Semantic Watermarks for Detecting Cheating in Online Database Exams

Stefan Brass[1] Alexander Hinneburg[2]

Abstract:

Due to the COVID-19 pandemic, we were forced to conduct two exams for a database course as online exams. An essential part of the exams was to write non-trivial SQL queries for given tasks. In order to ensure that cheating has a certain risk, we used several techniques to detect cases of plagiarism. One technique was to use a kind of "watermarks" in variants of the exercises that are randomly assigned to the students. Each variant is marked by small discrimination points that need to be included in submitted solutions. Those markers might go through undetected when a student decides to copy a solution from someone else. In this case, the student would reveal to know a "secret" that he cannot know without the forbidden communication with another student. This can be used as a proof for plagiarism instead of just a subjective feeling about the likelihood of similar solutions without communication. We also used a log of SQL queries that were tried during the exam.

Keywords: SQL; Plagiarism; Online Exams; Cheating; Academic Integrity

1 Introduction

The COVID-19 pandemic made classical, proctored exams for large classes impossible. Therefore online exams suddenly became interesting in spite of doubts that students might cheat. We limit our discussion to the case when only the communication between students during the exam must be prevented. Otherwise the exam is "open book".

It is often argued that the grade distribution of the online exam results is not very different from normal exams. Therefore, the percentage of good marks achieved by cheating cannot be very large. However, this argument disregards the possibility that students participate in the exams who are not well prepared. Maybe, they would have learnt more, or taken a later exam, if they had not relied on the possibility of cheating. Conversely, the first online exams were prepared without much experience, and the exam setting was also new for the students. It cannot be excluded that some good students got not so good grades. Whoever is responsible for an exam has the obligation to create a setting in which the honest students are not the fools.

The paper [Ja21] reports the results of an online questionaire answered by 1608 German students. 45.9% of the participants admitted „Exchanging ideas with others about possible

[1] Martin-Luther-University, Computer Science, Germany brass@informatik.uni-halle.de
[2] Martin-Luther-University, Computer Science, Germany hinneburg@informatik.uni-halle.de

doi:10.18420/BTW2023-30

answers during an examination“ at least once in an online exam. The studies cited in [Di03] are in the same range or higher (but may include homework assignments). The paper [DS17] proposes statistical tests to check whether cheating occurred in an exam based on the grade distribution, and also using the normal dependency of the grade on attendance of classes and the number of completed homework assignments. The results showed that cheating did occur in most of the investigated online exams. In [Eh21], conversely bad exam results of some students with good performance in the developed RA/SQL tutoring system were noticed, leading to the question of plagiarism in the homeworks. [Cl20] mentions exam questions appearing on commercial websites and being rapidly answered by tutors of these services (still during the exam). See also [LC15]. In our online exam in February 2022, we detected 14 students who participated in the exchange of solutions (out of 114, i.e. 12%).

Approaches that have been used to prevent cheating in online exams include software solutions like exam browsers and video supervision [FMG19, Ba17]. Since the student is at home and can prepare the setup, software solutions and video supervision do not really prevent cheating [Bi15]. Other counter measures against cheating include the preparation of large question pools from which questions are randomly assigned to individual exams. This prevents cheating between particular pairs of students who agreed to close collaboration during online exams. However, question pools have limited effects in larger collaborative groups of students. Furthermore, building such a question pool is a lot of work. Since questions of an online exam can easily be copied by the students, they also cannot be reused in the near future.

It is important that there is at least a certain risk involved in cheating to counter the negative consequences for academic integrity. Discussions among students in anonymous online forums show that most students thought that the risk is near zero and it would be stupid not to use the chances offered by online exams [An22]. A good method against cheating in online exams should fulfill the following criteria: (a) the method should increase the risk of being caught for the involved students, (b) it should increase the time needed for the students to cheat in an undetected way, (c) it should be easily applicable and the lecturer, who designs the exams, should not need to spend much additional time for creating and checking the exercises.

We propose a method called semantic watermarks that fulfills the above mentioned criteria. The method works for all types of tasks, where a written answer is required that must include specific semantic pieces from the task description, e.g. programming and design tasks.

The remainder of the paper is structured as follows: In Section 2, we discuss related work about cheating in online exams. Section 3 introduces the semantic watermark method in a formal way, discusses practical issues and ways to strengthen semantic watermarks. Section 4 describes applications of the method in real exams. The results on SQL queries are compared with those from a log-file analysis in Section 5. Section 6 gives some additional practical advice on what to do before and after the exam. Section 7 concludes the paper.

2 Related Work

So far, there has not been a lot of research specifically on SQL plagiarism detection (e.g., the literature overview [NJK19] contains no entry for SQL). The paper [RC05] suggests some methods for detecting "similar" queries, like comparing the queries with ignoring case and non-essential white space, and a "histogram" method that looks at specific seldom characteristics of the query, such as trailing (invisible) whitespace in the lines, or inconsistent case in keywords. The paper contains important ideas, but unfortunately, as also [KS19] notices, not enough details are given so that one can exactly reproduce the method. [KS19] have implemented the first two methods of [RC05], and tested them for a coursework dataset (with lots of plagiarism) and an exam dataset (with basically no plagiarism). It turned out that even in the exam dataset, many queries were classified as being plagiarized (and are therefore "false positives"). As [KS19] noted, this might be due to the relative short length of queries in both datasets. They suggest that for queries of over 200 characters, the algorithms might work, but they have not enough data to reach a definite conclusion. The queries in our exam have an average length of 250 characters, and 62% of the queries are at least 200 characters. Nevertheless, in our tests with such similary-based methods, either only very few of the actual cheaters are found, or so many quries are detected as similar that the actual cheaters are like a needle in a haystack (assuming that the methods presented in this paper give a good approximation of the "actual cheaters").

The teaching situation in [SSS18] is special, because he students have to develop also a database schema, which gives a much larger space of possible solutions. The methods of [MJ08, Si13] are applicable only for Microsoft Access (at least, students need to submit binary database files). The method of [DH05] also uses a "watermark" for detecting plagiarized Java programs, but needs to be able to change files on the computer of the student. A well-known plagiarism checker for Java programs is JPlag [PMP02]. A general literature overview on cheating in online exams is [NMA22]. Watermarks for relational data were studied, e.g., in [AHK03, La08, GA11]. During manual grading, copied solutions are often detected because of the same strange mistakes. For classifications of SQL mistakes and style/preformance problems, we refer to [BG06, TSV18, MAF21].

Different versions of exercises have been used for a long time, and there are also techniques where randomly chosen numbers are inserted into the task description of an exercise. Our "watermark method" works with parameterized task descriptions, too. However, we use this for detecting cheating attempts. Before, the main goal was that students get no points for an exercise if they copy a solution, because ideally, every student has a different exercise. One often cannot distinguish whether the student copied a solution or whether he simply made a mistake. In our case, the idea is that the watermarks appear in the submitted solution and can be used as a proof for cheating. Of course, if it was visible that a submitted solution was for a different exercise variant, a cheating attempt was detected even without having a fancy name for this method. However, our contribution still is to systematically define this method, give suggestions for not eye-catching differences, test the method in several exams, and compare it with other methods for detecting plagiarized SQL-queries.

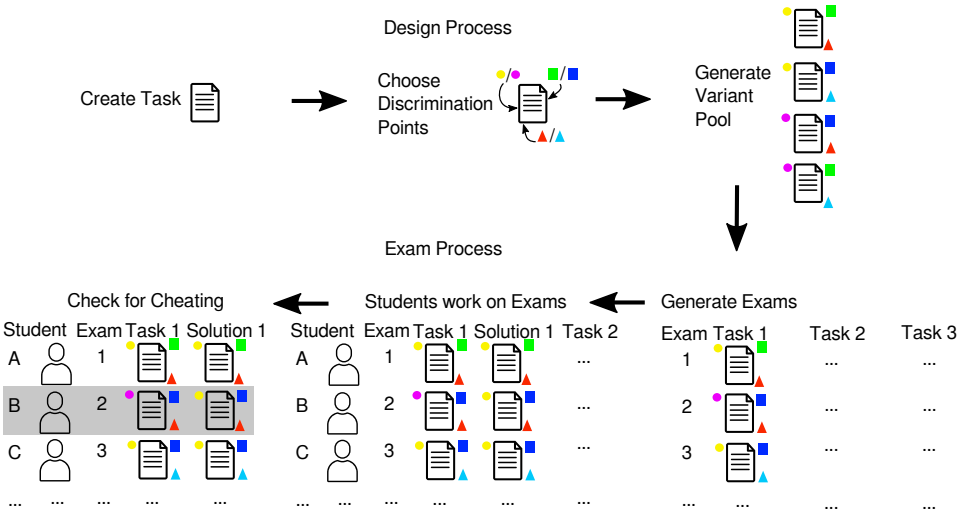

Fig. 1: Overview of the semantic watermark method

3 Semantic Watermarks

The method of semantic watermarks is sketched in Figure 1. At the beginning, the lecturer designs a specific task, e.g. the creation of an SQL query to retrieve specific data that is described in the task's text. In a next step, one or more discrimination points are chosen, which are short pieces in the task text that can be replaced by different values to generate other but sematically similar variants of the task. In Figure 1, the discrimination points are represented by different icon shapes and the different values for the discrimination points are represented by different colors. A discrimination point could be a constant that needs to be used in the `WHERE` clause. We discuss several criteria how to choose good discrimination points and the corresponding values. The discrimination points allow the generation of a pool of variants of the task that differ only at those points. Most online exam software packages like ILIAS or MOODLE allow to use such questions pools to randomly generate exams. A particular student sees only one of the variants of the task in the exam and submits a solution based on that particular textual task description. The discrimination points are designed in such a way that a complete solution needs to include values for all discrimination points mentioned in the task. Therefore, a student could be identified to cheat, when the solution does not match in the values of all discrimination points with the corresponding task description. In this case, the student would reveal to know information that he or she cannot know without the forbidden communication with another student (because the value was not contained in his/her exam version). In Figure 1, student B cheated in the submitted solution, because the yellow dot did not appear in corresponding task description.

3.1 Definition and Preconditions

The formal definition of the semantic watermarks method starts with a given task t that is described by a text or a picture. Given a task t, a number of n discrimination points needs to be chosen.

Each discrimination point is associated with a set of several values that are called the domain D_i for the respective point, $1 \leq i \leq n$. Each domain needs to have at least two different values: $|D_i| \geq 2$ for $1 \leq i \leq n$. The values in a particular domain should identify the corresponding discrimination point. However, `NULL` values are not allowed in those domains. The choice of discrimination points for the task t defines a mapping $\tau(d_1, \ldots, d_n)$ that is called the parameterized task. A parameterized task is a mapping that takes a vector $\vec{d} = (d_1, \ldots, d_n)$ from the cross product of the respective domains $D_1 \times \ldots \times D_n$ of the discrimination points. Such a vector $\vec{d}$ is called a discrimination vector. A parameterized task outputs a task description, which can be a text or a picture. Usually, the original task t equals the output of τ for some discrimination vector $\vec{d}$. The pool of variants of the parameterized task τ is generated by choosing a set of discrimination vectors and mapping them to texts or images via τ.

Each student sees the output t' of the parameterized task τ based on a particular discrimination vector. Based on this description she or he develops a solution s for t'. The solution s needs to include values for the discrimination points that forms another discrimination vector that is from the cross product of extended domains $\hat{D}_1 \times \ldots \times \hat{D}_n$ with $\hat{D}_i = D_i \cup \{\texttt{NULL}\}$ for $1 \leq i \leq n$. The original domains are extended by `NULL` values because a student may not submit a complete solution. Thus, certain aspects including some values for discrimination points might be missing. Given the discrimination vector $\vec{v}^t$ responsible for the task description shown to the student and the discrimination vector $\vec{v}^s$ included in his/her solution, a cheating attempt is detected if and only if there is a discrimination point i, $1 \leq i \leq n$, such that $v_i^t \neq v_i^s$ and $v_i^s \neq \texttt{NULL}$ (i.e. the values are both defined and different).

The definition of semantic watermarks requires at least one discrimination point. However, semantic watermarks with only a single discrimination point are not very useful, even when the number of values for this point is large. It is likely that students, who want to cheat, would spot this single point by chance and use the correct value for their variant of the task. Therefore, multiple discrimination points are preferable and would lower this chance. The number of values per discrimination point does not need to be large. In fact two different values would suffice, which would create binary discrimination points.

When a pool of variants of a task is generated, the discrimination vectors of the semantic watermark should be chosen from the cross product of the domains $D_1 \times \ldots \times D_n$ in such a way that the discrimination vectors have pairwise Hamming distances equal or larger than two. If the value for more than one discrimination point is wrong, the evidence is very strong. Furthermore, the larger the minimal pairwise Hamming distance, the lower the risk that cheating students find all discrimination points.

3.2 Practical Issues

The design of effective discrimination points requires to balance two conflicting goals: (i) the discrimination points need to have distinct and clearly different values that could not be mixed by chance and (ii) the discrimination points should not be striking and eye-catching to a casual observer. Furthermore, since students were shown the expected result of the query with respect to a given database state and could try every query, we choose the discrimination point values in such a way that they do not influence the query result for this particular state (at least not in an eye-catching way).

A simple case of a discrimination point is a condition of the form $A > c$. Then we could choose several constants $c_1 < c_2 < \cdots c_m$ as differentiating values, such that the example table contains no data values between c_1 and c_m. The constant c_i of a different exercise variant could be a good indicator of cheating if: (1) The constants are sufficiently different, i.e. differening characters should not be adjacent on the keyboard, and ideally the constants should differ in more than one character. (2) They should also not appear in the given database state, so that the student cannot know them. The differences are not so easy to spot if first and last character are the same. Also many-to-many relationships offer possibilities for differences in the query without differences in the result. E.g., if the query asks for the courses taken by a particular student, it might be that two students took the same courses.

"Redundant conditions" are another possibility to design discrimination points. Suppose the main query condition is C. Now variant j of the exercise ($1 \le j \le m$) could require in addition that the data satisfies C'_j. Therefore, the correct query condition is $C \wedge C'_j$. But if the given database state does not contain data satisfying $C \wedge \neg C'_j$, this condition does not change the answer in the test state. However, there is a tradeoff: This also means that the given example state does not test for the missing condition C'_j.

Another option to construct discrimination points are "`ORDER BY`" specifications, where two different result columns happen to give the same sequence of table rows. For instance, in one variant we could ask the students to order employees by `HIREDATE` and in another variant by `EMPNO`, when both cases lead to the same order of employees.

Result column names could be used as discrimination points (we used this a lot). In a similar way, the task might ask for string constants as part of the `SELECT`-clause that can be used as discrimination points. For instance, the students might have to construct a string with some fixed part and a data value with the string concatenation operator `||`. Or the query might require the `UNION` operator, and a discrimination point uses a fixed value for a result column (e.g., some kind of summary row). However, the differences between the different values of such a discrimination point should not be to small.

We checked for cheating attempts first with `grep`, then with SQL scripts using `LIKE` and `SIMILAR TO`. In both cases, the values for the discrimination points must be quite specific. Of course, with a full SQL parser (or manual checking), the range of possible values increases.

3.3 Risks, Costs, and Countermeasures

Even when a student is able to spot all discrimination points, this operation consumes some amount of time. Thus, cheating still comes at a certain cost. This cost might be larger than expected because students, who did not explicitly prepare for this method, will probably exchange only solutions to the exercises. Checking the solution of another person for small differences to the assigned task is much harder than comparing the task descriptions.

If the students know in advance that this method will be applied, they will also exchange the task descriptions and use programs like `diff` to spot the discrimination points. An effective counter measure for this behavior would be to render the task descriptions as non-textual images that could not be automatically compared by `diff` programs. Alternatively, one could add many unessential differences, such as doubling some spaces between words.

4 Applications in Online-Exams

We applied semantic watermarks in three exams for an introductory database course — two online exams with over a hundred participants each (in March 2021 and February 2022) and one smaller proctored exam (repetion exam in July 2022).

Besides the watermarks, there were two obviously different versions of each SQL exercise in both online exams. Probably, students expected different exercise versions, and maybe, after having detected the obvious differences, some did not search for additional, much smaller differences.

For space reasons, we show only the results of the exam in February 2022 (114 participants). We detected 7 cheating attempts of 6 students using the watermarks method:

Student	Task	DPs	Hamming Distance	DPs wrong	DPs adapted	DPs omitted	Group Size
A	SQL (NOT EXISTS)	3	2	3	0	0	15
B	SQL (GROUP BY)	2	2	2	0	0	16
C	Logical Design	4	2	1	1	0	7
D	Rel. Algebra	4	2	1	0	1	6
E	Rel. Algebra	4	2	1	1	0	5
F	Rel. Algebra	4	2	1	0	1	5
F	BCNF	3	2	2	0	0	15

The columns show the following data: “DPs” is the number of discrimination points used in the particular task, “Hamming Distance” is the minimal Hamming distance between two versions of the same task with respect the number of discrimination points, “DPs wrong” is the number of discrimination points in the student’s answer that do not match the task description, “DPs adapted” is the minimal number of discrimination points that the student

found and adapted to match his/her assigned task description, "DPs omitted" is the number of discrimination points that were omitted in the student's answer and last the "Group Size" column shows the number of students who worked on the same variant as the student with the cheating attempt but included discriminations points in the correct way.

Thus, six students were caught because they submitted solutions with wrong watermarks. Manual analysis revealed the original authors of the two SQL queries. One further student had a very similar solution to one that was known to be passed between students (he got the same exercise variant as the original author, therefore, one cannot say whether he would have detected the watermark). We did not try to search for the original authors of the non-SQL exercises. In these exercise types, there was less space for "personal style". If we had used more values for the discrimination points, so that each discrimination vector is used only very few times (ideally once), that would have helped to identify the original authors. Five more students were caught with the log file analysis explained in Section 5. In total, 14 students were detected as cheating (out of 114, i.e. 12%).

As a test for false alarms, we applied semantic watermarks in the SQL tasks of a proctored exam (repetition exam with 19 participants in July 2022). Due to proctoring in a single room, cheating was quite unlikely. There were actually two queries with wrong watermarks, however, in each case, only one out of two discrimination points was wrong. In the first case, the student wrote "Informatik" instead of "Bioinformatik". Since both are programs of study in our department, the discrimination point values were no secret in this case. In the second case, the student sorted by course title instead of the ID. Again, the columns are known to the student, and they might be intermixed. If a student might use a wrong discrimination value simply by mistake (without communication), the evidential value is small. However, two or more such wrong discrimination values would again be a strong evidence.

5 Verification by Log-File-Analysis

In the February 2022 exam, we worried that the students might already know the watermark method, because we applied it already in the exam of March 2021. Therefore, we added a different method: In all our exams, the students have the opportunity to test their queries with a given example database using a web-interface (Adminer). The same setup with a single "read only" database account for all students was used for the lab sessions and homeworks before. We told the students very clearly that (1) all accesses to our server during the exam are logged, and (2) the data will be analyzed for cheating attempts. However, it seems that many students did not forsee what actually can be done with the data. We matched the tried queries with the queries that were actually submitted in the exam. This gives the IP addresses used by many (not all) students. It might be that students share an IP address (e.g., if they live in the same student residence). However, if the same query Q (submitted by student S_1) is tried from two different IP addresses I_1 and I_2, and I_1 is used for other queries submitted by S_1, and I_2 is used for other queries submitted by S_2, it seems that S_1 passed his query Q to S_2, and S_2 tried it unchanged, and later modified it or decided to develop his/her

own solution. Of course, the query Q must be sufficiently complex so that it is extremely unlikely that the two students developed it independently.

We have no space to explain this method in detail, but it gives us a chance to verify the quality of the watermark method: (1) Are innocent students wrongly accused of cheating? (2) How many cheating students were not caught by the watermark method? The log analysis detected three clusters of students who exchanged SQL queries:

Cluster ID	Number of Students	Students with wrong Watermark	Students
1	3	1	B, G, H
2	3	1	D, I, J
3	5	1	A, K, L, M, N

In some cases, one can see in the log of an IP address, how the complex query Q suddenly appears and then is modified by changing the names of tuple variables, the case of SQL keywords, and so on. Even though it might be conceivable that S_1 and S_2 independently developed the same query Q, this process of obfuscation is a clear sign of guilt. In summary, the log analysis has hardened the case against the students who had the wrong watermark in their SQL queries, and made clearer who contributed to the cheating attempt by passing a solution.

6 Practical Advice

A common advice is "An ounce of prevention is worth a pound of cure" [Di03]. One certainly should make very clear before the exam that checks for cheating attempts will be done, and the consequences of being caught will be unpleasant. One must assume that more than a few students base the decision to cheat on a risk-benefit analysis. Everybody should understand that the risk is definitely not zero. It would also be unfair to set up a kind of "trap" for cheating students without informing them about such dangers. The goal must be that the students are more likely to overestimate the actual risks than to underestimate them.

In particular, it must clear that the professor really cares about academic integrity. The students must ask themselves whether they are able to outsmart the professor. If this paper is read by all of our students, they might be able to avoid being detected by these methods (although it would cost them at least some time). However, if a professor has once done things like the log analysis, there is the risk that other unforseen methods are used next time.

Of course, one must respect data privacy laws and should contact the legal department of the university to be safe. One cannot tell the students the exact methods that will be used for detecting cheating attempts in their exam. Probably any algorithm for plagiarism detection can be avoided by the students as soon as they know it. It can be considered ethically

questionable when the used methods are not completely transparent. But situations, in which somebody decides to break the rules just because he can do it, often have no ethically completely satisfying solutions — precisely because the rules do not help anymore.

It is also important how one communicates the discovery of the cheating attempt to the students. We first tried to keep the details of our method secret (so that we might apply it again). That probably was a mistake. When a student gets such a message, he/she must decide whether to admit breaking the rules or not. After denying the cheating attempt, one cannot later change one's mind without also admitting that one lied before (which is very difficult). Therefore, the full evidence should be given to the student from the beginning. We also got the advice (from a non-lawyer) to ask the student whether he/she wants us to deliver the evidence. The answer "yes" might be understood as an agreement for data processing. For data privacy laws, it is also important to inform each student in a separate email, not containing the names of any other students, even if that would be an important part of the evidence. This might also have the advantage that students can individually decide whether to admit the cheating attempt.

One criticism of our work was also that we catch the "small criminals", but so far we cannot do anything against really big cases of cheating — such as letting a paid expert write the entire exam. In future work, we plan to compare the programming style used in previous homeworks with the style in the exam (see also [MJ15, ORKS18]). It might be possible to do an additional oral exam (with full identification) in a few very suspicious cases. Of course, the possibility of such additional exams must be announced in advance.

7 Conclusions

Online exams seem to make cheating much simpler and less risky. However, due to the COVID-19 pandemic, they were often the only option. But even after the pandemic, new teaching methods and conveniences developed during the pandemic will remain: The expectations for online teaching are rising. If online exams could be done with trust in the integrity of the result, it would be possible to rethink the entire examination process. The techniques presented here might also be used for homeworks.

We used small variations in the exercises (nearly "invisible") to prove that a solution was copied (because it contained the wrong "watermark"). For SQL queries, we verified the detected cheating attempts with data from the log of SQL queries tried during the exam. This also showed that only about half of the forbidden communication between the students was detected with the watermark technique. However, when the watermark technique shows a case of plagiarism, we know that the query from another student was really used for the query that was finally submitted (not only tried and thrown away). An interesting aspect of our work is that we wrote nearly the entire analysis in SQL (including recursive queries). More information is available from: `[https://users.informatik.uni-halle.de/~brass/sqlplag/]`. There will also be a longer version of this paper on `[https://arxiv.org/]`.

Bibliography

[AHK03] Agrawal, Rakesh; Haas, Peter J.; Kiernan, Jerry: Watermarking relational data: framework, algorithms and analysis. The VLDB Journal, 12:157–169, 2003. https://doi.org/10.1007/s00778-003-0097-x.

[An22] Anonymous Authors: Betrug in Online-Klausuren (cheating in online exams), January 2022. https://www.wiwi-treff.de/Lernen-and-Klausuren/Taeuschungsversuch/Betrug-in-Online-Klausuren/Diskussion-88316.

[Ba17] Bawarith, Razan; Basuhail, Abdullah; Fattouh, Anas; Gamalel-Din, Shehab: E-exam Cheating Detection System. International Journal of Advanced Computer Science and Applications (IJACSA), 8(4):176–181, 2017. https://dx.doi.org/10.14569/IJACSA.2017.080425.

[BG06] Brass, Stefan; Goldberg, Christian: Semantic errors in SQL queries: A quite complete list. The Journal of Systems and Software, 79(5):630–644, 2006. https://doi.org/10.1016/j.jss.2005.06.028, https://dbs.informatik.uni-halle.de/sqllint/.

[Bi15] Binstein, Jake: On Knuckle Scanners and Cheating — How to Bypass Proctortrack, Examity, and the Rest. Blog, January 2015. https://jakebinstein.com/blog/on-knuckle-scanners-and-cheating-how-to-bypass-proctortrack/.

[Cl20] Clark, Ted M.; Callam, Christopher S.; Paul, Noel M.; Stoltzfus, Matthew W.; Turner, Daniel: Testing in the Time of COVID-19: A Sudden Transition to Unproctored Online Exams. Journal of Chemical Education, 97(9):3413–3417, 2020. https://pubs.acs.org/doi/pdf/10.1021/acs.jchemed.0c00546.

[DH05] Daly, Charlie; Horgan, Jane: A Technique for Detecting Plagiarism in Computer Code. The Computer Journal, 48(6):662–666, 2005. https://doi.org/10.1093/comjnl/bxh139, https://www.researchgate.net/publication/31104286.

[Di03] Dick, Martin; Sheard, Judy; Bareiss, Cathy; Carter, Janet; Joyce, Donald; Harding, Trevor; Laxer, Cary: Addressing student cheating: definitions and solutions. ACM SIGCSE Bulletin, 35(2):172–184, June 2003. https://doi.org/10.1145/782941.783000.

[DS17] D'Souza, Kelwyn A.; Siegfeldt, Denise V.: A Conceptual Framework for Detecting Cheating in Online and Take-Home Exams. Decision Sciences Journal of Innovative Education, 15(4):370–391, 2017. https://doi.org/10.1111/dsji.12140, https://www.researchgate.net/publication/319967019.

[Eh21] Ehrlinger, Christina; Fritsch, Thomas; Fruth, Michael; Lehner, Franz; Scherzinger, Stefanie: Toolunterstützung für den Übungsbetrieb in der Datenbanklehre: Erfahrungen mit der Software Praktomat (Tool support for database labs: Experiences with the Software Praktomat). Datenbank-Spektrum, 21:91–98, 2021. https://doi.org/10.1007/s13222-021-00374-y.

[FMG19] Foltýnek, Tomáš; Meuschke, Norman; Gipp, Bela: Academic Plagiarism Detection: A Systematic Literature Review. ACM Computing Surveys, 52(6), 2019. https://doi.org/10.1145/3345317.

[GA11] Gross-Amblard, David: Query-preserving watermarking of relational databases and Xml documents. ACM Transactions on Database Systems, 36(1):1–24, 2011. https://doi.org/10.1145/1929934.1929937.

[Ja21] Janke, Stefan; Rudert, Selma C.; Petersen, Änne; Fritz, Tanja M.; Daumiller, Martin: Cheating in the wake of COVID-19: How dangerous is ad-hoc online testing for academic integrity? Computers and Education Open, 2(100055), 2021. https://doi.org/10.1016/j.caeo.2021.100055.

[KS19] Kleerekoper, Anthony; Schofield, Andrew: The False-Positive Rate of Automated Plagiarism Detection for SQL Assessments. In: UKICER: Proceedings of the 1st UK & Ireland Computing Education Research Conference. ACM, pp. 1–6 (Article No.: 6), 2019. https://doi.org/10.1145/3351287.3351290, https://www.researchgate.net/publication/335480205.

[La08] Lafaye, Julien; Gross-Amblard, David; Constantin, Camelia; Guerrouani, Meryem: Watermill: An Optimized Fingerprinting System for Databases under Constraints. IEEE Transactions on Knowledge and Data Engineering, 20:532–546, 2008. https://doi.ieeecomputersociety.org/10.1109/TKDE.2007.190713.

[LC15] Lancester, Thomas; Clarke, Robert: The implications of Plagiarism and Contract Cheating for the Assessment of Database Modules, 2015. https://www.slideshare.net/ThomasLancaster/the-implications-of-plagiarism-and-contract-cheating-for-the-assessment-of-database-modules-teaching-learning-and-assessment-of-databases-2015.

[MAF21] Miedema, Daphne; Aivaloglou, Efthimia; Fletcher, George: Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. In: Proceedings of the 17th ACM Conference on International Computing Education Research. ACM, pp. 355–367, 2021. https://doi.org/10.1145/3446871.3469759.

[MJ08] McCart, James A.; Jarman, Jay: A Technological Tool to Detect Plagiarized Projects in Microsoft Access. IEEE Transactions on Education, 51(2):166–173, 2008. https://ieeexplore.ieee.org/document/4455466.

[MJ15] Mirza, Olfat; Joy, Mike: Style Analysis for Source Code Plagiarism Detection. In: Plagiarism across Europe and Beyond — Conference Proceedings. European Network for Academic Integrity (ENAI), pp. 53–61, 2015. https://academicintegrity.eu/conference/proceedings/2015/Mirza_Style.pdf.

[NJK19] Novak, Matija; Joy, Mike; Kermek, Dragutin: Source-code Similarity Detection and Detection Tools Used in Academia: A Systematic Review. ACM Transactions on Computing Education, 19(3):1–37, 2019. https://doi.org/10.1145/3313290.

[NMA22] Noorbehbahani, Fakhroddin; Mohammadi, Azadeh; Aminazadeh, Mohammad: A systematic review of research on cheating in online exams from 2010 to 2021. Education and Information Technologies, 27:8413–8460, 2022. https://doi.org/10.1007/s10639-022-10927-7.

[ORKS18] Opgen-Rhein, Julia; Küppers, Bastian; Schroeder, Ulrik: An Application to Discover Cheating in Digital Exams. In (Joy, Mike; Ihantola, Petri, eds): Proc. of the 18th Koli Calling International Conf. on Computing Education Research. ACM, pp. 1–5 (Article No.: 20), 2018. https://doi.org/10.1145/3279720.3279740.

[PMP02] Prechelt, Lutz; Malpohl, Guido; Phlippsen, Michael: Finding Plagiarisms among a Set of Programs with JPlag. Journal of Universal Computer Science, 8(11):1016–1038, 2002. https://doi.org/10.3217/jucs-008-11-1016.

[RC05] Russell, Gordon; Cumming, Andrew: Online Assessment and Checking of SQL: Detecting and Preventing Plagiarism. In: 3rd Workshop on Teaching, Learning and Assessment in Databases (TLAD), in 22nd British National Conference on Databases. HE Academy for Information and Computing Sciences, pp. 1–6 (Article No.: 6), 2005.
https://www.researchgate.net/publication/252140116.

[Si13] Singh, Anil: Detecting Plagiarism in MS Access Assignments. Journal of Information Systems Education, 24(3):177–180, 2013.
https://jise.org/Volume24/n3/JISEv24n3p177.html.

[SSS18] Scerbakov, Nikolai; Schukin, Alexander; Sabinin, Oleg: Plagiarism Detection in SQL Student Assignments. In (Auer, Michael E.; Guralnick, David; Simonics, Istvan, eds): Teaching and Learning in a Digital World, Proc. of the 20th Conf. on Interactive Collaborative Learning (ICL). Springer, pp. 321–326, 2018.
https://www.researchgate.net/publication/319942726.

[TSV18] Taipalus, Toni; Siponen, Mikko; Vartiainen, Tero: Errors and Complications in SQL Query Formulation. ACM Transactions on Computing Education, 18(3):1–29, 2018.
https://doi.org/10.1145/3231712.