4	CC

Tabla 1. Diseño del experimento

Todas las consultas contienen cláusulas JOIN con tres tablas. En cada una de las sesiones, la primera consulta (Q1 ó Q2) es la más simple, incluyendo una cláusula WHERE y ORDER BY. La segunda (Q3 ó Q4) incluye además GROUP BY y HAVING.

La Tabla 2 muestra para cada consulta el número de mutantes no equivalentes generados junto con el número de reglas de cobertura utilizadas (criterio CC).

¹ SQLMutation está disponible a través de un interfaz web y como servicio web en <http://in2test.lsi.uniovi.es/sqlmutation/>

	Q1	Q2	Q3	Q4
Mutantes no equivalentes	89	106	125	122
Reglas de cobertura (CC)	18	18	20	22

Tabla 2. Número de mutantes y reglas de cobertura para cada consulta SQL

Para la realización de las pruebas, todos los alumnos utilizaron una herramienta denominada SQTtest (<http://in2test.lsi.uniovi.es/sqltest/>) que permite introducir desde un único formulario los datos de entrada y visualizar los resultados. En el caso de utilizar la técnica CC, además existe la posibilidad de evaluar la cobertura de los casos de prueba y visualizar cuáles son las situaciones no cubiertas.

4. Resultados del experimento

Tras la realización de las dos sesiones del experimento, se generaron los mutantes para todas las consultas y se evaluó el “mutation score” (M) obtenido por cada una, así como la cobertura conseguida (C). La Figura 2 muestra el diagrama de cajas correspondiente a la variable dependiente (Score). La leyenda correspondiente a cada una de las cajas representa la variable independiente que se está midiendo (M, C), la consulta (Q1 a Q4) y técnica utilizada (CC, OT) en el diseño de los casos de prueba.

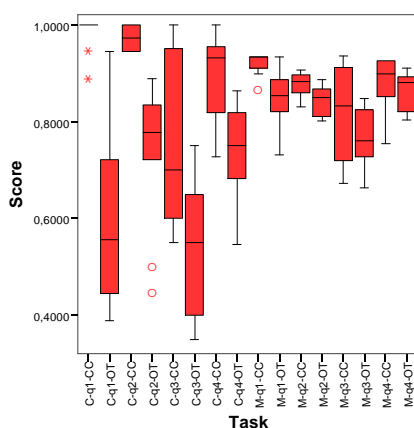


Figura 2. Diagrama de cajas correspondiente a las consultas Q1 a Q4

En la primera mitad de la figura se puede apreciar cómo para los casos diseñados utilizando CC se obtiene un porcentaje mucho mayor de cobertura C, lo cual es lógico puesto que estas pruebas han sido elaboradas utilizando este mismo criterio como guía.

La observación de los valores del “mutation score” (M) en la segunda mitad de la Figura 2 muestra que aparentemente se obtiene un mayor valor en las pruebas realizadas utilizando el criterio CC que en las realizadas utilizando OT, aunque las diferencias son mayores en las consultas Q1 y Q2 que en el resto.

Con el objeto de determinar si estas diferencias son estadísticamente significativas, para cada una de las consultas se realizará una prueba T para muestras independientes con un factor (la técnica utilizada: CC u OT) y variable dependiente M. La Tabla 3 muestra los resultados del estadístico (t), el grado de significación (p) y la diferencia de medias.

	t	p	Diferencia de medias
Q1	3,624	0,004	7,08%
Q2	2,471	0,024	1,22%
Q3	1,142	0,272	3,78%
Q4	0,685	0,502	2,15%

Tabla 3. Resultados del análisis para Q1 a Q4

Se aprecia que tanto en Q1 como Q2 la diferencia es significativa. Hay que hacer notar que la distribución de M en Q1 no cumple el requisito de normalidad (la mayor parte de los valores de M son iguales). Por ello se realiza un test no paramétrico alternativo (Mann-Whitney) que muestra que la diferencia es significativa con $p=0,002$. Ello permite rechazar la hipótesis nula de que ambas medias son diferentes.

Pero en las consultas Q3 y Q4, aunque los valores medios de M son superiores en ambos casos cuando se utiliza CC, esa diferencia no es significativa. Se pueden buscar diferentes justificaciones a ello que se discuten a continuación.

Una primera causa puede ser la fiabilidad de los casos de prueba realizados ya que estas consultas son más complejas, realizadas al final de cada sesión, y no siempre se completan. En la Figura 2 se puede apreciar que el porcentaje de cobertura C utilizado al aplicar la técnica CC tiene

mucho mayor rango de variación y valores inferiores para Q3 y Q4 que para Q1 y Q2. Además las reglas que no han sido cubiertas son las últimas en ser utilizadas para realizar las pruebas, por lo que no se puede considerar que estas pruebas no hayan sido completadas en su totalidad. Estos resultados deben aplicarse con precaución, aunque esta influencia afecta en principio por igual a ambos grupos.

Sin embargo, se ha encontrado otra justificación más plausible relacionada con el propio diseño de la técnica de cobertura CC. Examinando en detalle las situaciones cubiertas y los mutantes que quedan vivos se aprecia que se produce un enmascaramiento de los defectos que son capaces de detectar los casos de prueba.

Estas consultas (Q3 y Q4) tienen una cláusula HAVING que selecciona filas tras haber realizado JOIN, seleccionado con WHERE y agrupado con GROUP BY. Cuando se usa el criterio CC, el sujeto experimental centra su atención en cumplir las reglas de cobertura que se le presentan. La técnica CC no considera la existencia de la cláusula HAVING para la evaluación de la cobertura en cláusulas JOIN y WHERE. Por tanto, es posible que se cubran situaciones en éstas que cumplen las reglas de cobertura, pero que luego son filtradas por el HAVING, y por tanto no se traducen en un fallo detectable. Sin embargo, cuando se utiliza OT, el sujeto experimental centra su atención en toda la consulta y se tiene menor tendencia a este fenómeno. Es decir, muchas de las reglas que se cumplen utilizando el criterio CC no se traducen en una salida diferente, y por tanto, no matan determinados mutantes. El enmascaramiento es más grave en Q4, debido a que además de HAVING tiene una cláusula WHERE (que no tiene Q3).

Para comprobar lo anterior se añadirán dos consultas más denominadas Q5 y Q6 que son una copia de las Q3 y Q4 a las que se les ha eliminado la cláusula HAVING. Estas consultas son ejecutadas con los mismos casos de prueba diseñados para Q3 y Q4, y el valor de M se evaluado. Los resultados de los valores de M se presentan en la Figura 3.

Gráficamente se puede apreciar que cuando se compara CC con OT, las diferencias parecen mayores en Q5 y Q6 que en Q3 y Q4 respectivamente.

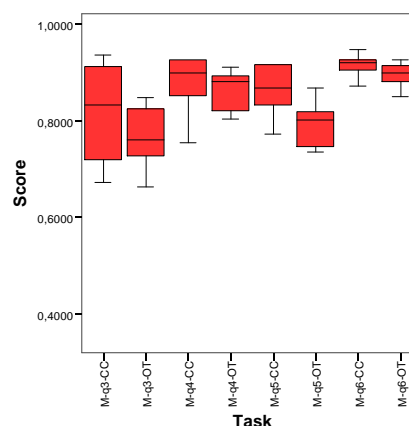


Figura 3. Diagrama de cajas comparativo entre Q3, Q4 y Q5, Q6.

Una nueva prueba T proporciona los resultados que se presentan en la Tabla 4.

	t	p	Diferencia de medias
Q5	12,966	0,008	6,50%
Q6	1,947	0,067	2,04%

Tabla 4. Resultados del análisis para Q5 y Q6

Por tanto, en ambos casos se rechaza la hipótesis nula (si bien en el caso de Q6 de forma marginal, pues $p=0,067$).

Esto sugiere una mejora de la implementación del criterio de cobertura CC, de forma que cuando se evalúe la cobertura del JOIN y WHERE en una consulta que tiene HAVING, se debe imponer una restricción adicional para que la cláusula HAVING sea cierta.

5. Conclusiones

En este artículo se ha mostrado empíricamente mediante un experimento controlado que el uso sistemático de un criterio de cobertura estructural para SQL para el diseño de casos de prueba consigue que se obtengan casos de pruebas capaces de detectar más defectos que cuando se utilizan otros criterios convencionales.

El examen de los casos en los que las diferencias no eran significativas ha permitido

asimismo detectar un punto de mejora en el propio criterio aplicable a las consultas que tienen cláusula HAVING.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Programa Nacional de I+D+i del Ministerio de Educación y Ciencia y fondos FEDER, con los proyectos IN2TEST (TIN2004-06689-C03-02), IN2QUANT (TIN2004-06689-C03-01) y la Red para la Promoción y Mejora de las Pruebas en Ingeniería del Software: RePRIS (TIN2005-24792-E).

Referencias

- [1] P.L. Bowen, F.H. Rohde, Further evidence of the effects of normalization on end-user query errors: an experimental evaluation, *International Journal of Accounting Information Systems* 3(4) (2002) 255–290.
- [2] S. Brass, C. Goldberg, Semantic Errors in SQL Queries: A Quite Complete List. *Journal of Systems and Software* 79(5) (2006) 630-644.
- [3] N. Juristo, A.M. Moreno, S Vegas, Reviewing 25 Years of Testing Technique Experiments, *Empirical Software Engineering* 9(1-2) (2004) 7–44.
- [4] K.N. King, A.J. Offutt. A Fortran Language System for Mutation-Based Software Testing. *Software Practice and Experience* 21(7) (1991) 686-718.
- [5] N.F. Moratinos, M.J. Suárez-Cabal, J. Tuya, Evaluación de la investigación en el campo de pruebas de aplicaciones con base de datos, Taller sobre Pruebas en Ingeniería del Software (PRIS 2006), Sitges, 2006.
- [6] P. Reisner, Use of Psychological Experimentation as an Aid to Development of a Query Language, *IEEE Transactions on Software Engineering* 3(3) (1977) 218-229.
- [7] K. L. Siau, H.C. Chan, K.L. Wei, Effects of Query Complexity and Learning on Novice User Query Performance With Conceptual and Logical Database Interfaces, *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 34(2) (2004) 276-281.
- [8] M.J. Suárez-Cabal, J. Tuya, Using a SQL Coverage Measurement for Testing Database Applications, 12th ACM SIGSOFT Symposium on Foundations of Software Engineering. *ACM Software Engineering Notes* 19(6) 2004, 253-262.
- [9] M.J. Suárez-Cabal, Mejora de casos de prueba en aplicaciones con bases de datos utilizando medidas de cobertura de sentencias SQL. Tesis doctoral, Universidad de Oviedo, 2006.
- [10] J. Tuya, M.J. Suárez-Cabal, C. de la Riva, SQLMutation: a Tool to Generate Mutants of SQL Database Queries, Second Workshop on Mutation Analysis (Mutation 2006), Raleigh, NC, 2006.
- [11] J. Tuya, M.J. Suárez-Cabal, C. de la Riva, Mutating Database Queries, *Information and Software Technology*, 49(4) (2007) 398-417.