Actes de l'atelier "Apprendre la Pensée Informatique de la Maternelle à l'Université"
Brest, France, June 13, 2023

Enseigner SQL en NSI

Emmanuel Desmontils Laura Monceaux
LS2N, 2 rue de la Houssinière, 44322 Nantes, France
emmanuel.desmontils@univ-nantes.fr, laura.monceaux@univ-nantes.fr

RÉSUMÉ

Dans cet article, nous présentons une démarche systématique d'exploration des situations didactiques qui peuvent être envisagées pour enseigner SQL en lycée. En effet, partant du constat que les propositions, aussi bien dans la littérature, les manuels scolaires ou les ressources en ligne, se contentent de proposer des exercices uniquement de la forme "écrire une requête qui recherche...", nous cherchons à diversifier les types de questions. A la suite de ces réflexions, nous développons une application web permettant à un enseignant de proposer des exercices sur diverses bases de données.

ABSTRACT

Teaching SQL

In this article, we present a systematic approach to exploring the didactic situations that can be considered to teach SQL in high school. Indeed, starting from the observation that proposals, both in literature, textbooks or online resources, offer exercises only in the form "write a query that find...", we proposes to diversify the types of questions. Following these reflections, we are developing a web application allowing a teacher to offer exercises on various databases.

MOTS-CLÉS : Didactique en informatique, Bases de données, SQL, NSI.

KEYWORDS: Computer Science Didactic, Databases, SQL.

1 Introduction

L'enseignement des bases de données est présent dans les curricula universitaires depuis très longtemps et, depuis 2019, constitue un chapitre à part entière du programme de NSI [1] de terminale ((MENJ, 2019), annexe page 5). On constate que l'enseignement s'axe sur la compréhension et l'usage des bases de données en général ainsi que sur le questionnement et la modification des données d'une base de données relationnelle. La modélisation d'une base de donnée n'est pas au programme de cette formation, contrairement à toutes les initiations aux bases de données de l'enseignement universitaire. C'est regrettable, car la modélisation permet de mieux comprendre la structure du modèle relationnel, d'améliorer l'efficacité des requêtes SQL produite et de rendre plus autonomes les élèves dans le cadre de la mise en place des projets (part importante des activités de NSI).

Une difficulté particulière en bases de données est le langage SQL utilisé pour exploiter et manipuler les données. C'est un nouveau langage pour les élèves de NSI, mais, en plus, c'est un nouveau paradigme. En effet, le langage de requête SQL n'est pas très complexe en soit, surtout sur le programme de terminale, mais ce changement de paradigme peut troubler voire poser des problèmes

1. "Numérique et Sciences Informatiques" : spécialité qui peut être choisie par des lycéens (cycle secondaire général) en France pour une durée hebdomadaire de 4h en première et de 6h en terminale

de compréhension fondamentaux (Sadiq et al., 2004; Ahadi et al., 2015). Nous proposons donc un ensemble de onze situations didactiques visant à faciliter la découverte et l'apprentissage de ce langage.

Dans la suite de cet article, nous aborderons d'abord l'enseignement de SQL (section 2). Puis, nous présenterons une catégorisation des situations didactiques pour enseigner SQL (section 3). Enfin, nous présenterons l'outillage associé que nous développons (section 4) avant de conclure.

2 Enseignement de SQL

SQL (Structured Query Language) est un langage apparu en 1974 permettant de décrire, renseigner et interroger une base de données relationnelle. SQL comporte quatre parties : le langage de définition de données (LDD), le langage de manipulation de données (LMD), le langage de contrôle de données (LCD) et le langage de contrôle des transactions (LCT). Comme mentionné dans (MENJ, 2019), seule la partie LMD est au programme NSI, allant des requêtes de mise à jour à l'interrogation d'une base de données.

Les travaux autour de l'enseignement de SQL se situent sur deux axes :

- l'étude des erreurs selon des critères qualitatifs (Sadiq et al., 2004; Reisner, 1981; Smelcer, 1995; Miedema et al., 2021) et quantitatifs (Ahadi et al., 2015; Reisner, 1981) mettant en évidence des difficultés linguistiques, syntaxiques (SQL) et sémantiques.
- l'organisation d'entrainements ou d'évaluations avec des réflexions sur l'organisation des activités (entrainements ou évaluation) ou la manière dont saisir les requêtes SQL.

C'est sur ce deuxième axe que notre attention s'est portée. Les travaux abordent peu les types de question que l'on peut présenter pour ces activités. (Reisner, 1981) propose six tâches utilisées pour "mesurer la facilité d'utilisation d'un langage de requête" : "query writing", "query reading", "query interpretation", "question comprehension", "memorization" (être capable de mémoriser et reproduire une base de données) et "problem solving" (étant donné un problème et une base de données, proposer les intentions pour résoudre en problème). Cependant, l'auteur ne développe pas leurs caractéristiques et propriétés, son travail étant axé sur la comparaison des différents langages de requête de l'époque (SQL, QBE, SQUARE & TABLET) pour des développeurs professionnels.

Nous proposons dans cet article, une catégorisation en onze situations didactiques pour permettre l'enseignement de SQL pour des élèves de NSI (voire des étudiants de premier cycle universitaire) où les quatre premières tâches de (Reisner, 1981) sont présentes. Il est nécessaire de considérer ces situations didactiques pour enseigner SQL au regard de la complexité des opérateurs. Plusieurs travaux ont montré la difficulté de certains opérateurs comme le distinct, la jointure (en particulier l'auto-jointure), les sous requêtes (en particulier synchronisées) ou les regroupements (hors NSI) (Kearns et al., 1997; Kleerekoper and Schofield, 2018; Mitrovic, 1998; Renaud and Van Biljon, 2004; Ahadi et al., 2015; Miedema et al., 2021). Il faudra donc considérer que la complexité des situations didactiques que nous allons présenter sera pondérée par les opérations relationnelles qui seront mises en œuvre. En combinant la complexité des situations avec la complexité des opérateurs, il est alors possible de proposer un ordre des questions d'un exercice.

3 Situations didactiques en Base de Données

L'enseignement des bases de données n'est pas récent. Cependant, force est de constater que les exercices proposés ont souvent la même forme : "Donnez une requête qui ...". Dans cette section, nous allons proposer une classification des questions possibles sur SQL. Pour illustrer celle-ci, nous utiliserons la base de données de (Chrisment et al., 2008) (p.62) qui permet de recenser les fleuves et les espaces maritimes dont le modèle relationnel est présenté ci-dessous (les clé primaires sont soulignées et la clés étrangères précédées de #) :

- Pays(id_p,nom_p, superficie, nbhab) ;
- EspaceMaritime(id_em, nom_e, type) ;
- Fleuve(id_f, nom_f, longueur, #id_p, #id_em) ;
- Cotoie(#id_p, #id_em) ;
- Parcourt(#id_p,#id_f, distance).

Trois éléments entrent en jeu lors dans la construction de questions sur SQL :

- l'intention (I) visée par la requête : texte en langage naturel décrivant ce que l'on cherche dans la base de données ;
- la table (T) résultat de la requête : ensemble des tuples ;
- la requête SQL qui peut prendre deux formes : complète (R) ou partielle (r).

L'*intention* donne le besoin fonctionnel de la recherche (ce que l'on veut) et non une traduction en langage naturel de la requête. Une intention est donc un élément clé lors de la construction et la manipulation de la base de données. Elle participe à la construction du schéma (on doit pouvoir faire...) et, bien évidemment, elle est implémentée sous la forme d'une (voire plusieurs) requêtes SQL permettant d'extraire des informations de la base vers l'application. A noter que (Reisner, 1981) nomme cela *question*, mais, dans notre contexte, il y a ambiguïté sur ce terme avec les questions d'un exercice qui englobe l'intention, les attendus, etc. Par exemple, une intention liée à la base de données exemple peut être la phrase : "Le nom des fleuves de longueur supérieure à 1000 Km". La requête partielle répondant à cette intention pourrait être [2] :
"Select ? ? ? ? ? From Fleuve Where longueur > 1000 ;"

A partir des trois éléments, nous avons considéré les différentes combinaisons possibles en faisant varier ce qui est proposé à l'élève et ce qui est attendu de l'élève. Nous avons mis en évidence onze situations didactiques potentiellement intéressantes dont font partie quatre situations de (Reisner, 1981). Nous avons choisi de coder chaque situation sous la forme "Entrées"→"Sortie", où chaque information peut être un élément de l'interrogation d'une base de données : I, T, R ou r. Les entrées sont les informations données à l'élève et la sortie est l'information qu'il devra produire. Quand une entrée est entre crochet, celle ci est optionnelle. Par exemple, "[I]R→T" décrit une situation où l'intention (éventuellement) et la requête sont données à l'élève et celui ci doit donner la table résultat. Pour certaines situations, il est possible d'ajouter une étape "⇒T" qui permet d'obtenir le résultat en exécutant la requête sur le SGBD-R, permettant à l'élève de valider sa production en sortie.

Dans la suite de cette section, nous supposerons que l'élève a connaissance du modèle relationnel et, en cas d'attente de la table résultat, un exemple d'instance de la base de données. Nous évaluerons les différents cas au regard de la complexité de la situation.

2. "? ? ? ?" est la partie à compléter pour répondre à la question.

3.1 Différentes situations didactiques proposées

Transformer un besoin fonctionnel en une requête SQL : I[T]→R

Cette situation didactique est, de loin, la plus courante dans la littérature, les cours en lignes, etc (voir section 3.2). Sans la table, c'est la situation "query writing" de (Reisner, 1981). L'enseignant propose une intention I à l'élève, accompagnée ou non de la table résultat T, et ce dernier doit trouver une requête R qui permet d'y répondre. Il doit être capable de transformer un besoin fonctionnel en une requête SQL.

Par exemple, on peut donner à l'élève l'intention suivante « les fleuves de longueur supérieure à 1000 Km » et il devra fournir la requête suivante en sortie :
"Select nom_f From Fleuve Where longueur > 1000 ;"

Attention, il faut, dans cette situation, insister sur la vérification de la requête. Ce n'est pas parce qu'une requête donne la table résultat attendue T qu'elle est bonne ! De même, ce n'est pas parce qu'une requête ne donne pas de résultat qu'elle est fausse. Donc, il faut proposer des situations qui suivent le déroulement IT1→R⇒T2 avec T2=T1 quand c'est possible.

Mais d'autres situations didactiques pourraient être mises en place pour l'élève avant d'être capable de transformer le besoin fonctionnel en une requête SQL.

Traduire le résultat d'une intention : I→T

Dans cette version, plus simple que la précédente, on demande la table résultat T d'une intention I. C'est la situation "question comprehension" de (Reisner, 1981). Elle est un peu utilisée chez (Balabonski et al., 2020) (voir section 3.2). L'élève n'a pas à connaitre SQL, mais doit proposer les tuples que la requête SQL répondant à l'intention devrait retourner. Cette situation permet d'amener l'élève à traduire un besoin en un ensemble d'actions (parfois informelles). Ces actions servent ensuite à l'enseignant pour introduire le langage SQL et ses différents opérateurs.

Pour faciliter le travail de l'élève, on pourrait décomposer les situations présentées. Par exemple, I→T peut être un travail préalable à la forme I[T]→R, ce qui donnerait I→T→R. L'enseignant demande d'abord la table réponse T à une intention I avant de demander la requête elle-même R. L'élève peut ainsi prendre le temps de comprendre la base de données avant de construire la requête. Cette solution peut être intéressante dans le cas de requête ne donnant aucun résultat. Les élèves peuvent avoir un blocage sur ce type de requête pourtant correcte. Cependant, pour les convaincre, il faut sans doute revenir sur le schéma précédent à savoir I→T1→R⇒T2 (en espérant que T1=T2). Cette démarche peut être généralisée à la vérification de toute situation de ce type.

Comprendre le fonctionnement des opérateurs utilisés par une requête : [I]R→T

La situation suivante est de mettre à disposition de l'élève une requête SQL bien formée R. L'élève doit, à partir d'une instance de la base de données qui lui est proposée, donner la table résultat T de cette requête. Sans l'intention, c'est la situation "query interpretation" de (Reisner, 1981). Cette situation est un peu utilisée chez (Balabonski et al., 2020) (voir section 3.2). L'objectif est de l'amener à comprendre le fonctionnement des opérateurs utilisés par la requête. Le motif de cette situation est "Soit la base de données suivante, donner le résultat de la requête... [qui recherche...]". Par exemple, "Donner la table résultat de la requête suivante qui recherche les fleuves de longueur supérieure à 1000 Km" pour la requête "Select..." (forme IR→T)

Cette situation est aussi intéressante pour amener l'élève à comprendre l'architecture de la base de

données relationnelle. Il faut proposer des requêtes simples, mais aussi des jointures clé-primaire/clé-étrangère pour l'amener à "parcourir" les tables. Dans cette situation, comme dans la précédente, l'enseignant peut amener l'élève à vérifier sa réponse en suivant la forme : IR→T1, R⇒T2 en espérant T1=T2.

Dans cette famille de situations, il est aussi possible de proposer une requête et plusieurs tables possibles. L'élève doit choisir celle qui correspond au résultat de la requête. Cette situation est plus facile que la précédente et peut être proposée en amont.

Découvrir les différents opérateurs disponibles dans une requête SQL : Ir→R & Tr→R

Cette situation est assez classique en programmation, beaucoup moins en bases de données. Elle est totalement absente des ouvrages consultés et est présente rarement en ligne (voir section 3.2). L'idée est de proposer une requête à trous. L'élève doit proposer des solutions. La requête seule est une situation assez difficile, et pas nécessairement intéressante. Il faut donc l'accompagner soit de l'intention I, soit de la table résultat T. Cette approche permet de découvrir les différents opérateurs disponibles dans une requête SQL.

La situation suivante propose de découvrir simplement la projection en proposant pour l'intention "Les fleuves qui se jettent dans une mer" la requête suivante à compléter : "Select ? ? ? ? ? From Fleuve Where longueur > 1000 ;"

Selon le niveau des élèves, il est possible de compliquer un peu cette situation en proposant aussi les opérateurs inutilisés. La place libre doit alors le rester. Par exemple, sur l'exemple précédent, la situation devient : "Select ? ? ? ? ? From Fleuve Where longueur > 1000 Order By ? ? ? ? ? ;"

On peut aussi fournir la table résultat T et la requête précédente (sans l'intention), mais cela est assez délicat. Il est assez complexe de trouver la bonne requête, juste en ayant à sa disposition le résultat, car, bien évidemment, plusieurs requêtes peuvent donner le même résultat.

Comprendre le besoin fonctionnel d'une requête : R[T]→I

Sans la table, c'est la situation "query reading" de (Reisner, 1981). Nous avons ici une situation un peu particulière. Une requête R est proposée (éventuellement avec la table résultat T) et les élèves doivent trouver l'intention I sous-jacente. L'objectif est d'amener les élèves à comprendre le besoin fonctionnel d'une requête pour éventuellement la corriger. La présence de la table (ou la possibilité donnée à l'élève d'exécuter la requête) permet d'aider à trouver cette intention. Cette table peut-être fournie dans un second temps en cas de difficulté à trouver la bonne intention (aide progressive).

En pratique, cette situation n'est pas évidente à mettre en place, car beaucoup d'élèves vont formuler la requête en langage naturel et non énoncer son intention. Par exemple, pour la requête "Select nom_f From Fleuve Natural Join Espace-Maritime Where type = 'mer' ", les élèves diront *"on fait la jointure entre Fleuve et Espace-maritime, on ne garde que les mers et on projette le nom du fleuve."* au lieu d'exprimer l'intention *"Nom des fleuves qui se jettent dans une mer"*.

Se trouver dans une situation où il y a ambiguïté : T→R

Cette situation est particulièrement complexe à gérer dans le cas général, pour l'élève comme pour l'enseignant. Il faut vraiment se trouver dans une situation où il n'y a pas d'ambiguïté. Une table résultat T est proposée et l'élève doit trouver la (une ?) requête permettant de l'obtenir. Souvent plusieurs solutions existent, et il n'est pas toujours simple de les anticiper. Devant la machine, il est possible de permettre à l'élève de vérifier sa proposition : T1→R⇒T2, T1=T2. Cette situation peut être scénarisée par un travail individuel initial sur plusieurs tables, suivi d'un débat dans la classe

permettant d'appréhender mieux la structure de la base de données et les besoins fonctionnels.

3.2 Situations didactiques utilisées dans les ouvrages et les applications

Au regard des propositions que nous avons faites, nous avons parcouru les ouvrages disponibles aux épreuves de CAPES NSI de 2021. Pour les exercices en SQL, nous constatons, comme beaucoup de sites sur le Web, que la très grande majorité des questions (90% à 100%) est de la forme I→R (Table 1). (Balabonski et al., 2020) est un peu plus original en proposant d'autres formes (30%) : R→T et R→I.

Ouvrage \ Situation	R-T	R-I	IT-R	I-R
(Connan et al., 2020)		1		9
(Bonnefoy and Petit, 2020)			1	10
(Balabonski et al., 2020)	5	4		20
(Bays, 2020)				68

TABLE 1 – Types de questions dans les ouvrages à destination des élèves

Des outils ont été proposés pour aider à la formation à SQL : SQL-Tutor (Mitrovic, 1998), SQLator (Sadiq et al., 2004), AsseSQL (Prior and Lister, 2004), SQL-KnoT (Brusilovsky et al., 2008, 2010), SQLSandbox (Desmontils, 2010), BlocklySQL (Pöhner et al., 2019) ou SQheLper (Jacobs and Jaschke, 2021) par exemple. Cependant, ces outils permettent de construire des requêtes SQL (pour certains en programmation par blocs) sur des questions qui restent sur le format "Donnez la requête qui ..." (I→R). Leur effort porte sur la construction de la requête, l'organisation de tests, l'évaluation ou la vérification des requêtes proposées par les étudiants. Le même constat peut-être fait sur une très grande majorité de cours en ligne pour NSI ou d'autres niveaux qui proposent des exercices de la forme I→R. Nous proposons donc un nouvel outil pour enseigner SQL qui couvre les différentes situations présentées.

4 L'application Apprendre-SQL

Notre objectif est de proposer une application (appelée Apprendre-SQL [3]) pour faire des exercices (ensemble de questions) plus variés qu'avec les autres outils existants. Une question est associée à une requête (éventuellement choisie dans une base de requêtes) selon une des situations didactiques vues en section 3.

Un enseignant peut installer une base de données (fichier SQL compatible avec Sqlite) mais aussi la décrire, proposer une base de requêtes SQL et constituer des exercices dont chacune des questions correspondra à une situation didactique particulière sur une requête de sa base. Dans un exercice, les questions sont présentées dans un ordre de complexité tenant compte de la situation choisie et de la complexité intrinsèque de la requête travaillée.

Quand l'élève réalise un exercice, il a, selon la situation didactique de la question, des retours instantanés. Par exemple, si la requête proposée par l'élève ne donne pas le résultat attendu (situation

3. `https://gitlab.univ-nantes.fr/ls2n-didactique/asql`

de la forme [X]Y→R), l'outil lui précise que la sortie attendue est mauvaise. A l'issu de l'exercice, deux traces sont produites. La première trace restitue toutes les actions réalisées par l'élève afin d'étudier sa manière d'aborder chaque question et donc l'apprentissage du SQL. La seconde correspond uniquement aux résultats finaux produits par l'élève que l'enseignant récupère et corrige.

5 Conclusion

Dans ce papier, nous avons présenté un ensemble de situations didactiques pour introduire le langage SQL. Ces situations permettent de développer chez l'élève des compétences liées à la manipulation d'une base de donnée. Les situations proposées sont axées sur la partie recherche du langage.

Comme pour les requêtes de recherche, il est possible de proposer des situations d'enseignement pour des modifications de la base de données. L'objectif, en plus de faire comprendre le mécanisme de ces requêtes, est de sensibiliser à la gestion de la cohérence de la base de données. Cela nécessite une bonne compréhension du modèle relationnel et des différentes contraintes, pas toujours visibles sur le modèle lui-même.

Il existe plusieurs référentiels de compétences en informatique comme dans (Selby and Woollard, 2013) ou même dans le bulletin officiel (annexe à (MENJ, 2019)). Nous avons commencé à explorer les compétences spécifiques des différentes situations présentées, mais cela reste encore à préciser. On constate en première analyse, et c'est rassurant, que le standard I→R est la modalité qui semble la plus complète. Cependant, cela montre aussi que, pour mettre en place de la progressivité, ce n'est pas forcément judicieux de commencer par cette catégorie.

Les situations didactiques présentées peuvent être mise en œuvre en NSI, mais aussi en premier cycle universitaire pour des mises à niveau, aussi bien par l'application "Apprendre-SQL" que, plus simplement, dans le cadre d'exercices de travaux dirigés sans machine. Nous avons commencé une expérimentation informelle dans cette dernière modalité en mise à niveau en 3e année de licence MIAGE (20 étudiants) sur l'algèbre relationnelle. Les premiers retours des étudiants (en particulier ceux ayant déjà eu une introduction aux bases de données) sont très positifs. D'autres expérimentations (avec et sans l'application) vont être lancées en classe de NSI et en licence à la rentrée 2023.

Références

Ahadi, A., Prior, J., Behbood, V., and Lister, R. (2015). A quantitative study of the relative difficulty for novices of writing seven different types of sql queries. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 201–206.

Balabonski, T., Conchon, S., Filliâtre, J.-C., and Nguyen, K. (2020). *Numérique et sciences informatiques : 24 leçons avec exercices corrigés*. Ellipses.

Bays, S. (2020). *Spécialité Numérique et sciences informatiques*. Ellipses.

Bonnefoy, J.-C. and Petit, B. (2020). *Numérique et sciences informatiques Tle*. Ellipses.

Brusilovsky, P., Sosnovsky, S., Lee, D. H., Yudelson, M., Zadorozhny, V., and Zhou, X. (2008). An open integrated exploratorium for database courses. *AcM SIGcSE Bulletin*, 40(3) :22–26.

Brusilovsky, P., Sosnovsky, S., Yudelson, M. V., Lee, D. H., Zadorozhny, V., and Zhou, X. (2010). Learning SQL programming with interactive tools : From integration to personalization. *ACM Transactions on Computing Education (TOCE)*, 9(4) :1–15.

Chrisment, C., Pinel-Sauvagnat, K., Teste, O., and Tuffery, M. (2008). *Bases de données relationnelles : Concepts, mise en oeuvre et exercices*. Collection Informatique. Hermes Science Publications.

Connan, G., Petrov, V., Rozsavolgyi, G., and Signac, L. (2020). *Prépa bac, Spécialité NSI, Tle générale*. Hatier.

Dagiene, V., Sentance, S., and Stupuriené, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, 28(1) :23–44.

Desmontils, E. (2010). SQLSandbox. `http://www.desmontils.net/SQLSandbox/`.

Jacobs, S. and Jaschke, S. (2021). SQheLper : A block-based syntax support for SQL. In *2021 IEEE Global Engineering Education Conference (EDUCON)*, pages 478–481. IEEE.

Kearns, R., Shead, S., and Fekete, A. (1997). A teaching system for SQL. In *Proceedings of the 2nd Australasian conference on Computer science education*, pages 224–231.

Kleerekoper, A. and Schofield, A. (2018). SQL tester : an online SQL assessment tool and its impact. In *Proceedings of the 23rd annual ACM conference on innovation and technology in computer science education*, pages 87–92.

MENJ (2019). Programme de l'enseignement de spécialité de numérique et sciences informatiques de la classe terminale de la voie générale. Bulletin officiel spécial 8, Ministère de l'Éducation nationale et de la Jeunesse.

Miedema, D., Aivaloglou, E., and Fletcher, G. (2021). Identifying SQL misconceptions of novices : Findings from a think-aloud study. In *Proceedings of the 17th ACM Conference on International Computing Education Research (ICER 2021)*. ACM.

Mitrovic, A. (1998). Learning SQL with a computerized tutor. In *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, pages 307–311.

Pöhner, N., Schmidt, T., Greubel, A., Hennecke, M., and Ehmann, M. (2019). BlocklySQL : A new block-based editor for SQL. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education*, pages 1–2.

Prior, J. C. and Lister, R. (2004). The backwash effect on SQL skills grading. *ACM SIGCSE Bulletin*, 36(3) :32–36.

Reisner, P. (1981). Human factors studies of database query languages : A survey and assessment. *ACM Computing Surveys (CSUR)*, 13(1) :13–31.

Renaud, K. and Van Biljon, J. (2004). Teaching SQL—which pedagogical horse for this course ? In *Key Technologies for Data Management : 21st British National Conference on Databases, BNCOD 21, Edinburgh, UK, July 7-9, 2004. Proceedings 21*, pages 244–256. Springer.

Sadiq, S., Orlowska, M., Sadiq, W., and Lin, J. (2004). SQLator : an online SQL learning workbench. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 223–227.

Selby, C. and Woollard, J. (2013). Computational thinking : the developing definition. University of Southampton (E-prints).

Smelcer, J. B. (1995). User errors in database query composition. *International Journal of Human-Computer Studies*, 42(4) :353–381.