IMPLEMENTING TOPOLOGICAL INTEGRITY CONSTRAINTS ON TEMPORAL DATABASES

CHUNG PHAM VAN

Faculty of Information Technology, Industrial University of Ho Chi Minh City;

pchung@iuh.edu.vn

Abstract. Checking integrity constraints in real-time the database is an important field of investigation. In this paper, we implement topological integrity constraints depending on which user has chosen an integrity constraint on different times on many existing integrity constraints of temporal databases. We suggested using the Hamiltonian paths in directed graphs order to implement a test program on data simulated by a college's real data. The object needs to satisfy integrity constraints in temporal databases as students who have enrolled in the subjects. Moreover, the program also monitors the learning process and advises students to choose courses in the school's training program.

Keywords. Full-state sequence, topological integrity constraints, transition graph, version graph.

1 INTRODUCTION

Checking integrity constraints in temporal databases is a problem that many people study. There have been some approaches to this issue, such as:

• Doucet and et al. [1], [2] performed the process of checking many integrity constraints by sorting versions and data cohesion on the object-oriented model. Time logic identifies constraints and then transforms into revisions

• Cordeiro RLF and et al. [3], [4] improved the old method by offering a number of methods such as time query language defined valid areas of constraints, versions of constraint distinguished over time, the unbounded points of data are found and version language represented the evolution of schemas.

In this paper, we implemented a test program using the Hamiltonian paths in directed graphs approach as in [5], [6],[7] to check topological integrity constraints in a temporal database. This approach has studied how to check multiple integrity constraints of systematic data over time imposed on a temporal database, instead of constraint on multiple database versions like [3].

Using a data structure is a graph to build tasks to check multiple integrity constraints, called shortlisted: Topological integrity constraint (TIC). The integrity constraints are full-state sequences (FSS) that exist in the temporal database that an object must satisfy FSS during the time updating. Also, every object at some points can follow one of many integrity constraint versions. At different times released these versions and they are valid when updating data for the objects. Each version is a full-state sequence, it is a Hamiltonian path in a direct transition graph (TG).

In the temporary database, there are many versions of integrity that are valid when of update and some versions that are outdated or newly released. This is quite complicated, to solve this problem systematically, we use the version diagram (VG) to make and update versions over time, each vertex of VG is a version. ; Each edge of VG indicates the link between the two versions.

An object first recorded in the database will choose a version (called original version) in existing versions and must follow the constraints of that version for a fixed period of time in real-time. However, during this time, an object can be converted to a certain version (based on the set of rule (file) written in the database, this file is updated over time), but eventually, the object must return to the original version to end the process of satisfying the constraint and end the data update overtime for it.

This test program can help students register for courses according to the credit system of each semester in a college. Students can query academic results over time. The rest of the article includes: Section 2 presents some data structures such as transition graph, version graph, the rule set, and their relationships. Section 3 presents procedures for checking topological integrity constraints. Section 4 implements and finally the conclusion and future works.

2 CREATING DATA STRUCTURES

This section presents the data structures used in implementing test programs, including transition graph, version graph and set of rules.

2.1 Transition graph

Each constraint version is described by a template with statements (see [9]), then converted into a directed transition graph, which has a starting vertex (outgoing edges only) and an end vertex (no outgoing edges). Each vertex represents a state of the object, each a directed edge indicates the next constraint state that the object must follow. The starting vertex is the first constraint for the object and the end vertex is the last constraint. For example, in Figure 1, representing a version with six constraints, s_0 is the first constraint, s_5 is the last constraint.

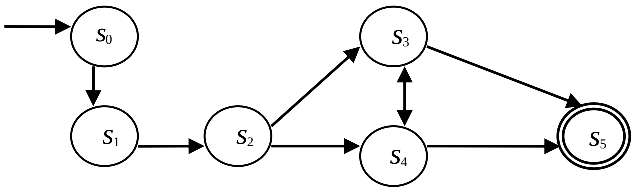


Figure 1: Transition graph [8]

The transition state graph must satisfy: There is at least one path from begin vertex to end vertex and through all the remaining vertices, each vertex only passes once, can check this with the cost of linear-time [7], [10]. Such paths are Hamiltonian paths in directed graphs [11], we call them full-state sequences (FSS) and use the procedure SEARCHING_SEQUENCE [7] to search them. For example, in Figure 1 there are two full-state sequences: s_0, s_1, s_2, s_3, s_4, s_5 and s_0, s_1, s_2, s_4, s_3, s_5.

2.1 Version graph

In temporal multi-version databases, there may be multiple versions of integrity constraints. Use version graphs (VB) to represent existing and effective versions, it is also updated over time, meaning there are deleted versions and new versions are created. Each version has many FSS as mentioned in 2.1. In addition, VB is considered as a hyper-graph in addition to managing versions, it also links to transition graph (TG) to check the multi-version integrity constraints. In Figure 2, depicting a VB with six vertices corresponding to six existing and effective versions, each edge has a direction to show an object in the database being executed in this version, can be transferred to another version. Note that each version in Figure 2 will have a corresponding TG.

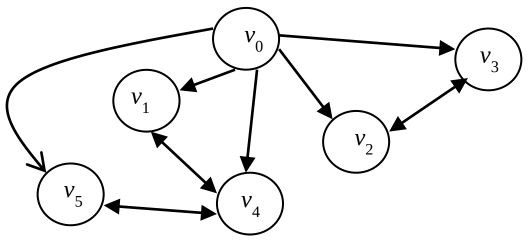


Figure 2. Version graph

An object in the database at time t_1 is executing the s_i version constraint, and by the time t_2 has executed a certain state in s_i, it can switch to another version s_j, if this is written in the set of rules, the form of rules is written as if-then clauses. This ruleset is updated when new versions are released or deleted.

3 THE PROCEDURE CHECKING INTEGRITY CONSTRAINTS

Some of the following procedures are implemented to check multiple topological integrity constraints [6].

3.1 Procedures on Version graph

- **Inserting a version**

```
procedure INSERT-VERSION ($v_i$: version, $VG$: version graph, $TG$: transition graph)
begin
  insert the new transition rules into set of rules;
  insert vertex $v_i$ into $VG$;
  based on the new transition rules relating to $v_i$ to insert the paths between $v_i$ with other vertices in
  $VG$; create $TG$;
  perform procedure SEARCH-SEQUENCE;
  update new rule into set of rules;
end;
```

- ***Deleting a version***

  Deleting a version according to the DELETE-VERSION procedure needs to satisfy:
  i) An objects in that version have completed the last state in the corresponding TG.
  ii) In VB there is at least one version other than the version that needs to be deleted.

```
procedure DELETE-VERSION ($v_i$: version, $TG$: transition graph)
begin
  if met two conditions (i) and (ii) then
  begin
    delete the edges go out and go to $v_i$, then delete $v_i$;
    delete the corresponding rules in set of rules;
    delete $TG$ corresponds to $v_i$;
  end
end;
```

3.2 Procedures on transition graph

- ***Inserting data of the object***

In the procedure INSERT-DAT, the data of object *Ob* is inserted at the top of the v_n version, and *Ob* has completed the constraint at vertex si under version v_m, and must satisfy a rule in the set of rules. This is the main problem when inserting data.

```
procedure INSERT-DATA ($s_i$, $s_j$: vertex, $v_m$ ,$v_n$ : version, $Ob$: object, $TG$: transition graph corresponding to $v_m$)
    /*Data of $Ob$ is inserted to vertex $s_j$ under version $v_n$ */
begin
  if vertex $s_i$ in $TG$ no outgoing edge then rejection message
  else /*vertex $s_k$ is a vertex in $TG$ of version $v_m$ */
  if (there is direct path from $v_m$ to $v_n$ in $VG$ and check in $SRs$,if met and $Ob$ never once had
              transferred to $s_j$)
     then insert data of $Ob$ to vertex $s_i$ of $TG$;
     else rejection message
end;
```

- ***Deleting data of the object***

  Use the procedures as shown in [5].

3.3 Object checking procedures enforce multiple version integrity constraints

Based on full-state sequences (FSS) to track objects satisfies integrity constraints. The first time an object chooses one of FSS to do integrity constraints and must execute in the order of the selected sequence. However, because of multiple integrity constraints, an object can enforce constraints in intermediate states (except beginning state and end state in the initially selected FSS), it can pass through executing another state in another version, as long as a set of rules allows.

- *Mark the constraints that the object has finished executing*

Each constraint in FSS will be marked with the MARKED_ELEMENTS procedure when an object is executed.

```
procedure MARKED_ELEMENTS
/*input: full-state sequences $\sigma_i$ of an object Ob,
 list C contains integrity constraints from users selected */
begin
 for each element $c_i$ in C
   for each $\sigma_i$
    mark elements in $\sigma_i$ corresponding to $c_i$ ;
    delete the contents of C;
end ;
```

- List of constraints that objects can chooses

Procedure LIST_INTEGRITY CONSTRAINTS lists the constraints that an object can select to execute when the user queries.

```
procedure LIST_INTEGRITY CONSTRAINTS
/*input: full-state sequences $\sigma_i$ of an object Ob,
  output: List C contains integrity constraints from
  users selected */
begin
  create an empty list L;
  for each $\sigma_i$ of an object Ob
     copy elements are not marked in $\sigma_i$ to L;
   keep the various elements in L;
   if L is empty then
     message "Ob has fully executed";
   else
     print L;
end ;
```

- **List of constraints that the object has satisfied**

Procedure CHECKING, list the constraints that an object has satisfied when the query or message object has fully satisfied constraints.

```
procedure CHECKING
/*input: full-state sequences $\sigma_i$ of an object Ob */
begin
 create an empty list Temp;
 if the elements in a sequence of $\sigma_i$ have been marked then message
 "It has completed ";
 else for each $\sigma_i$
     copy the marked elements to Temp;
     remove duplicate elements in Temp; show Temp;
end;
```

4 IMPLEMENTING

This section implements a test program for students to enroll in the semester credit-based courses of information technology faculty of a college. First students choose a training program according to a specific discipline, then they can choose the subjects in other programs when the training program allows (set of rules). The tool selected to build the program is Windows Form C # (Visual. Net 2010) and SQL Server 2008R2.

4.1 Relational schemas

- ***Schemas to check TIC***
  PHIEN_BAN_SV (MaSV, MaMH, StartDate,MaPB, EndDate, KetQua),
  PHIEN_BAN (MaPB, EffectiveDate, EndDate),
  MON_HOC (MaMH, MaPB, TenMH),
  STATE_SEQUENCE (MaSV, FullStateSequence, MaPB).
- ***Other schemas for faculties, student registration and student learning outcomes***
  KHOA(MaKhoa, TenKhoa)
  LOP(MaLop, TenLop, MaKhoa, HeDaoTao, SiSo)
  SINH_VIEN(MaSV, HoTenSV, Nam, NgaySinh, NoiSinh, NamHoc, NamKT, Pass)
  MON_THAY_THE(MaMH, PbCu,MaMH_Moi, PbMoi)
  CHON_CHUONG_TRINH(MaSV, MaPB, NgayChon,NamHoc)
  DANG_KY(MaDKy, MaSV, NgayDKy, HocKy)
  CHUONG_TRINH_D_KY(MaDKy, MaMH,MaPB, NgayBDHoc, NgayKTHoc)

4.2 Interface and tasks

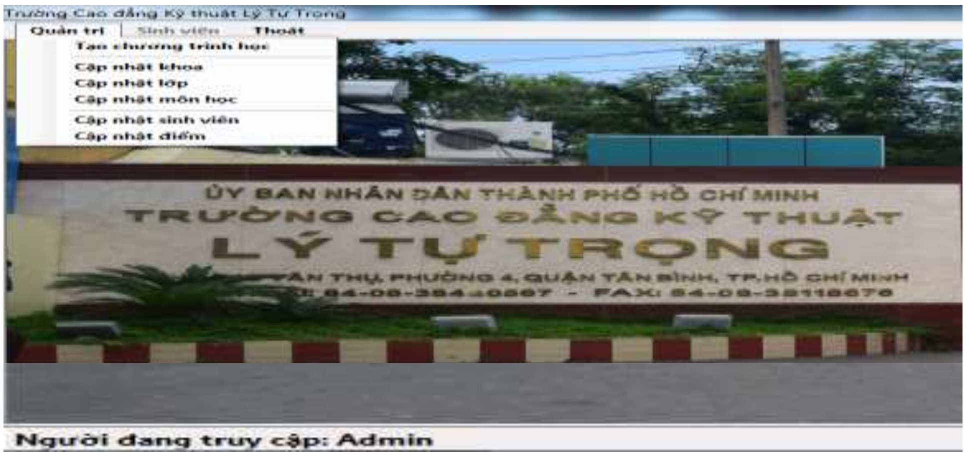


Figure 3. Interface for administrator

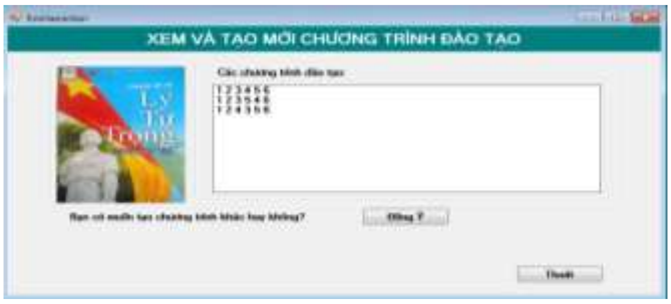
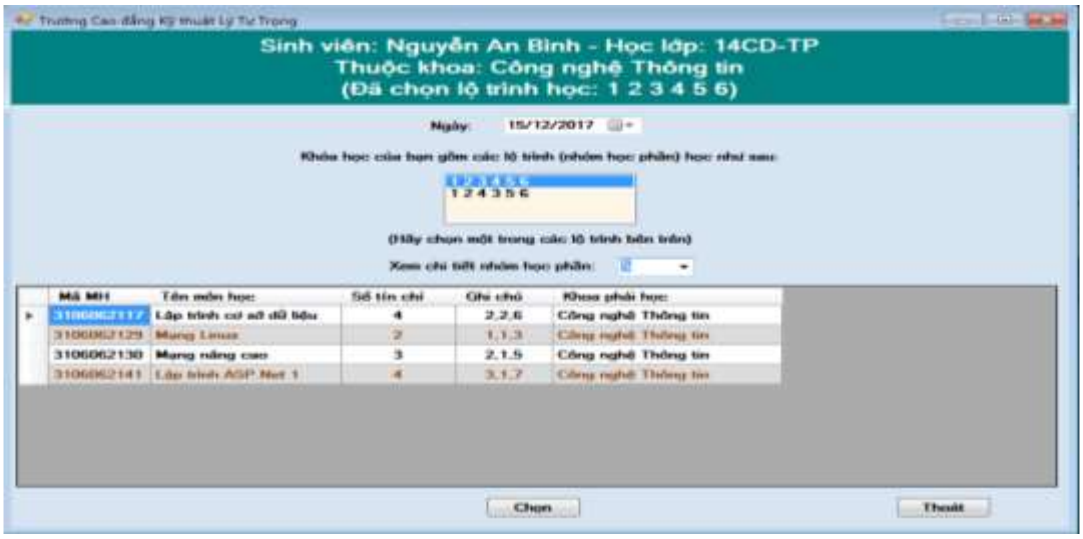

XEM VÀ TAO MỚI CHƯƠNG TRÌNH ĐÀO TAO
Các chương trình đào tạo
Bạn có muốn tạo chương trình khác hay không?
Đồng ý
Thoát

Figure 4. Creating the new versions

In Figure 4, performing the version update task and updating into the set of rules, if it is the new version, also insert into the MON_THAY_THE schema.



Sinh viên: Nguyễn An Bình - Học lớp: 14CD-TP
Thuộc khoa: Công nghệ Thông tin
(Đã chọn lộ trình học: 1 2 3 4 5 6)

Ngày: 15/12/2017

Khóa học của bạn gồm các lộ trình (nhóm học phần) học như sau:

1 2 3 4 5 6
1 2 4 3 5 6

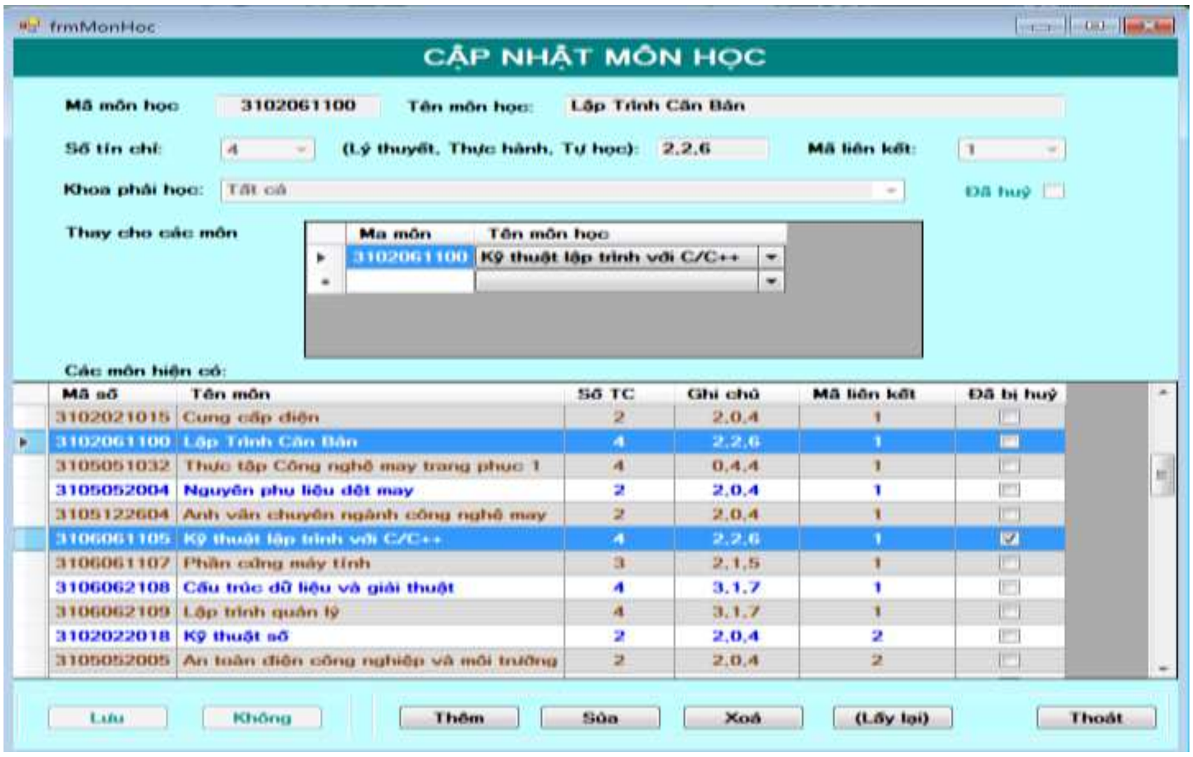
(Hãy chọn một trong các lộ trình bên trên)

Xem chi tiết nhóm học phần: 2

Mã MH	Tên môn học	Số tín chỉ	Ghi chú	Khoa phải học
3106062117	Lập trình cơ sở dữ liệu	4	2,2,6	Công nghệ Thông tin
3106062129	Mạng Linux	2	1,1,3	Công nghệ Thông tin
3106062130	Mạng nâng cao	3	2,1,5	Công nghệ Thông tin
3106062141	Lập trình ASP.Net 1	4	3,1,7	Công nghệ Thông tin

Figure 5. Students choose the training program.

In Figure 5, students choose a training program (version) in many of the college's programs that are now in effect and studied throughout the process until graduation.



CẬP NHẬT MÔN HỌC

Mã môn học: 3102061100 Tên môn học: Lập Trình Căn Bản

Số tín chỉ: 4 (Lý thuyết, Thực hành, Tự học): 2,2,6 Mã liên kết: 1

Khoa phải học: Tất cả Đã huỷ

Thay cho các môn

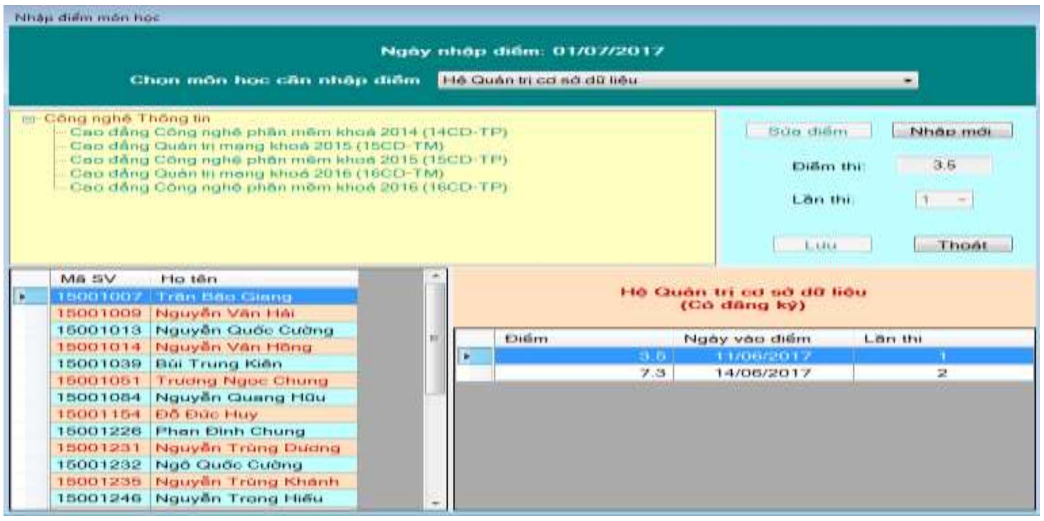
Ma môn	Tên môn học
3102061100	Kỹ thuật lập trình với C/C++

Các môn hiện có:

Mã số	Tên môn	Số TC	Ghi chú	Mã liên kết	Đã bị huỷ
3102021015	Cung cấp điện	2	2,0,4	1	☐
3102061100	Lập Trình Căn Bản	4	2,2,6	1	☐
3105051032	Thực tập Công nghệ may trang phục 1	4	0,4,4	1	☐
3105052004	Nguyên phụ liệu dệt may	2	2,0,4	1	☐
3105122604	Anh văn chuyên ngành công nghệ may	2	2,0,4	1	☐
3106061105	Kỹ thuật lập trình với C/C++	4	2,2,6	1	☑
3106061107	Phần cứng máy tính	3	2,1,5	1	☐
3106062108	Cấu trúc dữ liệu và giải thuật	4	3,1,7	1	☐
3106062109	Lập trình quản lý	4	3,1,7	1	☐
3102022018	Kỹ thuật số	2	2,0,4	2	☐
3105052005	An toàn điện công nghiệp và môi trường	2	2,0,4	2	☐

Figure 6. Students register for new subjects

The functions in Figure 6 perform checking integrity constraints on full-state sequences.


Nhập điểm môn học

Ngày nhập điểm: 01/07/2017

Chọn môn học cần nhập điểm: Hệ Quản trị cơ sở dữ liệu

- Công nghệ Thông tin
  - Cao đẳng Công nghệ phần mềm khoá 2014 (14CD-TP)
  - Cao đẳng Quản trị mạng khoá 2015 (15CD-TM)
  - Cao đẳng Công nghệ phần mềm khoá 2015 (15CD-TP)
  - Cao đẳng Quản trị mạng khoá 2016 (16CD-TM)
  - Cao đẳng Công nghệ phần mềm khoá 2016 (16CD-TP)

Điểm thi: 3.5

Lần thi: 1

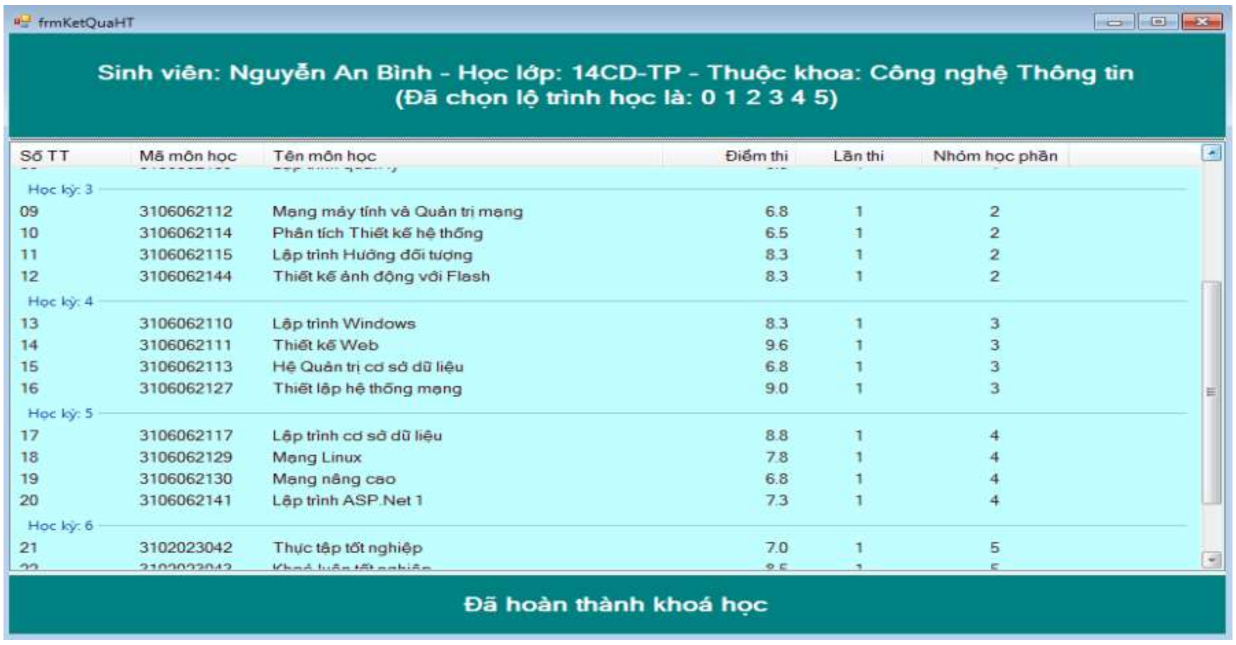
Mã SV	Họ tên
15001007	Trần Bảo Giang
15001009	Nguyễn Văn Hải
15001013	Nguyễn Quốc Cường
15001014	Nguyễn Văn Hồng
15001039	Bùi Trung Kiên
15001051	Trương Ngọc Chung
15001084	Nguyễn Quang Hữu
15001154	Đỗ Đức Huy
15001226	Phan Đình Chung
15001231	Nguyễn Trùng Dương
15001232	Ngô Quốc Cường
15001235	Nguyễn Trùng Khánh
15001246	Nguyễn Trọng Hiếu

Hệ Quản trị cơ sở dữ liệu
(Có đăng ký)

Điểm	Ngày vào điểm	Lần thi
3.5	11/06/2017	1
7.3	14/06/2017	2

Figure 7. The function of updating subject results for students

The functions in Figure 7 update the results. If the score of the subject of an object (the student) is satisfactory when it has completed constraints on a state (subject) in FSS but otherwise, it has not completed the constraints. The student must then re-enroll in this subject until the required results are obtained.


frmKetQuaHT

Sinh viên: Nguyễn An Bình - Học lớp: 14CD-TP - Thuộc khoa: Công nghệ Thông tin
(Đã chọn lộ trình học là: 0 1 2 3 4 5)

Số TT	Mã môn học	Tên môn học	Điểm thi	Lần thi	Nhóm học phần
Học kỳ: 3					
09	3106062112	Mạng máy tính và Quản trị mạng	6.8	1	2
10	3106062114	Phân tích Thiết kế hệ thống	6.5	1	2
11	3106062115	Lập trình Hướng đối tượng	8.3	1	2
12	3106062144	Thiết kế ảnh động với Flash	8.3	1	2
Học kỳ: 4					
13	3106062110	Lập trình Windows	8.3	1	3
14	3106062111	Thiết kế Web	9.6	1	3
15	3106062113	Hệ Quản trị cơ sở dữ liệu	6.8	1	3
16	3106062127	Thiết lập hệ thống mạng	9.0	1	3
Học kỳ: 5					
17	3106062117	Lập trình cơ sở dữ liệu	8.8	1	4
18	3106062129	Mạng Linux	7.8	1	4
19	3106062130	Mạng nâng cao	6.8	1	4
20	3106062141	Lập trình ASP.Net 1	7.3	1	4
Học kỳ: 6					
21	3102023042	Thực tập tốt nghiệp	7.0	1	5

Đã hoàn thành khoá học

Figure 8. Query all students' learning results

In Figure 8, perform the CHECKING procedure to check whether the object has fully satisfied the TIC or listed the subjects that have passed (training requirements).

CONCLUSIONS

This paper has used graph structure and the Hamiltonian paths in directed graphs to track the process of checking multiple topological integrity constraints on the temporal database. The problem of finding the Hamiltonian paths in a directed graph is NP-complete [11]. But in this program, the complexity of the procedure SEARCH_SEQUENCE is only $O(k^{n-2})$, (see details in [7]).

Versions are training programs that are released over time. In a school, the training of one discipline usually lasts 3 to 4 years. The current trend, the training program will change each year, so students can learn new subjects instead of studying subjects follow the old training program. To make this systematically, the article has implemented a test program on data simulated by the current training program at a college of information technology faculty. This program manages students who enroll in credit and annuity academic programs and monitor their academic progress as well as advising students to register for courses.

The program uses structures: transition state graph, version graph and set of rules that allow students to transfer to equivalent subjects of other programs (students do not have to wait to save time study), and use the orderly Hamiltonian paths on the transition graph to build the full range of constraints that objects in the database must follow in real-time.

The program initially has prospects, encouragement, and suggestions from colleagues, future research direction is:

- Implementing programs for other faculties, supplementing the detailed curriculum of the subject, reference books, teaching teachers to help students consider the general curriculum.
- Develop procedures to check for inconsistency or duplication of a set of rules.

REFERENCES

[1] Doucet A., Fauvet M., Ganҫarski S., Jomier G., Monties S. Using database versions to implement temporal integrity constraints, In Workshop, Constraint database and applications, Proceedings, Greece, (1997) 219-233.

[2] Doucet A., Monties S. Versions of integrity constraints in multiversion databases, in book title: Database and expert systems applications, (1997) 252-261.

[3] Cordeiro, R. L. F., Santos C. S., Edelweiss N. Integrity constraint for temporal versioned model: classification, modeling and verification, in WTDBD, Proceedings Brasilia, Brazil, (2004) 67-72.

[4] Cordeio, R. L. F., Santos C., Edelweiss N. – TVCL Temporal version constraints language (2006), Available at: http://citeseerx.ist.psu.edu/viewdoc

[5] Chung Pham Van, Phong Vu Thanh, Checking temporal integrity constraints in temporal databases depending on user selection, Journal science and technology, 52 (4A) (2014) 160-169.

[6] Chung Pham Van, Multiple Versions Topological Integrity Constraints Imposed on Objects in Real Time Databases, The International Conference on Advanced Technology and Sustainable Development ICATSD2016.

[7] Chung P.V. Phong V.T., Checking Topological Intergity Constraints Impose on Objects in Real Time Databases, The Conference on Information and Computer Science (NICS2015), Publisher IEEE 2015.

[8] Michael Gertz, Udo Lipeck, "Temporal" Integrity constraints in temporal databases, in Proceedings of the International workshop on temporal databases Sept 1995.

[9] Chung P., Tuan Anh D. Implementing a query sublanguage for temporal clinical database systems, Proc. of MUSIC2005, Petaling Jaya, Nov. (2005)14-26.

[10] Pascal Welke, Simple Necessary Conditions for the Existence of a Hamiltonian Path with Applications to Cactus Graphs, Informatik III, University of Bonn, Germany, 5 Sep 2017.

[11] Robert Sedgewick, Kevin Wayne. Algorithms (4th edition.). Addison-Wesley Professional, ISBN 978-0-321-57351-3, (2011), pp. 566-585.

Ngày nhận bài:31/07/2019

Ngày chấp nhận đăng:01/10/2019