**TRƯỜNG ĐẠI HỌC KỸ THUẬT - CÔNG NGHỆ CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**Luận Văn**

**Xây dựng hệ thống điều khiển truy xuất**

**phân tích chính sách và tích hợp đánh gía chính sách**

**trên nền tảng SMT Solver**

**GVHD: Hà Xuân Sơn**

**---o0o---**

**SVTH : Triệu Phương Nam(1500380)**

**Lê Ngọc Tiến Thành (1501131**)

**Mục lục**

[**LỜI CAM ĐOAN** 8](#_Toc9630201)

[**LỜI CẢM ƠN** 9](#_Toc9630202)

[**MỞ ĐẦU** 10](#_Toc9630203)

[**I.** **Lý do chọn đề tài** 10](#_Toc9630204)

[**II.** **Mục tiêu và phạm vi nghiên cứu** 10](#_Toc9630205)

[**1.** **Mục tiêu** 10](#_Toc9630206)

[**2.** **Phạm vi nghiên cứu** 11](#_Toc9630207)

[**3.** **Ý nghĩa khoa học và thực tiễn nghiên cứu** 11](#_Toc9630208)

[**Chương 1** **TỔNG QUAN** 13](#_Toc9630209)

[**I.** **Tổng quan tài liệu** 13](#_Toc9630210)

[**1.** **Nghiên cứu trong nước** 13](#_Toc9630211)

[**2.** **Nghiên cứu ngoài nước** 13](#_Toc9630212)

[**II.** **Đối tượng và phương pháp nghiên cứu** 13](#_Toc9630213)

[**1.** **Đối tượng nghiên cứu** 13](#_Toc9630214)

[**2.** **Phương pháp nghiên cứu** 13](#_Toc9630215)

[**Chương 2** **CƠ SỞ LÝ THUYẾT** 15](#_Toc9630216)

[**I.** **Access Control là gì ?** 15](#_Toc9630217)

[**1.** **Khái niệm** 15](#_Toc9630218)

[**2.** **Các loại access control :** 16](#_Toc9630219)

[**2.1.** **DAC (Discretionary Access Control)** 16](#_Toc9630220)

[**2.2.** **MAC (Mandatory Access Control)** 17](#_Toc9630221)

[**2.3.** **RBAC (Role-based Access Control)** 18](#_Toc9630222)

[**2.4.** **ABAC (Attribute-based access control ):** 21](#_Toc9630223)

[**II.** **GIỚI THIỆU VỀ XACML** 24](#_Toc9630224)

[**1.** **Tìm hiểu XACML** 24](#_Toc9630225)

[**1.1.** **Giới thiệu XACML** 24](#_Toc9630226)

[**1.2.** **Cấu trúc XACML** 24](#_Toc9630227)

[**2.** **Thành phần của XACML** 28](#_Toc9630228)

[**2.1.** **Rule**: 29](#_Toc9630229)

[**2.2.** **Target** 30](#_Toc9630230)

[**2.3.** **Effect** 30](#_Toc9630231)

[**2.4.** **Condition (*Điều kiện* )** 30](#_Toc9630232)

[***2.5.*** **Biểu thức *obligation*(nghĩa vụ)** 30](#_Toc9630233)

[***2.6.*** **Biểu thức *advice (tư vấn)*** 30](#_Toc9630234)

[**2.7.** **Policy** 31](#_Toc9630235)

[**2.8.** **Mục tiêu chính sách (target policy)** 31](#_Toc9630236)

[**2.9.** **Rule-combining algorithm-identifier ( thuật toán kết hợp quy tắc)** 31](#_Toc9630237)

[**2.10.** **Biểu thức Obligation (nghĩa vụ)** 32](#_Toc9630238)

[**2.11.** **Biểu thức advice (tư vấn)** 32](#_Toc9630239)

[**2.12.** **Policy set (bộ chính sách)** 32](#_Toc9630240)

[**2.13.** **Policy-combining algorithm (Thuật toán kết hợp chính sách)** 32](#_Toc9630241)

[**2.14.** **Biểu thức Obligation (nghĩa vụ)** 33](#_Toc9630242)

[**2.15.** **Biểu thức Advice** 33](#_Toc9630243)

[**III.** **GIỚI THIỆU VỀ LOGIC VỊ TỪ** 34](#_Toc9630244)

[**1.** **The Logic of XACML** 34](#_Toc9630245)

[***1.1.*** **Xacml** 34](#_Toc9630246)

[***1.2.*** **Three-Valued Lattice (Lưới ba giá trị)** 34](#_Toc9630247)

[***1.3.*** **The Match Evaluation** 35](#_Toc9630248)

[***1.4.*** **Condition Evaluation** 35](#_Toc9630249)

[***1.5.*** **Extended Values** 36](#_Toc9630250)

[***1.6.*** **Biểu thức rút gọn** 36](#_Toc9630251)

[**IV.** **Kỹ thuật viết XML bằng ALFA** 37](#_Toc9630252)

[**1.1.** **History ALFA** 37](#_Toc9630253)

[**1.2.** **Concepts** 38](#_Toc9630254)

[**1.3.** **Tổng quan về các phần tử XACML và các phần tử ALFA tương ứng** 39](#_Toc9630255)

[**1.4.** **Comments** 39](#_Toc9630256)

[**1.5.** **Namespace** 39](#_Toc9630257)

[**1.6.** **Rule** 40](#_Toc9630258)

[**1.7.** **Policy** 41](#_Toc9630259)

[**1.8.** **Policy Set** 42](#_Toc9630260)

[**V.** **SMT-Z3** 43](#_Toc9630261)

[**1.** **SMT là gì ?** 43](#_Toc9630262)

[**2.** **Z3- microsoft ?** 43](#_Toc9630263)

[**3.** **Kết hợp SMT-z3** : 43](#_Toc9630264)

[**VI.** **Giới thiệu về NoSQL** 51](#_Toc9630265)

[**VII.** **Hệ cơ sở dữ liệu MongoDB** 53](#_Toc9630266)

[1.1. Thiết kế lược đồ 55](#_Toc9630267)

[1.2. Nhúng hay Tham chiếu 56](#_Toc9630268)

[1.3. Lựa chọn chỉ mục 58](#_Toc9630269)

[1.4. Chỉ mục 58](#_Toc9630270)

[1.5. Các khái niệm cơ bản 58](#_Toc9630271)

[1.6. Chỉ mục hỗn hợp các khóa 60](#_Toc9630272)

[1.7. Chỉ mục thưa thớt 60](#_Toc9630273)

[1.8. Chỉ mục duy nhất 61](#_Toc9630274)

[1.9 Giá trị lặp lại 61](#_Toc9630275)

[1.10 Xóa chỉ mục 61](#_Toc9630276)

[1.11 ReIndex 62](#_Toc9630277)

[1.12 Sao chép 63](#_Toc9630278)

[1.13 Truy vấn 66](#_Toc9630279)

[**VIII.** **Thiết kế cơ sở dữ liệu:** 68](#_Toc9630280)

[**1.1.** **Sơ đồ cơ sở dữ liệu:** 68](#_Toc9630281)

[**1.2.** **Chi tiết cơ sở dữ liệu:** 69](#_Toc9630282)

[**2** **HIỆN THỰC HỆ THỐNG** 74](#_Toc9630283)

[***I.*** **Cài đặt hệ thống:** 74](#_Toc9630284)

[***II.*** **Evalue policy** 77](#_Toc9630285)

[***III.*** **Giải thuật của SMT-z3** 79](#_Toc9630286)

[***IV.*** **Giải thuật về NegVaule:** 81](#_Toc9630287)

[***V.*** **Giải thuật về query Analysis** 81](#_Toc9630288)

[***VI.*** **Giải thuật gen policy:** 83](#_Toc9630289)

[***VII.*** Giải thuật genRequest: 83](#_Toc9630290)

[***VIII.*** Giải thuật Request Json: 84](#_Toc9630291)

[**3** **ĐÁNH GIÁ KẾT QUẢ** 85](#_Toc9630292)

[**4.1.** **Kết quả đạt được:** 85](#_Toc9630293)

[**4.1.1.** **tạo cơ sỡ dữ liệu trên mockaroo** 85](#_Toc9630294)

[**4.1.2.** **So sánh với các nghiên cứu trước đây** 85](#_Toc9630295)

[**4.1.3.** **Kết quả đánh giá với Mongodb** 88](#_Toc9630296)

[**4.2.** **Kết Luận** 89](#_Toc9630297)

[**Tài liệu tham khảo** 90](#_Toc9630298)

**Mục lục Hình Ảnh**

[**Hình 1: Minh họa bộ sưu tập 7**](#_Toc532977265)

[**Hình 2.Mô hình Master – Slave hai nút 12**](#_Toc532977266)

[**Hình 3. Mô hình Master – Slave bốn nút 12**](#_Toc532977267)

[**Hình 4.Mô hình Replica Sets hai nút 13**](#_Toc532977268)

[**Hình 5.Mô hình Replica Sets hai nút 13**](#_Toc532977269)

[**Hình 6.Server chính trở thành server cấp 2 13**](#_Toc532977270)

[**Hình 7. Sơ đồ XML Tree 16**](#_Toc532977271)

[**Hình 8.XML 16**](#_Toc532977272)

[**Hình 9. Kiến trúc xử lý XACML 18**](#_Toc532977273)

[**Hình 10.Các khối trong kiến trúc xử lý XACML 20**](#_Toc532977274)

[**Hình 11. Cấu trúc chính sách XACML 21**](#_Toc532977275)

[**Hình 12 Thành phần PIP phù hợp CSDL NoSQL 31**](#_Toc532977276)

[**Hình 13 Giải pháp hỗ trợ phân tích chính sách XACML và ứng dụng nhận biết lỗ hổng suy diễn 31**](#_Toc532977277)

[**Hình 14.Biểu thức biểu diễn chính sách 32**](#_Toc532977278)

[**Hình 15 Ví dụ Policy với 1 PolicySet, 5 Policy,12 Rule và 28 Constraint 34**](#_Toc532977279)

# 

**Mục lục bảng**

[**Bảng 1: tóm tắt cấu trúc XACML 19**](#_Toc532758950)

[**Bảng 2. Three-Valued Lattice (Lưới ba giá trị) 24**](#_Toc532758951)

[**Bảng 3. Extended Values 25**](#_Toc532758952)

[**Bảng 4. các phần tử XACML và các phần tử ALFA tương ứng 28**](#_Toc532758953)

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**Cần Thơ, ngày … tháng … năm 2019**

**Giảng viên hướng dẫn**

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**……………………………………………………………………………………………**

**Cần Thơ, ngày … tháng … năm 2019**

**Giảng viên phản biện**

**LỜI CAM ĐOAN**

Chúng em gồm có Lê Ngọc Tiến Thành, Triệu Phương Nam xin cam đoan rằng luận văn là công trình nghiên cứu của bản thân chúng em dưới sự hướng dẫn của Th.S Hà Xuân Sơn

Các kết quả công bố trong luận văn là trung thực và không sao chép từ bất kỳ công trình nào khác.

***Cần Thơ, ngày … tháng … năm 2019***

***Sinh viên thực hiện Sinh viên thực hiện***

*(Ký và ghi rõ họ tên) (Ký và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Luận văn là môn học đánh giá khả năng làm việc nhóm của sinh viên và khả năng làm việc dựa vào hướng dẫn của giáo viên, tạo điều kiện cho sinh viên nghiên cứu, tính tự lập, đánh dấu sự trưởng thành vủa sinh viên, là quá trình phấn đấu, học tập và rèn luyện của sinh viên trên con đường trở thành một kỹ sư công nghệ thông tin. Quá trình làm luận văn, quá trình học tập, tích lũy và kiểm tra lại kiến thức đã học và đem những kiến thức đó áp dụng thực tế.

Không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian thực hiện, nhóm đã nhận được sự hướng dẫn tận tình của Ths. Hà Xuân Sơn cùng các thầy cô trong khoa Công nghệ Thông tin và bạn bè để nhóm hoàn thành tốt luận văn một cách thuận lợi và gặt hái được nhiều kiến thức mới. Với lòng biết ơn sâu sắc nhất nhóm xin gửi lời cảm ơn chân thành đến quý thầy cô khoa Công nghệ Thông tin, trường Đại học Kỹ thuật Công nghệ Cần Thơ. Đặc biệt chúng em muốn gửi lời cảm ơn sâu sắc nhất đến thầy Hà Xuân Sơn, thầy đã tận tình hướng dẫn nhóm em trong suốt quá trình học tập và thực hiện luận văn.

Bước đầu đi vào làm báo cáo luận văn, nhóm em còn nhiều bỡ ngỡ. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo chắc chắn không tránh khỏi những thiếu sót trong cách hiểu, lỗi trình bày nên rất mong quý Thầy, Cô bỏ qua. Nhóm em rất mong nhận được ý kiến đóng góp của quý Thầy, Cô để nhóm em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn các báo cáo sắp tới.

Nhóm em xin chân thành cảm ơn!

**MỞ ĐẦU**

1. **Lý do chọn đề tài**

* Thông tin nhạy cảm là dữ liệu cần được bảo vệ dưới mọi hình thức, có nhiều hướng tiếp cận để bảo vệ dạng dữ liệu này như mã hoá dữ liệu, ẩn chi tiết dữ liệu, phân quyền người dùng.
* Trong các hướng tiếp cận trên thì phân quyền người dùng đang được phát triển mạnh mẽ từ các mô hình cơ bản như DAC mô hình điều khiển truy xuất tuỳ quyền, MAC mô hình điều khiển truy xuất bắt buộc, RBAC mô hình điều khiển truy xuất dựa trên vai trò và gần đây nhất là ABAC mô hình điều khiển truy xuất dựa trên thuộc tính. So với các mô hình trước đây thì ABAC có được ứng dụng rộng rãi và có tính bảo mật cao hơn các mô hình trước.
* Với khả năng làm mịn các chính sách bảo mật và linh hoạt với tất cả hệ thống (Cloud computing, Distributed network, Mobile network v.v…). Một trong các mô hình hiện thực của ABAC là XACML với phiên bản hiện tại là 3.0. Các hướng tiếp cận trước đây xoay quanh các vấn đề quản lý chính sách, phân tích chính sách, phát triển trên môi trường chính sách động, và đặc biệt là đánh giá chính sách.
* Có rất nhiều hướng tiếp cận để giải quyết bài toán đánh giá chính sách như Decision Diagrame, Statistic, Decision Tree, SAT và hướng tiện tại là SMT. Hướng tiếp cận này cho phép làm phẳng cấu trúc dạng thừa kế theo định dạng XML thành các biểu thức điều kiện cụ thể liên kết thông qua các phép toán logic (và, hoặc, phủ định, tồn tại). Đề tài này thiết lập cơ chế tôi ưu hoá biểu thức điều kiện sinh ra từ chính sách của hệ thống. Tăng hiệu xuất đánh giá chính sách.

1. **Mục tiêu và phạm vi nghiên cứu**
   1. **Mục tiêu**

Mục tiêu nghiên cứu chính của luận văn xây dựng hệ thống điều khiển truy xuất, đánh giá, phân tích dựa vào SMT Solver. Để thực hiện mục tiêu này, luận văn hướng đến các mục tiêu cụ thể cần đạt được như sau:

* Tìm hiểu các kiến thức, quy trình về Access Control.
* Tìm hiểu các kiến thức, quy trình của XACML.
* Tìm hiểu các kiến thức, phân tích Logic Vị từ.
* Tìm hiểu các kiến thức, quy trình của ALFA.
* Tìm hiểu các kiến thức SMT-Z3.
* Đưa ra cơ chế đánh giá chính sách (evaluation policy) .
* Đo đạt thời gian evaluation policy so khớp với các open source khác.
* Tìm hiểu tạo dữ liệu thực tế trên mockaroo.
* Tìm hiểu về cơ sở dữ liệu, phân tích mongodb.
* Kết hợp evalutation policy với mongodb
  1. **Phạm vi nghiên cứu**

Nhằm giới hạn phạm vi nghiên cứu theo như mục tiêu đề ra, luận văn chỉ tập trung phân tích, đánh giá các yếu tố nằm trong phạm vi sau:

* Luận văn tập trung nghiên cứu giái thuật về phân tích policy dựa vào SMT-Z3, và giải thuật mongodb kết hợp phân tích policy.
* Luân văn sẽ nghiên cứu trong case study student. Cụ thể hơn, chúng em lựa một chọn quy trình sinh viên trong lớp học để đánh giá sinh này có học giáo viên không. Từ đó, đánh giá permit, deny case study trả về mongodb để hệ thống đưa ra cơ sỡ dữ liệu. Hệ thống được phát triên backend cho console sau này cố gắng tạo library cho web springMVC .
  1. **Ý nghĩa khoa học và thực tiễn nghiên cứu**

**3.1 Ý nghĩa khoa học:**

Vận dụng những kiến thức đã học để xây dựng một hệ thống Access Control dựa vào SMT-Z3. Xây dựng các giải thuật xử lý phân tích, đánh giá policy kết hợp mongodb để chứng minh Access Control dựa vào SMT-Z3.

* 1. **Ý nghĩa thực tiễn**
* Các hệ thống, ứng dụng ngày càng nhiều dẫn đến dữ liệu người dùng các tăng pháp sinh ra bigdata, dẫn đến cần một hệ thống điều khiển truy xuất hiểu quả hơn. Từ đó có thể thấy được tầm quan trọng và nhiều lợi ý như:
* Giúp Admin kiểm soát được người dùng vào hệ thống, đảm bảo hệ thống an toàn, tránh trường hợp xấu xảy ra.
* Giúp người dùng ngăn cảng các hành vi cố ý truy cập hay vô tình truy cập được vào một tập không mong muốn đánh cấp thông tin người dùng ngày ra hiểu hệ quả xấu.
* Ngăn trặn các hành vi truy cập chiếm quyền hệ thống, chia sẽ tập dữ liệu không cho phép.

# **TỔNG QUAN**

1. **Tổng quan tài liệu**
2. **Nghiên cứu trong nước**

Có rất nhiều hướng tiếp cận để giải quyết bài toán đánh giá chính sách như Decision Diagrame, Statistic, Decision Tree, SAT và hướng tiện tại là SMT. Nhưng ở Việt Nam cho đến thời điểm này chưa có công trình nghiên cứu nào của các tác giả trong nước bàn luận một cách tập trung và có hệ thống về việc xây dựng và phát triển Access control analysis with SMT **Solver**. Tuy nhiên có các bài báo cáo, nghiên cứu của các thạc sĩ và tiến sĩ riêng lẽ về Access control with SMT and big data[ 1],…

1. **Nghiên cứu ngoài nước**

Ngược lại với trong trong nước, ở ngoài nước rất sôi nổi nghiên cứu về các Access control analysis policy ví dụ như bài báo “Về việc sử dụng giải quyết smt để đánh giá chính sách xacml ”[2], và bài” Phân tích chính sách xacml với smt”[3], “Phân tích chính thức các chính sách xacml bằng smt”[4], đây là những bài báo mới nhất SMT ,.. trong những năm gần đây.

1. **Đối tượng và phương pháp nghiên cứu**
2. **Đối tượng nghiên cứu**

* Luận văn tập trung nghiên cứu các kỹ thuật xử lý thời gian, các công việc bắt đầu từ việc viết các các policy, đến phân tích policy, đánh giá policy trả kết quả về mongodb, từ mongodb đưa ra cơ dỡ liệu.
* Đề xuất ra một test case study student để test thử mô hình Access control with SMT-Z3 để giải quyết bài toán xây dưng phân quyền trong trường học.
* Điều này định hướng trong tương lai, hệ thống được xây dựng có thể trở thành các hệ thống được dùng trong web security, hay phân quyền trong các hệ thống dữ liệu big data ngày nay.

1. **Phương pháp nghiên cứu**

* Đề tài áp dụng hai phương pháp nghiên cứu đó là: nghiên cứu lý thuyết, trực tiếp xây dựng test case, giải thuật xử lý phân tích, đánh giá cũng như giải thuật nhận đánh giá trả ra cơ sở dữ liệu.
* Nghiên cứu lý thuyết để tìm hiểu các thông tin liên quan đến các vấn đề cần giải quyết liên quan đến hệ thống Access control hay SMT\_Z3.
* Xây dựng giải thuật phân tích, đánh giá policy trên version 1, 3 XACML của các trường hợp test case đề ra.
* Kết hợp các vấn đề nghiên cứu và lý thuyết để thu thập và phân tích, pháp triển mô hình Access Control with SMT-z3 . Từ đó phân tích thiết kế hệ thống hoàn chỉnh để hay một hệ thống Access control , phân tích đánh giá với SMT solver cùng với mongodb.

# **CƠ SỞ LÝ THUYẾT**

1. **Access Control là gì ?**
   1. **Khái niệm**

* ***Kiểm soát truy cập (Access control):****.* dùng để chỉ các hoạt động nhằm hạn chế việc truy cập vào một tòa nhà, một văn phòng,… hay nói rộng ra là một chủ thể. Nó chỉ cho phép những người được ủy quyền mới có thể truy cập, những người không được cấp quyền sẽ không được phép truy cập.
* Access control có thể được thực hiện bởi sức của con người như người bảo vệ, người lễ tân hoặc bằng sức cơ như khóa và chìa khóa hoặc áp dụng khoa học kỹ thuật tiên tiến để ứng dụng như hệ thống điều khiển truy cập sử dụng đầu đọc thẻ từ, đầu đọc vân tay, khuôn mặt, password,…
* Hiện nay với sự phát triển của công nghệ thì thuật ngữ access control được ngầm hiểu để chỉ về hệ thống kiểm soát sử dụng các bộ điều khiển, [**đầu đọc thẻ từ**](http://soyal.com.vn/dau-doc-the/) để điều khiển sự truy cập của con người khi ra vào 1 tòa nhà, văn phòng, cơ quan, và mới nhất access control được ứng dụng công nghệ thông tin, phô biến nhất là trong security server.
* Một hệ thống điều khiển truy cập có 3 cách kiểm soát khác nhau: truy cập chính sách (access control *policy),* mô hình kiểm soát truy cập (access control *model)* và cơ chế kiểm soát truy cập (access control *mechanism).*
* **Access control gồm:** 
  + **Bảo vệ các đối tượng (Protect objects):** bảo vệ các tài nguyên hệ thống
    - Ví dụ: tài nguyên bộ nhớ, tập tin, thư mục, tài nguyên phần cứng, phần mềm, bảng, các bộ, ..v.v..
  + **Chủ thể (subjects)**: các đơn vị hoạt động yêu cầu truy cập đến tài nguyên,
    - Ví dụ: người sử dụng, chủ sở hữu, chương trình, ..v.v..
  + **Chế độ truy cập (access mode)**: các kiểu truy cập
    - Ví dụ: đọc / select, viết / cập nhật, thực thi.
* **Chức năng của access control**
* Cấp phép hoặc từ chối phê duyệt sử dụng các tài nguyên xác định cho các chủ thể.
* Kiểm soát được các đối tượng đang hoạt động hay các đối tượng có thể bị truy cập bởi các hoạt động khác.
* **Ai có thể cấp quyền và thu hồi quyền truy cập:**
* *Centralized* administration (người quản trị trung tâm): nhân viên an ninh.
* *Decentralized* administration (người quản trị phân cấp).
* *Hierarchical decentralization (phân cấp theo thứ bậc)*: nhân viên an ninh, quản trị hệ thống phòng ban, người quản trị Windows
* *Ownership based (quyền sở hữu)*: có thể cấp quyền truy cập cho những người khác truy xuất dữ liệu của mình
* *Cooperative authorization (Ủy quyền)*: ủy quyền cho các chủ thể khác
  1. **Các loại access control :**
  2. **DAC (Discretionary Access Control)**
* Cho biết chủ thể nào có thể truy cập kiểu gì đến các đối tượng CSDL.
* Có những nguyên tắc để 1 chủ thể có thể tùy ý cấp quyền hay lấy lại quyền cho/ từ 1 chủ thể khác.
* Subject có thể Action để Object
* Subject có thể grant khác Subject
* Dựa trên người dùng và nhóm

|  |  |  |
| --- | --- | --- |
| 1.Cấp cho Dino bài viết tạo sự cho phép: | Subject | Dino |
| Action | Create |
| Object | Article |
| 2.Dino có thể tạo bài viết ngay bây giờ và cấp quyền này cho người khác. | | | | | |
| 3.Dino cấp cho James để tạo ra các bài báo | Subject: | James |  |  |  |
| Action | Create |  |  |  |
| Object | Article |  |  |  |
| 4.James có thể tạo bài viết ngay bây giờ. | | | | | |

* Thí dụ

Bảng 1: ví dụ về Discretionary Access Control

* 1. **MAC (Mandatory Access Control)**

**Mỗi** Object **và** Object **đều có một tập các thuộc tính bảo mật.**Bất cứ khi nào một đối tượng cố gắng truy cập một đối tượng, một quy tắc ủy quyền được thực thi bởi nhân hệ điều hành sẽ kiểm tra các thuộc tính bảo mật này và quyết định xem việc truy cập có thể diễn ra hay không. Bất kỳ hoạt động nào của bất kỳ đối tượng nào trên bất kỳ đối tượng nào đều được kiểm tra theo bộ quy tắc ủy quyền (còn gọi là Policy) để xác định xem hoạt động có được phép hay không.

Với Mandatory Access Control, chính sách bảo mật này được kiểm soát tập trung bởi quản trị viên chính sách bảo mật; Người dùng không có khả năng ghi đè chính sách và, ví dụ, cấp quyền truy cập vào các tệp sẽ bị hạn chế.

* Subject có thể Action để Object
* Object có thể Action bằng Subject
* Dựa trên người dùng và nhóm

Thí dụ

|  |  |  |  |
| --- | --- | --- | --- |
| 1.Cấp cho Dino bài viết tạo sự cho phép | Subject | Dino | 3.Dino có thể tạo bài viết ngay bây giờ. |
| Action | Create |
| Object | Article |
| 2.Hãy để điều có thể được tạo ra bởi Dino | Subject | Article |
| Action | Created |
| Object | Dino |

Bảng 2: ví dụ về Mandatory Access Control

* 1. **RBAC (Role-based Access Control)**

– RBAC khác với access control lists (ACL), được sử dụng trong các hệ thống kiểm soát truy cập tùy ý truyền thống, ở chỗ **nó gán quyền cho các hoạt động cụ thể có ý nghĩa trong tổ chức** , thay vì cho các đối tượng dữ liệu cấp thấp. Ví dụ, một access control lists có thể được sử dụng để cấp hoặc từ chối quyền truy cập ghi vào một tệp hệ thống cụ thể, nhưng nó sẽ không cho biết cách thay đổi tệp đó. Trong một hệ thống  RBAC-based, một hoạt động có thể là 'tạo giao dịch tài khoản tín dụng' trong một ứng dụng tài chính hoặc 'điền vào hồ sơ kiểm tra mức đường huyết' trong một ứng dụng y tế.

* Subjectlà một Roletrong đó có Permissioncủa ActionđểObject
* Có thể thực hiện kiểm soát truy cập bắt buộc (MAC) hoặc kiểm soát truy cập tùy ý (DAC).
* (Người dùng hoặc nhóm) -role-allow-object
* Khái niệm
  + Subject ( Chủ thể)
  + Role( Vai trò )
  + Permission( Giấy phép )
  + Operation ( Hoạt động )

Nhóm vs Vai trò

* Nhóm: tập hợp người dùng
  + Dino, James và Liam là thành viên của Tổ chức Meifamly.
* Vai trò: tập hợp các quyền
  + Nhà văn là một vai trò, có thể tạo, cập nhật bài viết.
  + Vai trò có thể được áp dụng cho người dùng và nhóm.

Các quy tắc cơ bản của RBAC là:

* Người dùng chỉ có thể thực hiện một thao tác nếu có vai trò được gán cho chủ thể.
* Xác định và xác thực không được coi là hoạt động.
* Tất cả các hoạt động của người dùng được thực hiện thông qua các hoạt động.

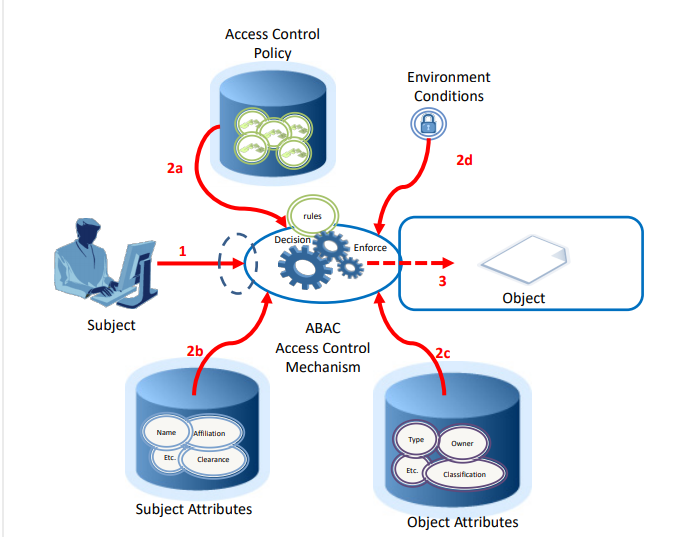
Thí dụ

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.Đặt quyền write articlevà tên manage article | Permission | Name | | write article | 2.Đặt một Vai trò được đặt tên Writer, cho write articlephép nó và một Vai trò được đặt tên Manager, cho manage articlephép nó . CEO có tất cả các quyền. | Role | Name | Writer |
| Operations | Object | | Article | Permissions | write article |
| Action | | Created | Name | Manager |
| Object | | Article | Permissions | manage article |
| Action | | Updated | Name | CEO |
| Object | | Article | Permissions | write article |
| Action | | Read | manage article |
| Name | | manage article |
| Operations | Object | | Article |
| Action | | Delete |
| Object | | Article |
| Action | | Read |
| 3.Writer Đóng vai Dino | Subject | Dino | |
| Roles | Writer | |
|
| 4.Dino có thể tạo bài viết ngay bây giờ | | | | | | | | |
| Đưa James Writervà Managervai trò | | | Subject | James |
| Roles | Writer |
|  | Manager |
| 5.James có thể tạo và xóa bài viết ngay bây giờ | | | | | | | | |

Bảng 3: ví dụ về Role-based Access Control

* 1. **ABAC (Attribute-based access control)**

Không giống như điều khiển truy cập dựa trên vai trò (RBAC), sử dụng các vai trò được xác định trước mang một nhóm đặc quyền cụ thể liên quan đến chúng và đối tượng được gán, **sự khác biệt chính với ABAC là khái niệm về các chính sách thể hiện một bộ quy tắc Boolean phức tạp có thể đánh giá nhiều thuộc tính khác nhau.**

* Subject ai là xxx có thể Action đến Object đó là xxx trong Environment.
* Khái niệm
  + Policies: tập hợp các thuộc tính để thể hiện những gì có thể xảy ra và không được phép.
  + Attributes
    - Subject
      * Tuổi, giải phóng mặt bằng, bộ phận, vai trò, chức danh công việc.
    - Action
      * Đọc, xóa, xem, phê duyệt
    - Resource
      * Loại đối tượng (hồ sơ bệnh án, tài khoản ngân hàng,), bộ phận, phân loại hoặc độ nhạy, vị trí.
    - Contextual (environment)
      * Các thuộc tính xử lý thời gian, vị trí hoặc các khía cạnh động của kịch bản kiểm soát truy cập.

Hình 1: Ảnh minh họa về **Attribute-based access control**

1. Subject requests access to object.

2. Access Control Mechanism evaluates.

a) Rules,

b) Subject Attributes,

c) Object Attributes,

d) Environment Conditions to compute a decision.

3. Subject is given access to object if authorized.

Thí dụ

* Dino, người trong Bộ phận Sản phẩm với tư cách là Nhà văn có thể tạo và cập nhật bài viết, thẻ nào là công nghệ và phần mềm ở chế độ nháp và kết nối đến từ Đài Loan trong khoảng thời gian từ 2017-12-01 đến 2017-12-31.

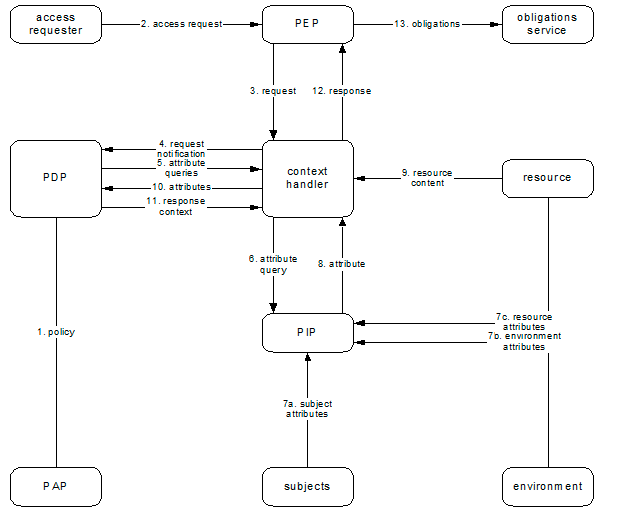
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Subject | | Action | Resource |  | Contextual |  |
| Name | Dino | create | Type | Article | Location | Taiwan |
| Department | Product | update | Tag | technology | StartTime | 2017-12-01 |
| Role | Writer | software | EndTime | 2017-12-31 |
|  | | | Mode | draft |  | |

Bảng 4: ví dụ về **Attribute-based access control**

* Tiêu chuẩn
  + XACML (Ngôn ngữ đánh dấu kiểm soát truy cập có thể mở rộng)

1. **GIỚI THIỆU VỀ XACML**
   1. **Tìm hiểu XACML**
   2. **Giới thiệu XACML**

* eXtensible Access Control Markup Language được viết là XACML.
* XACML là một tài liệu từ Oasis được hình thành để củng cố sự nổ lực các bên liên quan như IBM và Trường đại học Milan. Nó được sử dụng để kết hợp với SAML và đem đến một ý nghĩa lớn trong việc tiêu chuẩn hóa các quyết định kiểm soát truy cập cho tài liệu XML. XACML còn được gọi là XACL được dùng để xác định có nên cho phép truy cập đến nguồn tài nguyên hay là toàn bộ một tài liệu, hay trên nhiều tài liệu hay một phần của tài liệu.
* XACML phiên bản 2.0 đã được chấp nhận như một tiêu chuẩn OASIS cùng với sáu cấu hình của XACML: SAML 2.0, XML Digital Signature, Privacy Policy (chính sách bảo mật), Hierarchical Resource (phân cấp tài nguyên) và RBAC (Role-Based Access Control). XACML là một tiêu chuẩn bổ sung của OASIS để đưa ra các quyết định việc điều khiển truy cập
* XACML phiên bản 3.0 có các cấu hình cho các quy định Kiểm soát xuất khẩu (EC), Kiểm soát sở hữu trí tuệ (IP), chuẩn hóa các lược đồ thuộc tính và hơn thế nữa, ví dụ như cho phép các tổ chức có triển khai XACML khác nhau cộng tác dễ dàng hơn. Với XACML 3.0, cấu hình SAML hiện có cũng được nâng cao.
  1. **Cấu trúc XACML**
* Mô hình XACML:



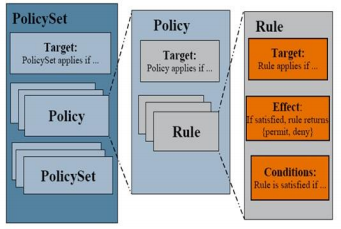
Hình 2. Kiến trúc xử lý XACML

* Bảng viết tắt:

|  |  |  |  |
| --- | --- | --- | --- |
| Tên viết tắt | Tên tiếng anh | Tên tiếng việt | Mô tả |
| PEP | Policy Enforcement Point | Điểm thực thi chính sách | Thực hiện kiểm soát truy cập bằng cách yêu cầu quyết định và thực thi các quyết định ủy quyền |
| PAP | Policy Administration Point | Điểm quản lý chính sách | Tạo và lưu trữ chính sách bảo mật |
| PDP | The Policy Decision Point | Điểm quyết định chính sách | Nhận, xem xét yêu cầu. Sau đó áp dụng các chính sách cùng với việc đánh giá các chính sách đó rồi trả về PEP |
| PIP | Policy Information Point | Điểm thông tin chính sách | Là nguồn gốc của các giá trị thuộc tính hoặc các dữ liệu cần thiết để đánh giá chính sách |
|  | Context Handler | Bối cảnh xử lý | Xác định để chuyển đổi các yêu cầu theo định dạng gốc của nó với hình thức XACML và chuyển đổi các quyết định ủy quyền theo hình thức XACML sang định dạng gốc |

Bảng 1: tóm tắt cấu trúc XACML

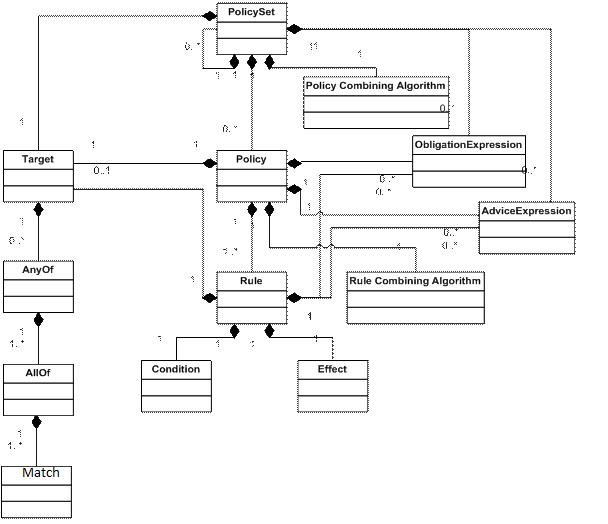
* Mô hình hoạt động theo các bước sau:

1. ***PAPs*** viết  ***policies*** (chính sách) and ***policy sets (bộ chính sách)*** và cung cấp chúng cho ***PDP.*** Các ***policies*** (chính sách) hoặc ***policy sets***  (bộ chính sách )thể hiện ***policy***  hoàn chỉnh cho mộ ***targe (mục tiêu)***được chỉ định.
2. Người yêu cầu ***access***  (***truy cập)*** gửi yêu cầu ***access*** (***truy cập***) vào ***PEP***
3. Các PEP gửi yêu cầu cho access (truy cập) đến context handler (xử lý bối cảnh) ở định dạng yêu cầu có nguồn gốc của nó, tùy chọn bao gồm attributes của **subjects**, **resource**  , ***action***, **environment**  và các thể loại khác.
4. ***Context handler***  (xử lý bối cảnh) xây dựng một  ***context (bối cảnh***) yêu cầu XACML, ùy chọn thêm các thuộc tính và gửi nó đến  ***PDP***
5. Các ***PDP*** yêu cầu bất kỳ thêm **subjects**, **resource**  , ***action***, **environment**  và các thể loại khác (không hiển thị) **attributes**  từ ***context handler***
6. ***Context handler***  yêu cầu các **attributes**  từ một ***PIP***
7. Các ***PIP*** có được các yêu cầu **attributes**
8. Các ***PIP*** trả về yêu cầu ***các* attributes**  cho ***Context handler***
9. Tùy chọn, ***Context handler*** bao gồm **resource**   trong ***Context***
10. ***Context handler*** gửi các **attributes**  được yêu cầu và (tùy chọn) **resource**   tới ***PDP*** . Các ***PDP*** đánh giá ***policies*** .
11. Các ***PDP*** trả về phản ứng ***Context*** (bao gồm cả ***authorization decision (quyết định uỷ quyền)*** ) để ***Context handler***
12. ***Context handler*** dịch  ***Context*** đáp ứng sang định dạng phản hồi gốc của PEP . Trình xử lý ngữ cảnh trả về phản hồi cho PEP .
13. Các ***PEP*** đáp ứng các ***obligations***
14. Nếu ***access***  được cho phép, thì ***PEP*** cho phép ***access***  vào **resource**  ; nếu không, nó từ chối ***access***
    1. **Thành phần của XACML**

Hình 3:Các khối trong kiến trúc xử lý XACML

Một XACML bao gồm 3 thành phần cơ bản sau:

* Rule
* Policy
* Policy Set



Hình 4: Cấu trúc chính sách XACML

1. **Rule**:

Một Rule là đơn vị cơ bản nhất của Policy. Nó có thể tồn tại trong sự cô lập trong những tác nhân chính của miền XACML. Để trao đổi các Rule giữa các tác nhân chính, chúng phải được đóng gói trong Policy. Một Rule có thể được đánh giá dựa trên nội dung của nó. Các thành phần chính Của Rule:

* Một **target**.
* Một **effect**.
* Một **condition.**
* Biểu thức ***obligation***
* Biểu thức ***advice***

1. **Target**

* Các **target** xác định tập hợp các yêu cầu mà các **Rule** được thiết kế để áp dụng trong các hình thức của một biểu thức logic trên ***các* attributes**  trong yêu cầu. Phần tử <Condition> có thể tinh chỉnh thêm khả năng ứng dụng được thiết lập bởi **target**. Nếu **Rule** được dự định áp dụng cho tất cả các thực thể của một loại dữ liệu cụ thể, thì thực thể tương ứng sẽ bị bỏ qua khỏi **target**. Một XACML ***PDP*** xác minh rằng các kết quả được xác định bởi ***target***  được thỏa mãn bởi các **attributes**  trong ***Context*** yêu cầu .
* Phần tử <Target> có thể không có trong <Rule> . Trong trường hợp này, **Target** của <Rule> giống với đích của phần tử <Policy>element.

1. **Effect**

* Các ***effect***  của ***Rule*** chỉ ra ***Rule***-writer dự định hậu quả của một đánh giá "True" cho ***Rule*** . Hai giá trị được cho phép: " "Permit " và "Deny". Ngoài ra còn có hai các giá trị: “NotApplicable” và “Indeterminate”

1. **Condition (*Điều kiện* )**

* ***Điều kiện*** biểu diễn một biểu thức Boolean để tinh chỉnh khả năng áp dụng của ***quy tắc*** ngoài các biến ***predicates*** ***(vị ngữ) được*** ngụ ý bởi ***mục tiêu*** của nó . Do đó, nó có thể vắng mặt.

1. **Biểu thức *obligation*(nghĩa vụ)**

* ***Các*** biểu thức ***obligation*** có thể được thêm vào bởi người viết ***Rule***.
* Khi một ***PDP*** đánh giá một ***Rule*** chứa ***obligation*** biểu thức, nó đánh giá ***obligation*** biểu thức thành ***obligation*** và trả về một số những ***obligation*** với ***PEP*** trong ứng phó với ***bối cảnh*** .

1. **Biểu thức *advice (tư vấn)***

* ***Các*** biểu thức ***advice*** có thể được thêm vào bởi người viết ***Rule*** .
* Khi một ***PDP*** đánh giá một ***Rule*** chứa ***advice*** biểu thức, nó đánh giá ***advice*** biểu thức hành ***advice*** và trả về một số những ***advice*** cho ***PEP*** trong ứng phó với ***Context*** .

1. **Policy**

Từ mô hình luồng dữ liệu, người ta có thể thấy rằng ***các Rule*** không được trao đổi giữa các thực thể hệ thống. Do đó, ***PAP*** kết hợp ***các Rule***  trong một ***Policy*** . Một **Policy** bao gồm bốn thành phần chính:

* Target
* A ***rule-combining algorithm***-identifier (thuật toán kết hợp thuật toán- quy tắc).
* set Rule
* Biểu thức ***obligation***
* Biểu thức ***advice***

1. **Mục tiêu chính sách (target policy)**

* Phần tử XACML <PolicySet> , <Policy> hoặc <Rule> chứa phần tử <Target> chỉ định tập hợp các yêu cầu mà nó áp dụng. Các <Target> một <PolicySet> hoặc <Policy> có thể được công bố bởi người viết của <PolicySet> hoặc <Policy> , hoặc nó có thể được tính từ <Target> yếu tố của <PolicySet> , <Policy> và <Rule> các phần tử chứa nó.
* Một thực thể hệ thống tính toán một <Target> theo cách này không được định nghĩa bởi XACML, nhưng có hai phương thức logic có thể được sử dụng. Trong một phương thức, phần tử <Target> của <PolicySet> hoặc <Policy> bên ngoài ("thành phần bên ngoài") được tính như là sự kết hợp của tất cả các phần tử <Target> của tham chiếu <PolicySet> , <Policy> hoặc < Rule > các phần tử ("thành phần bên trong").

1. **Rule-combining algorithm-identifier ( thuật toán kết hợp quy tắc)**

* Các ***thuật toán quy tắc kết hợp*** quy định cụ thể thủ tục theo đó các kết quả đánh giá các thành phần ***Rule*** được kết hợp khi đánh giá các ***Policy***, tức là ***decision*** (***quyết định)*** giá trị đặt trong phản ứng ***context***của ***PDP*** là giá trị của các ***Policy***, theo quy định của **thuật toán kết hợp quy tắc**. Một ***Policy*** có thể đã kết hợp các tham số ảnh hưởng đến hoạt động của ***thuật toán kết hợp quy tắc*** .

1. **Biểu thức Obligation (nghĩa vụ)**

* ***Các*** biểu thức **Obligation**  có thể được thêm vào bởi người viết ***policy***.
* Khi một ***PDP*** đánh giá một ***policy*** có chứa **Obligation** biểu thức, nó đánh giá **Obligation**  biểu thức thành **Obligation** và trả về một số những **Obligation** với ***PEP*** trong ứng phó với ***context***

1. **Biểu thức advice (tư vấn)**

* ***Các*** biểu thức ***advice*** có thể được thêm vào bởi tác giả của ***policy***.
* Khi một ***PDP*** đánh giá một ***policy*** có chứa ***advice*** biểu thức, nó đánh giá ***advice*** biểu thức thành ***advice*** và trả về một số những ***advice*** cho ***PEP*** trong ứng phó với ***context.***

1. **Policy set (bộ chính sách)**

Một ***Policy set*** bao gồm bốn thành phần chính:

* Target;
* A ***policy-combining algorithm***-identifier (***chính sách kết hợp thuật toán*** –định danh)
* A policy set;
* Biểu thức Obligation
* Biểu thức***advice***

1. **Policy-combining algorithm (Thuật toán kết hợp chính sách)**

* ***Các Policy-combining algorithm quy định cụ thể thủ tục theo đó các kết quả đánh giá các thành*** phần  ***Policy*** được kết hợp khi đánh giá ***Policy set***, tức là các Quyết định giá trị đặt trong phản ứng ***context*** của ***PDP*** là kết quả của việc đánh giá ***policy set*** , theo quy định của **Policy-combining algorithm** . Một ***Policy set*** có thể kết hợp các tham số ảnh hưởng đến hoạt động của **Policy-combining algorithm**

1. **Biểu thức Obligation (nghĩa vụ)**

* Tác giả của ***Policy set***  có thể thêm các biểu thức ***obligation*** vào ***Policy set*** ngoài các biểu thức có trong các ***Rule***  thành phần , ***các Policy*** và ***các Policy set***.
* Khi một ***PDP*** đánh giá một ***Policy set***.có chứa ***các* Obligation** biểu thức, nó đánh giá **Obligation** biểu thức thành **Obligation** và trả về một số những **Obligation** với ***PEP*** trong phản ứng của nó ***Context***

1. **Biểu thức Advice**

* ***Các*** biểu thức ***Advice*** có thể được thêm vào bởi người viết của ***Policy set***..
* Khi một ***PDP*** đánh giá một ***bộ sách*** chứa ***Advice*** biểu thức, nó đánh giá ***Advice*** biểu thức thành ***Advice*** và trả về một số những ***Advice***  cho ***PEP*** trong ứng phó với ***Context***
  1. **The Logic of XACML**

1. **GIỚI THIỆU VỀ LOGIC VỊ TỪ**
   1. **Xacml**

Việc đánh giá các thành phần XACML bắt đầu từ đánh giá hàng loạt và nó được tiếp tục lặp đi lặp lại cho đến khi đánh giá **PolicySet**. Các giá trị **Match**, **AllOf**, **AnyOf** và **Target** đều khớp nhau, không khớp hoặc không xác định. Giá trị có thể không xác định nếu có lỗi trong quá trình đánh giá để không thể đưa ra quyết định tại thời điểm đó. Việc đánh giá **Rule** phụ thuộc vào Đánh giá **Target** và Đánh giá **Condition**. Thành phần **Condition** là một tập hợp các công thức mệnh đề mà mỗi công thức được đánh giá là đúng, sai hoặc không xác định. Điều kiện trống luôn được đánh giá là đúng. Kết quả của **Rule** là có thể áp dụng, không áp dụng hoặc không xác định. **Rule** áp dụng có hiệu lực hoặc từ chối hoặc cho phép. Cuối cùng, việc đánh giá **Policy** và **PolicySet** được dựa trên thuật toán kết hợp mà kết quả có thể được áp dụng (với hiệu ứng của nó hoặc từ chối hoặc cho phép), không áp dụng hoặc không xác định.

* 1. **Three-Valued Lattice (Lưới ba giá trị)**

Chúng tôi sử dụng logic có giá trị ba để xác định giá trị đánh giá XACML. Chúng tôi định nghĩa L3 =(V3, ≤ )là mạng ba giá trị trong đó V3 là tập hợp {>, I, ⊥} và ⊥ ≤ I ≤ T.

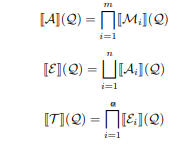
Chúng tôi sử dụng ký hiệu J.K để ánh xạ các phần tử XACML vào các giá trị đánh giá của chúng. Các đánh giá các thành phần XACML với các giá trị trong V3 được tóm tắt trong Bảng

|  |  |  |  |
| --- | --- | --- | --- |
| V3 | Match and Target value | Condition value | Rule, Policy and PolicySetvalue |
| T | Match | True | Applicable (either deny or permit) |
| ⊥ | not match | false | not applicable |
| I | indeterminate | indeterminate | indeterminate |

Bảng 2: **Three-Valued Lattice (Lưới ba giá trị)**

* 1. **The Match Evaluation**

Để M là một trận đấu, A = M1 ∧. . . ∧ Mm là một AllOf, E = A1 ∨. . . ∨ An là một AnyOf, T = E1 ∧. . . ∧ Eo là mục tiêu và Q là yêu cầu. Sau đó, các đánh giá của AllOf, AnyOf và Target như sau:



Hình5: the match evaluation

Tóm lại, chúng ta có thể đơn giản hóa việc đánh giá mục tiêu như sau:



Hình 6: the match evaluation basic

* 1. **Condition Evaluation**

Chúng tôi xác định hàm đánh giá có điều kiện eval là một hàm tùy ý để đánh giá

Điều kiện để giá trị trong V3 cho một thành phần Yêu cầu Q. Đánh giá

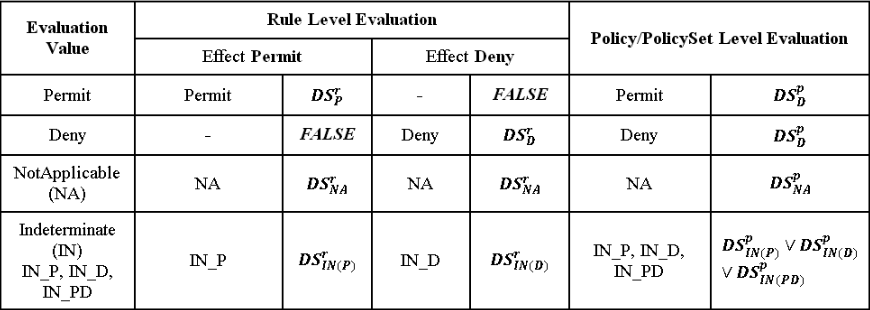
Điều kiện được định nghĩa như sau:



Hình 7: Condition evaluation

* 1. **Extended Values**

Chúng được biểu diễn qua bảng sau:



Bảng 3. **Extended Values**

**DS**: biểu thức

**R**: rule

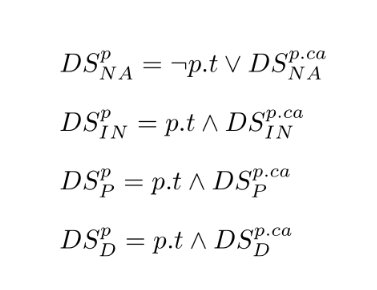
**P**: Permit: Cho phép

**D**: Deny: Từ chối

**NA**: Not Applicable: Không áp dụng

**IN**: Indeterminate: Không xác định

* 1. **Biểu thức rút gọn**

 Được biểu diễn qua hình sau:

Hình 8: biểu thức rút gọn

**Chú thích:**

**p.t**: là Target

**p**: là Permit (cho phép)

**in**: là Indeterminate (erro)

**Na**: là NotApplicable (không liên qua)

**D**: là Deny ( không cho phép)

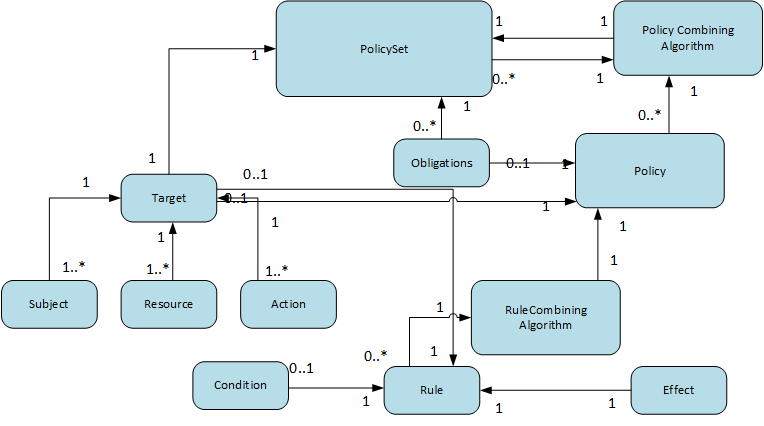
**p.ca:** làpolicyCombining Algorithm

1. **Kỹ thuật viết XML bằng ALFA**
2. **History ALFA**

* XACML là ngôn ngữ đánh dấu kiểm soát truy cập có thể phát triển, sử dụng XML làm ngôn ngữ mã hóa chính của nó. Các nhà phát triển luôn phải vật lộn để viết XML và do đó một ký hiệu mới, nhẹ hơn, là cần thiết. Nhà nghiên cứu Axiomatics, Pablo Giambiagi, do đó thiết kế ALFA, Ngôn ngữ Axiomatics cho phép.
* ALFA ánh xạ trực tiếp vào XACML. ALFA chứa các phần tử cấu trúc giống như XACML, ví dụ như PolicySet, Policy và Rule

1. **Concepts**

Ngôn ngữ ALFA, là một biểu diễn mã giả của XACML, kế thừa cấu trúc tổng thể và các khái niệm về ngôn ngữ XACML.



Hình 9: mô hình mô phỏng ALFA

1. **Tổng quan về các phần tử XACML và các phần tử ALFA tương ứng**

hinh

|  |  |  |
| --- | --- | --- |
| XACML | ALFA | Description |
| PolicySet | policyset |  |
| Policy | policy |  |
| Rule | rule |  |
| Condition | condition |  |
| Effect | Either of permit or deny. | There is no Effect element in ALFA. The user can directly insert the value. |
| Target | target |  |
| AnyOf | or |  |
| AllOf | and |  |
| Policy Combining Algorithm | apply |  |
| ObligationExpression | obligation |  |
| AdviceExpression | advice |  |
| RuleCombiningAlgorithm | apply |  |

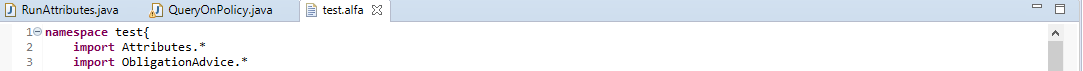
Bảng 4. các phần tử XACML và các phần tử ALFA tương ứng

1. **Comments**

Comments mã ALFA bằng cách đánh dấu chúng bằng ‘//’ hoặc ‘/\*..\*/

1. **Namespace**

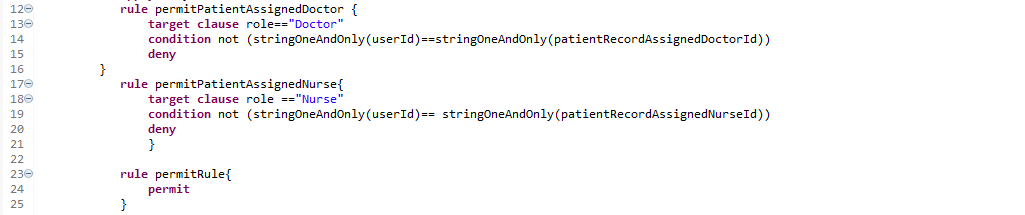
Các phần tử XACML như các Policy, các Policysets và các Rule được đưa ra các định danh trừu tượng (các tên) được thu thập vào các không gian tên. Không gian tên được sử dụng để thu thập các phần tử liên quan theo cấu trúc tên được chia sẻ. Ở một từ khác, không gian tên được sử dụng để khai báo một phạm vi chứa một tập các đối tượng liên quan. Không gian tên là loại phạm vi duy nhất tồn tại trong ALFA. Tất cả các số nhận dạng phải là duy nhất trong ngữ cảnh của một phạm vi nhất định.



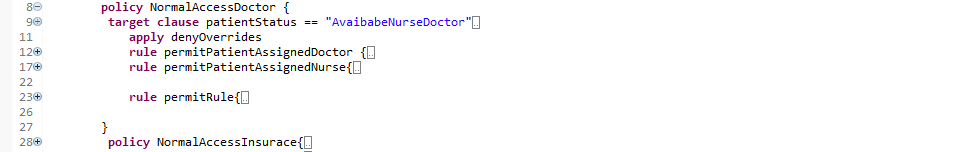
Hình 10: NameSpace của ALFA

1. **Rule**

* Một chính sách có chứa một hoặc nhiều quy tắc, yếu tố cơ bản nhất để xác định các chính sách kiểm soát truy cập.
* Quy tắc được khai báo với quy tắc từ khóa. Mỗi quy tắc phải chứa hiệu ứng của quy tắc, hoặc là giấy phép hoặc từ chối.
* Quy tắc có thể chứa mục tiêu và / hoặc điều kiện và điều này xác định liệu quy tắc có áp dụng cho yêu cầu hay không. Nếu quy tắc không có điều kiện hoặc mục tiêu, thì quy tắc sẽ áp dụng cho mọi yêu cầu.
* Quy tắc có thể được đặt tên rõ ràng sau từ khóa quy tắc trong khai báo. Với tên này, bạn có thể tham khảo quy tắc từ các chính sách và trình biên dịch sẽ sử dụng tên này khi tạo ID quy tắc XACML.Nếu khai báo quy tắc không xác định tên, trình biên dịch sẽ tự động tạo một ID duy nhất cho quy tắc.

Hình 11: Rule của ALFA

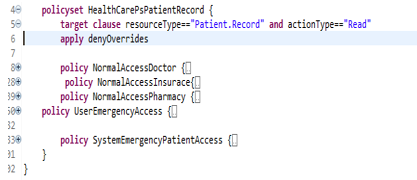
1. **Policy**

* Các chính sách được sử dụng để thu thập nhiều quy tắc. Chính sách được khai báo bằng chính sách từ khóa và có thể được xác định bằng tên, được nêu sau từ khóa. Đặt tên cho một chính sách là tùy chọn, nhưng nó rất hữu ích để định nghĩa nó cho các chính sách mức cao nhất kể từ vùng tên và tên chính sách COULD được sử dụng để xác nhận tên của tệp XACML XML được trình biên dịch tạo ra.
* Chính sách có thể a target and/or a condition. Chúng được sử dụng để xác định xem chính sách có áp dụng cho yêu cầu hay không. Nếu không có mục tiêu hoặc điều kiện, thì chính sách sẽ áp dụng cho mọi yêu cầu.

Hình 12: policy của ALFA

1. **Policy Set**

* Policyset là tập hợp các policyhoặc các policyset khác. Một bộ chính sách được khai báo với từ khóa policyset.
* Policy set có thể chứa a target and/or a condition. Mục tiêu và điều kiện xác định xem tập hợp chính sách có áp dụng cho yêu cầu hay không. Nếu không có mục tiêu hoặc điều kiện, thì tập hợp chính sách sẽ áp dụng cho tất cả các yêu cầu.



Hình 13: policy của ALFA

1. **SMT-Z3**
   * + 1. **SMT là gì ?**

SMT xây dựng một mô hình nội bộ của XACML chính sách được đưa ra làm đầu vào bằng cách trích xuất thông tin liên quan và cơ cấu chính sách. Mô hình này thể hiện các ràng buộc về khả năng áp dụng cho từng yếu tố chính sách (nghĩa là các quy tắc, chính sách và bộ chính sách) cũng như hiệu lực của các quy tắc (Permit hoặc Deny) và thuật toán kết hợp cho các policyvà policySet. Các ràng buộc cho một thành phần chính sách được xây dựng từ các hàm, thuộc tính XACML và các giá trị của chúng xuất hiện trong target và conditions của nó. Đánh giá của họ phân vùng không gian chính sách (tất cả các yêu cầu có thể) thành ba không gian con khác nhau (nghĩa là Applicable, Indeterminate, NotApplicable).

Mô hình nội bộ được sử dụng để tạo các công thức chính sách mã hóa các không gian quyết định của policy, tức là các điều kiện theo đó một quyết định truy cập được trả về bởi đánh giá policy ( là Permit, Deny, Indeterminate and NotApplicable).

* + - 1. **Z3- microsoft ?**

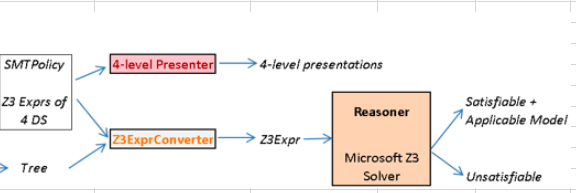
Z3 là một nhà tiên tri tiên tiến của Microsoft Research. Nó có thể được sử dụng để kiểm tra sự thỏa mãn của các công thức logic qua một hoặc nhiều lý thuyế. Z3 cung cấp một trận đấu hấp dẫn cho các công cụ phân tích và xác minh phần mềm do một số phần mềm phổ biến xây dựng ánh xạ trực tiếp vào các lý thuyết được hỗ trợ.

Z3 là một công cụ cấp thấp. Nó được sử dụng tốt nhất như là một thành phần trong ngữ cảnh của các công cụ khác yêu cầu giải các công thức logic. Do đó, Z3 trưng bày một số phương tiện API để thuận tiện cho các công cụ ánh xạ vào Z3, nhưng không có trình soạn thảo độc lập hoặc phương tiện lấy người dùng làm trung tâm để tương tác với Z3. Cú pháp ngôn ngữ được sử dụng ở phía trước ưu tiên sự đơn giản trái ngược với sự thuận tiện về ngôn ngữ.

* + - 1. **Kết hợp SMT-z3** :

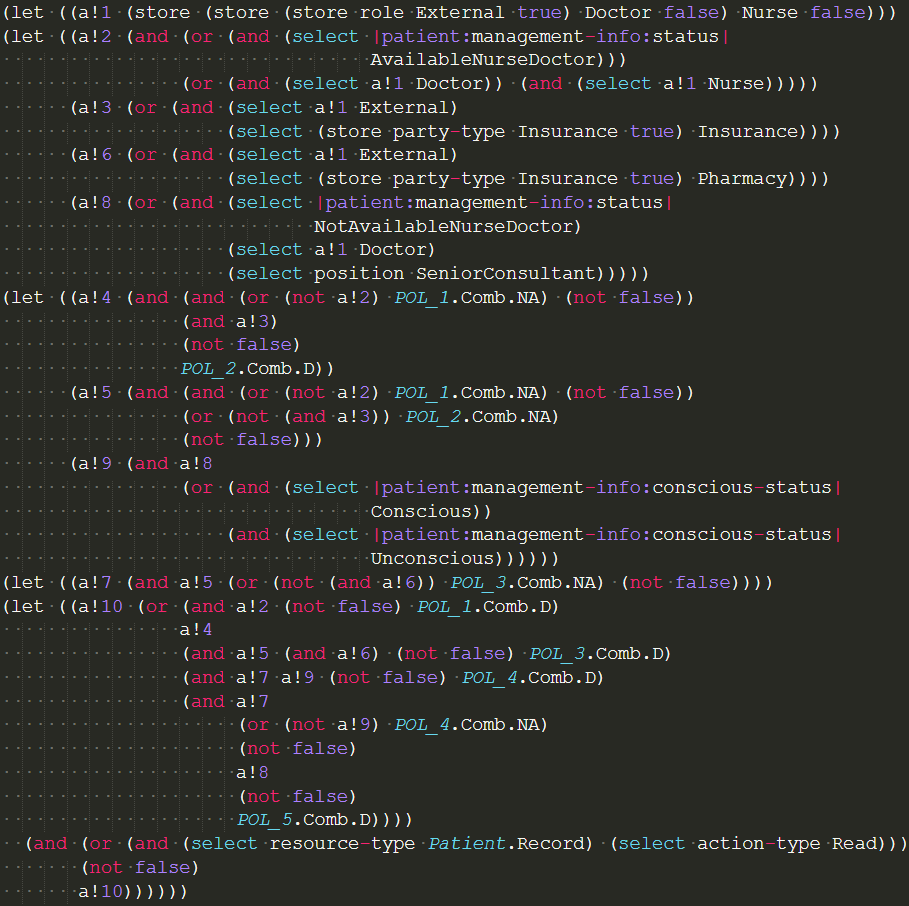
Để kết hợp SMT-z3 vào bài toán evaluation policy ta cần có Query- **[Analysis:](https://research.tue.nl/en/publications/analysis-of-xacml-policies-with-smt)**

Khi đã có policy thì SMT-Policy sẽ chia bài toán thành 4 vùng không gian (ShowAtPolicyOnly, ShowAtPolicyWithTargetCombination, ShowAtRule, ShowAtConstraint) và policyFormulas. Còn z3 sẽ dựa vào policyFormulas và Query- **[Analysis](https://research.tue.nl/en/publications/analysis-of-xacml-policies-with-smt)** [sinh ra Tree . Cuối cùng z3 và tree trả ra kết quả Permit or Deny or Indeterminate or NotApplicable](https://research.tue.nl/en/publications/analysis-of-xacml-policies-with-smt)



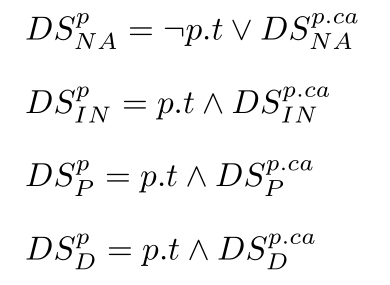
Hình 14: Kết hợp SMT-z3 vào XACML

* Giải pháp phân tích chính sách đặc tả XACML ở nhiều mức độ
* Ý tưởng chính ở đây được kế thừa một hệ thống có sẵn là tạo ra một bộ công cụ cho phép biểu diễn các vùng không gian decision của các đối tượng PolicySet, Policy dựa theo từ vùng không gian decision của các đối tượng con cho đến mức độ thấp nhất là các Applicable Constraint. Ý tưởng này giúp cho việc biểu diễn vùng không gian decision của đối tượng trở nên linh hoạt tùy theo yêu cầu của người phân tích. Đặc biệt khi người phân tích chỉ muốn quan sát một cách tổng quan về cấu trúc chính sách cũng như các sự kết hợp, không muốn bị chi phối bởi các biểu thức ràng buộc thì phương pháp này hoàn toàn hỗ trợ điều đó như ví dụ ở Hình 13. Biểu thức ở ví dụ này được thu giảm lại chỉ còn một vài điều kiện kết hợp với các biến thể hiện các vùng không gian như:
* *POL\_1.Comb.NA*: vùng không gian Not Applicable khi kết hợp các vùng không gian Not Applicable con từ các Rule của Policy 1.
* *POL\_5.Comb.D*: vùng không gian Deny khi kết hợp các vùng không gian Deny con từ các Rule của Policy 5.



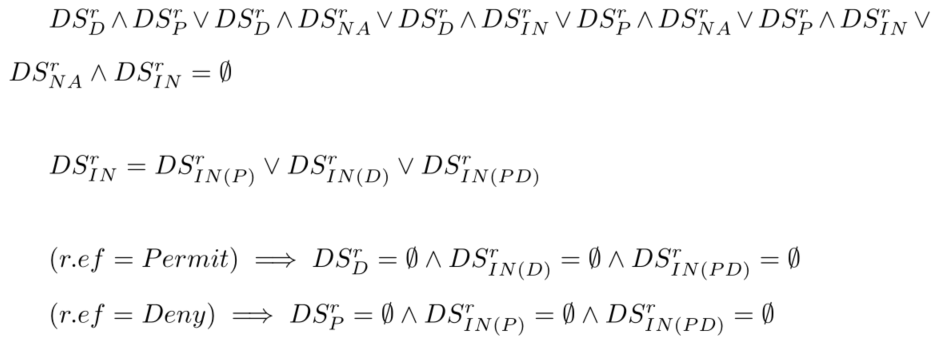
Hình 15: Biểu thức biểu diễn chính sách

* Cách biểu diễn này làm che dấu đi sự phức tạp bên trong các Policy 1, Policy 2... giúp Administrator có thể tập trung sự quan sát vào góc nhìn mới là mối quan hệ giữa các Policy với nhau ảnh hưởng như thế nào đến kết quả đánh giá câu Request.s
* Bên cạnh đó công cụ cũng cho phép thu giảm biểu thức không gian decision bằng cách gán giá trị vào một thuộc tính cụ thể có mặt trong biểu thức, từ đó thu giảm biểu thức trở nên gọn hơn giúp người phân tích tập trung hơn vào những đối tượng chính. Việc gán giá trị vào một thuộc tính cụ thể cũng tương tự như việc gom nhóm các request khác nhau. Khi đó biểu thức sẽ chỉ còn thu gọn lại những đối tượng Policy, Rule nào đang liên quan trực tiếp đến nhóm Request đang xét, từ đó sẽ có những quyết định phù hợp liên quan đến việc công việc quản trị như thêm, chỉnh sửa hoặc xóa các chính sách.
* Giải pháp được triển khai dưới dạng công thức như sau:
  + Từ khái niệm vùng không gian của Policy đưa ra trong bài báo của [<turkmen2017formal>], đề tài bổ sung thêm một biểu diễn mới dựa trên thành phần Target và Combination của Policy đó.



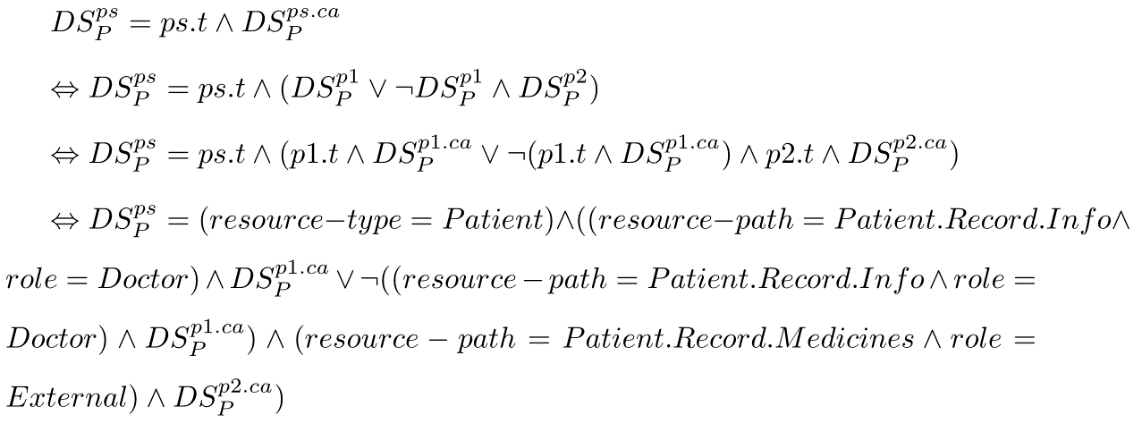
Hình 16: biểu thức rút gọn

* Khi kết hợp các biểu thức không gian với nhau, ta có thể áp dụng các hằng đẳng thức dưới đây để giúp thu gọn hơn các các biểu diễn.



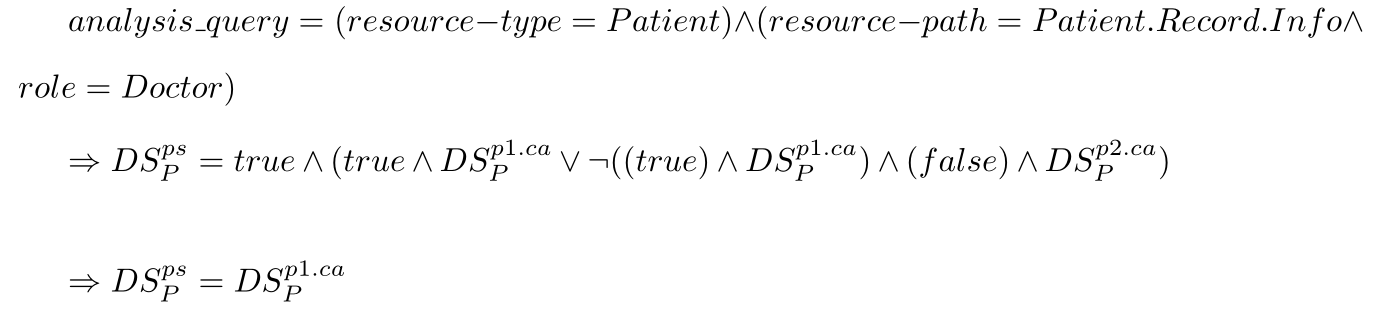
Hình 17: biểu thức kết họp vùng không gian với nhau

* Ví dụ dưới đây phân tích từ vùng không gian Permit của một PolicySet.



Hình 18: ví dụng phân tích vùng không gian Permit

* Và áp dụng bước analyze query với phép gán các giá trị cho một số thuộc tính, ta còn lại biểu thức ngắn gọn cuối cùng cho thấy từ việc PolicySet phụ thuộc vào 2 Policy ban đầu, giờ chỉ còn lại mỗi vùng không gian Combination Permit của Policy 1.



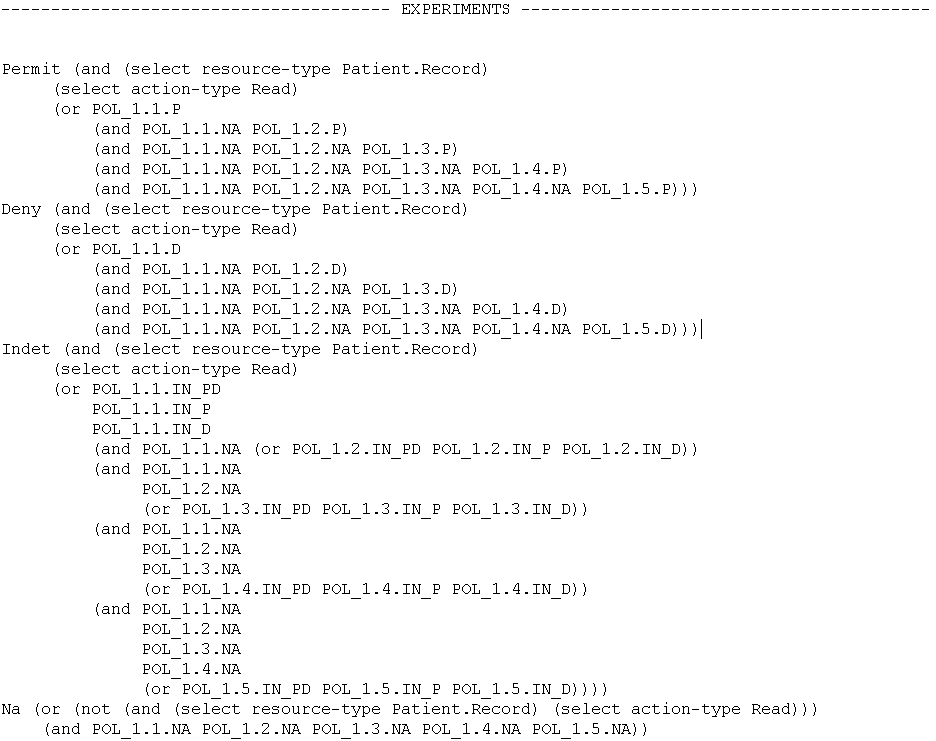
Hình 19: áp dụng câu analyze query với phép gán

* Ta có thể biểu diễn 4 vùng không gian của chính sách ở 4 cấp độ từ một chính sách gốc phức tạp ban đầu ở dạng XML như ở Hình 4‑4. Chi tiết chính sách được thể hiện ở Bảng 0‑2 health-care.afla, Phụ lục A.



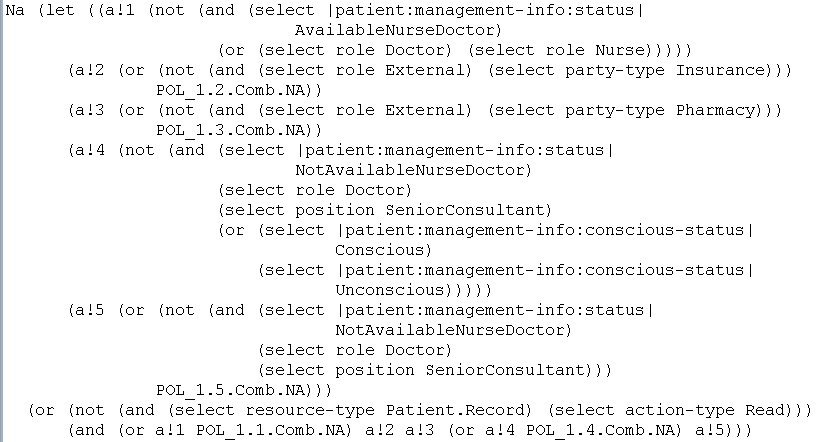
Hình 20: Ví dụ Policy với 1 PolicySet, 5 Policy,12 Rule và 28 Constraint

* Mức (1): Phân tích ở mức Policy cho chính sách trên



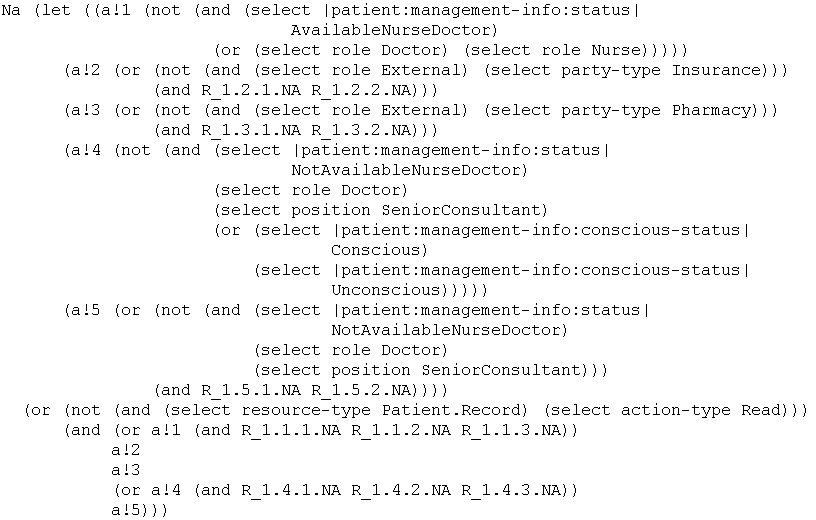
Hình 21: ví dụng phân tích mức policy

* Mức (2): Phân tích ở mức PolicyTarget và PolicyCombination



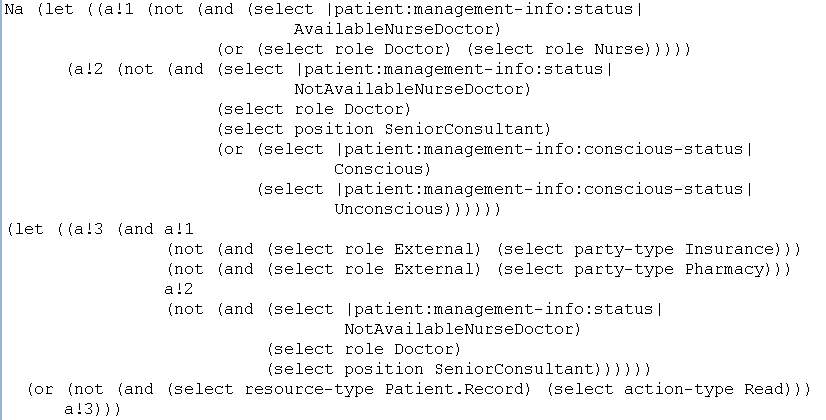
Hình 22: phân tích mức PolicyTarget và PolicyCombination

* Mức (3): Phân tích ở mức Rule



Hình 23: phân tích ở mức Rule

* Mức (4): Và cấp độ mịn nhất là ở mức Constraint



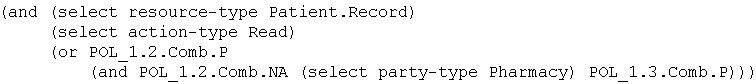
Hình 24: Phân tích ở mức Constraint

* Và với các biểu thức ở các cấp độ khác nhau này, ta có thể thực hiện thêm các câu query phân tích bằng việc thay thế các giá trị vào biểu thức. Từ đó ta có được biểu thức đơn giản cuối cùng nhưng vẫn không làm mất đi ngữ nghĩa của chính sách ban đầu.
* Ví dụ ở đây ta thực hiện câu query:

|  |
| --- |
| (  \\REPLC (  (role = External) \\WEDGE  (\\NEG (role = Doctor \\WEDGE role = Nurse)))  (\\REPLC (party-type = Insurance) P\_1.SHORT)  ) |

Hình 25: Câu query analyze

* Biểu thức cho vùng không gian Permit sẽ được thu gọn lại như hình dưới đây:



Hình 26: Biểu thức cho vùng không gian Permit

1. **Giới thiệu về NoSQL**

* Hệ quản trị cơ sở dữ liệu quan hệ dựa trên SQL đã thống trị hầu hết các hệ quản trị cơ sở dữ liệu. Tuy nhiên, thời gian gần đây, một cách tiếp cận mới đã bắt đầu biết đến là NoSQL, tạo ra sự thay thế cho các hệ quản trị cơ sở dữ liệu quan hệ truyền thống.
* NoSQL còn có nghĩa là Non-Relational - không ràng buộc. Tuy nhiên, thuật ngữ đó ít phổ dụng hơn và ngày nay người ta thường dịch NoSQL thành Not Only SQL - Không chỉ SQL. NoSQL ám chỉ đến những cơ sở dữ liệu không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm
* Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các cơ sở dữ liệu quan hệ nguồn mở nhỏ nhưng không sử dụng SQL cho truy vấn.
* Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL khi Johan Oskarsson của Last.fm muốn tổ chức một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: một thế hệ CSDL không ràng buộc, phân tán, nguồn mở, khả năng mở rộng theo chiều ngang, có thể lưu trữ, xử lý từ một lượng rất nhỏ cho tới hàng petabytes dữ liệu trong hệ thống có độ chịu tải, chịu lỗi cao với những đòi hỏi về tài nguyên phần cứng thấp.
* Một số đặc điểm nhận dạng cho thế hệ CSDL mới này bao gồm: schema-free, hỗ trợ mở rộng dễ dàng, API đơn giản, nhất quán cuối (eventual consistency), không giới hạn không gian dữ liệu,...
* Sau đây là danh sách các CSDL NoSQL:

1. Wide Column Store / Column Families: Hadoop/HBase  –  Apache,  BigTable  –  Google,  Cassandra  - Facebook/Apache, Hypertable - Zvents Inc/Baidu, Cloudera, SciDB, Mnesia, Tablets,…
2. Key-Value Store/Tuple store
   * 1. Key/value cache in RAM: memcached, Citrusleaf database, Velocity, Redis, Tuple space,...
     2. Key/value save on disk: Memcachedb, Berkeley DB, Tokyo Cabinet, Redis,...
     3. Eventually Consistent Key Value Store: Amazon Dynamo, Voldemort, Dynomite, KAI, Cassandra, Hibari, Project Voldemort,…
     4. Ordered key-value store: NMDB, Memcachedb, Berkeley DB,...
     5. Distributed systems: Apache River, MEMBASE, Azure Table Storage, Amazon Dynamo,...
3. Document Store: Apache Jackrabbit, CouchDB, IBM Lotus Notes Storage Format (NSF),  MongoDB,  Terrastore,  ThruDB, OrientDB, RavenDB,...
4. Graph Database: Neo4J, Sones, AllegroGraph, Core Data, DEX, FlockDB, InfoGrid, OpenLink Virtuoso,...

* Tuy cùng mang những đặc điểm chung của NoSQL nhưng mỗi CSDL NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau. Ví dụ:
* MongoDB và Redis là những lựa chọn tốt cho việc lưu trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên.
* Hadoop, một CSDL dạng tự do, phân tán làm tốt công việc lưu trữ các dữ liệu lớn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.
* Memcachedb, một CSDL nhất thời chóng tàn, tuyệt vời trong lưu trữ các phiên làm việc web, các khóa, và các con số thống kê ngắn hạn.
* Cassandra và Riak (các lưu trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn.
* Để tìm hiểu sâu hơn về các CSDL hiện đại NoSQL, chúng ta đi nghiên cứu chi tiết CSDL đặc trưng là MongoDB.

1. **Hệ cơ sở dữ liệu MongoDB**

* Trong những gương mặt góp phần làm suy tàn đế chế SQL thì MongoDB nổi lên là một CSDL đáng tin cậy và dễ dùng nhất. Mongo viết bằng C++. Nó thích hợp cho các ứng dụng tầm trung trở lên. Nếu tỉ lệ lượng dữ liệu ghi vào CSDL của ứng dụng lớn hơn lượng đọc thì đây càng là lựa chọn hợp lý.
* MongoDB là một CSDL có khả năng mở rộng, hiệu suất cao, mã nguồn mở và hướng văn bản.
* Trước khi đi vào tìm hiểu kỹ hơn về MongoDB, chúng ta làm quen với một số khái niệm cơ bản của MongoDB:
* Văn bản (Document) là đơn vị cơ bản của dữ liệu trong MongoDB, nó tương đương với một dòng trong CSDL quan hệ.
* Bộ sưu tập (Collection) có thể được coi như tương đương với một bảng.
* MongoDB có thể lưu trữ nhiều CSDL độc lập, mỗi CSDL này có các bộ sưu tập và điều khoản riêng của mình.
* MongoDB đi kèm với một trình tiện ích JavaScript đơn giản nhưng mạnh mẽ, nó hữu ích trong quản trị và thao tác dữ liệu.
* Mỗi văn bản có một khóa đặc biệt, đó là “\_id”, nó là duy nhất trong bộ sưu tập của văn bản.

**Văn bản**

Văn bản là một khái niệm quan trọng trong MongoDB. Văn bản bao gồm tập hợp các khóa với các giá trị tương ứng.

Ví dụ: {"greeting" : "Hello, world!"}

Văn bản trên gồm một khóa là “greeting”, với giá trị là “Hello, world!”. Các văn bản có thể chứa nhiều cặp khóa/giá trị.

Ví dụ: {"greeting" : "Hello, world!", "foo" : 3}

Một số lưu ý:

* Các cặp khóa/ giá trị trong văn bản được sắp xếp. Văn bản trên sẽ khác với văn bản sau:

{"foo" : 3, "greeting" : "Hello, world!"}

* Khóa trong văn bản là một chuỗi
* MongoDB phân biệt chữ hoa chữ thường
* Văn bản trong MongoDB không được chứa những khóa giống nhau. Ví dụ văn bản sau là không hợp lệ.

{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}

**Bộ sưu tập**

Bộ sưu tập là một nhóm các văn bản. Nếu văn bản tương đương với dòng trong CSDL quan hệ thì bộ sưu tập tương đương với bảng.

Bộ sưu tập là một Schema-Free, nghĩa là các văn bản có hình dạng khác nhau có thể cùng được lưu trữ trong 1 bộ sưu tập.

Ví dụ các văn bản sau có thể cùng được lưu trong một bộ sưu tập:

{"greeting" : "Hello, world!"}

{"foo" : 5}

Bộ sưu tập được xác định bởi tên của nó là một chuỗi UTF-8

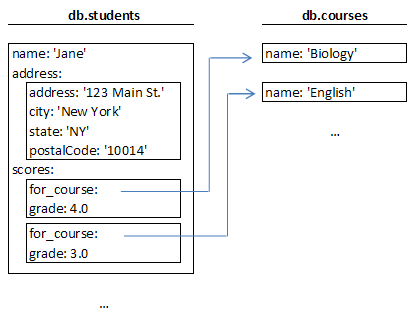
Các đặc trưng của MongoDB:

* Lưu trữ hướng văn bản: Văn bản theo phong cách JSON với những lược đồ động đơn giản.
* Hỗ trợ chỉ mục đầy đủ: chỉ mục trên bất kỳ các thuộc tính
* Tính sao lặp và tính sẵn sàng cao: mở rộng
* Auto-sharding: mở rộng theo chiều ngang mà không ảnh hưởng đến chức năng
* Truy vấn: đa dạng, truy vấn dựa trên văn bản
* Cập nhật nhanh:
* Map/Reduce
* GridFS: lưu trữ file với bất kỳ kích cỡ nào mà không làm phức tạp ngăn xếp
* Hỗ trợ thương mại: hỗ trợ doanh nghiệp, đào tào, tư vấn

### Thiết kế lược đồ

* Với MongoDB, chúng ta ít phải “chuẩn hóa” hơn so với khi làm việc với lược đồ quan hệ vì trong MongoDB không có khái niệm liên kết (join). Nói chung, với mỗi đối tượng (object) mức cao nhất, ta sẽ có một bộ sưu tập (collection) dữ liệu.
* Một bộ sưu tập không phải cho tất cả các lớp (class), thay vào đó, các đối tượng sẽ được nhúng vào đó.

Hình 1 minh họa có 2 bộ sưu tập: students và courses. Các văn bản student được nhúng văn bản address và văn bản score. Trong đó, văn bản Score được tham chiếu đến Courses.



Hình 1: Minh họa bộ sưu tập

So sánh với lược đồ quan hệ: ta cần lưu Score vào bảng riêng và dùng khóa ngoài liên kết với Student.

### Nhúng hay Tham chiếu

* Một câu hỏi quan trọng trong thiết kế lược đồ Mongo là: “Đối tượng này có cần một bộ sưu tập của riêng nó không hay nên nhúng vào trong các đối tượng trong các bộ sưu tập khác?” Trong cơ sở dữ liệu quan hệ, mỗi tiểu mục có thể trở thành một bảng riêng biệt. Trong Mongo, nó không được khuyến cáo, việc nhúng các đối tượng hiệu quả hơn nhiều. Chúng ta cũng có thể đặt ra câu hỏi “Tại sao tôi không muốn nhúng đối tượng này?”
* Tại sao tham chiếu lại chậm. Ta xem ví dụ sau. Chúng ta có một đối tượng Student và cần thực hiện:
* print( students.address.city );
* Phép toán này sẽ luôn được thực hiện nhanh nếu Address là một đối tượng nhúng, và được lưu ở RAM nếu Student được lưu ở RAM.
* Tuy nhiên, với truy vấn:
* print( students.scores[0].for\_course.name );

Nếu đó là lần đầu truy cập đến khóa này thì trình tiện ích phải thực hiện truy vấn:

* students.scores[0].for\_course = db.courses.findOne({\_id:\_course\_id\_to\_find\_});

***Các luật cơ bản***

* Các đối tượng “lớp thứ nhất” là các đối tượng ở mức cao nhất, có bộ sưu tập của riêng mình.
* Các đối tượng miêu tả chi tiết các mục thường được nhúng
* Các đối tượng mà theo mô hình đối tượng có chứa quan hệ nói chung nên được nhúng
* Quan hệ nhiều – nhiều thường được tham chiếu.
* Các bộ sưu tập chỉ với một vài đối tượng có thể tồn tại một cách an toàn giống như bộ sưu tập riêng lẻ, được lưu trữ nhanh chóng trong bộ nhớ máy chủ ứng dụng.
* Các đối tượng nhúng khó khăn để tham chiếu hơn là các đối tượng mức cao.
* Sẽ khó khăn hơn để có một cái nhìn mức hệ thống đối với các đối tượng nhúng. Ví dụ: Sẽ dễ thực hiện truy vấn tìm 100 sinh viên có điểm cao nhất hơn nếu Score không bị nhúng.
* Nếu dữ liệu được nhúng lớn, có thể đạt đến giới hạn kích thước của một đối tượng.
* Nếu hiệu suất là quan trọng, hãy nhúng.

***Một số ví dụ***

* + C
* Nghe

Đọc ngữ âm

- Customer/Order/ Order Line-Item: Customers, Orders nên có một bộ sưu tập riêng. Line-Items nên là một mảng các mục cần mua và được nhúng trong đối tượng Order.

* Hệ thống Blog: Posts cần có bộ sưu tập riêng. Post Author có thể có bộ sưu tập riêng hoặc nếu đơn giản chỉ là địa chỉ mail của tác giả thì cho thành một trường trong Posts. Comments được nhúng trong Posts.

### Lựa chọn chỉ mục

* Một khía cạnh thứ hai khi thiết kế lược đồ là việc lựa chọn chỉ mục. Việc đánh chỉ mục làm cho việc thực hiện truy vấn nhanh hơn. Một truy vấn bình thường cần vài phút, có thể được thực hiện ngay lập tức với việc sử dụng chỉ mục.
* Trong MongoDB:
* Trường \_id được đánh chỉ mục tự động
* Những trường mà theo đó các khóa được tìm kiếm nên được đánh chỉ mục
* Những trường sắp xếp nói chung nên được đánh chỉ mục
* Lưu ý rằng việc thêm vào chỉ mục chỉ làm chậm quá trình ghi vào bộ sưu tập mà không làm chậm quá trình đọc. Vì vậy, sử dụng nhiều chỉ mục với những bộ sưu tập mà tỉ lệ read:write cao. Với những bộ sưu tập mà ghi nhiều hơn đọc, sử dụng chỉ mục là rất tốn kém.

### Chỉ mục

* Chỉ mục làm tăng hiệu suất truy vấn lên rất nhiều. Điều quan trọng là nghĩ xem xét tất cả các loại truy vấn cần trong ứng dụng để xác định những chỉ mục liên quan. Khi đã xác định xong, việc tạo ra các chỉ mục trong MongoDB là khá dễ dàng.

### Các khái niệm cơ bản

* Chỉ mục là một cấu trúc dữ liệu, thu thập thông tin về giá trị của các trường trong các văn bản của một bộ sưu tập. Cấu trúc dữ liệu này được sử dụng trong tối ưu truy vấn Mongo để sắp xếp nhanh các văn bản trong một bộ sưu tập.
* Chúng ta có thể khởi tạo chỉ mục bằng cách gọi hàm ensureIndex() và cung cấp một văn bản với một hoặc nhiều khóa để đánh chỉ mục. Ví dụ đánh chỉ mục cho trường name trong students

db.students.ensureIndex({name:1});

* Hàm ensureIndex() chỉ khởi tạo chỉ mục nếu nó chưa tồn tại. Để kiểm tra việc tồn tại chỉ mục trên bộ sưu tập students, ta có thể chạy hàm db.students.getIndexes().
* Khi một bộ sưu tập được đánh chỉ mục trên một khóa nào đó, truy cập ngẫu nhiên trên biểu thức truy vấn có chứa khóa đó sẽ được thực hiện rất nhanh. Nếu không được đánh chỉ mục, MongoDB phải soát tất cả các văn bản để kiểm tra giá trị của khóa đó trong truy vấn.

***Chỉ mục mặc định***

Một chỉ mục luôn luôn được tạo ra là \_id. Chỉ mục này là đặc biệt và không thể bị xóa. Chỉ mục \_id là duy nhất cho các khóa của nó.

***Các khóa nhúng***

Với MongoDB chúng ta thậm chí có thể đánh chỉ mục trên các khóa bên trong văn bản nhúng. Ví dụ

db.students.ensureIndex({"address.city": 1})

***Văn bản như là khóa***

Các trường được đánh chỉ mục có thể là bất kỳ loại nào, bao gồm cả văn bản.

***Mảng***

Khi giá trị của trường được đánh chỉ mục của văn bản là một mảng. MongoDB đánh chỉ mục mỗi phần tử của mảng đó.

### Chỉ mục hỗn hợp các khóa

Ngoài chỉ mục khóa đơn, MongoDB còn hỗ trợ đánh chỉ mục hỗn hợp nhiều khóa. Giống như đánh chỉ mục cơ bản, chúng ta sử dụng hàm ensureIndex() để khởi tạo chỉ mục.

db.things.ensureIndex({j:1, name:-1});

Khi khởi tạo một chỉ mục, số đi cùng với khóa là hướng của chỉ mục, 1: tăng dần, -1: giảm dần. Hướng không ảnh hưởng đến việc truy cập ngẫu nhiên nhưng quan trọng nếu bạn đang làm các truy vấn sắp xếp hoặc phân loại trên chỉ mục hỗn hợp.

Nếu chúng ta có một chỉ mục hỗn hợp trên nhiều trường, chúng ta có thể sử dụng nó để truy vấn trên các tập hợp con đầu của các trường đó. Ví dụ ta có chỉ mục trên (a, b, c), ta có thể sử dụng nó để truy vấn trên (a), (a, b), (a, b, c).

### Chỉ mục thưa thớt

* Chỉ mục thưa thớt là chỉ mục mà chỉ bao gồm các văn bản có trường được đánh chỉ mục. Bất kỳ văn bản nào bị thiếu trường đánh chỉ mục thưa thớt đều không được lưu vào trong chỉ mục. Các chỉ mục là thưa thớt vì bị thiếu những văn bản không có giá trị của trường được đánh chỉ mục.
* Chỉ mục thưa thớt, theo định nghĩa, là không đầy đủ và hoạt động khác với chỉ mục đầy đủ. Khi sử dụng chỉ mục thưa thớt để sắp xếp, một vài văn bản trong bộ sưu tập sẽ không được trả về. Đó là do chỉ những văn bản được đánh chỉ mục mới được trả về.

db.people.ensureIndex({title : 1}, {sparse : true})

db.people.save({name:"Jim"})

db.people.save({name:"Sarah", title:"Princess"})

db.people.find({title:{$ne:null}}).sort({title:1})

// returns only Sarah

### Chỉ mục duy nhất

* MongoDB hỗ trợ đánh chỉ mục duy nhất, đảm bảo rằng không có văn bảo nào được chèn mà giá trị của khóa được đánh chỉ mục lại trùng với văn bản đã tồn tại. Để tạo ra một chỉ mục đảm bảo ràng không có 2 văn bản có cùng giá trị cho 2 trường firstname và lastname ta làm như sau:

db.things.ensureIndex({firstname: 1, lastname: 1}, {unique: true});

***Khóa bị thiếu***

Khi một văn bản được lưu vào bộ sưu tập với việc đánh chỉ mục duy nhất, bất kỳ khóa được đánh chỉ mục nào bị thiếu sẽ được chèn vào với giá trị null. Vì vậy, không được phép chèn nhiều văn bản bị thiếu cùng một khóa được đánh chỉ mục

db.things.ensureIndex({firstname: 1}, {unique: true});

db.things.save({lastname: "Smith"});

// Next operation will fail because of the unique index on firstname.

db.things.save({lastname: "Jones"});

### Giá trị lặp lại

Chỉ mục duy nhất không cho phép một khóa có giá trị nhân bản. Nếu bạn muốn đánh chỉ mục bằng mọi giá, hãy giữ văn bản đầu tiên trong CSDL và xóa tất cả các văn bản có giá trị bị nhân bản, thêm tùy chọn dropDups

db.things.ensureIndex({firstname : 1}, {unique : true, dropDups : true})

### Xóa chỉ mục

Xóa tất cả các chỉ mục trên bộ sưu tập:

db.collection.dropIndexes();

Xóa chỉ mục đơn:

db.collection.dropIndex({x: 1, y: -1})

Chạy trực tiếp như một lệnh mà không cần hỗ trợ:

// note: command was "deleteIndexes", not "dropIndexes", before MongoDB v1.3.2

// remove index with key pattern {y:1} from collection foo

db.runCommand({dropIndexes:'foo', index : {y:1}})

// remove all indexes:

db.runCommand({dropIndexes:'foo', index : '\*'})

### ReIndex

Lệnh reIndex sẽ xây dựng lại tất cả các chỉ mục cho bộ sưu tập.

db.myCollection.reIndex ()

/ / giống như:

db.runCommand ({reIndex: 'myCollection'})

Thông thường, điều này là không cần thiết. Chúng ta có thể làm điều này nếu kích thước bộ sưu tập đã thay đổi đáng kể hoặc không gian đĩa được sử dụng bởi các chỉ mục có vẻ lớn bất thường.

reIndex sẽ bị chậm với các bộ sưu tập lớn.

Các lệnh sửa CSDL tái tạo lại tất cả các chỉ mục trong CSDL.

**Lưu ý**

* Chỉ mục trong MongoDB phân biệt chữ hoa chữ thường
* Chỉ số thông tin được lưu giữ trong bộ sưu tập system.indexes, chạy db.system.indexes.find () để xem ví dụ.

***Hiệu suất chỉ mục***

Việc đánh chỉ mục thực hiện rất nhanh. Cập nhật được thực hiện nhanh hơn vì MongoDB có thể tìm thấy các văn bản cần cập nhật rất nhanh chóng. Tuy nhiên, với việc sử dụng chỉ mục, khi ghi dữ liệu vào bộ sưu tập, các khóa sau đó phải được thêm vào trường chỉ mục. Như vậy, chỉ mục chỉ tốt cho bộ sưu tập có số lượng đọc nhiều hơn rất nhiều số lượng ghi. Đối với các bộ sưu tập chú trọng ghi, việc sử dụng chỉ mục, trong một số trường hợp, có thể phản tác dụng. Hầu hết các bộ sưu tập đều chú trọng đọc, vì vậy mà chỉ mục là tốt trong hầu hết các tình huống.

***Sử dụng Sort() mà không cần chỉ mục***

Chúng ta có thể sử dụng sort() để trả về dữ liệu được sắp thứ tự mà không cần phải dùng chỉ mục nếu dữ liệu cần trả về là nhỏ (< 4 MB). Đối với những trường hợp này tốt nhất là sử dụng hàm limit() và sort() cùng nhau.

### Sao chép

Có lẽ công việc quan trọng nhất của bất kỳ quản trị viên MongoDB là đảm bảo sao cho sao chép được thiết lập và hoạt động đúng. Sao chép có thể được sử dụng hoàn toàn để dự phòng và toàn vẹn dữ liệu hoặc có thể được sử dụng cho mục đích cao hơn như mở rộng đọc, sao lưu nóng,…

MongoDB hỗ trợ sao chép dữ liệu không đồng bộ giữa các máy chủ. Tại một thời điểm, chỉ có 1 máy chủ hoạt động để ghi (primary hay master).

Có hai hình thức sao chép.

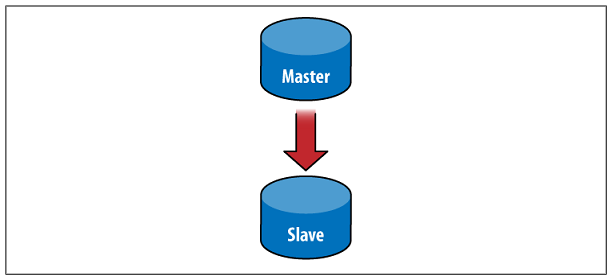
\* Master-Slave Replication

\* Replica Sets.

**Master-Slave Replication**

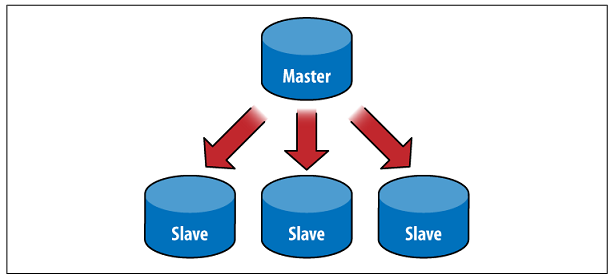
Sao chép Master-slave là mô hình sao chép phổ biến nhất được hỗ trợ bởi MongoDB. Mô hình này rất linh hoạt và có thể được sử dụng để sao lưu, dự phòng, mở rộng đọc, …

Hình 2.2 minh họa mô hình Master – Slave bao gồm 2 nút, một nút làm Master, nút còn lại làm Slave



Hình 2. Mô hình Master – Slave hai nút

Hình 2 minh họa mô hình Master – Slave bao gồm 4 nút, một nút làm Master, 3 nút còn lại làm Slave



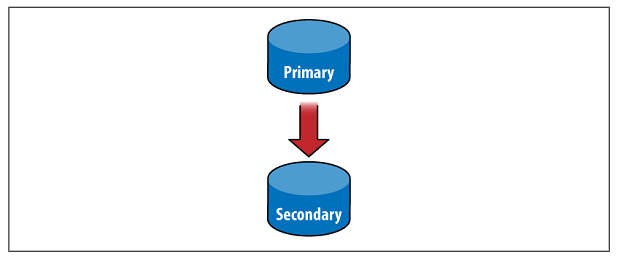
Hình 3. Mô hình Master – Slave bốn nút

Để thiết lập cần khởi động nút master và một hoặc nhiều nút slave, các nút này đều biết địa chỉ của nút master. Để khởi động master, chạy mongod --master. Để khởi động slave, chạy mongod --slave --source master\_address, trong đó master\_address là địa chỉ của nút master vừa được khởi động

**Replica Sets**

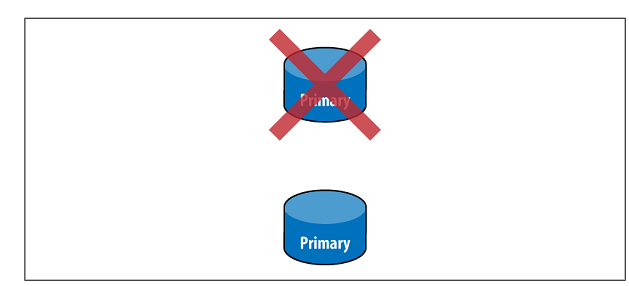
Replica Sets là một cụm master-slave tự động chịu lỗi. Replica Sets không có một master cố định: một master được bầu chọn và có thể thay đổi đến nút khác nếu master bị sập [1].

Hình 2.4 mô phỏng mô hình Replica Sets gồm 2 nút.



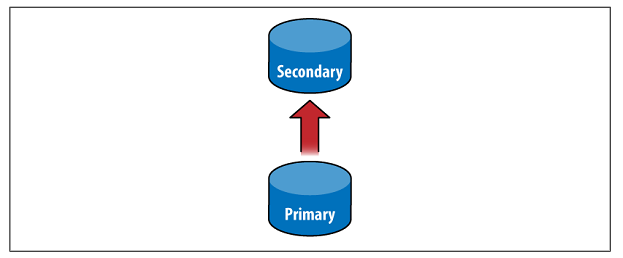
Hình 4. Mô hình Replica Sets hai nút

Khi server chính chết, server cấp 2 chở thành server chính (hình 4).



Hình 5. Mô hình Replica Sets hai nút

Nếu server chính ban đầu hoạt động trở lại, nó trở thành server cấp 2 (hình 5).



Hình 6. Server chính trở thành server cấp 2

### Truy vấn

Một trong những tính năng tốt nhất của MongoDB là hỗ trợ truy vấn động. Hệ thống hỗ trợ truy vấn động không yêu cầu bất cứ chỉ mục nào để tìm dữ liệu. Người dùng có thể tìm dữ liệu với việc sử dụng bất kỳ tiêu chuẩn nào. Với CSDL quan hệ, truy vấn động là chuẩn hóa.

**Đối tượng biểu thức truy vấn**

MongoDB hỗ trợ một số các đối tượng truy vấn để lấy dữ liệu. Ví dụ, giả sử chúng ta muốn sử dụng trình MongoDB để trả về mọi văn bản trong bộ sưu tập *users.* Truy vấn sẽ được viết như sau:

db.users.find({})

Trong trường hợp này, lựa chọn (điều kiện) của chúng ta là trống, nó phù hợp với mọi văn bản trong bộ sưu tập. Chúng ta xem thêm một số ví dụ:

db.users.find({'last\_name': 'Smith'})

Ở đây, lựa chọn của chúng ta là tất cả các văn bản mà thuộc tính *last\_name* là *Smith.*

**Các tùy chọn truy vấn**

***Lựa chọn các trường***

Ngoài các biểu thức truy vấn, truy vấn MongoDB còn có thể thêm vào các tham số. Ví dụ, chúng ta muốn các số CMT của tất cả người có họ là *Smith*, ta có thực hiện truy vấn:

// lấy trường ssn của các văn bản có last\_name == 'Smith':

db.users.find({last\_name: 'Smith'}, {'ssn': 1});

// lấy tất cả các trường ngoại trừ trường thumbnail đối với tất cả các văn bản.

db.users.find({}, {thumbnail:0});

Chú ý rằng, trường \_id luôn luôn được trả về ngay cả khi không yêu cầu

***Sắp xếp***

Truy vấn MongoDB có thể trả về kết quả được sắp xếp. Để trả về tất cả các văn bản mà trường last\_name được sắp xếp theo thứ tự tăng dần, ta viết truy vấn sau:

db.users.find({}).sort({last\_name: 1});

***Bỏ qua và giới hạn***

MongoDB luôn luôn hỗ trợ bỏ qua và giới hạn để phân trang một cách dễ dàng. Ví dụ ta muốn bỏ qua 20 họ đầu tiên và giới hạn kết quả đến 10, ta viết truy vấn sau:

db.users.find().skip(20).limit(10);

db.users.find({}, {}, 10, 20); // giống như lệnh trên nhưng không rõ ràng.

***slaveOk***

Khi thực hiện truy vấn ở một hoặc nhiều bản sao, trình tiện ích gửi yêu cầu đến master, để thực hiện truy vấn đối với slave, truy vấn có thể chạy với tùy chọn slaveOk.

db.getMongo().setSlaveOk(); // cho phép truy vân slave

db.users.find(...)

***Con trỏ***

Các truy vấn CSDL được thực hiện với phương thức find(), với kỹ thuật này một con trỏ được trả về. Con trỏ sau đó được sử dụng lặp đi lặp lại để lấy tất cả các văn bản mà truy vấn trả về. Chúng ta có thể xem ví dụ sau:

> var cur = db.example.find();

> cur.forEach( function(x) { print(tojson(x))});

{"n" : 1 , "\_id" : "497ce96f395f2f052a494fd4"}

{"n" : 2 , "\_id" : "497ce971395f2f052a494fd5"}

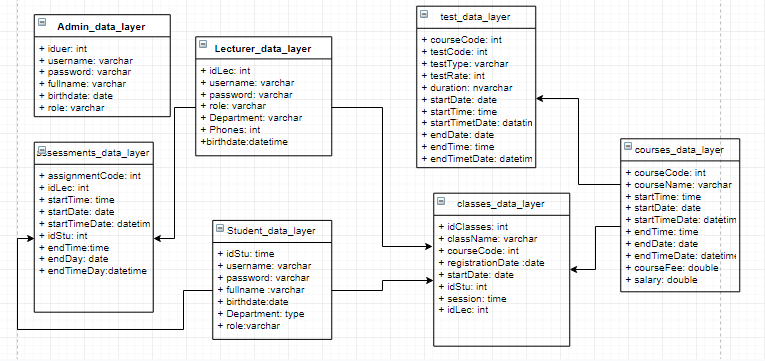
{"n" : 3 , "\_id" : "497ce973395f2f052a494fd6"}

>

Như vậy, MongoDB là một CSDL hướng văn bản, lưu trữ dữ liệu dưới cặp khóa/giá trị. Các đối tượng trong MongoDB thường được nhúng trong các đối tượng mức cao hơn để tăng tốc độ xử lý truy vấn. Để tăng tốc độ truy vấn, người ta cũng thường đánh chỉ mục cho những bộ sưu tập có tỉ lệ đọc:ghi cao. MongoDB thực hiện truy vấn để lấy dữ liệu thông qua các biểu thức truy vấn cùng các tham số cần thiết. Với những dự án mà tỉ lệ lượng dữ liệu ghi vào CSDL lớn hơn lượng đọc thì lựa chọn MongoDB sẽ mang lại hiệu quả cao.

Để làm quen với truy vấn trong MongoDB, người đọc có thể tham khảo một số lệnh tương đương với truy vấn SQL chuẩn mà chúng tôi trình bày trong Phụ lục của báo cáo này.

1. **Thiết kế cơ sở dữ liệu:**
2. **Sơ đồ cơ sở dữ liệu:**



Hình 7. Sơ đồ cơ dỡ liệu của test case studnet

1. **Chi tiết cơ sở dữ liệu:**

**1.Bảng Admin**

**Tên bảng : Admin\_data\_layer**

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| iduer | int |  | **V** |  |  |
| username | varchar |  |  |  |  |
| password | varchar |  |  |  |  |
| fullname | varchar |  |  |  |  |
| birthdate | date |  |  |  |  |
| role | varchar |  |  |  |  |

Bảng 1. Admin

**2.Bảng bài kiểm tra**

**Tên bảng :** test\_data\_layer

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| + courseCode | int |  | **V** |  |  |
| + testCode: | int |  |  |  |  |
| + testType: | varchar |  |  |  |  |
| + testRate: | int |  |  |  |  |
| + duration: | nvarchar |  |  |  |  |
| + startDate: | date |  |  |  |  |
| + startTime: | time |  |  |  |  |
| + startTimetDate: | datatime |  |  |  |  |
| + endDate: | date |  |  |  |  |
| + endTime: | time |  |  |  |  |

Bảng 2. Test\_data\_layer

**3.Bảng dữ liệu các khóa học**

**Tên bảng :** courses\_data\_layer

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| courseCode | int |  | **V** |  |  |
| courseName: | varchar |  |  |  |  |
| startTime: | time |  |  |  |  |
| startDate: | date |  |  |  |  |
| startTimeDate: | datetime |  |  |  |  |
| + endTime: time | date |  |  |  |  |
| + endDate: date | time |  |  |  |  |
| + endTimeDate: datetime | datatime |  |  |  |  |
| + courseFee: | double |  |  |  |  |

Bảng 3. Courses\_data\_layer

**4.Bảng giáo viên**

**Tên bảng :** Lecturer\_data\_layer

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| idLec: int | int |  | **V** |  |  |
| + username: | varchar |  |  |  |  |
| + password: | varchar |  |  |  |  |
| + role: | varchar |  |  |  |  |
| + Department: | varchar |  |  |  |  |
| + Phones: | int |  |  |  |  |
| +birthdate: | date |  |  |  |  |

Bảng 4. Lecturer\_data\_layer

**5.Bảng dữ liệu đánh giá**

**Tên bảng :** assessments\_data\_layer

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| assignmentCode: | int |  | **V** |  |  |
| idLec: | int |  |  |  |  |
| startTime: | time |  |  |  |  |
| startDate: | date |  |  |  |  |
| startTimeDate: | datetime |  |  |  |  |
| idStu: int | int |  |  |  |  |
| endTime: | time |  |  |  |  |
| endDay: | date |  |  |  |  |
| endTimeDay: | datetime |  |  |  |  |

Bảng 5. Assessments\_data\_layer

**6.Bảng sinh viên**

**Tên bảng :** Student\_data\_layer

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| idStu | int |  | **V** |  |  |
| username | varchar |  |  |  |  |
| password | varchar |  |  |  |  |
| fullname | varchar |  |  |  |  |
| birthdate | date |  |  |  |  |
| Department | varchar |  |  |  |  |
| role | varchar |  |  |  |  |

Bảng 6. Student\_data\_layer

**7.Bảng** lớp học

**Tên bảng :** classes\_data\_layer

**Mô tả: Quản lý**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Null** | **PK** | **FK** | **Mô tả** |
| idClasses | int |  | **V** |  |  |
| className: | varchar |  |  |  |  |
| courseCode: | int |  |  |  |  |
| registrationDate | date |  |  |  |  |
| startDate: | date |  |  |  |  |
| idStu: | int |  |  |  |  |
| session: | time |  |  |  |  |
| idLec: | int |  |  |  |  |

Bảng 7. Classes\_data\_layer

# **HIỆN THỰC HỆ THỐNG**

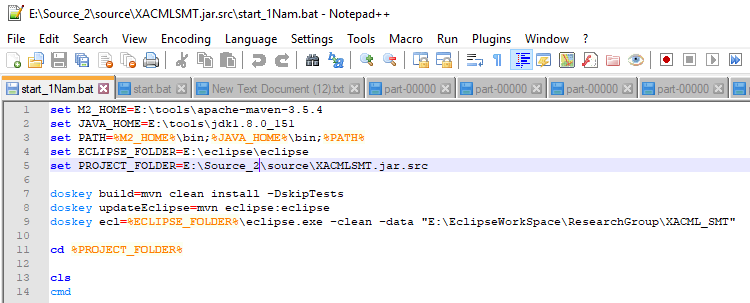
1. **Cài đặt hệ thống:**

1.Tạo folder Tools và copy Maven vào.

**2. Tạo file mở workspace hệ thống:**

Tạo folder research-group-workspace: dùng lưu trữ thông tin workspace của Eclipse.

Update path in start.bat or (start\_1Nam.bat)

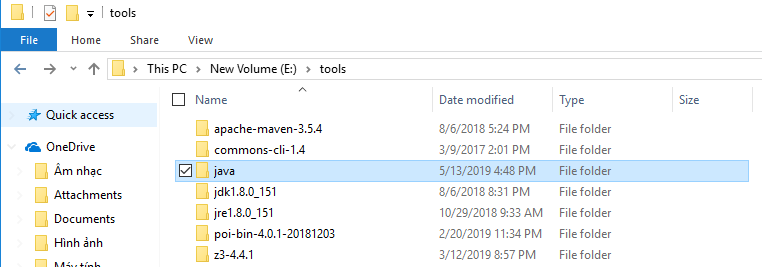


Hình 41: Update path in start

**3.Cài đặt thư viện Z3:**

Z3 4.4.1: <https://github.com/Z3Prover/z3>

Installation: copy Z3 bỏ vào folder Tools tương tự như Maven.

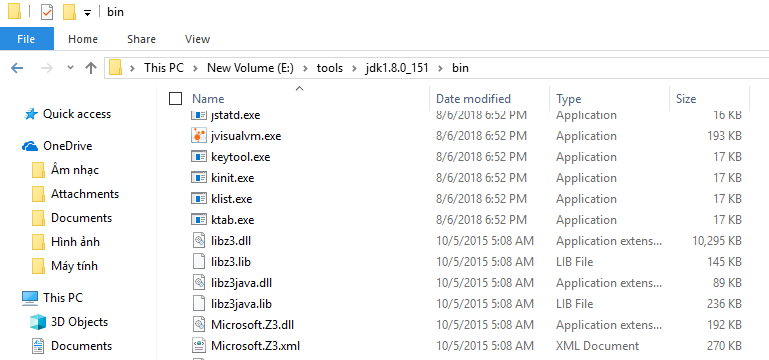


Hình 42 Thư mục skype

Maven Library:

- Giải thích: dùng để cài đặt file jar vào Maven Repository rename the jar file in "bin" folder to z3-4.4.1.jar go to the folder of Z3\bin mvn install:install-file -Dfile=z3-4.4.1.jar -DgroupId=com.microsoft -DartifactId=z3 -Dversion=4.4.1 -Dpackaging=jar.

Dll files: Copy DLL File into folder JAVA\_PATH\jre\bin.

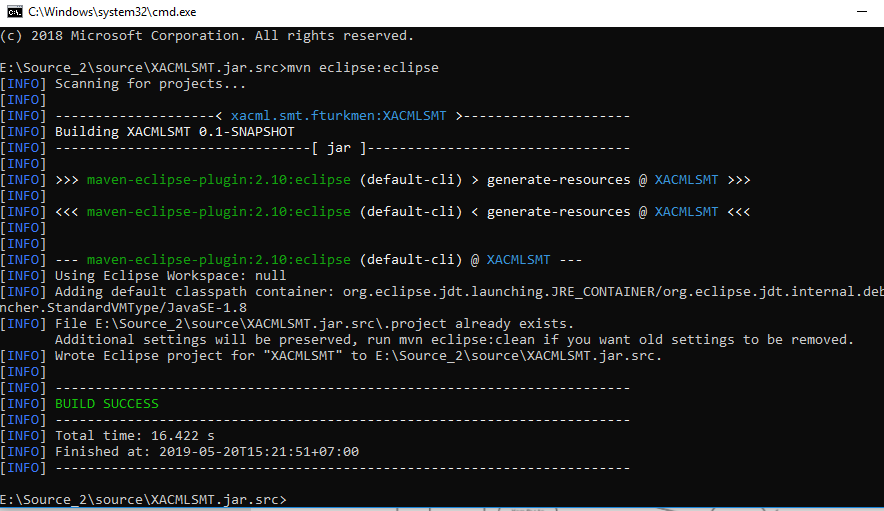


Hình 43: Thêm thư việc z3 vào JDK

**4 Cài đặt source code:**

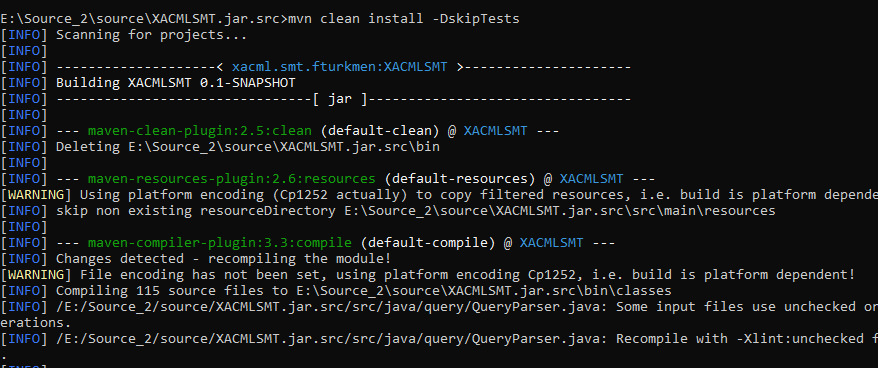
[Source code:](https://bitbucket.org/khiem111189/research_group/src/InternXACML/)

Build Eclipse: Vào PROJECT\_FOLDER path và chạy mvn eclipse:eclipse



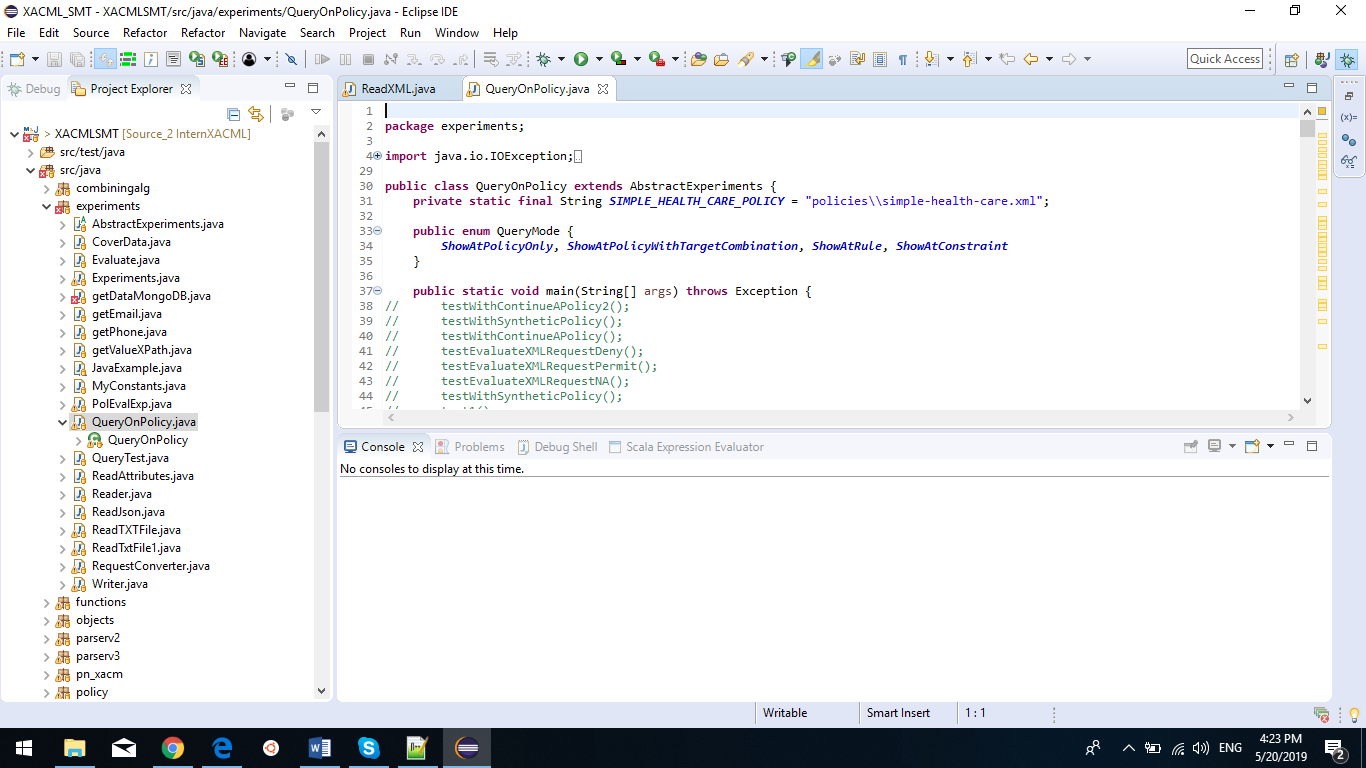
Hình 44: chạy mvn eclipse:eclipse

Build Code: Vào PROJECT\_FOLDER path và chạy mvn clean install –DskipTests



Hình 45: chạy mvn clean install –DskipTests

Import project to eclipse



Hình 46: import source vào eclipse

**5.Chạy chương trình**

Run: QueryOnPolicy: Test cases

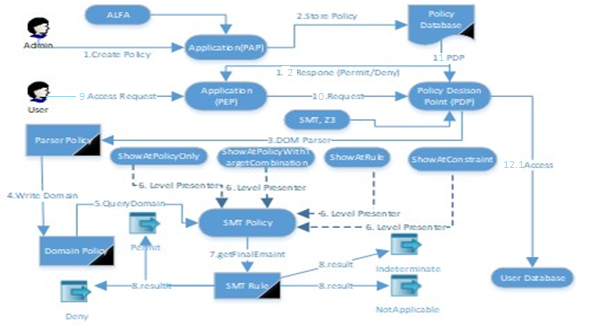
1. **Evalue policy**

Hình 2 mô tả mô hình đề xuất. Kiến trúc chứa các thành phần chính sau đây.

- Điểm thực thi chính sách (PEP): là thành phần đóng vai trò là người gác cổng cho tài nguyên số. Khi người dùng cố gắng truy cập tài nguyên, PEP sẽ mô tả các thuộc tính người dùng cho PDP, yêu cầu quyết định bảo mật và thực hiện quyết định đó.

- Điểm quyết định chính sách (PDP): là một thành phần của hệ thống kiểm soát truy cập dựa trên Chính sách quyết định có cho phép người dùng yêu cầu hay không dựa trên thông tin có sẵn (thuộc tính) và chính sách bảo mật.

- Điểm quản trị chính sách (PAP): là giao diện hoặc công cụ cho phép tạo và sửa đổi chính sách và bộ chính sách. Nó đi kèm với một bộ tính năng phong phú để giúp quản trị viên tạo Chính sách tuân thủ XACML 3.0 bằng ngôn ngữ ALFA.



Hình 47:Mô hình kiểm soát truy cập dựa trên thuộc tính hỗ trợ chính sách XACML đa cấp

Mô hình đề xuất của chúng tôi hoạt động như sau. Quản trị viên tạo Chính sách [Bước 1] bằng PAP. Các chính sách này sau khi được tạo sẽ được thêm vào cơ sở dữ liệu [Bước 2] có tên Cơ sở dữ liệu chính sách. Người dùng cuối tạo một yêu cầu truy cập sau đó gửi nó đến hệ thống. PEP nhận các yêu cầu này [Bước 4] Định dạng PEP yêu cầu người dùng và sau đó gửi chúng đến PDP [Bước 5]. PDP sẽ phân tích và tìm ra các chính sách phù hợp. Dựa trên chính sách đã chọn, liệu các Yêu cầu này có được phép truy cập [Bước 3] hay không. Trong mô hình đề xuất PDP tích hợp SMT-Z3 của chúng tôi được sử dụng để đánh giá các yêu cầu truy cập vào hệ thống thông qua Bộ chính sách được lấy từ Cơ sở dữ liệu chính sách. Khi Yêu cầu của người dùng yêu cầu quyền truy cập vào DOM Parsers [Bước 6], họ sẽ được chuyển đổi thành các cấp tương ứng: ShowAtPolicyOnly, ShowAtPolicyWithTargetCombination, ShowAtRule, ShowAtConstraint. Từ Chính sách của SMT sẽ được thực hiện thông qua Quy tắc SMT [Bước 10] để đánh giá. Sau khi thực hiện SMT-Z3, đánh giá sẽ trả về 1 trong 4 kết quả [Bước 11] như: Cho phép, Từ chối, Không xác định, Không áp dụng. Kết quả trả về Permit, tài khoản sẽ được truy cập dữ liệu cơ sở dữ liệu [Bước 12] Hoặc kết quả trả về Từ chối / Không xác định / Không áp dụng, tài khoản sẽ không thể truy cập dữ liệu. Đồng thời trả về thông báo cho người dùng cuối thông qua PEP. PEP sẽ hiển thị màn hình kết quả của phản hồi đó cho người dùng cuối.

1. **Giải thuật của SMT-z3**

Thuật toán 1 lấy phần tử cuối cùng bằng cách cấu hình. Đầu tiên, nó kiểm tra đúng khu vực được chọn (dòng 1), nếu nó đúng, kiểm tra xem chính sách có nằm trong (dòng 2) không. Quy tắc SMT bằng để thực hiện đánh giá chính sách phần tử. mặt khác (dòng 3) là một bộ chính sách để duyệt phần tử bộ chính sách (dòng 6). Chính sách là một enum có 4 giá trị thay đổi bao gồm bộ chính sách, chính sách, quy tắc. PolicyEuityType là một loại giá trị có chứa bộ chính sách, một chính sách nằm trong Quy tắc SMT.

Giá trị trả về của quy tắc SMT của id, ctx, c p temp. Fi rst, c d temp. Fi rst, c in p temp. Fi rst, c in d temp. Fi rst, c na. Rst.

**Algorithm 1** getFinalElementByConﬁguration() function

**Input**: 4 level presenter ShowAtPolicyOnly, ShowAtPolicyWithTargetCombination, ShowAtRule, ShowAtConstraint

**Output**: SMT của id, ctx, c p temp. Fi rst, c d temp. Fi rst, c in p temp. Fi rst, c in d temp. Fi rst, c na. Rst

1: if compare 1 in 4 level presenter then

2: if PolicyElementType equal PolicyElement.policy then

3: SMT Rule equal each element policy;

4: end if

5: if PolicyElementType equal PolicyElement.policy set then

6: SMT Rule equal each element policy set;

7: end if

8: endif

Thuật toán 2 là dòng 3 và dòng 6 giải thuật đệ quy lại Thuật toán 1,

Duyệt phần tử đầu tiên của chính sách hoặc bộ chính sách (dòng 1). Kiểm tra xem chiều dài phần tử có bằng 1 (dòng 2) không. Sau đó, Quy tắc SMT có chính sách loại, bộ chính sách (dòng 3) trả về Quy tắc SMT (dòng 9) nếu không 1 duyệt từng phần tử ListList (dòng 5) trả về Quy tắc SMT có loại của từng thành phần duyệt như chính sách, chính sách, quy tắc (dòng 6) trả về Quy tắc SMT (dòng 9)

**Algorithm 2**: AlgAllPairs() function

**Input**: Element List in policy set or policy

**Output**: SMT của id, ctx, c p temp. Fi rst, c d temp. Fi rst, c in p temp. Fi rst, c in d temp. Fi rst, c na. Rst

1: SMT Rule compare elementList ﬁrst in policy set or policy

2: if elementList.length equal 1 then

3: SMT Rule equal (get type policy set, policy);

4: else

5: for elementList.lengh; i++ do

6: SMT Rule equal (get type policy set or policy or rule);

7: end for

8: end if

9: return SMT Rule;

1. **Giải thuật về NegVaule:**

Tạo list<string[]> list nhận dự liệu từ file.txt và đọc 1 dòng rồi split(“=>”) thành mảng có key[0]”=>”value[1]. Tạo HashMap<String, Set> mapNewDomain. Khởi tạo PrinttWrite từ thư viện java write value vào filedomain.txt. duyệt list lấy key và vaule vào hashmap. Do mapNewDomain có value là kiểu collections Set nên dữ liệu nhập vào phải là conllections Set. Duyệt mapNewDomain wirte key và value vào filedomain.txt

**Algorithm 3** NegVaule

Input: file.txt key và value của policy parser

Output: HashMapNegValue

1: list *đọc file.txt* and split("=>")

2: PrintWriter(“filedoamin.txt”);

3: for list.size() - 1; i++ do

4: mapNewDomain.put(get(list element 0),list.add(Set file.txt));

5: value thuộc conllection set nên loại bỏ các phần tử cùng

6: endfor

7: For mapNewDomain do {

8: println mapNewDomain key and "=>" and mapNewDomain .list value;

9:endfor

1. **Giải thuật về query** [**Analysis**](https://link.springer.com/chapter/10.1007/978-3-662-46666-7_7)

(line 1) duyệt hashmap request xml .(line 2) duyệt hashmap negValue. (line 3) if key of hashmap request xml key equals hash map negvaule key then duyệt negVaule value (line 4).(line 5) truyền biến vào replaceof(giải thuật 2). (line 10) trả về reqlaceof

space() {

**Algorithm 4:** query [Analysis](https://link.springer.com/chapter/10.1007/978-3-662-46666-7_7)

**Input**: HashMapDomain là các key và value của policy parser, HashMapRequest là các key và value của requset, HashMapNegVaule là các key và value của policy và request

**Output**: String Query analysis

**1: for** mapReadAttributeRequetsXML do

2: **for** mapAttributeNegValue do

**3: if** mapAttributeRequetsXMLkey equal entryAttributeNegValueKey then

**4: for** AttributeNegValueValue do

5: *replaceOf*(strQuery, mapAttributeRequetsXMLkey, entryAttributeRequetsXMLValue, entryAttributeNegValueValue)

6: endfor

7: endif

8: endfor

9: endfor

10: **return** r*eplaceOf:*

Trong giai thuat 2 truyen bien tu gia thuat 1 vao (line 1). (line 2) if neglues length bigger 0 then gan bien vao chuoi (line 4) .(line 6) nguoc lai gan bien vao chuoi (line 7). (line 8) tra ve ket qua

**Algorithm 5 :** Reqlaceof

**Input**: String currentExpr, String var, String posValue, String... negValues Chuyền vào Reqlaceof

**Output**: "\\REPLC " + "(" + internalExpr + ")", currentExpr

1: if negValues bigger 0 the

2: gan internalExpc equal \\ WEDGE var= posValue [\\NEG](file:///\\NEG) \\ WEDGE var = negValue

3:else then

4: internalExpr equal *var =* posValue

5: end if

6: return "\\REPLC " + "(" + internalExpr + ")", currentExpr;

1. **Giải thuật gen policy:**

Đọc file excel (line 1) lấy key và value rồi truyền giá trị vào hashMap. (line 2) gán trị cần so sánh vào 1 biến string . (Line 3) đọc file xml mẫu sau đó write vào file xml other. (Line 4) đọc file xml other tìm giá trị giống biến string cần so sánh lấy giá trong hashMap thay vào .

**Algorithm 6: genPolicy**

**Input**: key và vaule của file excel

**Ouput**: sinh ra các file policy mẫu từ file gốc có key và value của file excel

1: read excel forward key and value vào hashMap

2: assign values ​​to compare to variables

3: read xml form and write into file xml other

4: read xml other to so sánh giá trị cần đổi

1. Giải thuật genRequest:

Đọc file excel (line 1) lấy key và value rồi truyền giá trị vào hashMap. (line 2) gán trị cần so sánh vào 1 biến string . (Line 3) đọc file xml mẫu sau đó write vào file xml other. (Line 4) đọc file xml other tìm giá trị giống biến string cần so sánh lấy giá trong hashMap thay vào

**Algorithm 6: genPolicy**

**Input**: key và vaule của file excel

**Ouput**: sinh ra các file requeyst mẫu từ file gốc có key và value của file excel

1 read excel forward key and value vào hashMap

2: assign values ​​to compare to variables

3: read xml form and write into file xml other

4: read xml other to so sánh giá trị cần đổi

1. Giải thuật Request Json:

# **ĐÁNH GIÁ KẾT QUẢ**

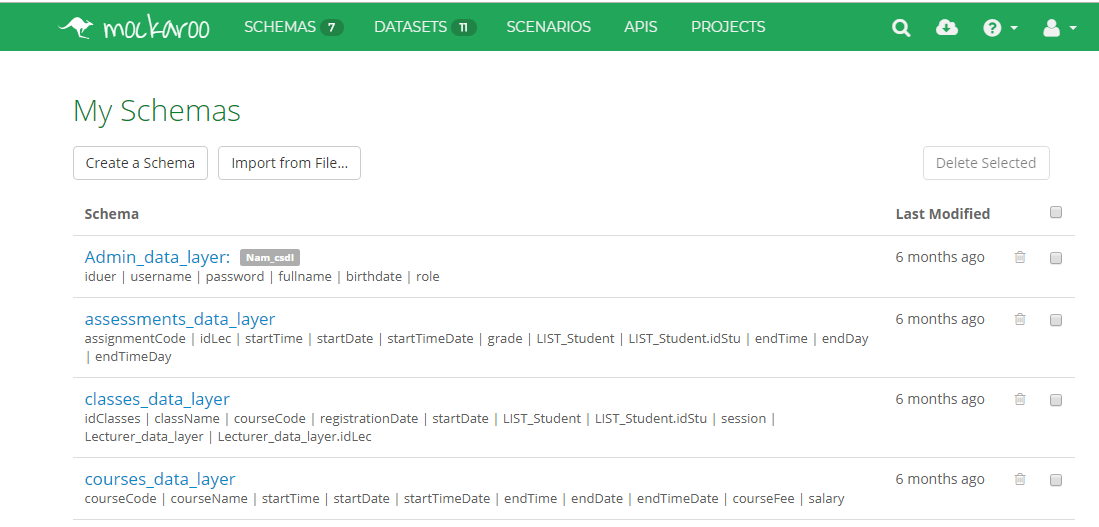
1. **Kết quả đạt được:**

Do phạm vi đề tài có hạn nên đề tài đạt được kết quả như sau:

Nắm được cấu trúc, nguyên lý XACMl, cách viết ALFA để viết policy cho các hệ thống lớn về quản lý phân quyền cũng như các hệ thống web security.

1. **tạo cơ sỡ dữ liệu trên mockaroo**

Biết các tạo dữ liệu lớn bằng mockaroo để xây dựng cơ sở dữ liệu thử nghiệm:



1. **So sánh với các nghiên cứu trước đây**

Ngoài ra nhóm còn tiến hành so sánh trực tiếp giữa nghiên cứu với:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| balana | | xEngine\_beta0.2 | | | sunxacml\_1.1 | | | sne | |
| Kmarket 2 | |  | policy\_test1 | continue-a |  | continue-a | policy\_test1 | continue-a-xacml3 | |
| read policy | 2 | xacml | 588 | 1408 | xacml&request | 486 | 206 | PDP | 2119 |
| request | 133 | request | 33 | 254 | setup PDP | 449 | 182 | request+response | 144096 |
| response | 128 | request Sun's PDP | 60 |  | response | 482 | 202 | request+response | 144 |
|  |  | results | 6 | 7 |  |  |  |  |  |

Kết quả đo element policy:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Policy level | Policy Set | Policy | Rule |
| KMarket2 | 2 | 1 | 3 | 12 |
| GEYSERS | 2 | 1 | 7 | 15 |
| continue-a-xacml3 | 8 | 111 | 266 | 298 |
| contrinue-a |  |  |  |  |
| itrust3 | 1 | 0 | 1 | 64 |
| policy\_test1 | 0 | 1 | 1 | 1 |

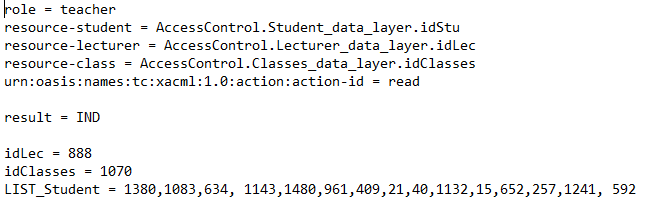
Kết quả đo đạt thời gian của luận văn

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | KMarket2 | continue-a-xacml3 | contrinue-a | GEYSERS | itrust3 | policy\_test1 |
| Parse Map | 120.25 | 569.75 | 562 | 164.5 | 316.25 | 266.75 |
| Parse Domain | 125.75 | 562.25 | 570.25 | 171.75 | 332.25 | 273.25 |
| quey | 132.75 | 586.75 | 582.75 | 183 | 2650 | 288 |
| ShowAtPolicyOnly | 74 | 1551 | -186 | 1098 | 2908 | 3057 |
| ShowAtPolicyWithTargetCombination | 78 | 2029 | 123 | 1133 | 2764 | 2737 |
| ShowAtRule | 88 | 2155 | 117 | 1035 | 2691 | 2987 |
| ShowAtConstraint | 88 | 1524 | -270 | 1238 | 1546 | 2701 |
| Thoi gian response | 94.5 | 490.25 | 640 | 909.25 | 3810.25 | 1527 |
| tg doc qua map,domain,leve | 555.25 | 4038.75 | 2301 | 2554.5 | 3401.75 | 5241.25 |
| tg doc qua map,domain | 378.75 | 1733.75 | 1715 | 519.25 | 316.25 | 1291.25 |

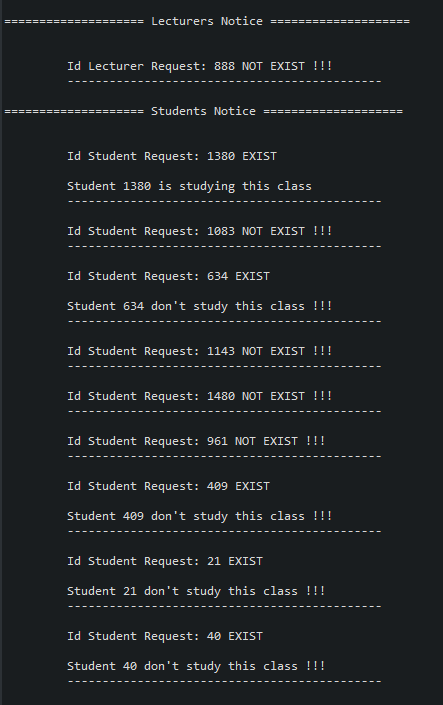
Ta thấy được các nghiên cứu trước đây chỉ tập chung đo một mẫu policy nhất định không phân tích cũng như đánh giá các loại policy khác hay các mẫu policy khác phiên bản. họ chỉ tập trung vào thời gian ra kết quả nhanh nhất mà không nghỉ về các version XACML hay nói khác hơn là các nghiên cứu ấy chỉ tập trung một mẫu policy nhất đinh. Còn luận văn của bon em áp dụng phân tích, đánh giá policy dựa vào SMT-z3 có thể phân tích đánh giá policy trên các mẫu policy khác nhau nên đánh ứng được mẫu policy mới nhất hay policy lỗi thời, cùng với ta biết rằng policy và request được viết dưới dạng xml nay luận văn của bọn có thể chuyển request thành dạng Json mà poliicy có nhận request và phân tích , đánh giá dưa ra kết quả.

1. **Kết quả đánh giá với Mongodb**

Phân tích policy đóng góp rất lớn cho sự pháp triển hệ thống sau này : việc phân tích policy trả ra kết quả là tiền đề để pháp triển một hệ thống phân tích chính sách cho dữ liệu bigdata. Luận văn này chúng em bắt đầu pháp triển từ mongodb. Từ file txt value chuyển cho mongodb xử lý.



Phần mongodb



1. **Kết Luận**

Tóm lại, luận văn này đã đề xuất một cơ chế quản lý chính sách XACML bằng cách phân tích mức độ khác nhau. Có bốn phân tích cấp độ khác nhau bao gồm ShowAtPolicyOnly, ShowAtPolicyWithTargetCombination, ShowAtRule và ShowAtConstraint. Theo yêu cầu của người dùng, mức độ được chọn mô tả như một biểu thức logic trừu tượng hoặc biểu thức logic chi tiết. Cách tiếp cận này không chỉ làm giảm khả năng sai sót trong quy trình thiết kế chính sách mà còn phá vỡ rào cản công nghệ giữa các nhà hoạch định chính sách và quản trị viên. Bộ giải SMTZ3 được sử dụng để phân tích và xác minh chính sách XACML do đó cải thiện hiệu suất của quy trình đánh giá. Luận văn này cũng đã thực hiện một cơ chế quản lý chính sách đối với chính sách XACML nhằm xác định quyền truy cập dữ liệu cho mỗi người dùng, ngăn chặn người dùng độc hại truy cập bất hợp pháp. Cuối cùng, chúng tôi đã triển khai và đánh giá hiệu suất của bốn chính sách phân tích cấp độ XACML khác nhau với bốn chính sách thực tế. Cách tiếp cận của chúng tôi có thể được áp dụng cho các mô hình dữ liệu lớn để quản lý quyền truy cập dữ liệu cá nhân trong các hệ thống lớn. Hiện tại, bài viết này đang được thực hiện trong một nghiên cứu trường hợp. Trong tương lai, chúng tôi sẽ tiếp tục phát triển một cơ chế quản lý hoàn chỉnh cho phức tạp hơn trên các hệ thống dữ liệu lớn.

# **Tài liệu tham khảo**

1. Tutorial on XACML: http://www.cs.odu.edu/~mukka/cs795sum15.net/Lecturenotes/day7/xacmltutorial.pptx

2.Modelling and Analysing Access Control Policies in XACML 3.0

http://orbit.dtu.dk/files/117690532/phd364\_Ramli\_CDPK.pdf

A Beginner’s Guide to XACML

https://www.axiomatics.com/

[1] H. X. Son, T. K. Dang, and F. Massacci, “Rew-smt: A new approach for rewriting xacml request with dynamic big data security policies,” in International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, 2017, pp. 501– 515.

[2] Turkmen, F., Demchenko, Y.: On the use of smt solving for xacml policy evaluation. In: Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on. pp. 539–544. IEEE (2016)

[3] Turkmen, F., den Hartog, J., Ranise, S., Zannone, N.: Analysis of xacml policies with smt. In: International Conference on Principles of Security and Trust. pp. 115–134. Springer (2015)

[4] Turkmen, F., den Hartog, J., Ranise, S., Zannone, N.: Formal analysis of xacml policies using smt. Computers & Security 66, 185–203 (2017)