DO YOU KNOW SQL?
ABOUT SEMANTIC ERRORS IN DATABASE QUERIES

Christian Goldberg
Martin-Luther-Universität Halle, Institut für Informatik
Von-Seckendorff-Platz 1
D-06099 Halle, Germany
goldberg@informatik.uni-halle.de
http://dbs.informatik.uni-halle.de/sqllint

ABSTRACT
We investigate classes of SQL queries which are syntactically correct, but certainly not intended, no matter for which task the query was written. For instance, queries that are contradictory, i.e. always return the empty set, are quite often written in exams of database courses. Current database management systems, e.g. Oracle, execute such queries without any warning.

In this paper, we give some statistics how often such semantic errors occurred in five analyzed exams at the University of Halle and discuss possible causes and solutions.

Keywords
SQL, databases, queries, semantic errors, statistics

1. INTRODUCTION
SQL is today the standard language for relational and object-relational databases. Application programs typically contain a relatively large number of SQL queries and updates, which are sent to the DBMS for execution. As any program code, SQL queries can contain errors.

Errors in SQL queries can be classified into syntactic errors and semantic errors. A syntactic error means that the entered character string is not a valid SQL query. In this case, any DBMS will print an error message because it cannot execute the query. Thus, the error is certainly detected and usually easy to correct.

A semantic error means that a legal SQL query was entered, but the query does not or not always produce the intended results, and is therefore incorrect for the given task. Semantic errors can be further classified into cases where the task must be known in order to detect that the query is incorrect, and cases where there is sufficient evidence that the query is incorrect no matter what the task is. Our focus is on this latter class, since there is often no independent specification of the goal of the query. For instance, consider this query:

```
SELECT *
FROM   EMP
WHERE  JOB = 'CLERK' AND JOB = 'MANAGER'
```

This is a legal SQL query, and it is executed e.g. in Oracle10g and DB2 V8.1 without any warning. But the condition is inconsistent: The query result will be always empty. Since nobody would use a database in order to get a certainly empty result, we can state that this query is incorrect without actually knowing the task of the query.

In [1] we gave an extensive list of more than 40 conditions that are strong indications of semantic errors. Of course, questions like the satisfiability are in general undecidable, but in [2], we propose a consistency check that can handle a surprisingly large subset of SQL (it uses Skolemnization with sorted Skolem functions, and

a few other tricks). This consistency check is also the basis for generating warnings for other semantic errors, so that a significant subset of SQL queries can actually be checked.

The given list of semantic error types in [1] basically results from the experience in correcting and grading a large number of exams and homework. Almost any error type appeared in an exam or homework, however, there were approximately as much semantic errors in written examination as in the students homework, although the students were given the opportunity to verify their homework with a real database. This shows that some students not even cross-checked the query result with what they originally expected -- especially, as typical semantic errors were discussed in detail during the lectures and exercises. In some cases the students obviously realized that something is wrong but didn't know exactly what and than tried to straighten it out by another semantic error. A case in point is the simultaneously occurrence of Error 27 (Missing join condition) and Error 37 (Many duplicates), which often tempts the students to add a **DISTINCT** instead of correcting the real error (see also Section 3).

To get a closer look at how and why semantic errors occur we analyzed the solutions of SQL exercises in written examinations. The analyzed exams and the appeared exercises are described in Section 2. Section 3 contains some statistics how often errors occurred in these exams. The results, possible causes and solutions are discussed in Section 4.

2. BASE DATA

We analyzed five exams of the course "Databases I" in winter term at the Martin-Luther-University of Halle. The course was taught by two different professors. After summer term 2003 the professorship changed to its present chair, Prof. Dr. Stefan Brass. Figure 1 gives a brief overview of all analyzed exams. It lists the lecturer, the number of SQL exercises, related points and total points for each exam. The exercises are divided into three groups according to their degree of difficulty: beginner, intermediate and advanced. The classification is as follows:

beginner The query contains only joins and simple conditions within the **WHERE**-clause.

intermediate The query contains at least a subquery with a simple (not grouped) aggregation function, an **NOT EXISTS**-subquery, a self join or an **UNION** of two simple (beginner) queries.

advanced The query contains at least **GROUP BY** with some exclusion (within the **HAVING**- or **WHERE**-clause) and an aggregation function, complex self joins or more than one **NOT EXISTS**-subquery.

A detailed description of all appeared exercises together with sample solutions can be found at the project web page [3]. In this paper only exams with at least 30 participants are taken into consideration due to space restrictions and statistical reasons. Further course material and exam exercises are also available from the project web page.

Exam	Lecturer	Exercises SQL	Points SQL	Points Total
Final 2002/03	Dr. Sattler	4	10	50
Midterm 2003/04	Prof. Dr. Brass	3	9	23
Final 2003/04	Prof. Dr. Brass	3	9	20
Final 2005/06	Prof. Dr. Brass	6	18	37
Final 2008/09	Prof. Dr. Brass	6	15	35

Figure 1: Overview of analyzed "Databases I" exams

Error	2002/03 67 participants A	B	C	D	Mid 03/04 153 part. E	F	G	Fin 03/04 148 part. H	I	J	2005/06 40 part. K	L	M	N	O	P	2008/09 53 part. Q	R	S	T	U	V	Difficulty Class 1	2	3	$\sum$
	1	3	2	2	1	1	2	3	2	2	1	1	2	3	2	3	1	2	3	2	3	3	1	2	3	
1	-	-	2	1	3	4	14	1	14	2	-	1	3	13	-	-	-	7	1	-	1	-	8	43	16	67
1a	-	1	-	-	-	-	-	4	-	-	-	-	2	-	2	1	-	-	-	-	5	9	-	4	20	24
2	4	-	-	1	2	-	5	2	2	-	3	5	-	1	-	-	2	3	3	3	2	3	16	14	11	41
3	8	-	-	4	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	19	-	8	5	19	32
4	-	-	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	2
5	1	-	-	-	-	-	2	11	9	1	1	-	2	4	1	3	-	-	-	-	3	-	2	15	21	38
6	-	-	-	-	-	-	25	2	5	2	4	1	2	5	7	10	-	8	1	-	11	13	5	49	42	96
7	-	-	-	-	5	-	8	-	-	1	-	-	5	5	-	1	-	-	-	-	-	-	5	9	6	20
8	-	1	3	-	2	2	6	4	1	1	-	6	6	-	-	1	-	-	8	4	3	-	10	21	17	48
9	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6	-	-	6
11	-	-	1	-	-	-	-	1	-	3	-	-	-	-	-	-	-	-	-	2	-	-	-	6	1	7
12	-	-	-	-	2	3	2	-	-	-	-	3	-	-	-	-	-	-	-	-	-	-	8	2	-	10
13	-	-	-	-	-	-	-	-	-	-	-	-	2	1	-	1	-	-	-	1	-	-	-	3	2	5
14	-	-	-	-	-	-	-	2	2	1	-	-	4	1	-	-	-	-	-	-	-	-	-	7	3	10
15	-	-	2	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	5	-	-	5	7	12
17	1	-	-	17	-	-	-	1	-	-	-	-	-	-	-	24	-	-	-	-	9	-	1	17	34	52
19	-	-	-	-	-	-	-	1	2	16	-	-	-	-	-	1	-	-	1	6	1	-	-	24	4	28
20	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1
21	-	1	1	-	-	-	-	1	1	1	-	-	-	-	-	-	-	-	-	-	9	-	-	3	11	14
22	1	-	1	-	-	-	-	1	1	-	-	-	2	-	1	-	-	-	-	2	-	-	1	7	1	9
23	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	1
25	-	-	1	1	-	-	-	3	-	1	-	-	-	-	-	2	-	-	1	-	-	-	-	3	6	9
26	-	-	-	-	-	-	-	-	-	-	-	-	1	-	11	-	-	-	-	-	10	18	-	12	28	40
27	14	17	14	6	1	3	29	2	5	-	1	-	12	11	1	11	1	-	2	-	3	4	20	67	50	137
28	-	-	-	5	-	-	-	-	6	-	-	2	4	1	-	1	-	1	-	-	-	-	2	16	2	20
30	1	-	-	2	-	-	-	1	3	-	-	-	-	2	-	-	-	-	-	-	-	-	1	5	3	9
31	-	-	-	1	-	5	1	-	-	-	-	-	2	1	1	3	-	-	-	-	-	-	5	5	4	14
34	-	-	-	-	11	-	-	-	-	-	1	5	-	-	-	-	-	-	-	-	-	-	17	-	-	17
36	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	1	-	-	-	-	2	1	3
37	-	-	-	-	-	46	-	-	-	-	25	1	19	-	3	2	14	-	4	-	1	-	86	22	7	115
39	6	2	3	1	-	-	-	-	-	-	-	-	2	1	-	-	-	-	-	1	-	1	6	7	4	17
Correct	22	2	10	-	104	37	30	17	45	54	6	10	3	7	5	3	27	41	29	29	3	-	40.7%	28.2%	13.4%	28.0%
Semantic	19	6	7	11	20	50	50	9	31	18	27	11	19	24	10	13	13	10	9	18	15	20	27.6%	22.6%	21.2%	23.7%
Syntax	15	20	18	13	7	17	9	82	30	57	3	3	1	3	4	1	8	-	9	3	4	3	10.5%	17.6%	26.9%	17.9%
Both	10	14	14	19	6	11	23	21	15	7	3	9	14	5	9	21	4	1	2	-	26	11	8.5%	13.3%	22.0%	14.2%
N.C.	-	19	15	14	4	8	32	13	4	4	-	-	-	-	7	2	-	-	1	-	3	17	2.4%	9.9%	12.1%	8.3%
W.T.	1	6	3	10	12	30	9	6	23	8	1	7	3	1	5	-	1	1	3	3	2	2	10.3%	8.4%	4.4%	7.9%

Figure 2: Error Statistics for five Exams
Difficulty Class (underneath the exercise capital): 1 - beginner, 2 - intermediate, 3 - advanced

3. STATISTICS

The results of the survey are shown in figure 2. The exercises are numbered with unique letters, e.g. the first exam contained four exercises (A, B, C and D). Underneath the class of difficulty is given for each exercise in form of numbers: 1 means beginner, 2 intermediate, 3 advanced. (For more information about the exams and classification, see previous section.)

Among the individual semantic errors we also counted whether syntactical errors occurred, an exercise was not solved or the solution was not correct for other reasons. Thus, figure 2 lists for each exercise the number of solutions with only semantic errors ("Semantic"), only syntactical errors ("Syntax"), both (semantic and syntactical errors), solutions that can only be detected as incorrect if the goal of the query is known (wrong task; "W.T.") and unsolved solutions (Not Counted; "N.C."). "N.C." also lists exercises that contained so severe syntax errors that a detailed analysis was not possible, which ran in the one-digit range, though. In addition to the number of errors, correct or not counted solutions etc. per exercise, the figure also lists the number per difficulty class.

The number of exam solutions that contained at least one semantic error is the sum of the entries "Semantic" and "Both". Of course we counted only semantic errors from the list given in [1]. i.e. that are detectable without knowing the task of the query and that are quite strong indications that the query will not (always) behave as intended, or is at least more complex than it needs to be. Error 1a is an exception only since it is new and not yet mentioned in [1], it means: Unnecessary outer query[1]. Style errors, also listed in [1], which but are a matter of taste, were not counted.

We did sometimes count several unrelated semantic errors in the same exercise, but in most cases they did not interact, thus almost all semantic errors (that did not occur simultaneously with syntactic errors) could have been found by our methods, described in [2] and [1]. Otherwise an error was counted only once in our statistics if it occurred more than once in the same exercise.

Some errors are mutually dependent or mutually exclusive. Such as Error 27 (Missing join condition) or Error 5 (Unused tuple variable), which almost always involve Error 37 (Many duplicates). Therefore, in our statistics Error 37 is only counted if it occured independent from Error 27 and 5. Joins may cause a number of semantic errors, which look quite similar. Error 5 (Unused tuple variable), Error 6 (Unnecessary join) and Error 27 (Missing join condition) differ in the fact whether the tuple variable is never been used in the query, there exists an equivalent query without an used join or a join condition just was forgotten. These errors may not be counted at the same time for the same tuple variable. In fact, they can occur within the same query if they affect different tuple variables.

4. EVALUATION

In the exams that were analyzed, the ten most often occurring semantic errors are (percentages are relative to all detected semantic errors):

Rank	Ratio	Semantic Error
1.	14.5 %	Error 27: Missing join condition
2.	12.8 %	Error 37: Many duplicates
3.	10.7 %	Error 6: Unnecessary join
4.	7.5 %	Error 1: Inconsistent condition
5.	5.8 %	Error 17: Unnecessary argument of **COUNT**
6.	5.4 %	Error 8: Implied, tautological or inconsistent subcondition
7.	4.6 %	Error 2: Unnecessary **DISTINCT**
8.	4.5 %	Error 26: Inefficient **UNION**
9.	4.2 %	Error 5: Unused tuple variable
10.	3.6 %	Error 3: Constant output column

Figure 3: The ten most often occurred semantic errors

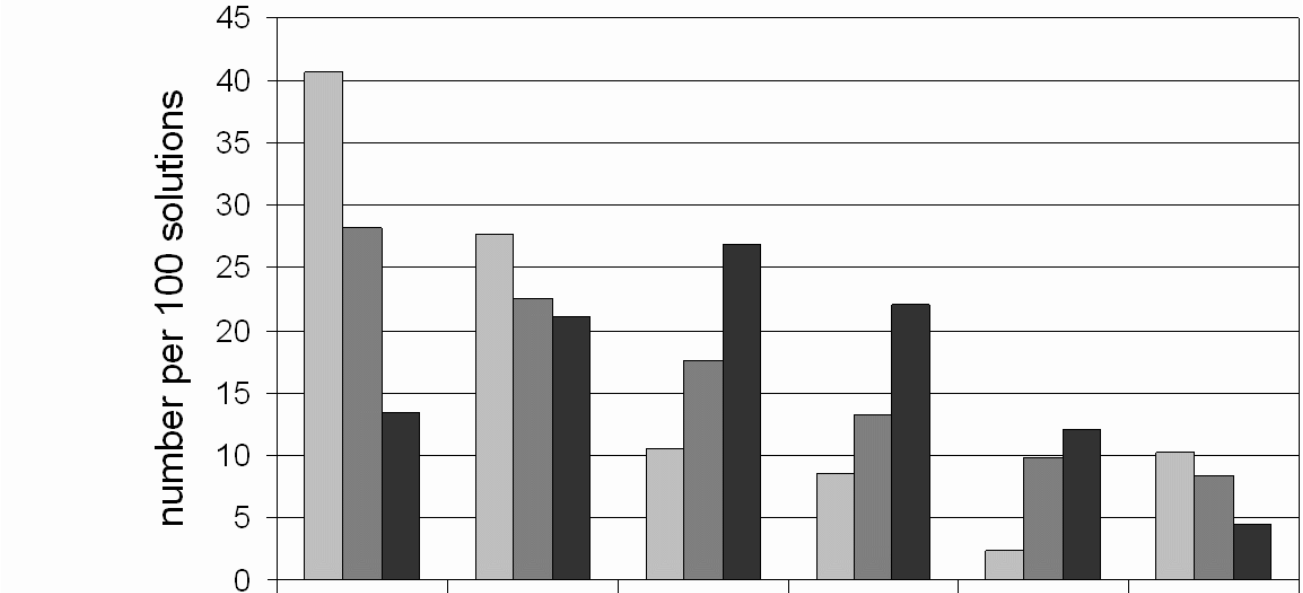
[1] An outer query is unnecessary if it contains only one subquery under **FROM** and if there is only one **GROUP BY** clause, either in the outer query or in the subquery. Then both queries should be combined into one to make it shorter and easier to read. The outer query is furthermore necessary if it aggregates a subquery with set operators.

As you can see, unnecessary complications relating to joins (Errors 27, 6 and 5) make up 29.4% of all semantic errors followed by unnecessary or forgotten **DISTINCT** (17.4%; Errors 37 and 2) and logical errors that cause unnecessary conditions or even whole unnecessary queries (12.9%; Errors 1 and 8). This means that these three classes add up to almost two-thirds of all semantic errors and need to be more investigated.

Under discussion with students one can sometimes get out, why errors occur. A frequent reason for Error 27 is that these people often don't know the difference between a join and a cartesian product in SQL. This becomes obvious by comments like “I thought, it will be joined if I type it under **FROM**”. Error 5 often evolves from writing down a tuple variable under from without finally using it. Unnecessary joins and errors with **DISTINCT** or many duplicates often arise from an insufficient acquaintance with the underlying database schema, whereas logical errors mostly result from difficulties in converting a problem into a query or programming language.

In the analyzed exams more than half of all solutions contained one or more semantical or syntactical error, nearly a quarter of all solutions contained only semantic errors. As the queries in exams could not be checked for syntactical errors and most of them would have been very easy to correct (e.g. false column names, double instead of single quotes, missing **GROUP BY**-terms etc.), with the techniques described in [2] in almost 38% of all cases one could have got a sensible error message, while a standard DBMS simply executes the query (with possibly a wrong result, which might or might not be detected). In contrast, the most semantic errors are therefore much more difficult to detect.

As one would expect, in syntactically relatively simple exercises (e.g., midterm exam in winter term 2003/04), there are many more correct solutions than solutions with syntax errors or not counted solutions. Interestingly, the easier the exercise the more there are detectable semantic errors (that not simultaneously occur with syntax errors) and solutions that can only be detected as incorrect if the goal of the query is known (wrong task). A possible reason is that there are more careless mistakes if the exercise is assumed to be easy.



	Correct	Sem.	Syntax	Both	N.C.	W.T.
beginner	40,71	27,67	10,47	8,5	2,37	10,28
intermediate	28,22	22,63	17,56	13,26	9,88	8,45
advanced	13,44	21,15	26,87	22,03	12,11	4,41

Figure 4: Normalized Error Distribution for Classes of Difficulty

To get a general idea about the numbers of solutions within each class of difficulty Figure 4 shows the distribution of the different types of solutions for the three classes. As the number of exercises per class and the number of participants per exam varied, it was impossible to compare absolute numbers of solutions. Hence, the figure lists the distribution for every class of difficulty normalized to 100 solutions of this class.

Again, it is not surprising that the most correct solutions occurred in exercises of the class beginner and the fewest in exercises of the class advanced and vice versa for e.g. the not counted solutions. There is also almost the same number of solutions with only semantic errors for every class of difficulty. This indicates fundamental comprehension problems with parts of SQL (see e.g. the distribution of Errors 1, 5, 6, 8 and 27 in Figure 2). Furthermore the number of solutions with syntax errors ("Syntax" and "Both") increases with the level of difficulty. It seems that here the students focused on finding a possible solution somehow and neglected the syntax. Perhaps they would have found some more syntax errors in their solutions if they had some more time to solve it. But mostly a lack of time resulted in an unsolved solution and not in syntax errors.

Based on talks with students after and before their examination, one can say that most of the students who made significantly more errors than others were inadequately prepared for the exam or did not even attend a lecture. Furthermore many students did not read the exercises very intently and often did not have enough experience in SQL. These problems should be easy to solve. Sometimes the students made less errors by using graphs to understand the connection of relations or simply by talking the problem over, which but isn't an option in an exam.

5. RELATED WORK

It seems that the general question of detecting semantic errors in SQL queries (as defined in [1]) is new. However, it is strongly related to the two fields of semantic query optimization and cooperative answering.

Semantic query optimization (see e.g. [7], [14], [6]), also tries to find unnecessary complications in the query, but otherwise the goals are different. As far as we know, no system prints a warning if the optimizations are "too good to be true". Also the effort for query optimization must be amortized when the query is executed, whereas for error detection, we would be willing to spend more time. Finally, soft constraints (that can have exceptions) can be used for generating warnings about possible errors, but not for query optimization.

Our work is also related to the field of cooperative query answering (see, e.g., [12], [9], [11]). However, there the emphasis is more on the dialogue between DBMS and user. As far as we know, a question like the possibility of runtime errors in the query is not asked. Also, there usually is a database state given, whereas we do not assume any particular state. For instance, the CoBase system would try to weaken the query condition if the query returns no answers. It would not notice that the condition is inconsistent and thus would not give a clear error message. However, the results obtained there might help to suggest corrections for a query that contains this type of semantic error.

The SQL Tutor system described in [16] and [17] discovers semantic errors, too, but it has knowledge about the task that has to be solved (in form of a correct query). In contrast, our approach assumes no such knowledge, which makes it applicable also for software development, not only for teaching.

Software testing techniques for database queries are developed e.g. in [4] and [20]. Further studies about errors in database queries, especially psychological aspects, are [19], [18], [15], [10] and [8].

6. CONCLUSIONS

There is a large class of SQL queries that are syntactically correct, but nevertheless certainly not intended, no matter what the task of the query might be. One could expect that a good DBMS prints a warning for such queries, but, as far as we know, no DBMS does this yet.

In this paper we gave a survey of how often semantic errors occurred in written exams and tried to find reasons for the most common errors. A detailed description of the analyzed exams with all appeared exercises and sample solutions can be found at the project web page [3]. This page also contains a prototype of the consistency test.

While our experience so far has almost only been with errors made by students, not professional programmers, most of the students will become programmers, and they will not immediately make fewer errors. Our experience in the cooperation with small- and medium-sized companies has also shown that database programming is often made by web developers without a thorough education in SQL. In the exam

solutions that we analyzed so far, 24% contained a semantic error of the type we consider in [1] (detectable without knowledge of the task), 18% contained a syntax error, 14% contained both, and 8% contained a semantic error that was only detectable if the task was known. Thus, with the techniques described in [2], in nearly a quarter of the cases one could have got a sensible error message, while a standard DBMS simply executes the query (with possibly a wrong result, which might or might not be detected).

7. REFERENCES

[1] Stefan Brass and Christian Goldberg. Semantic Errors in SQL Queries: A Quite Complete List. In: *Elsevier's Journal of Systems and Software 79(5)*, 2006.

[2] Stefan Brass and Christian Goldberg. Proving the Safety of SQL Queries. In: *Fifth International Conference on Quality Software (QSIC'05)*, IEEE Computer Society Press, 2005.

[3] Stefan Brass and Christian Goldberg. SQLLint: Detecting Logical Errors in SQL Queries. Project website: http://dbs.informatik.uni-halle.de/sqllint/

[4] M. Y. Chan and S. C. Cheung. Testing database applications with SQL semantics. *Proceedings of 2nd International Symp. on Cooperative Database Systems for Advanced Applications (CODAS'99)*, 364-375, 1999.

[5] Joe Celko: *Joe Celko's SQL for Smarties: Advanced SQL Programming, 2nd Ed*. Morgan Kaufmann, 1999.

[6] Qi Cheng, Jarek Gryz, Fred Koo, Cliff Leung, Linqi Liu, Xiaoyan Qian, and Bernhard Schiefer. Implementation of Two Semantic Query Optimization Techniques in DB2 Universal Database. *Proceedings of the 25th VLDB Conference*, 687-698, 1999.

[7] Upen S. Chakravarthy, John Grant, and Jack Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15:162-207, 1990.

[8] Hock C. Chan. The relationship between user query accuracy and lines of code. *Int. Journ. Human Computer Studies 51*, 851-864, 1999.

[9] Wesley W. Chu, M.A. Merzbacher, and L. Berkovich. The design and implementation of CoBase. In: *Proc. of ACM SIGMOD*, 517-522, 1993.

[10] Hock C. Chan, Bernard C.Y. Tan, and Kwok-Kee Wei. Three important determinants of user performance for database retrieval. *Int. Journ. Human-Computer Studies 51*, 895-918, 1999.

[11] Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. Cobase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 1996.

[12] Terry Gaasterland, Parke Godfrey, and Jack Minker. An Overview of Cooperative Answering. *Journal of Intelligent Information Systems 21:2*, 123-157, 1992.

[13] Sha Guo, Wei Sun, and Mark A. Weiss. Solving satisfiability and implication problems in database systems. *ACM Transactions on Database Systems 21*, 270-293, 1996.

[14] Chun-Nan Hsu and Craig A. Knoblock. Using inductive learning to generate rules for semantic query optimization. In: *Advances in Knowledge Discovery and Data Mining*, 425-445. AAAI/MIT Press, 1996.

[15] W.J. Kenny Jih, David A. Bradbard, Charles A. Snyder, and Nancy G.A. Thompson. The effects of relational and entity-relationship data models on query performance of end users. *Int. Journ. Man-Machine Studies*, 31:257-267, 1989.

[16] Antonija Mitrovic. A knowledge-based teaching system for SQL. In: *ED-MEDIA 98*, 1027-1032, 1998.

[17] Antonija Mitrovic, Brent Martin, and Michael Mayo. Using evaluation to shape its design: Results and experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, 12:243-279, 2002.

[18] Antonio Rizzo, Sebastiano Bagnara, and Michele Visciola. Human error detecting processes. *Int. Journ. Man-Machine Studies 27*, 555-570, 1987.

[19] Charles Welty. Correcting user errors in SQL. *International Journal of Man-Machine Studies 22:4*, 463-477, 1985.

[20] Jian Zhang, Chen Xu and S.-C. Cheung, Automatic Generation of Database Instances for White-box Testing. In: *Proc. of the 25th International Computer Software and Applications Conference (COMPSAC'01)*, Oct. 2001, IEEE Press, pp. 161-165.