Database Schemas Used in SQL University Courses – State of the Art

Georgi Tuparov(✉)

New Bulgarian University, 21 Montevideo str., Sofia 1618, Bulgaria
gtuparov@nbu.bg

Abstract. The body of knowledge in IT courses area includes topics about relational databases in which students have to obtain at least basic knowledge and skills in SQL. Also for the students in the areas of computer science majors SQL is an obligatory part of knowledge and skills. To be effective, SQL training in university courses has to use appropriate case studies (i.e. database schemas) which are semantically clear, simple, and suitable to demonstrate the major SQL functionality not only "on paper" but also in real DBMS environment (or DBMS sandbox). Moreover, the appropriate database state is important to demonstrate understandable results from SQL queries, which can be manually evaluated by students to compare with results of execution in real database environment.

In this paper some popular database schemas and states used in SQL training in university level courses are analyzed from these points of view. For this study, six schemas from popular database books used in university courses are considered. Two major streams in database schemas and states are distinguishable. One is focused on overall solution, that covers database design topics and SQL training, another is focused only on SQL training. It is found that a possible break point between these streams is to use database schema which is complex enough for demonstration of the design process, and after that to use simplified version of the database schema with reduced number of records for SQL training.

Keywords: Database schema · SQL · University courses

1 Introduction

Nowadays more and more students want to obtain knowledge and skills in Information technology (IT) as a part of their academic study. In general, the body of knowledge in IT courses area includes topics about relational databases in which students have to obtain basic knowledge and skills in SQL. For the students in the areas of computer science majors SQL is an obligatory part of knowledge and skills as is stated in ACM Curricula Recommendations [1].

In the last ten years we have a stable trend of rising number of young people in IT majors and minors. Lecturers in the database field now understand that part of the students simply follow the needs of more and more specialists in IT, without being deeply involved in the area. Moreover, contemporary development environments provide very

T. Zlateva and R. Goleva (Eds.): CSECS 2022, LNICST 450, pp. 196–204, 2022.
https://doi.org/10.1007/978-3-031-17292-2_16

rich tools to access, query and maintain relational databases, so students are not aware of the importance of SQL knowledge and skills.

Another general problem in nowadays teaching is that students spend more and more time in social networks preferably using smartphones. They believe that all information needed is available on the Internet and this – in combination with social distancing during COVID-19 crisis, changes the students' perception in ways of learning. In consequence these Internet channels of communication should be used to provide effective learning materials in appropriate formats. Practice during COVID-19 crisis showed that the simple usage of e-learning environments is not enough to increase the effectiveness of teaching and the quality of learning materials is a key point to achieve learning goals.

Focusing on SQL training, students have to be provided with appropriate case studies (i.e. database schemas) which are semantically clear, simple, and can be used to demonstrate the major SQL functionality not only "on paper" but also in real DBMS environment (or DBMS sandbox). Moreover, the appropriate database state is important to demonstrate understandable results from SQL queries, which can be easily evaluated manually by students to compare with results of execution in the DBMS. This will contribute to a deeper understanding of SQL functionality and some side effects in particular SQL implementation. Also, it will be very useful if the database schema used during the semester is used in quizzes and exams, eventually with minor changes.

2 Methodology Used

2.1 Collecting Data and Study Points

For this study six schemas from popular database books used in university courses are considered. Also some good database books are excluded due to lack of case study database schema and/or state used in examples of applying SQL to solve problems.

The database schemas and states study points are:

- Is this a case study in the book? Is the schema used in topics of conceptual modeling, relational model introduction and SQL examples?
- Database schema quality according to good practices – naming of attributes, primary and foreign key used, normalization.
- Is the schema and state usable to demonstrate major SQL functionality?
- Is the schema and state suitable to solve complex SQL queries?
- Is the schema and state useful for manual evaluation of SQL queries?
- Is DDL script provided for practical training?

2.2 Textual Representation of Relational Schemas

In this paper the following textual notation is used to describe relational DB schemas:

– Relation name – first letter is capital, i.e. Person.
– List of attributes are enclosed in parenthesis – (personID, name, phone)
– Primary key attribute is underlined – personID.

– Where FK is a foreign key in R1 relation, R2.PK is referred primary/candidate key i.e. FK → R2.PKey
– If the foreign key is composite, its attributes are enclosed in brackets [], i.e. R1.[FK1, FK2] → R2.[PK1, PK2].

3 Database Schemas and States Included in the Research

3.1 Supplier Database [2]

Supplier database [2] is the case study in the entire book and consists of three relations.

Relation **S** represents suppliers. Each supplier has a unique supplier number (sno), which is the primary key of this relation, and personal data like supplier's name and residence. Also a status value, representing ranking (preference level) among available suppliers is stored in the relation.

Relation P represents parts. Each part has a unique part number (pno), which is the primary key of the relation; and part's description properties like a name, color, weight; and city where the part is stored.

Relation SP represents shipments, which show which part is supplied (or shipped), by which supplier. It is assumed that one supplier can supply/ship a particular part only once. Each shipment has a supplier number (sno), a part number (pno), which are both foreign keys, corresponding to the primary keys of S and P, respectively and quantity of this part being shipped. Primary key is composite (sno, pno).

The database schema described in textual manner mentioned above is presented in Fig. 1:

S (sno, sname, status, city)
P (pno, pname, color, weight, city)
SP (sno->S.sno, pno->P.pno, qty)

Fig. 1. Supplier database schema [2] with foreign keys' references added

Database sample state used in the examples contains 23 records, five in S table, six in P table and 12 in SP table. One supplier doesn't have any shipments.

The database schema and state provided are very suitable to demonstrate full functionality of SQL. Naming conventions like using the same name for foreign key attributes as referred attributes, and database state allow to demonstrate full JOINs functionality. Simplicity of the schema is a problem for creating more complex queries, but is suitable for middle level training. Schema state allows easy manual evaluation of the queries. DDL script is not provided, but it is not a problem to implement database schema and state manually in relational DBMS environment.

3.2 Company Database [3]

Company Database [3] is used as case study in the entire book and implements the following business rules:

- The company has divisions, called departments. Each department has a unique name and number, several locations, and a particular employee who manages the department.
- One department may manage several projects, which are described with a unique project number and name, and location of project activities.
- The database stores each employee's personal data, including Social Security number and employee's supervisor (who is another employee). An employee can be assigned to only one department, but may be engaged in several projects, which may not be managed by the same department.
- The number of hours per week that an employee works on each project is logged into the database.
- The database keeps track of the dependents of each employee for insurance purposes, including each dependent's personal data and relationship to the employee.

Database schema consists of six relations and is presented in textual manner in Fig. 2:

Employee (fname, minit, lname, ssn, bdate, address, sex, salary,
super_ssn -> Employee.ssn, dno -> Department.dnumber)
Department (dname, dnumber, mgr_ssn->Eemployee.ssn, mgr_start_date)
Dept_location (dnumber->Department.dnumber, dlocation)
Project (pname, pnumber, plocation, dnum->Department.dnumber)
Works_on (essn->Employee.ssn, pno->Project.pnumber, hours)
Dependent (essn->Employee.ssn, dependent_name, sex, bdate, relationship)

Fig. 2. Company database schema [3] with foreign keys' references added

The schema represents interesting cases as weak entity type (Dependent relation), self-reference (super_ssn in Employee relation), projection of the composite attribute (fname, minit, lname in Employee relation) and flattening of the multivalued attribute (Dept_location relation). Foreign keys' names follow the conventions for different names with referred attributes.

The database state used in the SQL examples contains 40 records: eight in Employee table, three in Department, six in Projects, 16 in Works_on, and seven in Dependent. DDL script for schema creation and population with data is not provided, but it is not a problem to create a script using the book.

In general, the database schema and state are suitable to present major SQL functionalities, but foreign key naming convention doesn't allow some JOINs to be demonstrated. In fact, the database schema is not so complex, but in combination with the number of the records in the state it makes the manual evaluation of not so complex queries difficult. In this way, the results obtained from SQL query execution in DBMS will be not so easy to understand and check.

3.3 SaleCo Database [4]

SaleCo database [4] is used in the entire book and DDL scrips are provided for MS Access, MS SQL Server, MySQL and Oracle DBMS. The database model implements the following business rules:

- The database keeps data about customers, vendors, invoices and products.
- A customer may have many invoices, but one invoice belongs exactly to one customer.
- An invoice contains one or more invoice lines, and each invoice line belongs exactly to one invoice.
- Each invoice line consists of one product, and a particular product may be part of many invoice (as invoice line).
- A vendor may supply many products. Vendors may be added in the database before supplying any product yet.
- Products may be not supplied by a vendor, but if a product is vendor-supplied, it is supplied by exactly one vendor.

The database schema consists of five relations (Fig. 3):

Customer (cus_code, cus_lname, cus_fname, cus_initial, cus_areacode,
cus_phone, cus_balance)
Invoice (inv_number, cus_code->Customer.cus_code, inv_date)
Vendor (v_code, v_name, v_contact, v_areacode, v_phone, v_state, v_order)
Product (p_code, p_descript, p_indate, p_qoh, p_min, p_price, p_discount,
v_code->Vendor.v_code)
Line (inv_number->Invoice.inv_number, line_number, p_code->Product.p_code,
line_units, line_price)

Fig. 3. SaleCo database schema [4] with foreign keys' references added

The database sample state used in the examples contains 63 records, 10 in Customer table, eight in Invoice table, 11 in Vendor table, 16 in Product table, and 18 in Line table. Part of the customers don't have any invoices; part of the vendors also don't have any product supplied; some of the products are not vendor supplied and/or not ordered yet.

The SaleCo database schema and state are suitable to demonstrate major SQL functionality. Also complex queries can be demonstrated. Although the database schema is not complex, the number of the records in the state makes manual evaluation of complex (and not so complex) queries difficult and the reasons for the results obtained from SQL query execution in DBMS will be not so clear.

3.4 University Database [5]

University database [5] is a case study in the entire book. DDL script is provided for creation of the database, as well as two versions of scripts for population with data – short, used in book examples, and extended with more records. The database model implements the following business rules:

- One instructor belongs to zero or one department and one department may have zero or more instructors.
- One student may enroll in zero or more sections and may be attached to not more than one department. Also students may have advisor (instructor) attached.

- One course may have zero or more sections and may be attached to not more than one department.
- Each course may have zero or more courses as prerequisites.
- One section belongs to exactly one course and exactly one timeslot.

The database schema consists of 11 relations (Fig. 4) with a total of 137 records in the short version of the database state used in illustrative examples in the book:

Classroom (building, room_number, capacity)
Department (dept_name, building, budget)
Course (course_id, title, dept_name->Department, credits)
Instructor (ID, name, dept_name->Department, salary)
Section (course_id->Course.course_id, sec_id, semester, year,
[building, room_number]->Classroom.[building,room_number], time_slot_id)
Teaches (ID->Instructor.ID, [course_id, sec_id, semester, year]->
Section.[course_id, sec_id, semester, year])
Student (ID, name, dept_name->Department.dept_name, tot_cred)
Takes (ID->Student.ID, [course_id, sec_id, semester, year]->
Section.[course_id, sec_id, semester,year], grade)
Advisor (s_id->Student.ID, i_id->Instructor.ID)
Time_slot (time_slot_id, day, start_time, end_time)
Prereq (course_id->Course.course_id, prereq_id->Course.course_id)

Fig. 4. University database schema [5] with foreign keys reference added

The database schema is complex and is not easy to be understood, given the number of relations and complex foreign keys. Composite foreign keys are also not suitable for SQL coding. Moreover, not a single naming convention is used for foreign keys' names – one part follows the convention of the same names, another part – convention of usage of different names with corresponding primary/candidate keys which is a bad practice. Manual evaluation is not easy and in some cases could be not applicable for complex queries.

3.5 Movies DB [6]

Movies database [6] is used as a base case study example in the entire book. In fact, the entire database state is not included in the book and the examples are illustrated by partial states according to particular example. Movies database implements the following business rules:

- One movie is owned by one studio and has one producer.
- One movie star may have roles in zero or more movies.
- A movie executive may be a producer or studio president and has a certificate number.
- A studio may own zero or more movies and has exactly one president.

Database schema (Fig. 5) consists of 5 relations:

Movies (title, year, length, genre, studioName->Studio.name,
producerC#->MovieExec.cert#)
MovieStar (name, address, gender, birthdate)
StarsIn (movieTitle->Movies.title, movieYear->Movies.year,
starName->MovieStar.name)
MovieExec (name, address, cert#, netWorth)
Studio (name, address, presC#->MovieExec.cert#)

Fig. 5. Movies database schema [6] with foreign keys reference added

According to a free accessible DDL file in GitHub [7], the Movies database sample state contains 35 records: 10 in Movie table, five in MovieExec table, seven in MovieStar table, eight in StarsIn table, and five in Studio table. Part of the customers don't have any invoices; part of the vendors also don't have any product supplied; some of the products are not vendor supplied and/or not ordered yet.

Generally, the database schema is simple and understandable. Foreign keys' names don't follow any of the naming conventions, mentioned above, which is a bad practice. The database schema and state allow to demonstrate major functionality of SQL and complex queries can be created. The small number of the records in the state is suitable for manual evaluation of the results.

3.6 SCO Database [8]

SCO database [8] is used as a case study example in the entire book. SCO is an abbreviate for Salespeople-Customers-Orders, which are the names of the relations included in this schema (Fig. 6).

Salespeople (snum, sname, city, comm)
Customers (cnum, cname, city, rating, snum->Salespeople.snum)
Orders (onum, amt, odate, snum->Salespeople.snum, cnum->Customers.cnum)

Fig. 6. SCO database schema [8] with foreign keys reference added

Relation Salespeople denotes suppliers. Each salesperson has a unique number (snum), which is the primary key of this relation; name (sname); rating (rating) in orders; and location (city), where this salesperson resides.

Relation Customers denotes customers. Each customer has a unique number (cnum), which is a primary key of the relation; name (cname); rating (rating), representing some kind of ranking level among available customers; location (city), where this customer resides; and the number of the salesperson (snum) assigned to this customer.

Relation Orders denotes orders placed by the customers and fulfilled by the salespeople. Each order has unique number (onum); number of the customer (cnum), placed the order; number of the salesperson (snum), fulfilled the order; order date (odate); and amount (amt) of the order. It is not obligatory to the customers to place orders only to the salesperson attached.

The SCO database state used in the book examples consists of six records in Salespeople relation, eight records in Customers, and ten records in Orders. A DDL script is not provided, but the creation and population of the database is not a problem, according to its simplicity and small number of records used.

Schema and state are very simple, but suitable for full SQL functionality demonstration. An interesting point here is a relationship between Salespeople and Customers which allows creation of complex queries too. The simplicity of the schema and state makes manual evaluation very simple and supports better understanding of the results obtained.

4 Discussion

My teaching practice shows that students prefer to execute the SQL query in real DBMS environment (or DBMS sandbox) and after that to check the obtained result manually. Sometimes they don't understand why the obtained result differs from their manual evaluation. This situation makes simplicity of the schema and state a key point in effectiveness of SQL training. Another key point is the miniworld [3] presented in the database schema which has to be understandable for the students.

In fact, the database courses don't include only SQL training part, and the usage of one case study during the entire course is very suitable and (possibly) effective. In fact, such kind of overall solution suffer from some weaknesses. In example, to present process of conceptual modeling and relational mapping, a complex case is needed and as result the relational schema designed will be too complex. A good example with real data will fill designed database with many records. In opposite, the database schema and state for SQL training have to be simple and small to be effective for manual evaluation and will increase understanding of SQL semantics and execution.

Four of the discussed database schemas and states [3–6] are used as a case study to represent database design topics and for SQL training. The miniworlds presented in these databases are understandable by the students. Part of them [3, 5] suffer from database schema and state complexity which could reduce the effectiveness of SQL training. In particular, the database schema in [5] is too complex because of the composite foreign keys and could be simplified using surrogate keys in foreign keys constraints. Database schema and state presented in [4, 6] don't have such kind of weaknesses in general.

Database schemas and states presented in [2, 8] are focused mainly on SQL training. The miniworlds presented are simple and the semantic of the queries is easy to understand and manual evaluation is simple. Despite this simplicity, these database schemas and states can be used for training with complex queries.

5 Conclusions

In the books included in this study two major streams in database schemas and states are distinguishable. One is focused on overall solution, that covers database design topics and SQL training, another is focused only on SQL training. As is found in the considered schemas, overall solutions could suffer from complexity when they have to be used for SQL training. In the opposite, schemas focused only on SQL training are too simple for

database design and implementation topics. The possible solution is to use miniworld that will produce database schema which is complex enough for demonstration of the design process, and after that to use simplified version of the database schema with reduced number of records for SQL training.

References

1. ACM Curricula Recommendations. https://www.acm.org/education/curricula-recommendations. Accessed 31 May 2022
2. Date, C.J.: SQL and Relational Theory, 3rd edn. O'Reilly Media Inc. (2015)
3. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 7th edn. Pearson Education (2016)
4. Coronel, C., Morris, S.: Database Systems. Design, Implementation, & Management, 13th edn. Cengage Learning Inc. (2017)
5. Silberschatz, A., Korth, H.F., Sudarshan, S.: Database System Concept, 7th edn. McGraw Hill Education (2020)
6. Garcia-Molina, C., Ullman, J., Widom, J.: Database Systems: The Complete Book, 2nd edn. Pearson Education (2014)
7. GitHub. https://github.com/stelf/fmi-db/blob/master/scripts/SchemaMovies-MSSQL.sql. Accessed 31 May 2022
8. Gruber, M.: Mastering SQL, Sybex Inc. (2000)