Using a Semi-Realistic Database to Support a Database Course

Kwok-Bun Yue
Department of Computer Information Systems
University of Houston-Clear Lake
Houston, Texas, USA
yue@uhcl.edu

ABSTRACT

A common problem for university relational database courses is to construct effective databases for instructions and assignments. Highly simplified 'toy' databases are easily available for teaching, learning, and practicing. However, they do not reflect the complexity and practical considerations that students encounter in real-world projects after their graduation. On the other hand, production databases may contain too much domain nuances and complexity to be effectively used as a learning tool. Sakila is a semi-realistic, high quality, open source, and highly available database provided by MySQL. This paper describes the use of Sakila as a unified platform to support instructions and multiple assignments of a graduate database course for five semesters. Based on seven surveys with 186 responses, the paper discusses our experience using Sakila. We find this approach promising, and students in general find it more useful and interesting than the highly simplified databases developed by the instructor, or obtained from textbooks. We constructed a collection of 124 problems with suggested solutions on the topics of database modeling and normalization, SQL query, view, stored function, stored procedure, trigger, database Web-driven application development with PHP/MySQL, Relational Algebra using an interpreter, Relational Calculus, XML generation, XPath, and XQuery. This collection is available to Information Systems (IS) educators for adoption or adaptation as assignments, examples, and examination questions to support different database courses.

Keywords: Database design & development, Data modeling, Database management systems (DBMS), Normalization, Query language

1. INTRODUCTION

A common problem for university relational database courses is to construct effective databases for instructions and assignments. Limited by spaces and with the focus to provide easy-to-understand examples on core concepts, database textbooks usually contain highly simplified databases which are regarded by many as 'toy' databases (Elmasri and Navathe, 2010; Gillenson, 2011; Hoffer, Ramesh and Topi, 2013; Jukić, Vrbsky and Nestorov, 2013; Pratt and Adamski, 2011; Rob and Coronel, 2007; Watson 2005). Although effective in illustrating underlying principles, these simple examples do not reflect the complexity and practical considerations that are common and crucial in real-world problems. They would "under no circumstance prepare the students for the true feel and experience of the massive data sets and the database problems they will need to cope with once they graduate and work in the real world" (Jukić and Gray 2008).

However, using real-world production databases to teach database concepts is also not without challenges. Their intricate domain constraints, intrinsic complexity, and extensive practical considerations of technical, managerial, and organizational issues create a significant barrier for students to understand the problem. Their complexity can obscure the underlying principles and hinder effective learning, especially for novices. Students need to overcome a steep learning curve before being able to work on examples and assignments based on them. Thus, a well-designed 'semi-realistic' database with the right balance of complexity represents an appropriate compromise. It can allow students to quickly practice relational theory and techniques, while working in an environment resembling their future professional careers. This paper describes our five-semester experience in using Sakila, a well-designed and highly available semi-realistic database provided by MySQL, to support instructions and multiple assignments in a graduate database course. It describes a set of 128 Sakila-based problems we developed in diverse topics in databases. They can selectively be used to support different database courses. Our surveys show that this semi-realistic approach is both promising and effective.

The paper is organized as follows. Section 2 provides the background literature research. Section 3 describes Sakila and the database course. Section 4 describes the set of course materials with examples. Section 5 discusses our learned experience based on the survey results of studying the

effectiveness of using such a semi-realistic database. We draw our conclusions in Section 6.

2. BACKGROUND AND LITERATURE RESEARCH

Many IS educators have the impression that the vast majority of database examples and assignments used in database textbooks and courses tend to be highly simplified. This simplified approach helps to minimize complexity and highlight core principles to students. To evaluate the validity of this impression, we studied the Teradata University Network, a popular resource for supporting various kinds of database courses for both faculty members and students (Teradata University Network, 2013). It provides the Teradata Database DBMS, which is accessible via Teradata SQL Assistant through the Web (Jukić and Gray, 2008). Teradata Database DBMS contains a collection of databases used in various popular DBMS textbooks. We analyzed all textbook databases through Teradata SQL Assistant. In the collection, there are databases with large amount of overlapping schema and data. When given a choice, we selected the versions with the larger sizes. We obtained and analyzed information on 13 databases from the most recent versions of six textbooks (Gillenson, 2011; Hoffer, Ramesh and Topi, 2013; Jukić, Vrbsky and Nestorov, 2013; Pratt and Adamski, 2004; Rob and Coronel, 2007; Watson 2005). Table 1 contains the result. Note that the database db_watson contains all databases used in Watson (2005). Its 24 tables include at least seven independent databases and we selected only the largest one for inclusion in Table 1: qsale.

Database textbook	Database	# Tables	# Records	Average # records per table
Gillenson, 2011	db_fdms_ba	4	52	13.0
	db_fdms_qam	6	102	17.0
	db_fdms_hcl	6	71	11.8
	db_fdms_sbl	6	55	9.2
Hoffer, Ramesh and Topi, 2013	db_pvfc11_std	15	86	5.7
Jukić, Vrbsky and Nestorov, 2013	db_jukic_hafhmore	9	159	17.7
	db_jukic_zagimore	8	131	16.4
Pratt and Adamski, 2011	db_gts_pp	5	39	7.8
	db_gts_hb	6	173	28.8
	db_gts_amg	5	46	9.2
Rob and Coronel, 2007	db_robcor_ch07	8	113	14.1
	db_robcor_ch08	10	131	13.1
Watson 2005	db_watson: qsale	6	131	21.8
Average		**7.2**	**99.2**	**14.3**
Median		**6**	**102**	**13.1**

Table 1. Number of tables, number of records, and average numbers of records per table in 13 databases from six database textbooks from Teradata University Network

We identified several common characteristics:

- The databases are indeed small.
- An analysis of the relational schemas revealed that they are highly simplified and similar to the classical and popular employee/department/project and student/course/enrollment domains (Wagner, Shoop and Carlis, 2003).
- All databases contain only the basic tables and indices. None of them contains any useful advanced RDBMS features such as views, stored procedures, stored functions, or triggers.
- The textbooks use multiple databases as examples and exercises.

This confirms the common brief that databases from textbook tend to be based on 'toy' applications. On the other hand, IS educators also believe that highly simplified data domains do not adequately prepare students for the complexity of real world projects that they face after graduation (Jukić and Gray 2008, Wagner, Shoop and Carlis 2003). For this purpose, a common approach is to incorporate database projects based on real-world problems, such as partnering with industrial companies (Seyed-Abbassi, King and Wiseman, 2007), using realistic scientific datasets (Wagner, Shoop and Carlis, 2003), or reverse engineering existing databases (Enciso and Soler, 2013). Although these semester long projects are effective in providing realistic capstone experience in databases, these real-world domains are designed for production, and not for instructional purposes. They may contain too much complexity and domain knowledge for students to use efficiently as learning platforms to apply their database knowledge in design, programming, and query languages. Like many other IS educators, we attempted to incorporate realistic databases of the appropriate complexity into our courses. In particular, we sought to identify and use a balanced, semi-realistic database as a common vehicle for examples and assignments in diverse topics in relational databases: design and modeling, query languages, SQL, and database-driven Web development. Eventually, we selected and used Sakila for five semesters in a graduate database course.

3. SAKILA

Sakila is a sample database included in the standard installation package of MySQL, a popular and industrial strength open source DBMS (Sakila, 2013a; MySQL, 2013). It was "intended to provide a standard schema that can be

used for examples in books, tutorials, articles, samples, and so forth. Sakila sample database also serves to highlight the latest features of MySQL such as Views, Stored Procedures, and Triggers" (Sakila, 2013b). Because of the popularity of MySQL, Sakila enjoys good documentation and community support. It has been used for widely different purposes, such as testing and verification of MySQL (Cobb, et al., 2011), as exercises and test cases (Sakila, 2013b), and as sample datasets in database research (Bourennani, Guennoun and Zhu, 2010). Sakila is designed to support a DVD film rental application. Figure 1 shows the visual relation schema of Sakila (2013a). Table 2 shows the relations with attributes. Most of the tables are easy to understand and work with. Table 3 summarizes the basic statistics of Sakila we tabulated together.

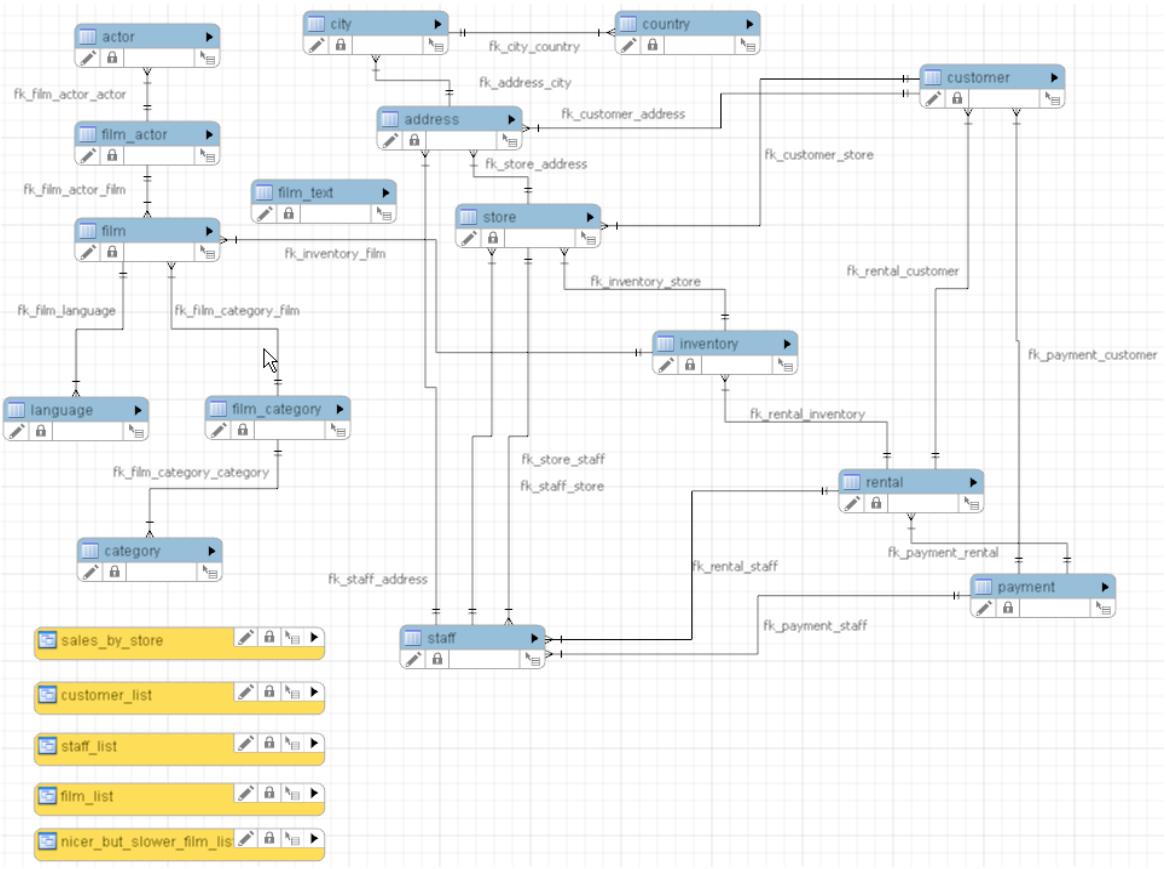


Figure 1. Visual Sakila Schema

Sakila Relation Schema
actor(actor_id, first_name, last_name, last_update)
address(address_id, address, address2, district, city_id, postal_code, phone, last_update)
category(category_id, name, last_update)
city(city_id, city, country_id, last_update)
country(country_id, country, last_update)
customer(customer_id, store_id, first_name, last_name, email, address_id, active, create_date, last_update)
film(film_id, title, description, release_year,language_id, original_language_id, rental_duration, rental_rate, length, replacement_cost, rating, special_features, last_update)
film_actor(actor_id, film_id, last_update)
film_category(film_id, category_id, last_update)
film_text(film_id, title, description)
inventory(inventory_id, film_id, store_id, last_update)
language(language_id, name, last_update)
payment(payment_id, customer_id, staff_id, rental_id, amount, payment_date, last_update)
rental(rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, last_update)
staff(staff_id, first_name, last_name, address_id, picture, email, store_id, active, username, password, last_update)
store(store_id, manager_staff_id, address_id, last_update)

Table 2. Sakila's relations and their attributes

Entities	Number
Tables	16
Total number of records	46,076
Average number of records per table	2,958
Medium number of records per table	585
Total number of foreign keys	23
Number of key indices (not counting primary keys)	25
Views	7
Stored Procedures	3
Stored Functions	3
Triggers	6

Table 3. Basic statistics of Sakila

Comparing to the textbook databases in Table 1, Sakila is significantly larger, more complex, and more realistic. Furthermore, it contains advanced RDBMS features, including views, stored procedures, stored functions, and triggers.

At the University of Houston-Clear Lake, all graduate students in Computer Information Systems (CIS) and Computer Science (CS) are required to take a graduate DBMS course. The course covers the usual topics in relational database, including modeling, design, querying, and normalization. It also includes other advanced topics such as Web database development, XML, etc. It is a little more technical than the typical database course in MIS. Our search for a suitable database of balanced complexity led us to experiment with Sakila.

4. USING SAKILA IN THE DATABASE COURSE

We started to use Sakila to support our course in the fall semester of 2011. Although it is also used in lecture examples and examinations, the main emphasis is to support assignments to provide students with realistic practices on relational theory and techniques. Table 4 summarizes our steadily increasing uses of Sakila in our homework assignments from one in fall 2011 to five in fall 2013.

Topic	Semester				
	F11	S12	F12	S13	F13
Modeling, Design and Normalization	√	√	√		
SQL Query			√	√	√
SQL View, Triggers, Stored Functions and Procedures	√	√	√	√	√
PHP MySQL Web Application		√	√	√	√
Full Text Search					√
Data Mining and Pattern Recognition				√	√
Relational Algebra					√
Relational Calculus					√
XML Generation by Scripts		√			
XPath and XQuery		√	√	√	√
Number of assignments based on Sakila	**1**	**3**	**4**	**4**	**5**
Total number of assignments in the semester	**8**	**8**	**8**	**8**	**9**
Percentage of assignments based on Sakila	**12.5%**	**37.5%**	**50%**	**50%**	**55.5%**

Table 4. Sakila-based assignments on different topics in the five semesters from Fall 2011 to Fall 2013

We refined and organized our Sakila-based problems from assignments, examinations, and examples into a collection. Every problem has a suggested solution. Table 5 summarizes the number and the perceived level of difficulty of the problems in various topics. The problem collection is available to IS educators through the Journal of Information Systems Education (JISE) and can easily be adapted as examples, assignments, examination questions, or quizzes to support different database courses.

Topic	Number of Problems			
	Easy	Medium	Hard	Total
Modeling, Design and Normalization	-	2	2	4
SQL Query	11	13	6	30
SQL View	5	5	-	10
SQL Stored Functions	7	1	2	10
SQL Stored Procedures	=	4	1	5
SQL Trigger	2	2	-	4
PHP MySQL Web Application	5	4	2	11
Relational Algebra	2	3	3	8
Relational Calculus	4	6	6	16
XML Generation by Scripts	-	2	-	2
XPath	12	6	3	21
XQuery	2	4	1	7
Total	**50**	**52**	**26**	**128**

Table 5. Numbers and levels of difficulties of Sakila-based problems

We briefly describe the problems in different topics with the examples below.

4.1 Modeling, Design and Normalization

Sakila documentation includes a structure diagram showing tables and their foreign key relationship. It does not include an ER diagram modeling the data requirements and assumptions. A significant problem is to reverse engineer a suitable data model using either ER diagram or UML class diagram in a way similar to Enciso and Soler (2013). Other problems are constructed to conduct normalization analysis and table redesign to enhance performance. An example of a problem of a medium level of difficulty on normalization analysis is described below.

> *Problem.* [Medium] Is the table film in first normal form? Why? If it is not in 1NF, describe a redesign to remove the violation of 1NF.

The problem requires the students to study the film table and find out that one of the columns, special_features, has a data type of set with four possible elements: "Trailers, Commentaries, Deleted Scenes, Behind the Scenes." Because the value is not atomic, it violates the first normal form. In pure relational theory, the use of any aggregate data type violates the first normal form. The problem highlights the need of balance between practical considerations in real-world database and theoretic consideration based on the relational model. This is usually absent in 'toy' databases. The redesign to eliminate 1NF violation includes creations of new tables such as special_feature and film_feature. The table redesign in this problem does not involve triggers but other redesign problems may need triggers.

4.2 SQL Query

A strong background in SQL query is crucial in any database course. Sakila provides plenty of opportunities to construct interesting query problems of various levels of difficulties and focuses. The following example shows that even a conceptually simple problem may involve many tables, six in this case.

> *Problem.* [Easy] List all actors who have appeared in a film rented by the customer "MARY SMITH". Note that the result should be sorted in ascending order of the first name and then last name of the actors.

Suggested Solution.

```
SELECT DISTINCT CONCAT(A.first_name, ' ',
   A.last_name) AS "ACTOR"
FROM ACTOR A JOIN FILM_ACTOR FA
        ON A.actor_id = FA.actor_id
 JOIN FILM F ON FA.film_id = F.film_id
 JOIN INVENTORY I ON F.film_id = I.film_id
 JOIN RENTAL R ON I.Inventory_id =
R.inventory_id
 JOIN CUSTOMER C ON R.customer_id =
C.customer_id
WHERE C.last_name = "SMITH"
AND C.first_name = "MARY"
ORDER BY A.first_name, A.last_name;
```

As another example of a more difficult query, the following problem requires the students to study the MySQL manual to learn to use the results of additional SELECT statements inside the FROM clause and the WHERE clause of a SELECT statement.

> *Problem.* [Hard] List the names of the actor with a number of film appearances within 10 that of the number of film appearances of the most prolific actor. In our relation instance, the most prolific actor, "GINA DEGENERRES", appears in 42 films. Thus, the query should list all actors that appear in 33 or more films.

```
+--------------------+-----------------+
| ACTOR              | Number of films |
+--------------------+-----------------+
| GINA DEGENERES     |              42 |
| WALTER TORN        |              41 |
…
| RIP CRAWFORD       |              33 |
| CAMERON ZELLWEGER  |              33 |
+--------------------+-----------------+
```

Suggested Solution.

```
SELECT afc.ACTOR, afc.N_FILMS AS "Number
of films"
```

```
FROM
 (SELECT DISTINCT CONCAT(A.first_name, ' ',
          A.last_name) AS "ACTOR",
          COUNT(FA.film_id) AS N_FILMS
  FROM ACTOR A, FILM_ACTOR FA
  WHERE A.actor_id = FA.actor_id
  GROUP BY A.actor_id, A.first_name, A.last_name
  ORDER BY COUNT(FA.film_id) DESC) AS afc
WHERE afc.N_FILMS + 9 >=
 (SELECT MAX(fcounts)
  FROM (SELECT count(*) as fcounts
      FROM actor a, film_actor fa
      WHERE a.actor_id = fa.actor_id
      GROUP BY a.actor_id) as temp
 );
```

Overall, the set of the 30 problems in SQL queries requires students to learn and understand a variety of general SQL techniques as well as specific MySQL features, such as group functions, simulation of the division operator of the relational algebra, regular expressions, full text search, and performance optimization.

4.3 Views, Stored Functions, Stored Procedures and Triggers

A major strength of Sakila compared to the textbook databases surveyed in Section 2 is its inclusion of advanced features such as views, stored subprograms, and triggers. As an example, the following problem asks students to write a stored function to return a feature vector of a customer. The function is then used in other problems in stored subprograms that serve as the basis of a simple data mining recommender application to find customers with shared tastes using a similarity vector. The suggested solution is not shown here because of space consideration.

> *Problem.* [Hard] Write a function, category_feature_length(customer_id INT), to compute the length of the category feature vector. Each element in the category feature vector is the number of the films the customer has rented in a category.

For example, for customer 1, the category feature vector is (2,2,6,5,2,4,1,1,1,4,2,2,2,2,1). She has rented 2, 2, 6 and 5 films in categories 1, 2, 3 and 4 respectively, and so on. For customer 20, it is (1,1,3,2,3,2,2,1,1,1,1,7,0,4,1). Note the 0 as the value of the 12th element as she has not rented any film in category 12.

The length of the vector is the square root of the sum of the square of each element.

□

4.4 Relational Algebra and Relational Calculus

Relational Algebra (RA) and Relational Calculus (RC) are core components of relational theory. However, most students do not have hands-on practice, partly because of a lack of suitable interpreters. We used the RA interpreter developed by Sunderraman (2010). The RA interpreter requires databases to be prepared in a certain format. Its queries use a special textual format (as opposed to the standard mathematical notation of RA). We prepared data files to populate the interpreter with Sakila and designed problems as illustrated in the example problem below. Since the interpreter does not support some RA operators such as division and logical disjunction, the solutions sometimes tend to be more complicated than those using the standard RA mathematical notation. This provides an example of the needs to work around unsupported features in the selected technology in real world environment.

> *Problem.* [Medium] List the customer id of customers who have rented film 1 but not film 21. □

Suggested Solution.

```
(PROJECT [CUSTOMER_ID] (RENTAL JOIN
  (PROJECT [INVENTORY_ID]
    (SELECT [FILM_ID = 1] (INVENTORY)))))
MINUS
(PROJECT [CUSTOMER_ID] (RENTAL JOIN
  (PROJECT [INVENTORY_ID]
    (SELECT [FILM_ID = 21] (INVENTORY)))));
```

Students also learn to be mindful of practical nuances. For an example, many textbook examples teach students to first perform a natural join to combine information from two tables and then extract the desired information through project and select. In the example above, that will not work. Projection must be performed first and a query like RENTAL JOIN INVENTORY returns an empty table. This is because besides the expected common attribute of inventory_id, both tables also have another common attribute, last_update, which have different meanings. If not removed by a projection first, the equality of last_update is incorrectly included in the JOIN operation. Sunderraman (2010) also developed an interpreter for RC. Because of time restriction, we have not used it in our problem set. Instead, we asked our students to construct both Tuple Oriented Relational Calculus and Domain Oriented Relational Calculus solutions using standard mathematical notation.

4.5 Database-Driven Web Application Development with PHP

Sakila provides sufficient details for non-trivial database-driven Web application development. We selected PHP as our language of choice as it is open source, popular, feature rich, and well supported. Since it is usually counted as one assignment out of a total of 8 to 9 assignments in the semester, the problem requirement is minimalistic with little demand on Web design. The focus is on embedded database programming in PHP and MySQL, as demonstrated by the example problem below. The suggested solution is not shown here because of space consideration.

> *Problem*. [Medium] Write a very simple toy PHP MySQL Web application to display category information of customers from Sakila for the DVD systems manager. The emphasis is on the PHP to MySQL database access component. Thus, the homework does not expect refined Web design. The minimum requirement is to allow the manager to select from the first character of the customer's last name from a drop down list. After the manager selects the character, it should lead to the page customer.php, which shows all customers in a list

format. For an example, if the character A is selected:

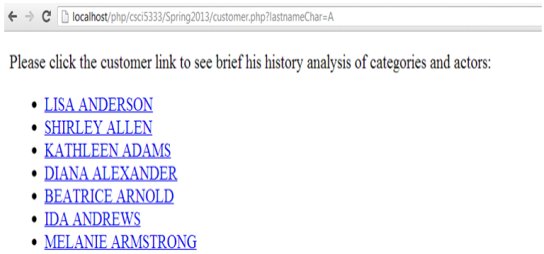
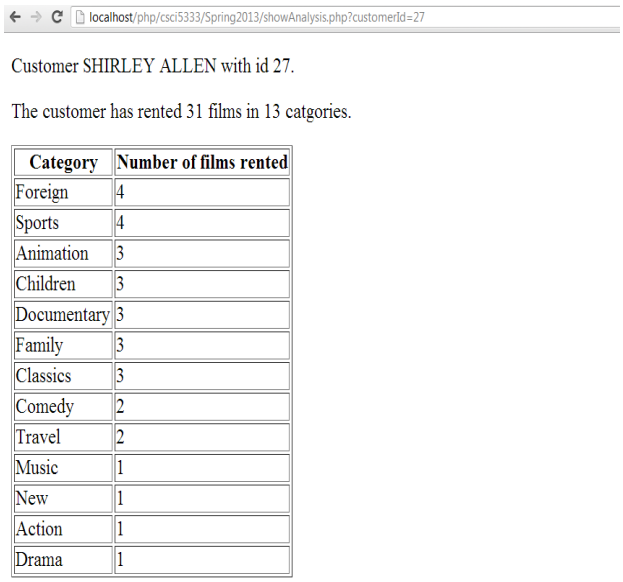


Figure 2. Web display example of customer.php with last name starting with the character 'A'

Clicking the link leads to another page, showAnalysis.php, to display the following information:



Category	Number of films rented
Foreign	4
Sports	4
Animation	3
Children	3
Documentary	3
Family	3
Classics	3
Comedy	2
Travel	2
Music	1
New	1
Action	1
Drama	1

Figure 3. Web display example of showAnalysis.php

4.6 XML Generation, XPath, and XQuery

XML provides a standard way to store meta-data about information (Elmasri and Navathe, 2010). It is especially useful in data transmission. Many database courses have a good treatment of XML. We designed two kinds of problems. The first is to write script programs to create XML content by reading data from Sakila.

> *Problem.* [Medium] Write a script program (such as a standalone PHP program) to access the Sakila database to generate an XML file: film.xml. Your program can be quick and dirty as the goal is the generation of XML content. Reasonable assumptions, such as no XML special characters in the data, can be made. A part of the film.xml is shown in Table 6.

We provided the students with a suggested solution in standalone PHP. Note that PHP is usually used in Web application but not in standalone mode. A suggested solution in other scripting languages such as Python or Perl can easily be developed instead. However, we do not want to incur more complexity to the course by adding another scripting language. The suggested solution is not included here because of space consideration.

```
<?xml version="1.0"?>
<films>
<film id="1">
   <title>ACADEMY DINOSAUR</title>
   <releaseYear>2006</releaseYear>
   <category>Documentary</category>
   <actor id="1">PENELOPE GUINESS</actor>
   …
   <actor id="198">MARY KEITEL</actor>
</film>
<film id="2">
…
</films>
```

Table 6. A portion of the file film.xml □

The second kind of problems is the construction of XPath and XQuery expressions for queries on the generated XML files. For example:

> *Problem.* [Medium] Write an XQuery expression to generate an XML document of actors and the number of film appearances of the actors using the format below. Note that it is in ascending order of actor id.

```
<?xml version="1.0" encoding="UTF-8"?>
<actors>
  <actor id="1">
    <numFilms>19</numFilms>
  </actor>
  ...
  <actor id="200">
    <numFilms>20</numFilms>
  </actor>
</actors>
```

Suggested Solution.

```
<actors>
   {for $id in fn:distinct-values(//film/actor/@id)
    order by xs:int($id)
    return
      <actor id="{$id}">
        <numFilms>
         { fn:count(//film[actor/@id=$id]) }
        </numFilms>
      </actor>
    }
</actors>
```

4.7 Other Topics

Sakila can also be used to support other database course objectives such as performance optimization and database security. For examples, we have assignments that ask students to provide several query solutions, such as using and not using join, using and not using subqueries, etc., and then compare their execution times to study their relative performance. Although not used in our classes, Sakila can

also be extended to support an encryption-based online customer account system.

5. RESULTS AND DISCUSSION

To gauge the effectiveness of the approach, we conducted student surveys in four semesters. The surveys do not represent a quantitative analysis in support of any hypothesis. However, they provide insights on the potential of our approach. Table 7 lists the averaged results of the 186 responses on seven surveys. We did not conduct a survey in the spring semester of 2013. Most responses use a scale of 5 with 3 being neutral and 5 being the most positive answer (for examples, very interesting or very useful). On the question on the level of difficulties, 5 means very difficult. Standard deviations are shown in parenthesis after the averages.

Survey	# Responses	Past Experience	Interestingness	Usefulness	Level of difficulty
Fall 2011	30	0.28 (0.74)	3.48 (1.16)	3.76 (1.16)	3.96 (1.02)
Spring 2012 Survey 1	18	0.88 (1.50)	4.12 (0.99)	4.24 (1.09)	3.41 (0.80)
Spring 2012 Survey 2	18	0.38 (0.81)	3.71 (1.16)	3.71 (1.05)	3.76 (0.97)
Fall 2012 Section 1	29	0.41 (0.87)	4.35 (1.06)	4.29 (1.10)	4.12 (0.60)
Fall 2012 Section 2	31	0.09 (0.43)	4.50 (0.91)	4.65 (0.57)	3.78 (0.79)
Fall 2013 Section 1	30	0.52 (1.33)	4.19 (0.93)	4.05 (1.24)	3.29 (0.72)
Fall 2013 Section 2	30	0.92 (1.20)	4.12 (0.99)	4.58 (0.58)	3.35 (0.56)
Average	186	0.50 (0.98)	4.07 (1.03)	4.18 (0.97)	3.67 (0.91)

Table 7. Average student responses on past experience (number of similar assignments done in the past), and interestingness, usefulness and level of difficulties of Sakila-based assignments. Standard deviations are shown in parenthesis after the averages.

The result indicates that students found that Sakila-based assignments are in general interesting and useful. The surveys also asked students to provide textual comments. Out of 186 responses, 148 provided textual comments. We analyzed these comments and count the occurrences of different perceptions. For example, the comment "Useful and helps students as we have to do it practically which will be helpful when working" is counted as a perception of both usefulness and providing real-world experience. Similarly, the comment "Assignments are interesting and are difficult. By practicing and doing hard work I can complete the assignment" is counted as a perception of both interestingness and difficulty. The result is shown in Table 8 below. It correlates well with the numeral responses.

Survey	# Responses	# Comments	Difficult	Interesting	Useful	R/W Exp.	Need help
F11	30	25	11	3	12	3	4
Sp12 #1	18	17	8	3	8	2	4
Sp12 #2	18	17	8	3	8	2	4
F12 #1	29	17	2	6	10	2	0
F12 #2	31	23	5	5	14	8	3
F13 #1	30	23	3	3	15	6	0
F13 #2	30	26	3	3	15	6	0
Total	**186**	**148**	**40**	**26**	**82**	**29**	**15**
Percentage		**80%**	**27%**	**18%**	**55%**	**20%**	**10%**

Table 8. Content analysis on comments showing the number of responses indicating in their written comments that the Sakila based assignments are difficult, interesting, useful, provided real-world experience, and students needed help to complete. The total number of students in classes and the number of responses are also shown.

It can be seen that the students had little prior experience and have done an average of only 0.5 similar assignments in the past. This may be one of the reasons why they found the assignments to be a little difficult (an average response of 3.67 on the level of difficulties, with a response of 3 being neutral). 27% of the textual comments mentioned that the assignments were difficult, and 10% indicated that they needed helps. PHP is the topic that was mentioned most as both hard and interesting. This is understandable as most students had no prior PHP experience and basically had only two weeks to learn enough to construct the solution.

Note that in more recent semesters, students perceived the assignments to be less difficult. No student explicitly said that they needed help in the most recent semester. This is likely due to accumulated instructor's experience and that Sakila is much more integrated into the course in more recent semesters (five assignments in fall 2013 as opposed to one assignment in fall 2011). This is a benefit of having multiple assignments on the same database. It seems that the students are able to learn quickly enough to be productive.

The students clearly found the assignments to be both interesting (4.07) and useful (4.18). Furthermore, the responses from more recent semesters (such as the average responses of 4.15 and 4.31 on interestingness and usefulness respectively on the two sections in fall 2013) are markedly better than earlier semesters (such as the responses of 3.48 and 3.76 on interestingness and usefulness respectively in fall 2011). Many students explicitly mentioned the benefits

of real-world experience. These responses are promising. Since the responses may be biased by the overall student perception of the general course and instructor quality, we also included survey questions to compare Sakila to database examples designed by the instructor and the textbook (Elmasri and Navathe, 2010). Table 9 shows the results.

Survey	Relative usefulness vs. Instructor's examples	Relative Interestingness vs. Instructor's examples	Relative usefulness vs. book's examples	Relative Interestingness vs. book's examples
F11	3.20	3.20	3.32	3.28
Sp12 #1	3.88	3.65	3.71	3.65
Sp12 #2	3.53	3.41	3.59	3.18
F12 #1	4.06	3.88	3.94	3.94
F12 #2	3.96	3.95	3.68	3.95
F13 #1	3.90	3.90	3.76	3.76
F13 #2	3.81	3.88	3.77	3.81
Average	3.76	3.70	3.68	3.65

Table 9. Comparing using Sakila to database examples by the instructor and the textbook. A response of 3 means 'about the same' and a response of 5 means 'much more interesting' or 'much more useful'.

Since a response of 3.0 represents a neutral perception, the survey result range of 3.65 to 3.76 on usefulness and interestingness thus shows that Sakila compares favorably to instructor and textbook examples in a consistent manner. This agrees with our teaching experience. In general, we find that Sakila offers several important advantages.

1. It has the right balance of complexity: not too simplistic, but also not bogged down by unnecessary domain intricacies.
2. It is open sourced. The SQL files for creating and populating the databases are available for study and modification.
3. It is well designed and documented, and highly available. It enjoys widespread support in the MySQL community.
4. It exposes students to practical considerations in realistic database without overwhelming them.
5. It includes advanced SQL features in views, stored procedures, stored functions and triggers.
6. Because the targeted application is easy to understand, it is not hard to set up assignments in multiple topics.
7. It can be used as a unified database to support multiple assignments to enhance learning efficiency and increase depth of knowledge.

All of these advantages are illustrated by the examples on various topics discussed in Section 4. Since an important goal of selecting Sakila is its semi-realistic nature, advantage #4 merits more discussion here. Sakila was designed not to be a production system, which usually requires much more complexity. However, it incorporates a lot of practical design decisions not generally available in 'toy' databases. Analyzing, understanding, and improving these practical aspects prepare students better for the real-world environment.

As another example of these practical considerations, one problem asks students to study two tables: film and film_text. They need to find out that the table film_text(film_id, title, description) is a subset of the table film. The values of the three columns in film_text are a copy of the same columns in the table film. The table film has additional columns to store other film properties. This is obviously a violation of the relational database design theory of avoiding unnecessary redundancy. However, in practice, Sakila was designed initially for Version 5.5 of MySQL. The table film uses the default and efficient MySQL database engine InnoDB, which does not support full text indexing and searching. To provide full text support on the columns title and description, they are copied to form a new table film_text. The new table uses the database engine MyISAM to support full text indexing and searching. The contents of the two tables are synchronized by triggers. This is an example of the kind of workaround common in real world projects but missing in idealized database examples. Furthermore, in the newer version of MySQL, full text indexing is supported by the InnoDB engine and there is no need for a separate table film_text. In fact, a few brighter students found this out independently. They ported the database to a newer MySQL with a simplified design to solve a Web application problem that requires full text searching. Better solutions made available by technology advances is common in the real world and students get a taste of practical database refactoring to improve performance and reliability (Ambler, 2003).

6. FUTURE WORK AND CONCLUSIONS

No single database can be used to satisfy all pedagogical needs of a database course. Sakila is no exception. This paper shows that a high quality semi-realistic database such as Sakila can effectively be used as a unifying platform to support multiple topics. However, Sakila is not without its limitations. Its targeted problem of DVD rentals is not a new and exciting application. Some of its data can also be more illustrative. For an example, the table film_category allows the possibility of a film to belong to multiple categories, which makes for more interesting query problems. However, the actual Sakila dataset does not take advantage of this flexibility, and every film belongs to only one category. Furthermore, Sakila may also miss some of the sophisticated design patterns in more complicated production databases.

To address some of these issues in future semesters, we are working on constructing a new instance of Sakila with richer and more interesting data. Also on the design board are assignments that use the Relational Calculus interpreter by Sunderraman (2010), and that redesign Sakila to satisfy

new requirements. We are also investigating porting Sakila to an object-oriented database and a NoSQL database.

7. ACKNOWLEDGEMENTS

We would like to thank our students for their interest, participation, and feedback on this experiment. Ms. Chloris Yue and scholars of the UHCL NSF Scholar Program (NSF Grant # 1060039) also provided invaluable suggestions and assistance.

8. REFERENCES

Ambler, S. (2003) Agile Database Techniques: Effective Strategies for the Agile Software Developer, Wiley Publishing, New York, NY.

Bourennani, F., Guennoun, M. and Zhu, Y. (2010) "Clustering Relational Database Entities Using K-means," Second International Conference on Advances in Databases Knowledge and Data Applications (DBKDA), Menuires, France, pp.143-148.

Cobb, J., Jones, J., Kapfhammer, G. and Harrold, M. (2011) "Dynamic invariant detection for relational databases," ACM Proceedings of the Ninth International Workshop on Dynamic Analysis (WODA '11), New York, NY, pp.12-17.

Elmasri, R. and Navathe, S. (2010) Fundamentals of Database Systems, 6th Edition, Addison-Wesley, Boston, MA.

Enciso, M. and Soler, E. (2013) "Teaching database design: A reverse engineering approach," 2013 IEEE Global Engineering Education Conference (EDUCON), Berlin, Germany, pp.474-480,

Gillenson, M. (2011) Fundamentals of Database Management Systems, 2nd Edition, Wiley Publishing, New York, NY.

Hoffer, J., Ramesh, V. and Topi, H. (2013) Modern Database Management, 11th Edition, Prentice-Hall, Upper Saddle River, NJ.

Jukić, N. and Gray, P. (2008) "Using real data to invigorate student learning," SIGCSE Bulletin, Vol. 40, No. 2, pp.6-10.

Jukić, N., Vrbsky, S. and Nestorov, S. (2013) Database Systems: Introduction to Databases and Data Warehouses, Pearson, Upper Saddle River, NJ.

MySQL (2013), Retrieved December 22, 2013 from http://www.mysql.com/.

Pratt, P. and Adamski, J. (2004) Concepts of Database Management, 5th edition, Cengage Learning, Stamford, CT.

Pratt, P. and Adamski, J. (2011) Concepts of Database Management, 7th edition, Cengage Learning, Stamford, CT.

Rob, P. and Coronel, C. (2007) Database Systems: Design, Implementation and Management, 8th Edition, Course Technology Press, Boston, MA.

Sakila (2013a), Sakila Sample Database, Retrieved December 22, 2013 from http://dev.mysql.com/doc/sakila/en/.

Sakila (2013b), Sakila Introduction, Retrieved December 22, 2013 from http://dev.mysql.com/doc/sakila/en/sakila-introduction.html.

Seyed-Abbassi, B., King, R. and Wiseman, E. (2007), "The Development of a Teaching Strategy for Implementing a Real-World Business Project into Database Courses," Journal of Information Systems Education, Vol. 18, No. 3, pp.337-343.

Sunderraman, R. (2010) Fundamentals of Database Systems Laboratory Manual, 2nd Edition, Retrieved December 21, 2013 from http://tinman.cs.gsu.edu/~raj/elna-lab-2010/.

Teradata University Network (2013), Retrieved December 21, 2013, from http://www.teradatauniversitynetwork.com/.

Wagner, P., Shoop, E. and Carlis, J. (2003). "Using scientific data to teach a database systems course," Proceedings of the 34th SIGCSE technical symposium on computer science education (SIGCSE '03), New York, NY, pp.224-228.

Watson, R. (2005) Data Management: Databases and Organizations, 5th Edition, Wiley Publishing, New York, NY.

AUTHOR BIOGRAPHY

Kwok-Bun Yue (B.S., M.Phil., Chinese University of Hong Kong, M.S., Ph.D., University of North Texas) is a Professor of Computer Information Systems and Computer Science at University of Houston-Clear Lake (UHCL). His research interests are in Internet computing, semi-structured data, and information systems and computer science education. He had published more than 30 technical papers. Dr. Yue is a recipient of s UHCL Distinguished Teaching Award, the UHCL Piper Award, the UHCL Alumni Association's Outstanding Professor Award and the UHCL Fellowship Award. He had served as a CTO of a startup company.

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096