

پردیس دانشگاهی

(عنوان سمینار)

رشته مهندسي ...............................

(نام نام خانوادگی دانشجو)

استاد راهنما:

(نام نام خانوادگی)

.....ماه سال

دانشكده مهندسي كامپيوتر



چکيده

امروزه اطلاعات مستقر بر روي وب به سرعت در حال افزايش است و حجم بسيار زيادي از اين اطلاعات در قالب XML ظاهر شده است. يكي از مزاياي اصلي استفاده از XML ، نمايش داده‌هاي غير ساختيافته است كه قابليت‌هاي بسياري را در اختيار كاربران مي‌گذارد. ويژگي غير ساختيافته بودن اطلاعات و انعطاف‌پذيري XML باعث همه‌گير شدن استفاده از آن شده و در بانك‌هاي اطلاعات نيز مورد توجه قرار گرفته است. بنابراين برقراري امنيت در مستندات XML يك نياز و بحث كليدي مي‌باشد. داده‌ها به هر شكلي كه ذخيره شوند بايد از تهديدهاي ممكن (سرقت، خرابكاري، دستكاري و مواردي از اين قبيل) محافظت گردند. براي جلوگيري از تهديدها، روش‌ها و مدل‌هايي را در بانك‌هاي اطلاعات طرح‌ريزي و پياده‌سازي نموده‌اند. مهمترين اين مدل‌ها، مدل كنترل دسترسي مي‌باشد. اين مدل خود مبتني بر روش‌هاي مختلفي مي‌باشد كه در بانك‌هاي اطلاعات گوناگون به كار گرفته مي‌شوند. در اين پايان‌نامه پس از بررسي روش‌هاي كنترل دسترسي، روشي جديد مبتني بر نگاشت بيتي و تابع دستور جهت كنترل دسترسي در بانك اطلاعات XML پيشنهاد شده است. در روش پيشنهادي سعي بر اين است كه كليه مشكلات و نواقص روش تابع دستور و نگاشت بيتي مرتفع گردد.

واژه‌هاي كليدي:بانك اطلاعات XML، كنترل دسترسي، امنيت، نگاشت بيتي.

فهرست مطالب

[فصل 1: مقدمه 2](#_Toc219340435)

[1-1- مقدمه 2](#_Toc219340436)

[فصل 2: مروري بر منابع و پيشينه‌ي تحقيق 2](#_Toc219340437)

[2-1- آشنايي با XML 2](#_Toc219340438)

[2-1-1- معرفي اجزاء اصلي XML 2](#_Toc219340439)

[2-1-2- مدل درختي XML 2](#_Toc219340440)

[2-1-3- مفهوم شِما در XML 2](#_Toc219340441)

[2-2- رابطه XML و پايگاه داده‌ها 2](#_Toc219340442)

[2-2-1- بانك اطلاعات پشتيبان XML 2](#_Toc219340443)

[2-2-2- بانك اطلاعات ذاتاً XML 2](#_Toc219340444)

[2-2-3- انواع ذخيره‌سازي در XML 2](#_Toc219340445)

[2-2-4- انواع زبان‌هاي پرس‌وجو در XML 2](#_Toc219340446)

[2-3- امنيت در بانك اطلاعات 2](#_Toc219340447)

[2-4- مدل‌هاي كنترل دسترسي در بانك اطلاعات XML 2](#_Toc219340448)

[2-4-1- مدل كنترل دسترسي محتاطانه 2](#_Toc219340449)

[2-4-2- مدل كنترل دسترسي الزامي 2](#_Toc219340450)

[2-4-3- مدل ليست كنترل دسترسي و مدل قابليت 2](#_Toc219340451)

[2-4-4- مدل كنترل دسترسي مبتني بر نقش 2](#_Toc219340452)

[2-4-5- مدل كنترل دسترسي مبتني بر تابع 2](#_Toc219340453)

[2-4-6- مدل كنترل دسترسي مبتني بر نگاشت بيتي 2](#_Toc219340454)

[2-5- نتيجه‌گيري 2](#_Toc219340455)

[فصل 3: روش تحقيق 2](#_Toc219340456)

[3-1- مقدمه 2](#_Toc219340457)

[3-2- مفاهيم و اصطلاحات 2](#_Toc219340458)

[3-3- بهينه‌سازي مكعب امنيت 2](#_Toc219340459)

[3-4- مدل پيشنهادي 2](#_Toc219340460)

[3-4-1- خط‌مشي كنترل دسترسي 2](#_Toc219340461)

[3-4-2- كنترل دسترسي 2](#_Toc219340462)

[3-4-3- معماري مدل پيشنهادي 2](#_Toc219340463)

[فصل 4: نتايج و تفسير آنها 2](#_Toc219340464)

[4-1- ارزيابي مدل پيشنهادي 2](#_Toc219340465)

[4-2- مقايسه مدل‌هاي كنترل دسترسي 2](#_Toc219340466)

[فصل 5: جمع‌بندي و پيشنهادها 2](#_Toc219340467)

[مراجع 2](#_Toc219340468)

[پيوست‌ها 2](#_Toc219340469)

فهرست اشکال

[شکل (2-1( نمونه‌اي از يك سند XML و اجزاء آن 2](#_Toc219341147)

[شکل (2-2( ساختار درختي سند XML 2](#_Toc219341148)

[شکل (2-3( نمونه‌اي از يك شماي XML در XML Schema 2](#_Toc219341149)

[شکل (2-4) مثالي از يك گراف نقش 2](#_Toc219341150)

[شکل (2-5) شماي مجوز شئ براي مثال نمونه 2](#_Toc219341151)

[شکل (2-6) گراف مجوز شئ براي مثال نمونه 2](#_Toc219341152)

[شکل (2-7) نمودار نوع مجوز براي مثال نمونه ذكر شده 2](#_Toc219341153)

[شکل (2-8) الگوريتم انتشار مجوز 2](#_Toc219341154)

[شکل (2-9) مثالي از ORF براي خط‌مشئ P1 2](#_Toc219341155)

[شکل (2-10) مثالي از SRF براي خط‌مشئ P1 2](#_Toc219341156)

[شکل (2-11) مثالي از GRF كه // را پشتيباني مي‌كند 2](#_Toc219341157)

[شکل (2-12) مثالي از SRF كه // را پشتيباني مي‌كند 2](#_Toc219341158)

[شکل (2-13) قطعه كدي جهت گزاره CustKey = $custID 2](#_Toc219341159)

[شکل (2-14) سيستم كنترل دسترسي مبتني بر تابع 2](#_Toc219341160)

[شکل (2-15) يك شاخص نگاشت بيتي براي مستندات XML 2](#_Toc219341161)

[شکل (2-16) مثالي از يك ديد محدود شده 2](#_Toc219341162)

[شکل (2-17) مكعب امنيت 2](#_Toc219341163)

[شکل (2-18) نمونه‌اي از مكعب امنيت با موضوعاتي از قبيل كاربران، آدرس‌هاي IP و نام‌هاي سمبليك 2](#_Toc219341164)

[شکل (3-1) نمونه‌اي از يك DTD سيستم آموزش 2](#_Toc219341165)

[شکل (3-2) قوانين مربوط به نقش دانشجو براي مجوز خواندن بدون انتشار مجوز 2](#_Toc219341166)

[شکل (3-3) قوانين مربوط به نقش دانشجو براي مجوز خواندن با انتشار مجوز 2](#_Toc219341167)

[شکل (3-4) معماري مدل پيشنهادي 2](#_Toc219341168)

[شکل (4-1) نمونه پرس‌وجوهاي اجرا شده جهت ارزيابي سرعت دسترسي به اطلاعات 2](#_Toc219341169)

[شکل (4-2) ارزيابي سرعت دستيابي به اطلاعات با مسيرهاي ساده 2](#_Toc219341170)

[شکل (4-3) ارزيابي سرعت دستيابي به اطلاعات با مسيرهاي داراي // و \* 2](#_Toc219341171)

[شکل (4-4) نمونه پرس‌وجوهاي اجرا شده جهت ارزيابي سرعت به‌روزرساني اطلاعات 2](#_Toc219341172)

[شکل (4-5) ارزيابي سرعت به‌روزرساني اطلاعات با دستورات XQuery در مدل پيشنهادي 2](#_Toc219341173)

[شکل (4-6) ارزيابي سرعت به‌روزرساني اطلاعات با دستورات XQuery در مدل آقاي يون 2](#_Toc219341174)

[شکل (4-7) حجم مكعب امنيت در مدل آقاي يون 2](#_Toc219341175)

فهرست جداول

[جدول (2-1) ماتريس وابستگي مجوز براي مثال نمونه 2](#_Toc219340937)

[جدول (2-2) نتايج ارزيابي يك تابع دستور 2](#_Toc219340938)

[جدول (2-3) نمايش نگاشت بيتي از كنترل دسترسي در سطح DTD 2](#_Toc219340939)

[جدول (2-4) نمايش نگاشت بيتي از كنترل دسترسي در سطح سند 2](#_Toc219340940)

[جدول (2-5) نمايش نگاشت بيتي از كنترل دسترسي در سطح مسير المان 2](#_Toc219340941)

[جدول (2-6) نمايش نگاشت بيتي از كنترل دسترسي در سطح محتوا 2](#_Toc219340942)

[جدول (3-1) روش محاسبه اندازة مكعب امنيت 2](#_Toc219340943)

[جدول (3-2) الگوريتم به‌روزرساني مكعب امنيت براي كاربر/مجوز جديد 2](#_Toc219340944)

[جدول (3-3) انواع مختلف مكعب‌هاي امنيت 2](#_Toc219340945)

[جدول (4-1) تعداد اقلام اطلاعاتي در يك سيستم آموزش نمونه 2](#_Toc219340946)

[جدول (4-2) حجم مكعب امنيت براي سيستم آموزش نمونه 2](#_Toc219340947)

[جدول (4-3) حجم مكعب امنيت بهينه براي سيستم آموزش 2](#_Toc219340948)

[جدول (4-4) نمونه‌اي از جدول نگاشت بيتي دسترسي در سطح سند در سيستم آموزش 2](#_Toc219340949)

[جدول (4-5) الگوريتم به‌روزرساني مكعب امنيت بهينه در مدل پيشنهادي 2](#_Toc219340950)

[جدول (5-1) مقايسه مدل‌هاي كنترل دسترسي 2](#_Toc219340951)

1. مقدمه
   1. مقدمه

اكثر نرم‌افزارهاي كاربردي تجاري، سيستم‌هاي تبادل داده، و شركت‌هاي مختلف داده‌هاي ساختيافته‌ي خود را در بانك‌هاي اطلاعات رابطه‌اي نگهداري مي‌كنند. بانك‌هاي اطلاعات رابطه‌اي نمونه خوبي از بانك‌هاي اطلاعات ساختيافته هستند. بانك‌هاي اطلاعات نيمه ساختيافته مشابه بانك‌هاي سنتي شِماي ثابتي ندارند. داده‌هاي نيمه ساختيافته، خود تعريف مي‌باشند و مي‌توانند مدل ناهمگون‌تري نسبت به داده‌هاي ساختيافته داشته باشند. زبان نشانه‌گذاري توسعه‌پذير XML تكنيك مناسبي براي مدل‌سازي چنين داده‌هايي است [1]، هر چند داده‌هاي ساختيافته و غيرساختيافته نيز مي‌تواند در قالب XML ذخيره شود. نمايش داده‌هاي غير ساختيافته به صورت XML قابليت‌هاي بسياري را در اختيار كاربران مي‌گذارد. ويژگي غير ساختيافته بودن اطلاعات و انعطاف‌پذيري XML و همچنين همه‌گير شدن استفاده از آن باعث شده است كه در بانك‌هاي اطلاعات نيز مورد توجه قرار گيرد. در اين ميان دسترسي به اطلاعات و امنيت اطلاعات مبادله شده بسيار مورد توجه مي‌باشد.

جهت برقراري امنيت در بانك اطلاعات XML مدل‌ها، مكانيزم‌ها و روش‌هايي وجود دارد. در حال حاضر تنها مدل‌هايي كه در برقراري امنيت در بانك اطلاعات XML استفاده مي‌شود مدل‌هاي كنترل دسترسي مي‌باشند. مطابقت مدل‌هاي كنترل دسترسي با مستندات XML و نيز چگونگي استفاده از اين مدل‌ها در بانك اطلاعات XML در اين پايان‌نامه تشريح شده است.

بيشتر كنترل‌هاي دسترسي مرسوم شامل ليست‌هاي كنترل دسترسي [2]، ليست‌هاي قابليت [3] و ماتريس‌‌هاي كنترل دسترسي [4] هستند. هنگامي كه درباره امنيت يك سيستم صحبت مي‌شود "كنترل دسترسي" جنبه‌هاي بسياري را شامل مي‌شود. كنترل دسترسي بايد رسا و قابل فهم باشد و بتواند امنيت دسترسي به داده‌هاي مستقر در يك مكان را پشتيباني نمايد. همچنين در بسياري از سيستم‌ها همچون سيستم‌هاي تراكنشي تجاري و بايگاني‌هاي پزشكي كه شامل داده‌هاي حساس مي‌باشند، كنترل دسترسي در پايين‌ترين سطح (عنصر و يا صفت) مورد نياز است.

كارهاي بسيار زيادي براي توصيف كنترل دسترسي روي مستندات XML از چند سال قبل انجام شده است. برخي از اين كوشش‌ها عبارتند از: تعريف و اجراي خط‌مشئ‌هاي كنترل دسترسي بر روي منابع XML [5]، كنترل دسترسي به مستندات XML توسط تعيين سطوح مجوزها و خط‌مشئ‌هاي انتشار مجوزها [6]، توصيف كنترل دسترسي براي اسناد XML كه ارتباط معنايي با هم دارند [22] و تعريف يك سيستم كنترل دسترسي در پايين‌ترين سطح براي مستندات XML [7]. كنترل دسترسي در پايين‌ترين سطح شامل توصيف موضوع‌هاي مجوز (كاربران يا گروه‌هاي كاربري و يا كامپيوترها)، اشياء مورد دسترسي (المان‌ها و محتواي المان‌ها) و تعيين مجوزهاي دسترسي مي‌باشد. تمركز در اين كار عموماً بر روي توصيف يك زبان براي تعيين محدوديت‌هاي كنترل دسترسي به مستندات XML و همچنين توصيف انواع مختلف خصوصيات و خط‌مشئ‌هاي مرتبط با آن مي‌باشد.

در اين پايان‌نامه روشي جديد مبتني بر نگاشت بيتي [8] و تابع دستور [9] جهت كنترل دسترسي در بانك اطلاعات XML پيشنهاد شده است. در روش پيشنهادي كنترل دسترسي را در پايين‌ترين سطح ايجاد مي‌كنيم. همچنين در روش پيشنهادي سعي شده است كه كنترل دسترسي بامعني ايجاد گردد. كنترل دسترسي با معني را با استفاده از الگوريتمي كه نقش‌ها را به نگاشت بيتي تبديل مي‌كند مهيا مي‌كنيم. در اين روش تلاش شده است كه مشكلات و نواقص روش تابع دستور و نگاشت بيتي مرتفع گردد.

در فصل اول اين پايان‌نامه پس از معرفي اجمالي بانك اطلاعات XML ، به امنيت در بانك اطلاعات XML مي‌پردازيم. در فصل دوم مدل‌هاي كنترل دسترسي كه براي بانك اطلاعات XML معرفي شده‌اند را بررسي مي‌كنيم. در فصل سوم روش پيشنهادي ارائه مي‌گردد. در فصل چهارم به پياده‌سازي و ارزيابي روش پيشنهادي مي‌پردازيم، و مقايسه‌اي بين مدل‌هاي مختلف كنترل دسترسي و روش پيشنهادي جهت استفاده در بانك‌هاي اطلاعات XML انجام خواهد گرفت. در فصل پاياني به نتيجه‌گيري و كارهاي آتي مي‌پردازيم.

1. مروري بر منابع و پيشينه‌ي تحقيق
   1. آشنايي با XML

زبان XML در سال 1997 توسط كنسرسيوم وب براي رفع محدوديت‌هاي موجود در HTML و اضافه كردن قابليت‌هاي جديد بوجود آمد [1]. تفاوت اصلي XML با HTML در اين است كه XML سعي دارد داده‌ها را طوري نشانه‌گذاري كند كه معناي آنها حفظ شود و در حاليكه HTML داده‌ها را طوري نشانه‌گذاري مي‌كند كه قابل نمايش براي مرورگرها باشد. در واقع XML يك استاندارد باز مي‌باشد و تاكيد آن بر روي معناي داده‌هاست در حاليكه تأكيد HTML بر نمايش داده مي‌باشد. XML. يك استاندارد پايه است كه وابستگي به رسانه‌ها، سيستم عامل‌ها و ديگر سيستم‌ها را ندارد. به منظور حفظ معناي داده‌ها، XML ابر داده[[1]](#footnote-1) كه توصيف كننده داده مي‌باشد را نيز همراه آنها ذخيره مي‌كند.

XML زير مجموعه ساده شده‌اي از زبان SGML مي‌باشد. زبان SGML يك زبان عمومي و پيچيده براي نشانه‌گذاري داده‌هاست كه در دهه 80 بوجود آمد و پدر زبان‌هاي نشانه‌گذاري محسوب مي‌شود. قابليت‌هاي زياد اين زبان آن را بيش از اندازه پيچيده كرده است، بطوري كه كنسرسيوم وب آن را بعنوان جانشين HTML مناسب ندانسته و تصميم گرفت زير مجموعه ساده شده‌اي از آن را با نام XML جانشين HTML كند. زبان XML در واقع يك ابر زبان ناميده مي‌شود. چرا كه كاربر بسته به نيازهايي كه دارد مي‌تواند با استفاده از آن زبان نشانه‌گذاري جديدي براي داده‌هايش ايجاد كند.

XML همچنين قادر است ساختار داده‌هاي ذخيره شده را نيز به‌همراه معناي آنها حفظ كند. اين زبان داراي هيچ برچسب از پيش تعريف شده‌‌اي نيست و تمامي برچسب‌ها برحسب نياز توسط كاربر تعريف مي‌شوند. قابليت‌هاي XML و اجزاء همراه آن، اين زبان را به زبان قابل حمل و استاندارد براي كاربردهاي مختلف تبديل كرده است.

فايل‌هاي XML داراي قابليت متني هستند به‌طوري كه مي‌توان آنها را در ويرايشگرهاي متني ويرايش كرد. يك فايل در XML از دو قسمت متن و علائم نشانه‌گذاري تشكيل شده است كه در قسمت متن داده‌هاي اصلي ذخيره مي‌شود و در قسمت علائم نشانه‌گذاري، ابرداده‌ توصيف كننده متن قرار مي‌گيرد. XML از يك سو با ذخيره فايل‌هايش در قالب متني و استفاده از علائم نشانه‌گذاري‏ فهم معناي داده‌هاي ذخيره شده را براي انسان ممكن مي‌سازد و از سوي ديگر با در اختيار قرار دادن اين فايل‌ها در يك قالب ساختيافته براي برنامه‌ها، پردازش آن را براي كامپيوتر ساده مي‌كند.

* + 1. معرفي اجزاء اصلي XML

جزء اصلي تشكيل دهنده XML ، عنصر يا المان[[2]](#footnote-2) نام دارد كه شامل نام و محتوي مي‌باشد. محتواي يك عنصر بين دو علامت نشانه‌گذاري خاص با نام‌هاي برچسب شروع و برچسب پايان محصور مي‌شود. روش برچسب‌گذاري XML همانند HTML است كه در اصل HTML و XML اين روش را از SGML به ارث برده‌اند.

برچسب شروع شامل نام عنصر است كه بين دو علامت > و < قرار گرفته و برچسب پايان داراي علامت / اضافه شده قبل از نام عنصر مي‌باشد. براي مثال رشته زير نمايش دهنده عنصر tel در XML مي‌باشد كه محتواي آن شماره تلفن 77292512-21-0098 مي‌باشد.

<tel>0098-21-77292512</tel> ‌

هر يك از عناصر مي‌توانند داراي چندين صفت باشند كه هر صفت از يك زوج نام و مقدار تشكيل شده است. در مورد صفت نيز نحوه نگارش همانند HTML مي‌باشد به اين صورت كه صفات يك عنصر در داخل برچسب شروع آن عنصر قرار مي‌گيرند و بين هر صفت و مقدار آن علامت "=" قرار گرفته و مقدار صفت نيز مابين كاراكترهاي مشخص كننده رشته محصور شده است. براي مثال عنصر tel در مثال قبلي مي‌تواند شامل صفتي با نام Preferred به صورت زير باشد كه محتواي آن true است.

<tel prefered="true">0098-21-77292512</tel> ‌

محتواي عناصر محدود به متن نيست و عناصر مي‌توانند شامل عناصر ديگر باشند كه آنها نيز مي‌توانند شامل متن و عناصر ديگر باشند. به عبارت ديگر يك فايلXML درختي از عناصر است كه محدوديتي برعمق آن وجود ندارد و عناصر مي‌توانند در اين درخت بطور دلخواه تكرار شوند. براي نمونه يك فايل XML در شكل 2-1 آورده شده است.

<address-book>

<entry>

<name>John Doe</name>

<address>

<street>34 Fountain Square Plaza</street>

<region>OH</region>

<postal-code>45202</postal-code>

<locality>Cincinnati</locality>

<country>US</country>

</address>

<tel preferred="true">513-555-8889</tel>

<tel>513-555-7098</tel>

<email href="jdoe@yahoo.com"/>

</entry>

<entry>

<name>

<fname>Jack</fname>

<lname>Smith</lname>

</name>

<tel>513-555-3465</tel>

<email href="mailto:jsmith@yahoo.com"/>

</entry>

</address-book>

عنصر/المان

صفت

متن/محتوي

نمونه‌اي از يك سند XML و اجزاء آن

هر سند XML، يك المان به عنوان المان بالاترين سطح (المان ريشه) خواهد داشت كه بقيه المانها درون آن قرار دارند. برچسب‌هاي شروع و پايان هميشه بايد با يكديگر وجود داشته باشند و فرزندان يك عنصر همواره بايد بطور كامل بين برچسب‌هاي شروع و پايان پدر خود محصور شده باشند. به عبارت ديگر نبايد برچسب‌هاي پايان فرزند بعد از برچسب پايان پدر خود ظاهر شود.

بايد توجه داشت كه كاراكترها در يك فايل XML مطابق استاندارد Unicode 2 مي‌باشند و بنابراين فايل‌هاي XML قادر به نمايش كاراكترهاي تمام زبان‌هاي زنده دنيا مي‌باشند كه اين يكي از مزاياي XML براي كاربردهاي مبادله داده مي‌باشد. البته به اين دليل كه كاراكترها در استاندارد Unicode به صورت دوبايتي كد مي‌شوند مي‌توان براي جلوگيري از افزايش حجم فايلهاي XML آنها را با استفاده از استانداردهاي ديگر مانند UTF-8 و يا ISO 8859-1 كه اولي مخصوص كاراكترهاي انگليسي و ديگري مخصوص كاراكترهاي لاتين است، نيز كد كرد.

* + 1. مدل درختي XML

ساختار سندهاي XML به صورت درختي است (البته در برخي موارد به صورت گراف نيز مشاهده مي‌شود). اين ساختار درختي به DOM (Document Object Model) معروف است. همچنين اين درخت، يك گراف فاقد حلقه (DAG) نيز مي‌باشد. در اين ساختار درختي، در ريشه درخت اولين گره از سند و در برگ‌هاي اين درخت، مقدارها و متن‌هاي موجود در سند XML قرار مي‌گيرند. فرزندان هر گره زير المانها و يا صفات آن المان هستند كه به ترتيب موجود در سند XML قرار مي‌گيرند و هر گره تنها يك پدر دارد (بجز ريشه). گره‌هاي مياني مي‌توانند عنصر و يا صفت باشند (در مدل درختي هر دوي اين‌ها يكي در نظر گرفته مي‌شوند). همه مفاهيم كلي درخت نظير همزاد[[3]](#footnote-3)، پدر، فرزند، جد[[4]](#footnote-4) و ... در مورد اين ساختار درختي صادق هستند. شكل 2-2 مدل درختي سند بالا را نشان مي‌دهد.



ساختار درختي سند XML

* + 1. مفهوم شِما در XML

همراه هر سند در زبان XML بايد يك ساختار بنام شِما[[5]](#footnote-5) باشد. هدف اين شِما اعتبارسنجي براي سندXML است، سندي را اصطلاحاً معتبر و يا خوش‌فرم[[6]](#footnote-6) مي‌گويند كه با شِماي خود مطابقت كند. از مهمترين ويژگي‌هاي XML اين است كه ساختيافته نيست، يعني الزامي وجود ندارد که شِماي يك سند XML مشخص باشد زيرا خود- تعريف است. با اين حال براي تبادل داده‌هاي XML، به خصوص در تفسير داده‌هاي دريافتي، شِما اهميت زيادي خواهد داشت. اصولاً شِماي يك سند را مي‌توان به دو صورت زير تعريف كرد:

* مبتني بر گرامر[[7]](#footnote-7)، كه بر اساس يكسري از علائم تعريف مي‌شود.
* ساختار درختي[[8]](#footnote-8)، كه اين ساختار شبيه به درخت DOM است.

دو مكانيزم اصلي تعريف شِما در XML عبارتند از:

1. Document Type Definition (DTD) ، كه بسيار رايج و مرسوم است.
2. XML Schema ، كه جديدتر و پيچيده‌تر از DTD مي‌باشد.

در DTD ساختار و شماي XML را از نظر المان‌هاي موجود، صفات و زيرالمان‌هاي هر المان مشخص مي‌نمائيم اما هيچ محدوديتي راجع به نوع داده‌ها تعيين نمي‌كنيم زيرا در XML تمام مقادير از نوع رشته مي‌باشند. شکل کلي DTD براي تعيين ساختار المان‌ها به صورت زير است:

<! ELEMENT element\_name (subelements\_specification)> ‌

كه در آن element\_name نام المان يا همان بر چسب مربوط به المان را نشان مي‌دهد و درون پرانتز، ليست نام زيرالمان‌هاي آن كه با يك کاما از هم جدا مي‌شوند مشخص مي‌گردد. در بخش subelements\_specification به جاي نام زير المان‌ها، مي‌توان از كلمه كليدي #PCDATA كه معرف رشته كاركتري بودن مقدار آن المان است (که براي همه المان‌هاي داراي مقدار، اينگونه خواهد بود)، استفاده کرد يا از EMPTY و يا ANY به ترتيب به معناي "فاقد زير المان" و "هر چيزي به عنوان زير المان".

اصول نگارشي در DTD شبيه به عبارات منظم است، لذا مي‌توان تعداد دفعاتي كه چنين الماني مي‌تواند در المان خارجي‌تر خود وجود داشته باشد را نيز نشان داد. نمادهاي زير را مي‌توان با توجه به مفهوم آنها در بخش subelements\_specification بكار برد:

| : ياي انتخابي

+ : يک يا بيشتر

\* : صفر يا بيشتر

براي مثال در زير يك قسمت از يك DTD آورده شده است:

<! ELEMENT edu (book\*)>

<! ELEMENT book (title, authors+, chapter\*, ref\*, CD\*)>

<! ELEMENT chapter (text | section)\*>

<! ELEMENT ref book>

<! ELEMENT title #PCDATA>

<! ELEMENT author #PCDATA>

<! ELEMENT section #PCDATA>

<! ELEMENT text #PCDATA>

براي تعيين ساختار صفت‌ها از شکل کلي زير استفاده مي‌شود:

<! ATTLIST element\_name (attributes\_specification)> ‌

attributes\_specification شامل نام و نوع صفت مورد نظر مي‌باشد. نوع صفت يا CDATA است يا يكي از اين سه مقدار ID ، IDREF ، IDREFS .

به عنوان مثال در زير نمونه‌اي از يك ساختار صفت نشان داده شده است.

<! ATTLIST price CDATA "$ 37.0"> ‌

XML Schema روش جديدي است كه W3C منتشر كرده است. اصول نگارشي آن تا حد زيادي شبيه به كدنويسي به زبان XML است. اين روش كه مشكلات روش DTD را رفع مي‌کند، بسيار پيچيده‌تر از آن مي‌باشد، به گونه‌اي كه هنوز به طور رايج از آن استفاده نمي‌شود. مهمترين امكانات تعبيه شده در XML Schema عبارتند از:

* مقادير مي‌توانند انواعي مثل string ، integer و ... داشته باشند.
* مي‌توان حداقل و حداكثر مقدار را نيز تعيين نمود.
* كاربر مي‌تواند انواع داده مورد نظر خود را تعريف نمايد.
* بر خلاف DTD، شکل کلي خودش نيز با XML مشخص مي‌شود.

شكل 2-3 نمونه‌اي از يك شماي XML كه با اين متدولوژي ايجاد شده است را نشان مي‌دهد.

<xsd:schema xmlns:xsd=<http://www.w3.org/2001/XMLSchema>>

<!-- this document describes the XML Schema namespace -->

<xsd:element name=“edu” type=“EduType”/>

<xsd:element name=“stud”>  
 <xsd:complexType>  
 <xsd:sequence>  
 <xsd:element name=“sname” type=“xsd:string”/>  
 <xsd:element name=“city” type=“xsd:string”/>  
 <xsd:element name=“avg” type=“xsd:decimal”/>  
 </xsd:squence>  
 </xsd:complexType>

</xsd:element>

….. definitions of customer and depositor ….

<xsd:complexType name=“EduType”>  
 <xsd:squence>

<xsd:element ref=“stud” minOccurs=“0” maxOccurs=“unbounded”/>

<xsd:element ref=“prof” minOccurs=“0” maxOccurs=“unbounded”/>

<xsd:element ref=“crs” minOccurs=“0” maxOccurs=“unbounded”/>

</xsd:sequence>

</xsd:complexType>

</xsd:schema>

نمونه‌اي از يك شماي XML در XML Schema

* 1. رابطه XML و پايگاه داده‌ها

زماني كه براي اولين بار XML ارائه شد، از آن به عنوان يك مدل داده‌اي براي مبادله اطلاعات ياد كردند. دليل اينكه براي مبادله اطلاعات از اين مدل درختي استفاده مي‌شد، اين بود كه XML به عنوان يك زبان بي‌طرف مطرح بود و بيشتر مدل‌هاي داده‌اي (درختي، رابطه‌اي و شئ‌‌گرا و ... ) مي‌توانستند داده‌هاي خود را به اين مدل تبديل كنند و بالعكس، از اين مدل‌ داده‌هاي خود را استخراج كنند.

پس از مدتي تعداد اين اسناد بسيار زياد شد و ايده‌اي كه مطرح شد اين بود كه خود سندهاي XML را به عنوان داده (و حتي در مواردي خود پايگاه داده) در نظر بگيريم. اين نگرش باعث شد كه دو نوع از اين بانك‌هاي اطلاعات پديدار شود [10]:

* + بانك اطلاعات ذاتاً XML[[9]](#footnote-9)
  + بانك اطلاعات پشتيبان XML [[10]](#footnote-10)
    1. بانك اطلاعات پشتيبان XML

بعد از اينكه XML فراگير شد و نياز به ذخيره‌سازي اين سندها بيش از پيش حس شد، بسياري از نرم‌افزارهاي شركت‌هاي معتبر در زمينه بانك‌هاي اطلاعات مانند Oracel ،MS SQL Server ، DB2 و ... سعي كردند كه تغييراتي در سندهاي XML بدهند و آنها را در همان جداول ذخيره كنند. در نتيجه به پايگاه داده خود يك لايه جديد اضافه كردند كه آن لايه مسئوليت تبديل داده‌هاي XML به داده‌هاي رابطه‌اي را برعهده داشت. كامپايلر اين لايه يك فاز اضافه نيز داشت كه پرس‌وجوهايي كه به زبان XPath و يا XQuery داده مي‌شد را تبديل به SQL مي‌كرد.

در اين بانك‌هاي اطلاعات، ذخيره‌سازي به صورت رابطه مي‌باشد. در اين نوع بانك‌هاي اطلاعات بايد داده‌هاي XML را به فرم رابطه تبديل نمائيم. مهمترين مزيت اين روش اين است كه از آن پس با يك سيستم بانك اطلاعات رابطه‌اي سروكار خواهيم داشت و از امكانات وسيع آن بهره خواهيم جست. عيب بزرگ اين روش سربار انجام تبديلات لازم براي داده‌ها و پرس‌وجوهاست. مي‌توان از مدل‌هاي ديگر مثل شئ‌گرا و شئ‌- رابطه‌اي نيز بجاي مدل رابطه‌اي استفاده نمود.‌

* + 1. بانك اطلاعات ذاتاً XML

پايگاه‌داده‌هاي قبلي براي ذخيره‌سازي داده‌هاي XML و همچنين براي پاسخ به پرس‌وجوهاي XML داراي مشكلاتي بودند. براي مثال زماني كه جواب يك پرس‌وجو تكه‌هايي از يك سند كه ساختارهاي متفاوتي داشتند باشد، لايه مبدل نمي‌تواند جواب‌ها را ادغام كند و به صورت جداگانه به كاربر بر‌گرداند. از اين‌رو تعدادي از بانك‌هاي اطلاعات با شماي درختي ساخته شدند. هدف اين بانك‌هاي اطلاعات توليد يك DBMS براي سندهاي XML بود كه همه قابليت‌هاي بانك اطلاعات رابطه‌اي را داشته باشد.

در اين بانك‌هاي اطلاعات ذخيره‌سازي اطلاعات به صورت غير رابطه‌اي بوده و اساساً مدل داده‌اي، XML مي‌باشد [25، 26]. در حال حاضر سيستمي تجاري با اين مشخصه رايج و مرسوم نمي‌باشد. البته نمونه‌هايي از قبيل TIMBER ، XTC و Qizx/db [11] ساخته شده و در بسياري از پروژه‌هاي كاربردي استفاده شده‌اند، اما هركدام داراي نواقص و مشكلاتي خاص مي‌باشند.

مزاياي بانك‌ اطلاعات ذاتاً XML بازيابي ساده اطلاعات، حفظ ساختار اصلي، عدم نياز به ترجمه و نگاشت، ماهيت نيمه‌ساختيافته اطلاعات و سرعت بالاي دستيابي مي‌باشد. مهمترين معايب اين گونه بانك‌هاي اطلاعات عبارتند از [11]:

1. پيچيدگي مديريت و نگهداري داده‌هاي غير XML .
2. عمل كنترل همزماني در يك سند به راحتي امكان‌پذير نمي‌باشد.
3. مشكل به‌ روزرساني همچنان بي‌پاسخ مانده است، به طوري كه برخي از اين بانك‌هاي اطلاعات طبق شرايط خاص اجازه به روزرساني را مي‌دهند.
4. پرس‌وجوها ناكارآمد و وقت‌گير هستند.
5. فقدان سيستم‌هاي تجاري رايج.
6. ساختار پيچيده با تودرتويي زياد.
7. كمرنگ بودن امنيت
   * 1. انواع ذخيره‌سازي در XML

سندهاي XML را مي‌توان با توجه به نوع ذخيره‌سازي اطلاعات به دو گروه تقسيم كرد. مبتني بر داده[[11]](#footnote-11) و مبتني بر متن[[12]](#footnote-12) . سند مبتني بر داده داراي عناصري است كه تا حد زيادي داراي نظم و ترتيب هستند و از يك ساختار پيروي مي‌كنند. براي مثال اگر داده‌هاي موجود در جداول را به صورت درخت XML در نظر بگيريم، اين ساختار يك ساختار منظم مي‌باشد.

سندهاي مبتني بر متن از هيچ قاعده و اصولي پيروي نمي‌كنند، براي مثال بسياري از صفحات اينترنتي داراي يك چنين ساختاري هستند. بانك‌هاي اطلاعات XML براي ذخيره و بازيابي اين نوع از اسناد داراي مشكلات زيادي هستند. براي مثال تعريف يك عنصر كه بيش از چند مگابايت حافظه نياز دارد و يا پاسخ به پرس‌وجوها در اين سند بسيار سخت خواهد بود.

* + 1. انواع زبان‌هاي پرس‌وجو در XML

يك تفاوت عمده ميان داده‌هاي XML و داده‌هاي بانك‌هاي اطلاعات رابطه‌اي وجود دارد. در بانك‌هاي اطلاعات رابطه‌اي وجود يك زبان استاندارد براي پرس‌وجو مانند SQL علاوه بر اين كه مقبوليت عام را بدست آورده است، براي تمامي پرس‌وجوها در اين پايگاه‌هاي داده كاربرد دارد. ولي در دنياي XML قضيه فرق مي‌كند. زبان‌هاي پرس‌وجوي زيادي مانند XML-QL ، XPath ، XQuery و XQL [11] وجود دارند، كه هيچ كدام از اينها نتوانسته‌اند به عنوان يك استاندارد كلي براي همگان مقبوليت عام داشته باشند. حتي W3C نيز نتوانسته يكي از اين زبان‌ها را به عنوان زبان پرس‌وجوي رسمي در سندهاي XML مطرح نمايد. ولي براي كساني كه با XML كار مي‌كنند دو زبان XQuery [23] و XPath [24] آشناتر و پذيرش گسترده‌تري داشته است. ما در ادامه به‌طور مختصر اين دو زبان را شرح مي‌دهيم.

* زبان پرس‌وجوي XPath

در اين زبان براي انتخاب هر داده از يك سند XML، از عبارت مسيري[[13]](#footnote-13) متناظر با آن داده استفاده مي‌شود. ساختار اين زبان بيشتر شبيه به آدرس‌دهي فايل‌ها در محيط سيستم عامل مي‌باشد. هر عبارت مسيري دنباله‌اي از مراحل است كه از المان ريشه شروع شده و با **/** از هم جدا مي‌شوند و نتيجة آن مجموعه مقادير (المانها يا صفات) متناظر با آن مسير خواهد بود. هر مرحله از عبارت مسيري، روي نتيجه مرحله قبل عمل مي‌كند. براي نمونه به پرس‌وجوي زير دقت كنيد.

Root//a /b /c[@age=25]

اين پرس‌وجو به دنبال cهايي مي‌گردد كه داراي صفتي بنام age با مقدار 25 باشند و همچنين اين cها فرزند b و نوه a كه اين گره نيز نسلي از گره ريشه است، باشند.

توابع و عملگرهاي زيادي در XPath پشتيباني مي‌شوند (مثل تابع count() ). با استفاده از نماد گذاري مرسوم در سيستم فايل‌هاي سيستم عامل، مي‌توانيم به هر يك از گره‌هاي فرزند، پدر، همزاد، جد يا نوة گره مرحله قبل برويم. مثلاً با // كه معني پرش از چندين مرحله را مي‌دهد مي‌توان تمام نوه‌هاي يك گره را بدست آورد و يا .. كه پدر يک گره را نشان مي‌دهد و مشابه اينها.

* زبان پرس‌وجوي XQuery

اين زبان بيشتر شبيه به SQL است و براي برنامه‌نويسان آشناتر است. ساختار يك پرس‌وجو در زبان XQuery (به نام عبارت FLWOR)، شامل بخش‌هاي زير مي‌باشد:‌

For … Let ... Where … Order by … Return … ‌

For، Where و Return به ترتيب معادل عبارات From، Where و Select در زبان SQL هستند و Let كه معادلي در SQL ندارد، براي تعريف متغييرهاي موقتي لازم بكار مي‌رود. در بخش For از عبارات مسيري استفاده مي‌شود و نتيجه محاسبه شده آنرا در يك متغير مي‌ريزند. با كمك بخش‌هاي Whereو Letداده خاص مورد نظر را از اين متغير جدا كرده و اين داده را در عبارت Return، توسط برچسب‌هاي مناسب و در قالب مورد نظر برمي‌گردانند.

توابع زيادي از جمله distinct()(براي حذف جواب‌هاي تكراري)، count()، sum()و ... در XQuery قابل استفاده مي‌باشند. در XQuery عملگر گروه‌بندي (group by) پشتيباني نمي‌شود اما مي‌توان با استفاده از عبارت FLWOR درون عبارت Return (يعني FLWOR تودرتو)، جواب‌هاي محاسبه شده را مطابق آنچه كه خواسته شده است گروه‌بندي نمود.

براي مثال كد ذكر شده در مثال قبل را به زبان XQuery مي‌نويسيم.

For $c in a/b/c

Let $a := $c/@age

Where $c/age =25

Return <c>$a</c>

* 1. امنيت در بانك اطلاعات

امنيت در بانك اطلاعات را محافظت از تلاش‌هاي تبهكارانه براي سرقت، تغيير و يا تخريب داده‌هاي بانك اطلاعات تعريف مي‌كنند [11]. امنيت از جمله مفاهيمي است كه بايد در تمام لايه‌هاي يك سيستم كامپيوتري اعم از سخت افزار، شبكه، سيستم عامل، بانك اطلاعات و ... آن را لحاظ نمود. تهديدات امنيت به دو صورت عمدي و غير عمدي هستند. دستة اول شامل تمام تهديداتي است كه نتيجه تخلف عمدي كاربران و يا برنامه‌ها است. اين كاربران مي‌توانند كاربران مجاز سيستم و يا كاربران خارجي (كه مي‌توانند به صورت غير مجاز به سيستم و منابع آن دسترسي داشته باشند) باشند. برنامه‌ها نيز مي‌توانند هر برنامه محلي و يا هر برنامه از راه دور (شامل ويروس‌ها يا اسب‌ها تراوا و...) باشند. دستة دوم شامل تهديداتي است که بر اثر ناآگاهي يا عدم دقت و کوتاهي افراد و نارسائي و کيفيت پايين برنامه‌ها رخ مي‌دهد.

هنگامي كه صحبت از يك بانك اطلاعات امن به ميان مي‌آيد معمولاً سه هدف زير در رابطه با آن مطرح مي‌شود:

**1- محرمانگي[[14]](#footnote-14) :** محرمانگي به صورت عدم دسترسي كاربران غير مجاز به اطلاعات تعريف مي‌شود. به عنوان مثال يك دانشجو نبايد اجازه مشاهدة اطلاعات ساير دانشجويان را داشته باشد.

**2- جامعيت[[15]](#footnote-15):** فقط كاربران مجاز مي‌توانند داده‌هاي مربوطه را تغيير دهند. به عنوان مثال دانشجويان مي‌توانند نمرات خود را ببيند اما اجازه تغيير آن را ندارند. جامعيت به صورت جلوگيري از تغيير، حذف و يا دخالت ناخواسته و نادرست در اطلاعات نيز تعريف مي‌شود.

**3- دسترس پذيري[[16]](#footnote-16):** مجوز كاربران مجاز نبايد به طور ناخواسته قطع شود. به عنوان مثال استاد درسي كه اجازه تغيير نمرات را دارد همواره بايد بتواند اين عمل را انجام دهد.

براي رسيدن به سه هدف فوق بايد سياست‌هاي امنيتيواضح و مشخصي تدوين گردد. به عبارت ديگر بايد به طور كامل و صريح روشن گردد که چه بخش يا بخش‌هايي از داده‌ها بايد محافظت شوند و چه كاربراني اجازه دسترسي به چه قسمتهايي از داده‌ها را دارند.

براي برقراري امنيت در سيستم بانك‌اطلاعات، ابتدا با كمك مكانيزم‌هاي تصديق هويت كاربر مثل كلمة عبور اطمينان حاصل مي‌كنيم كه كاربر وارد شده به سيستم يك كاربر مجاز مي‌باشد. البته اين امكان نيز وجود دارد که كاربري غير مجاز به نحوي بتواند مجوز ورود را به دست آورد (مثلاً از کلمة عبور يک کاربر مجاز استفاده کند). در مرحله بعد كه مطئمن هستيم كاربري كه وارد سيستم شده اجازه ورود داشته است، با مكانيزم‌هاي کنترل دسترسي (مجوز دهي به كاربر)، فقط مجوزهاي دسترسي به داده‌هاي به خصوصي كه مي‌تواند به آنها دسترسي داشته باشد را به او تخصيص مي‌دهيم. اما اين كاربر مجاز ممكن است روي همان داده‌هاي مجاز خود نيز تغييرات غير مجاز انجام دهد (مثل قرار دادن عددي منفي در صفت سن). مقابله با چنين تهديداتي به تضمين جامعيت سيستم برمي‌گردد كه با قرار دادن قيود جامعيت، استفاده از مكانيزم‌هايي مثل assertion، trigger و.... از داده‌هاي بانك‌اطلاعات مراقبت مي‌نمائيم.

در بانك‌هاي اطلاعات نامتمركز بخشي از امنيت بانك به تبادل اطلاعات و ذخيره‌سازي آنها بستگي دارد. به عبارت ديگر در يك بانك اطلاعات نامتمركز، به دليل انتقال داده‌ها در شبكه و به‌خصوص بين گره‌هاي مديريت بانك اطلاعات، مي‌بايست داده‌هايي كه بين گره‌هاي مختلف شبكه در تبادل هستند را از ديد افراد و كاربران مخفي نمود و با مكانيزم‌هايي داده‌ها را كد و رمز نگاري كرد. رمزنگاري امنيت كانال‌هاي ارتباطي را برقرار مي‌كند. ايده اصلي رمزنگاري داده‌ها استفاده از الگوريتم‌هاي رمز نگاري و نيز يك كليد رمز نگاري مخصوص مدير بانك‌اطلاعات است كه به صورت امن نگاه داشته مي‌شود. خروجي يك الگوريتم رمزنگاري، نسخه رمز شده داده‌هاي ورودي است. به همين ترتيب الگوريتم‌هاي رمز گشايي براي باز كردن داده‌هاي رمز شده و برگرداندن آنها به حالت اوليه است.

در بانك‌هاي اطلاعات XML و به‌خصوص بانك‌هاي نامتمركز، اغلب به دليل ذخيره‌سازي اطلاعات به فرمت رشته، عمل رمزنگاري داده‌ها جهت انتقال و همچنين ذخيره‌سازي بسيار الزامي مي‌باشد. در اينجا ما به بحث رمزنگاري وارد نمي‌شويم و فقط روش‌هاي رمز نگاري را كه به دو دسته كلي متقارن و نامتقارن تقسيم مي‌شوند را اجمالاً بيان مي‌كنيم.

* + **روش رمزنگاري متقارن:** در روش رمزنگاري متقارن، فرستنده و گيرنده از يك كليد سري مشترك براي رمزنگاري و رمزگشايي استفاده مي‌كنند. روشData Encryption Standard (DES) مثالي از رمز نگاري متقارن است. بديهي است كه براي دو طرف ناشناس توافق بر روي يك كليد سري مشترك دشوار است. بنابراين اين روش مورد استفاده كمتري دارد.
  + **روش رمزنگاري نامتقارن:** روش ديگر، روش رمزنگاري نامتقارن است. در اين روش هر فرد دو كليد در اختيار دارد، كليد عمومي كه آزادانه منتشر مي‌شود و كليد خصوصي كه به صورت خصوصي و كاملا محرمانه نگهداري مي‌شود. در اين روش، فرستنده داده‌ها را با كليد عمومي رمز كرده و به گيرنده ارسال مي‌كند. داده‌هاي رمز شده تنها با كليد خصوصي گيرنده قابل رمز گشايي هستند و از آنجاييكه كليد خصوصي هر شخص منحصر به فرد است و به طور كاملاً امن نگه داشته مي‌شود، شخص ديگري نمي‌تواند اين اطلاعات را رمزگشايي كرده و يا از آنها استفاده نمايد. اين روش از امنيت بالايي برخوردار است و مورد استفاده فراواني دارد. به‌عنوان نمونه الگوريتم‌‌هاي معروف RS2 ، MD4 و MD5 از اين روش استفاده مي‌كنند.

در قسمت بعدي مدل‌هاي كنترل دسترسي را بررسي خواهيم كرد. در اينجا فرض مي‌كنيم كه سنجش‌هاي اوليه براي تاييد اعتبار كاربر و همچنين ارسال مطمئن اطلاعات برروي شبكه انجام مي‌شود.

* 1. مدل‌هاي كنترل دسترسي در بانك اطلاعات XML

مديريت كنترل دسترسي يكي از اصول سيستم پايگاه داده‌ها مي‌باشد. مديريت كنترل دسترسي در بانك‌هاي اطلاعات توسط سرويس‌دهنده‌ها انجام مي‌پذيرد. كنترل دسترسي بنابر عوامل مختلف به سرعت تحول يافته است. مدل‌هاي كنترل دسترسي را مي‌توان به دو گروه سنتي و توسعه يافته تقسيم بندي نمود. در گروه سنتي دو مدل كنترل دسترسي محتاطانه[[17]](#footnote-17) و الزامي[[18]](#footnote-18) قرار دارند كه مدل‌هاي پايه و زيربنايي به حساب مي‌آيند. مدل‌هاي گروه توسعه يافته در واقع مدل‌هايي مي‌باشند كه بر پايه مدل‌هاي سنتي بنا شده و آنها را به نوعي توسعه داده و مشكلاتشان را رفع نموده‌اند. در ادامه به بيان اجمالي مدل‌هاي كنترل دسترسي مي‌پردازيم.

* + 1. مدل كنترل دسترسي محتاطانه

در اين مدل، دسترسي كاربران به بانك‌‌اطلاعات بر اساس هويت كاربران (ID) و قوانيني به نام اجازه ورود (مجوز) كنترل مي‌شود [11]. در اينجا براي هر كاربر (يا گروهي از كاربران) نوع دسترسي‌هاي آنها به هر شئ موجود در بانك مشخص مي‌شود. نوع اين اشياء بستگي به بانك اطلاعات دارد. در مدل رابطه‌اي، اشياء مي‌توانند رابطه‌ها (جداول)، ديدها، صفت‌ها و در صورت لزوم ركوردها باشند. در مدل شئ‌گرايي مي‌توانند شامل كلاس‌ها، اشيا و متدها باشند و در بانك اطلاعات XML مي‌توانند مستندات، مسيرها و عناصر باشند.

كنترل‌هاي قابل اجرا بر روي اشياء معمولاً در دو سطح صورت مي‌گيرد: يكي در سطح حساب كاربري كه امتياز توسط مدير بانك‌اطلاعات اعطا مي‌شود و ديگري در سطح اشياء كه امتياز اعطاء شده فقط براي بخشي از شئ خاص مي‌باشد. بر اين اساس معمولاً انواع مجوزهايي كه به يك كاربر داده مي‌شود به صورت زير است:

* در سطح داده‌ها:
  + *مجوز read :* اجازه خواندن داده‌ها را مي‌دهد ولي اجازه تغيير داده‌ها را نمي‌دهد.
  + *مجوز insert :* اجازه درج داده جديد را مي‌دهد.
  + *مجوز update :* اجازه تغيير داده‌ها را مي‌دهد.
  + *مجوز delete :* اجازه حذف داده‌ها را مي‌دهد.
* در سطح شِما:
  + *مجوز index*: اجازه ايجاد يا حذف شاخص‌ها را مي‌دهد.
  + *مجوز resource:* اجازه ايجاد رابطه‌هاي جديد را مي‌دهد.
  + *مجوز alteration*: اجازه اضافه يا حذف صفات رابطه را مي‌دهد.
  + *مجوز drop:* اجازه از بين بردن رابطه‌ها را مي‌دهد.

معمولاً يك مجوز به صورت يك سه تايي (subject, mode, object) نمايش داده مي‌شود كه بيان كننده روش دسترسي (mode) يك كاربر يا فرآيند (subject) به يك شئ ((object مي‌باشد. بنابراين هر درخواستي كه از طرف كاربر داده مي‌شود ابتدا مجوز آن بررسي شده و در صورت وجود سه تايي مربوطه، اجازه دسترسي داده مي‌شود. در اين مدل دو نوع خط‌مشئ تعريف مي‌شود:

1. *خط‌مشئ بسته:* در اين نوع خط‌مشئ دسترسي‌هايي اجازه داده مي‌شود كه مجوز صريح آنها موجود باشد و تصميم پيش فرض اين است كه دسترسي رد شود. اكثر سيستم‌ها از سياست‌هاي بسته پشتيباني مي‌كنند.
2. *خط‌مشئ باز:* در اين خط‌مشئ، مجوز منفي داده مي‌شود و در صورت وجود آن، اجازه دسترسي داده نمي‌شود و تصميم پيش فرض (در صورت عدم وجود مجوز منفي) اين است كه دسترسي قبول شود. سياست باز فقط در سيستم‌هايي استفاده مي‌شود كه حفاظت محدود نياز دارند و اكثر دسترسي‌ها اجازه داده مي‌شود.
   * 1. مدل كنترل دسترسي الزامي

براي حل مشكلات بالقوه موجود در روش كنترل دسترسي محتاطانه، از روش كنترل دسترسي الزامي استفاده مي‌شود [11]. معروف ترين مدل كنترل دسترسي الزامي، مدل Bell-LaPadula است. در اين مدل پديده‌هاي سيستم به چهار دسته زير مي‌شوند:

* *شئ*: پديده‌هاي فعالي كه اطلاعات را نگهداري مي‌كنند، مانند جداول، ركورها، صفت‌ها، ديدها، اشياء و کلاس‌ها.
* *فرايند*: پديده‌هايي كه در خواست هاي دسترسي به اشياء را مي‌دهند، مانند كاربران و برنامه‌ها.
* *كلاس‌هاي امنيتي*[[19]](#footnote-19): سطوح امنيتي دسترسي به هر شئ را مشخص مي‌كنند. سطوح امنيتي‌، به صورت‌هاي خيلي محرمانه (TS)، محرمانه (S)، سري (C)، طبقه بندي نشده (U) تعريف مي‌شوند و ترتيب آنها بصورت TS > S > C > U است.
* *حساسيت[[20]](#footnote-20):* حساسيت هر فرآيند در رابطه با يك كلاس امنيتي نمايش داده مي‌شود. در اين روش، كلاس دسترسي را به صورت زوج‌هاي مرتب از سطوح امنيتي و مجموعه گروه‌ها نمايش مي‌دهند (S,G1).

كلاس دسترسي C1 پايين تر از كلاس دسترسي C2 است (C1 >= C2) اگر و فقط اگر سطح امنيت C1 بزرگتر يا مساوي سطح امنيت C2 و گروه‌هاي C1 شامل همه گروه‌هاي C2 باشند. دو كلاس C1 و C2 را غير قابل مقايسه گويند اگر نه C1 >= C2 و نه C2 >= C1 برقرار باشد.

*سطح امنيت* از كلاس دسترسي يك شئ نشان دهنده حساسيت اطلاعات آن شئ مي‌باشد. همچنين نشان‌دهنده پتانسيل خرابي است كه از يك دسترسي غير مجاز به اطلاعات مي‌تواند نتيجه شود. سطح امنيت كلاس دسترسي يك كاربر نشان دهنده قابل اعتماد بودن كاربر است كه اطلاعات حساس را براي كاربران غير مجاز فاش نمي‌كند. كاربران با كلاس دسترسي مربوط به خود با سيستم ارتباط برقرار مي‌كنند و مي‌توانند تراكنشي با كلاس دسترسي خود يا كمتر از آن به سيستم بدهند. بطور نمونه كاربر با كلاس (S,0) مي‌تواند تراكنش‌هايي با كلاس‌هاي دسترسي (S,0) و (C,0) و (U,0) به سيستم بدهد.

مدل Bell-LaPadula دومحدوديت زير را در تمام دسترسي‌ها به اشياء بانك‌اطلاعات قائل است:

1. فرايند S اجازه دسترسي خواندن به شيء O را دارد اگر

Class(S) >= Class(O)

به عنوان مثال كاربر X با حساسيت TS، مي‌تواند جدولي با حساسيت C را بخواند. اما كاربر Y با حساسيت C، اجازه خواندن از جدولي با حساسيت TS را ندارد.

2. فرايند S اجازه نوشتن بر روي شيء O را دارد اگر

Class(O) >= Class(S)

ضعف اصلي سياست‌هاي الزامي دشواري كنترل آنها است، زيرا احتياج به تعريف و استفاده از طبقه بندي اشياء، كاربران و برنامه‌ها دارند. اين كار ممكن است هميشه ميسر نباشد. به علاوه دسترسي هاي تعيين شده، فقط بر اساس طبقه بندي اشياء و تراكنش‌هاي موجود در سيستم است و هيچ امكاني به كاربران براي واگذاري يا پس‌گرفتن مجوز به ديگر كاربران داده نمي‌شود. روشهاي ديگري پيشنهاد شده‌اند كه تركيبي از روش‌هاي معرفي شده قبلي است.

* + 1. مدل ليست كنترل دسترسي و مدل قابليت

مدل ليست كنترل دسترسي[[21]](#footnote-21) [2] و مدل قابليت[[22]](#footnote-22) [3] دو مدل قديمي براي كنترل دسترسي در زمان اجرا هستند. در مدل ليست كنترل دسترسي مجوزها براي اشياء تعيين مي‌گردند. در مقابل در مدل قابليت مجوزها براي موضوعاتي (كاربراني) كه قصد دسترسي به اشياء را دارند تعيين مي‌گردند.

نماد Rule(sub, act, obj) را براي توصيف قاعده كنترل دسترسي <sub, act, obj> استفاده مي‌كنند. فرض كنيد ACL(objk) ليست كنترل دسترسي براي شئ kام و Policy خط‌مشئ كنترل دسترسي شامل m شئ را نشان مي‌دهد. ACL(objk) و Policy به صورت زير بيان مي‌شوند:

ACL(objk):= {Rule(sub1, act1, objk), … , Rule(subn, actn, objk)}

Policy := {ACL(obj1), … , ACL(objm)}

كنترل دسترسي براي موضوع (كاربر) kام subk در مدل قابليت به‌صورت Cap(subk) نشان داده مي‌شود. Cap(subk) و Policy با s موضوع به‌صورت زير بيان مي‌شوند:

Cap(subk) := {Rule(subk, act1, obj1), … , Rule(subk, actn, objn)}

Policy := {Cap(sub1), … , Cap(subs)}

مدل‌هاي قابليت و ليست كنترل دسترسي مزايا و معايبي دارند. در ليست كنترل دسترسي، بررسي سريع مجوزها براي يك شئ امكان پذير است ولي براي به دست آوردن همه مجوزهاي يك كاربر، بايد تمام ليست‌هاي كنترل دسترسي بررسي شود. در روش قابليت، تعيين سريع امتيازهاي كاربر امكان پذير است ولي به دست آوردن همه دسترسي‌هاي ممكن به يك شئ نياز به بررسي كل قابليت‌هاي موجود دارد. همچنين اين روش‌ها در كارايي پس‌گرفتن مجوزها بر اساس حذف كاربر يا شئ تاثير مي‌گذارد.

اين دو مدل را مي‌توان به سادگي جهت بانك‌هاي XML استفاده كرد. به اين منظور كافيست كه هر سند و يا هر المان يا زير المان يك سند را يك شئ در نظر گرفته و بدين ترتيب خط‌مشئ مربوطه را ايجاد و كنترل دسترسي را اعمال كرد.

* + 1. مدل كنترل دسترسي مبتني بر نقش

در زمينه مديريت كنترل دسترسي به بانك‌هاي اطلاعات يا ديگر مجموعه داده‌هاي ساختيافته، علاوه‌بر مدل‌هاي سنتي (مدل‌هاي كنترل دستيابي محتاطانه و الزامي)، مدل كنترل دسترسي مبتني بر نقش[[23]](#footnote-23) نيز اضافه شده است [4،29]. مدل‌هاي مبتني بر نقش براي شبيه‌سازي ديگر مدل‌ها استفاده شده است. اين مدل همچنين الگوهاي ‌اختياري بيشتري را براي دسترسي به داده‌ها در اختيار مدير سيستم قرار مي‌دهد [30، 31].

مدل كنترل دسترسي مبتني بر نقش اولين بار توسط آقاي رابيتي و همكارانش جهت استفاده در بانك‌هاي اطلاعات شئ‌گرا مطرح گرديد [12]. اين مدل به راحتي قابل انطباق به مستندات XML و همچنين بانك اطلاعات XML مي‌باشد.

* مدل گراف نقش

مدل گراف نقش [13] يك مدل كنترل دسترسي عمومي است كه براي مديريت آسان واگذاري مجوزها به كاربران استفاده مي‌شود. اين گراف عموماً هنگامي استفاده مي‌شود كه تعداد كاربران و مجوزها بسيار زياد مي‌باشد. اين مدل مجوزها (امتياز‌ها) را براي نقش‌ها گروه‌بندي مي‌كند. همچنين مدل گراف نقش يك مدل نمودار گروه را ايجاد مي‌كند. اين مدل به كاربرها اجازه مي‌دهد به گروه‌هايي تقسيم شوند [14]. بنابراين، تخصيص دادن يك گروه به يك نقش اين اجازه را مي‌دهد كه هر كس تعداد اختياري از كاربران را به يك مجموعه بزرگ اختياري از امتيازها با يك عمل نسبت دهد. مدل گراف نقش قابليت‌هاي مشابهي با مدل‌هاي كنترل دسترسي مبتني بر نقش دارد [15، 35].

مدل گراف نقش كنترل دسترسي را در سه سطح مورد توجه قرار مي‌دهد: سطح مركزي تشكيل شده است از گراف نقشي كه نمايانگر روابط نقش- نقش بعنوان يك گراف نقش است؛ دو سطح ديگر گروه‌ها و امتيازها را توصيف مي‌كنند. گره‌ها در گراف نقش نمايانگر نقش‌ها هستند. لبه‌ها نمايانگر رابطه كوچكتري (is-junior) بين دو نقش مي‌باشند. يك نقش در واقع يك جفت مرتب شامل نام و مجموعه‌اي از امتيازهاي موثر مي‌باشد. نقش *ri* كوچكتر از نقش *rj* است اگر مجموعه امتياز موثر ri يك زير مجموعه از مجموعه امتياز موثر rj باشد. هر امتياز به صورت يك زوج مرتب (حالت دسترسي ، شئ) نشان داده مي‌شود. طبيعت شئ و حالت دسترسي به محيط بستگي دارد. براي مثال در يك بانك اطلاعات رابطه‌اي، اشياء ارتباطات و حالت‌هاي دسترسي عمل‌هاي مجاز درج، بروزرساني، حذف و غيره مي‌باشند. در محيط‌هاي پيچيده، بين امتيازها مي‌تواند مفهوم و معني خاصي برقرار باشد؛ به عنوان مثال امتيازي كه اجازه خواندن كل يك رابطه را مي‌دهد به طور ضمني اجازه خواندن تعدادي از تاپل‌ها را نيز مي‌دهد. در ادامه مطالب بيشتري براي بيان اين مفاهيم خواهيم داشت.

نقش‌ها در يك نمودار نقش بين دو نقش متمايز قرار گرفته‌اند: MinRole و MaxRole. نقش MaxRole تمام امتيازها در نمودار نقش را نمايش مي‌دهد و لازم نيست به كاربر يا گروهي نسبت داده شود. نقش MinRole نيز كمترين امتيازها را كه به هر كس در سيستم نسبت داده شده است را نشان مي‌دهد. مابين امتيازهاي مستقيم كه مستقيماً به يك نقش نسبت داده مي‌شوند و امتيازهاي مؤثر كه شامل امتيازهاي مستقيم و ارث‌برده شده از نقش‌هاي كوچكتر[[24]](#footnote-24) مي‌باشند، تمايز قايل مي‌شويم. امتيازهاي موثر براي نقش *r* توسط *r.rpset* علامت‌گذاري مي‌شود.

گراف‌هاي نقش مشخصه‌ها و صفات زير را دارند:

* داراي يك نقش MinRole است.
* داراي يك نقش MaxRole است.
* گراف‌هاي نقش داراي حلقه و دور نمي‌باشند.
* يك مسير از نقش MinRole به هر نقش *ri* وجود دارد.
* يك مسير از هر نقش *ri* به نقش MaxRole وجود دارد.
* براي هر دو نقش *ri* و *rj* ، اگر  آنگاه بايد مسيري از *ri* به *rj* وجود داشته باشد.

به طور قراردادي، ما نمودار (گراف) را به‌اين صورت رسم مي‌كنيم كه نقش MaxRole در بالاترين سطح و نقش MinRole در پايين‌ترين سطح و نقش‌هاي كوچكتر در سطحي پايين‌تر از نقش‌هاي بزرگتر قرار مي‌گيرند. براي اينكه حجم گراف كوچكتر و خواناتر گردد، لبه‌هاي گذرا را حذف مي‌كنيم.

عمليات قابل دسترس در گراف نقش در [13] فهرست شده‌اند. اين عمليات شامل اضافه و حذف كردن يك نقش، يك لبه و يك امتياز به/ از يك نقش مي‌باشند. اگر يك چرخه در گراف ايجاد گردد تمامي عمليات انجام شده لغو مي‌گردد. زمان اجراي همة اين موارد توسط يك چند جمله‌اي كه به اندازه گراف و مجموعه‌هاي امتياز بستگي دارد، محاسبه مي‌گردد. در شكل 2-4 نمونه‌اي از يك گراف نقش نشان داده شده است. اين يك گراف نقش براي اجراي مثال مرتبط با دانشجويان در يك دانشگاه مي‌باشد.

MinRole

MaxRole

UndergradChair

Admissions

Instructor

AcademicCounsellor

FinancialAid

UnderGradStudent

مثالي از يك گراف نقش

مدل گراف اجازه مي‌دهد كه مجموعه‌اي از كاربران را گروه‌بندي كنيم (گراف گروه‌بندي) [14]. به جهت سادگي مدل، هر كاربر منحصر بفرد به عنوان يك گروه با درجه بزرگي 1 مي‌باشد. به عنوان مثال، در اينجا ممكن است شخص منحصر بفردي مثل آليس كه استاد در مقطع كارشناسي مي‌باشد به نقش UnderGradChair نسبت داده شود. همچنين ممكن است كميته پذيرش شامل آليس و ديگر اعضاء هيئت علمي و يك دانشجو باشد. اين كاربرها مي‌توانند در يك گروه در نمودار گروه‌بندي قرار گيرند و گروه مي‌تواند به نقش Admissions نسبت داده شود.

لبه‌ها در گراف گروه‌بندي توسط رابطه زير مجموعه بين دو گروه تعيين مي‌شوند. نقش كاربر[[25]](#footnote-25) و يا نقش گروه[[26]](#footnote-26) هنگامي واگذار مي‌گردد كه يك كاربر يا گروه به يك نقش نسبت داده شود. در [14] مدل‌سازي گروه‌ها با يك رويكرد مبتني بر شئ انجام شده است چنان‌كه گروه‌ها مي‌توانند مشخصه‌هايي داشته باشند كه ممكن است در واگذاري نقش گروه استفاده شوند.

* مدل مجوز بانك اطلاعات شئ‌گرا

مدل ارائه شده توسط رابيتي و همكارانش [12] يك مدل كنترل دسترسي محتاطانه براي بانك‌هاي اطلاعات شئ‌گرا مي‌باشد. او مجوز را با سه عنصر مدل مي‌كند:



كه S نشاندهنده مجموعه‌اي از موضوعات (در اينجا منظور از موضوع، كاربر مي‌باشد)، O مجموعه‌اي از اشياء و A اختيارات و مجوزهاي ممكن (حالت‌هاي دسترسي) در سيستم مي‌باشند. براي هركدام از S، O و A ها، يك شبكه ايجاد مي‌گردد. شبكه موضوعات يك نمودار نقش مي‌باشد كه يك شبكه نقش ناميده مي‌شود.

نمودار نقش ارائه شده توسط رابيتي خيلي شبيه به مدل نمودار نقش مي‌باشد. داشتن يك نقش super-user كه مرتبط با MaxRol در مدل نمودار نقش و يك حداقل نقش كه مرتبط با MinRole مي‌باشد.

با توجه به قسمت A از يك مجوز، رابيتي و همكارانش يك نمودار نوع مجوز (ATG) را تهيه كردند كه اجازه مي‌دهد مفاهيم مجوزها و يا عمليات سيستم را مدل كند. براي مثال، ممكن است يك سيستم داشته باشيم كه هميشه مجوز نوشتن روي يك شئ O را دارد و به طور ضمني بيان بكند مجوز خواندن همان شئ O را هنگامي كه اين عمليات مولفة A از يك مجوز را مي‌سازد. در اين صورت، لبة write → read بايستي در نمودار نوع مجوز باشد. ما اين ارتباط را ضمني مي‌ناميم.

در قسمت O از مجوز، رابيتي و همكارانش دو نمودار را مورد بحث قرار دادند: شماي مجوز شئ (AOS) كه دانه‌بندي‌هاي[[27]](#footnote-27) ممكن تعريف شده توسط شما براي بانك اطلاعات شئ‌گرا را مورد بررسي قرار مي‌دهد، و گراف مجوز شئ (AOG) كه نمونه‌هاي شئ واقعي در بانك اطلاعات را در نظر مي‌گيرد. لبه‌ها در AOG يك رابطه شمول[[28]](#footnote-28) را نشان مي‌دهند. بسته به اينكه OODB چگونه مجموعه‌هاي اشياء را اداره مي‌كند ممكن است با گره‌هايي در هر دو نمودار AOS و AOG و با مجموعه‌هايي از اشياء از يك نوع سروكار داشته باشيم. براي مثال AOS در يك سيستم بانك اطلاعات دانشجويي، يك گره براي تعريف كلاس از يك كلاس دانشجو دارد، همچنين مجموعه‌اي از همة نمونه‌هاي دانشجو. در AOG، يك گره براي تعريف كلاس و يكي براي جمع آوري وجود دارد و احتمالاً گره‌هاي بسياري براي نمونه‌هايي از كلاس دانشجو. (كلاس در زمان اجرا مي‌بايست در دسترس باشد تا با خواندن آن ساختار شئ قابل فهم باشد).

علاوه بر اين سه نمودار، مدل رابيتي يك ماتريس به نام ماتريس وابستگي مجوز[[29]](#footnote-29) (AAM) دارد كه نشان مي‌دهد آيا زوج مرتب (نوع دسترسي ، نوع شئ) معتبر مي‌باشد يا خير. براي مثال، عمل ايجاد نمونه از روي كلاس تنها براي نمايش دادن تعريف‌هاي كلاس اشياء مفهوم دارد و براي نمايش دادن اتصال‌ها مثل يك شئ دانشجو، مفهوم ندارد.

مدل رابيتي همچنين شامل يك راه متمايز جهت انتشار دسترسي‌ها در گراف مجوز شئ (AOG) مي‌باشد. براي مثال، امتياز خواندن يك شئ در يك بانك اطلاعات شئ‌گرا بطور ضمني امتياز خواندن مقادير صفت‌هاي آن شئ را مي‌دهد. چنين عملي را *a.down* مي‌ناميم. بعضي از امتيازها، مثل قابليت اعطاي يك امتياز روي يك شئ نبايد تكثير و انتشار يابد و a.nil مي‌باشد. ديگر عمليات مثل قابليت خواندن شِما، به سمت بالا در AOG انتشار مي‌يابد. انواع مجوز بدين گونه به سه مجموعه تفكيك شده‌اند: *a.up* ، *a.down* و *a.nil* كه بستگي به اين دارد كه چطور در گراف مجوز شئ انتشار مي‌يابد. اين موضوع نيز بايد توسط طراح امنيت همچون بخشي از مجموعه مشخصات يك سيستم تعيين گردد.

مدل رابيتي علاوه بر مجوزهاي منفي، مفاهيم مجوزهاي قوي و ضعيف را نيز دارد. مدل گراف نقش تنها امتيازهاي مثبت را دارد. در پياده‌سازي براي مستندات XML مفهوم مجوز قوي و ضعيف مشابه مدل رابيتي نداريم.

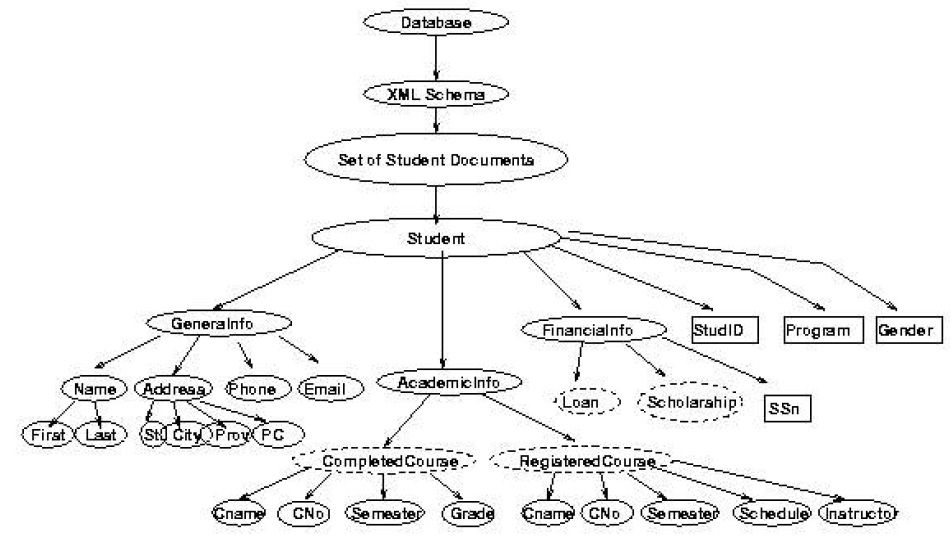
* ارائه مدل براي مستندات XML

امتيازها در بانك اطلاعات XML از يك شئ و يك حالت دسترسي (خواندن، نوشتن و ...) تشكيل شده‌اند. در اينجا مستندات XML با يك شماي XML تعريف شده است كه بخش شئ يك امتياز مي‌تواند هر بخشي از بانك اطلاعات XML باشد، از يك مشخصه انفرادي يا مقدار المان تا مجموعه‌اي از مستندات، مجموعه‌اي از المانها و يا مجموعه‌اي از زير المانها و غيره.

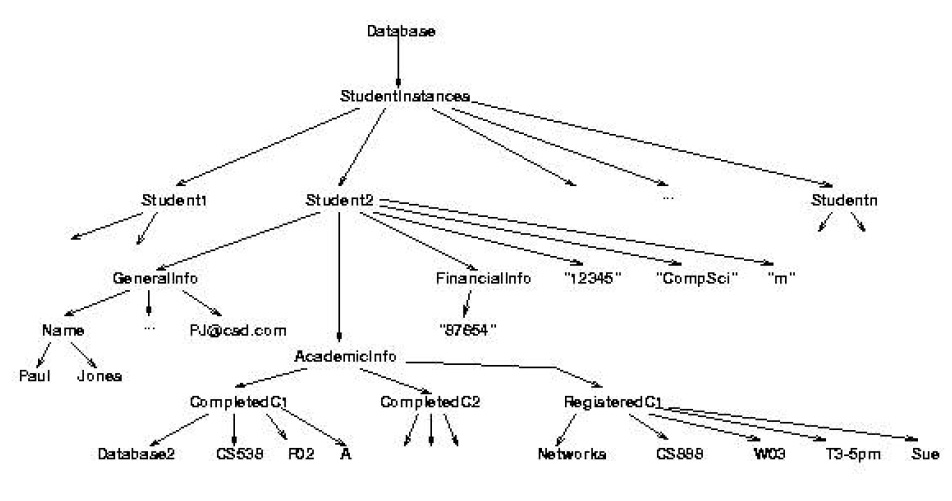
* اشياء

شماي XML، مستنداتي با ساختار سلسله‌مراتبي شامل صفات و المانها را تعريف مي‌كند. المانها مي‌توانند 0 و يا چندين بار در المان پدرشان ظاهر شوند. صفات مي‌توانند حداكثر يك بار ظاهر مي‌شوند. المانها مي‌توانند زيرالمانهاي تودرتو تا هر سطحي داشته ‌باشند. مرجع‌ها و نشانه‌هاي بين المانها نيز مي‌توانند تعريف شوند. شئ‌هايي كه كنترل دسترسي را بر روي آن‌ها اعمال مي‌كنند شامل مستنداتي هستند كه با يك شماي XML مطابقت دارند. شماي XML مي‌تواند براي ساخت AOS در مدل رابيتي بكار رود. مستندات XML، AOG را مي‌سازند. يك AOS و AOG نمونه در شكل 2-5 و 2-6 نشان داده ‌شده‌اند.

در AOS شكل 2-5، صفات را با مستطيل، المانها با بيضي و المانهايي كه اجباري به رخ دادن و ظاهر شدن آنها حتي يك بار نيست را به وسيله بيضي با خطهاي منقطع در المان پدرشان مشخص كرده‌ايم. در AOG مقادير و ارزش صفات را در "" قرار مي‌دهند و مقادير (ارزش) المانها را برابر با مقدار رشته‌اي خودشان بدون برچسب نشان مي‌دهند.



شماي مجوز شئ براي مثال نمونه



گراف مجوز شئ براي مثال نمونه

بخش شئ يك امتياز به وسيله يك عبارت درXPath مشخص خواهد شد. عبارات XPath مي‌توانند به صورت المانهاي منحصربفرد يا زيرالمانهايي نظير يك دانشجو يا مجموعه‌اي از المانها نظير تمامي نام‌هاي دانشجويان مشخص گردند.اين عبارات بصورت زير بيان مي‌شوند:

//Student[@StudentID = "12345"]

//FirstName

* حالت‌هاي دسترسي

معمولاً حالتهاي دسترسي براي مجموعه‌اي از مستندات XML در ساير كارها بسيار محدود مي‌باشد. براي مثال در [16]، عمل‌هاي مورد توجه تنها مروركردن[[30]](#footnote-30) (مثلاً خواندن) و تاليف كردن[[31]](#footnote-31) مي‌باشند. تأليف شامل تغييراتي در متن موجود بعلاوه افزودن به تعداد المانهاي مجموعه است. عمده كار ما با خواندن و فقط نوشتن در انتهاي مستند مي‌باشد. هر كسي ممكن است عمليات متعددي را بر روي برخي قسمت‌ها و يا تمامي بخش‌هاي مستندات XML متصور شود؛ مثل خواندن، تغيير در محتوي (به‌روز رساني)، اضافه كردن زير المان‌ها يا صفات (توسعه سند) و ديگر عمليات ممكن. در مثالي كه در ادامه ذكر خواهد شد فقط خواندن، به‌روز رساني و توسعه را در نظر گرفته‌ايم. در شكل 2-7 نمودار نوع مجوز (ATG) براي اين سه عمل نشان داده شده است.

Extend

Update

Read

نمودار نوع مجوز براي مثال نمونه ذكر شده

لبة كشيده شده از Extend به Update به اين معني است كه گرفتن امتياز Extend براي يك شئ O همان نقشي را كه امتياز Update براي وي دارد، بطور ضمني به او مي‌دهد. معمولاً هر كسي مي‌تواند چندين حالت دسترسي متفاوت كه براي يك نرم‌افزار در نمودار نوع مجوز معين شده است را دارا باشد. اين سه حالت دسترسي به عمل a.down تعلق دارند.

با درنظر گرفتن نمودار نوع مجوز و شماي مجوز شئ ، ماتريس وابستگي مجوز كه ارتباط بين عمليات و انواع داده‌ها را بيان مي‌كند، بدست مي‌آيد. اساساً هر كدام از برگ‌ها در شماي مجوز شئ كه زير المان قابل توسعه نداشته باشد، نمي‌تواند حالت دسترسي توسعه تعريف شده در آنها را بگيرد. به عبارت ديگر صفات نمي‌توانند توسعه يابند. ماتريس وابستگي مجوز براي مثال نمونه در جدول 2-1 نشان داده شده است.

ماتريس وابستگي مجوز براي مثال نمونه

|  |  |  |  |
| --- | --- | --- | --- |
| **granule** | **read** | **update** | **extend** |
| SetOfStudentDocuments, Student, GeneralInfo, AcademicInfo, FinancialInfo, Name, Address, CompletedCourse, RegisteredCourse | T | T | T |
| First, Last, St, City, Prov, PC, Cname, CNo, Semester, Grade, Schedule, Instructor, Loan, Scholarship, SSN, StudID, Program, Gender | T | T | F |

* انتشار امتيازها

در اينجا به عنوان نمونه نقش AcademicCounsellor را در شكل 2-4 در نظر مي‌گيريم. كاربران اين نقش بايد قادر باشند كه بسياري از داده‌هاي مربوط به دانشجويان همچون همة داده‌هاي GeneralInfo ، CompletedCourses و صفت‌ها را مشاهده كنند. آنها بايد بتوانند كارنامه دانشجويان در دوره‌هاي مختلف را براي توسعه RegisteredCourses بگيرند. از طرفي اين كاربران نبايد اطلاعات مالي دانشجويان را بخوانند و امكان تغيير و يا توسعه در اطلاعات دانشجويان داشته باشند. همة مجوزها در هر گره AOS مي‌تواند تعيين شود، ولي مي‌توانيم از انتشار مجوزها براي كاهش امتيازاتي كه عملاً تعريف خواهد شد، استفاده كنيم. الگوريتم انتشار مي‌تواند از AOG (و محدوديت‌هاي مطرح شده در قسمت بعد) براي قراردادن همة امتيازات مرتبط در يك نقش، استفاده كند. الگوريتم گراف نقش همچنين متقاعد مي‌كند كه هر نقش بزرگتر (به عنوان مثال MaxRole و UnderGradChair) اين مجوزها را همچون مجوزهاي مؤثر ارث مي‌برد.

* محدوديت‌ها

محدوديت‌ها در اينجا شامل راه‌هايي براي متوقف كردن انتشار مجوزها به زير المان‌ها مي‌باشد. در اينجا متذكر مي‌شويم كه اين محدوديت‌ها شبيه مجوزهاي منفي در مدل رابيتي عمل مي‌كنند. مجدداً نقش AcademicCounsellor را در شكل 2-4 در نظر مي‌گيريم. كاربرها به اين نقش اختصاص مي‌يابند كه نه‌تنها اجازه ندارند CompletedCourses را توسعه دهند، بلكه قادر به خواندن اطلاعات مالي FinancialInfo نيز نيستند. اگر محدوديتي را مشخص كنيم كه كاربران نتوانند اطلاعات مالي را بخوانند، از آنجا كه بروزرساني بطور ضمني مجوز خواندن در ATG را مي‌دهد، نفي امتياز خواندن بطور ضمني امتياز بروزرساني و همچنين امتياز توسعه را نفي مي‌كند. به عبارت ديگر، هنگامي‌كه محدوديت‌ها تعيين مي‌شوند، آنها در جهت خلاف لبه‌ها در گراف ATG انتشار مي‌يابند.

نكته قابل توجه ديگر در مورد محدوديت‌ها اينست كه آنها براي هر نقش مشخص شده‌اند. براي مثال نقش FinancialAid مي‌تواند FinancialInfo را براي دانشجويان توسعه دهد، امّا نمي‌تواند AcademicInfo را بخواند (و بنابراين نمي‌تواند بروزرساني و يا توسعه دهد). مجموعه محدوديت‌ها براي نقش *r* را با *Cr* نشان مي‌دهيم. محدوديت‌ها در *Cr* به همان روش مجوزها نشان داده مي‌شوند. آنها شامل يك زوج مرتب (o, a) هستند كه o يك شئ تعريف شده و a يك حالت دسترسي معتبر در گراف ATG مي‌باشد. محدوديت‌ها بر مجوزها اولويت دارند، خواه تعيين شوند و خواه بوسيله انتشار ايجاد گردند.

شكل 2-8 نسخة XML از الگوريتم انتشار را نشان مي‌دهد [13]. اين الگوريتم همة مجوزهاي ضمني را با در نظر گرفتن AOS، AOG، ATG، AAM، يك مجوز، يك نقش r، و محدوديت‌هاي *Cr* توليد مي‌كند. مجوزهاي توليد شده همة مجوزهاي مثبت هستند و همچون مدل گراف نقش اجازه وجود مجوز منفي در يك نقش را نمي‌دهد. بخصوص اين نسخه فقط انتشار رو به پايين در AOG را در نظر مي‌گيرد. الگوريتم دو مجموعه را مي‌سازد: مجموعه مجوزهاي ضمني كه توسط يك گره معين ايجاد مي‌شود و مجموعه مجوزهاي ضمني كه توسط محدوديت‌ها ايجاد مي‌شود (شبيه مجوزهاي منفي عمل مي‌كند).

**Algorithm:** PrivPropagation(RG, *r*, *p*, ATG, AOG, AAM, *Cr*)

Input: RG = 〈R, →〉 (the role graph),

*r*, /\* the role to which a privilege is to be added \*/

*p*, /\* the privilege, (a, o), to be added to role *r* \*/

ATG, /\* the authorization type graph \*/

AOG, /\* the authorization object graph \*/

AAM, /\* the authorization association matrix \*/

*Cr* /\* the constraints for role *r* \*/

Output: SetPri, the set of all privileges implied by *p*,

except for those prohibited by constraints in *Cr*.

Note: This algorithm assumes that all access modes are in a.down.

Method:

begin

if *p* ∈ Effective(*r*) /\* *p* already in *r* – do nothing \*/

then return;

SetPri ← {(a, o)}

while changes are made to SetPri do

(ak, ok) ← a privilege from SetPri

/\* using the ATG \*/

for all ai ∈ {ak} ∪ {aj |ak implies aj}

if (ai, ok) ∉ SetPri and (ai, ok) is OK by AAM

then SetPri ← SetPri ∪ {(ai, ok)}

/\* using the AOG \*/

for all oi ∈ {ok} ∪ {oj |ok contains oj}

if (ak, oi) ∉ SetPri and (ak, oi) is OK by AAM

then SetPri ← SetPri ∪ {(ak, oi)}

SetCons ← *Cr*

while changes are made to SetCons do

(ak, ok) a privilege from SetCons

/\* using the ATG \*/

for all ai ∈ {ak} ∪ {aj |aj implies ak}

if (ai, ok) ∉ SetCons

then SetCons ← SetCons ∪ {(ai, ok)}

/\* using the AOG \*/

for all oi ∈ {ok} ∪ {oj |ok contains oj}

if (ak, oi) ∉ SetCons

then SetCons ← SetCons ∪ {(ak, oi)}

SetPri ← SetPri \ SetCons

end.

الگوريتم انتشار مجوز

* مشكلات و نواقص مدل براي مستندات XML

در روش مدل گراف نقش معمولاً مجوزها در شرايطي از شئ خاص تعيين مي‌شوند. مكانيزمي براي تطبيق داده‌هاي دانشجو با كاربر دانشجو در زمان اجرا وجود ندارد. يك پيشنهاد كه اغلب در چنين شرايطي داده مي‌شود نقش اختصاصي براي هر دانشجو و هر استاد مي‌باشد كه مجوزهاي پايه را براي دانشجو يا استاد واقعي (وابسته به اشياء واقعي كه كنترل مي‌شوند) ارث مي‌برد. حال آنكه اين راه حل بصورت تئوريك قابل قبول مي‌باشد، اما در عمل چنين نيست. از آنجا كه در يك دانشگاه به عنوان نمونه 1600 استاد و بالغ بر 25000 دانشجو وجود دارد، يك گراف نقش با اين همه نقش خصوصي امكان‌پذير نيست. يك راه حل امتحان شده اينست كه نقش‌هاي پارامتريك داشته باشيم كه بعضي از صفات كاربر (به عنوان مثال شماره دانشجو) مي‌تواند با صفات شئ مربوطه براي اختصاصي كردن مجوزهاي قابل دسترس در نقش كاربر واگذار شده، تطبيق داده شود. به هر حال يكي از معايب اين مدل پيچيدگي گراف نقش مي‌باشد كه نيازمند تحقيقات جديد در بهبود دادن به آن مي‌باشد.

جنبه ديگر اين مبحث كه بايد بدان توجه نمود اين است كه چگونه حضور محدوديت‌ها ممكن است گراف نقش را تغيير دهد. مشخصات گراف نقش بعد از هر عمل تجديد نظر مي‌شود. اگر به عنوان مثال گفته شود كه نقش Admissions محدوديت بيشتري نسبت به نقش FinancialAid دارد و همچنين محدوديت (//GeneralInfo/Name, read) را دارد، سپس بعضي از مجوزها كه بصورت طبيعي ارث برده شده‌اند بايد توسط اين محدوديت پذيرفته نشوند. در مدل گراف نقش ذكر شده در شكل 2-4، اگر مجوزهاي نقش FinancialAid مناسب زيرمجموعه‌اي از نقش Admissions نباشند، بايد حذف گردند. بنابراين نياز است كه طراحان گراف نقش بسيار دقيق محدوديت‌ها را مشخص نمايند.

* + 1. مدل كنترل دسترسي مبتني بر تابع

مدل‌هاي كنترل دسترسي محتاطانه، ليست كنترل دسترسي و كنترل دسترسي مبتني بر نقش، كنترل دسترسي رسا و با معني را برروي مستندات XML فراهم مي‌نمايند. اين رويكردها معمولاً اعطا و لغو مشخصه‌هاي كنترل دسترسي، مكانيزم انتشار كه به‌وسيله آن عنصرهاي توليد شده احكام را از پدرشان به ارث مي‌برند، و روش‌هاي حل برخورد داده‌ها به علت همپوشاني را توسط كنترل دسترسي چندگانه پشتيباني مي‌كنند [16، 32]. از آنجا كه اين مدل‌ها كنترل دسترسي را بوسيله پيمايش مستندات XML در زمان اجرا انجام مي‌دهند، اجراي اين مدل‌ها هزينه‌هاي محاسبات سنگيني را تحميل مي‌كنند، بخصوص براي مستندات XML داراي لايه‌هاي بسيار زياد با قواعد حجيم كنترل دسترسي با معني.

ايده ايجاد كنترل دسترسي رسا و با معني در [7 ، 17] پيشنهاد شده است. اين روش‌ها براي جستجوي گره‌هاي دسترسي كنترل شده يا رفع چك كردن دسترس‌پذيري غير ضروري در زمان اجرا مؤثر هستند. اين تحقيقات تلاش مي‌كنند تا كارائي كنترل دسترسي رسا را بهبود بخشند. امّا از آنجا كه معمولاً روي بهينه‌سازي مبتني بر مستندات تمركز دارند [33]، بانك‌هاي اطلاعات XML با تكرار بروزرساني مستندات يا قواعد كنترل دسترسي ممكن است متحمل هزينه‌هاي غير قابل قبول شوند.

مدل كنترل دسترسي مبتني بر تابع يك مدل رسا و با معني مي‌باشد. اين مدل قابل كاربرد در مدل‌هاي كنترل دسترسي موجود براي مستندات XML مي‌باشد. نوآوري اين مدل كنترل دسترسي در مقياس‌پذيري و كارائي بالا مي‌باشد. ايده كليدي در اين مدل كد كردن قواعد كنترل دسترسي و همچنين يك مجموعه از توابع دستور كه جداگانه ارزيابي دسترسي واقعي را اجرا مي‌كنند، مي‌باشد. علاوه براين، توابع دستور براي نگهداري در حافظه و تسهيل در بروزرساني گروه‌بندي شده‌اند.

* خط‌مشئ كنترل دسترسي در مدل مبتني بر تابع

در مدل كنترل دسترسي مبتني بر تابع خط‌مشئ كنترل دسترسي شامل يك مجموعه از قواعد 3 تايي با ساختار < شئ ، عمل مجوز ، موضوع > (<Subject, Permission Action, Object>) مي‌باشد [18].

موضوع يك پيشوند دارد كه به نوع آن اشاره مي‌كند، همچون شناسه شخص (uid)، نقش (role) و گروه (group). علامت "+" براي قاعده اعطاء و علامت "-" براي قاعده لغو بكار مي‌رود. مقدار هر عمل مي‌تواند يكي از عمليات خواندن (r يا R)، بروزرساني، ايجاد و يا حذف كردن باشد. قاعده‌اي با +R يا –R مجوزهاي خواندن (اعطاء يا لغو) را به زير شاخه‌هاي گره‌اي كه انتخاب شده است منتشر مي‌كند. به عنوان مثال (uid:Ali , +r , /a) مشخص مي‌كند كه كاربر Ali اجازه دسترسي به مسير المان /a را دارد. اما دسترسي به /a/b به‌طور ضمني تا زماني كه اجازه انتشار به شاخه‌هاي پايين /a داده نشده لغو گرديده است. علاوه بر اين، طبق اصل سازگاري لغو رو به پايين كه تصريح مي‌كند كه فرزندان يك گره غير قابل دسترس، غير قابل دسترس مي‌باشند، يك وابستگي دسترسي بين پدران و فرزندان وجود دارد. بنابراين آشكار است كه –r و –R معادل هستند. علاوه براين براي بيشينه كردن امنيت داده‌ها 1) ناسازگاري و تضاد دسترسي را با اصل اولويت گرفتن عدم پذيرش حل مي‌كنند؛   
2) عمل پيش فرض لغو مجوز را روي مسيرها كه كنترل دسترسي آشكار يا ضمني مشخص شده ندارند، بكار مي‌رود.

* كنترل دسترسي مبتني بر تابع

مدل كنترل دسترسي مبتني بر تابع شامل يك تابع دستور، يك جدول نگاشت ، نتايج ارزيابي و يك الگوريتم ارزيابي مي‌باشد [9]. در اين رويكرد، خط مشئ كنترل دسترسي در مجموعه‌اي از توابع دستور در يك زبان برنامه‌نويسي گنجانده شده است. توابع دستور شامل توابع ORF، SRF و GRF مي‌باشند.

* توابع دستور

اين توابع در واقع مولفه اصلي سيستم مي­باشند که کار بررسي مجوز دسترسي به عهده آنها است. در مدل مبتني بر تابع، سه نوع از توابع دستور را تعريف مي‌كنند:

تعريف: تابع دستور شئ محور (ORF) تابعي مي‌باشد که کنترل دسترسي روي يک مسير يا يک شي را انجام مي‌دهد. به اين صورت که در زمان اجرا اين تابع يک موضوع را به عنوان ورودي دريافت مي‌كند و مقداري را به عنوان خروجي بر مي‌گرداند که مشخص کننده حق دسترسي آن موضوع به مسير مي‌باشد.

تعريف: تابع دستور موضوع محور (SRF) تابعي مي‌باشد که براي يک موضوع ايجاد مي‌گردد. اين تابع يک شي را به عنوان ورودي دريافت مي‌كند و نتيجه ارزيابي را به فراخواننده بر مي‌گرداند. در اين تابع قوانين دستور مشخص شده براي آن شي قرار مي‌گيرند.

تعريف: تابع دستور عمومي (GRF) تابعي است كه روي شئ يا موضوع شاخص‌بندي نشده است. يك تابع GRF ممكن است هنگامي‌كه دسترسي تعيين مي‌گردد روي موضوع و يا شئ ايجاد گردد. هنگامي که يک قانون مربوط به چندين شئ باشد نمي‌توان آنرا در يک ORF قرار داد. لذا آنرا در تابع GRF قرار مي‌دهيم. به عنوان مثال وقتي شئ مربوط به قانون مسيري باشد که شامل // است، آن قانون در GRF پياده‌سازي مي‌شود. زيرا چنين قانوني مربوط به گره‌هايي است که ممکن است در سطوح مختلف فايل XML باشند. براي نمونه مسير /a//c مي‌تواند شامل گره‌هاي /a/c و گره‌هاي /a/b/c باشد.

مفهوم ORF به ACL نزديك است، درصورتيكه SRF مي‌تواند به رويكرد قابليت، نگاشت شود. بهرحال توابع دستور نتايج ارزيابي را برمي‌گردانند كه مدل‌هاي سنتي را به مدل كنترل دسترسي مبتني بر تابع توسعه مي‌دهند.

* ارزيابي قابليت دسترسي

هنگاميكه يک موضوع، درخواستي را براي دسترسي به يک مسير دارد، ابتدا سيستم دسترسي موضوع به مسير را ارزيابي مي‌کند و بر اساس آن مقدار اعطاء يا رد را بر مي‌گرداند. مقدار "اعطاء" مشخص مي‌کند که دسترسي موضوع به مسير درخواست شده بلامانع است و مقدار "رد" مشخص مي‌کند که موضوع اجازه دسترسي به مسير درخواستي را ندارد. چهار نوع از نتايج ارزيابي در جدول 2-2 نشان داده شده ‌است.

نتايج ارزيابي يك تابع دستور

|  |  |
| --- | --- |
| نتيجه ارزيابي | تاثير دسترسي |
| GRANT\_ON\_NODE | +r |
| GRANT\_ON\_SUBTREE | +R |
| DENY | -R |
| UNDECIDED | Nothing |

در جدول 2-2، مجوز مثبت توسط GRANT\_ON\_NODE و GRANT\_ON\_SUBTREE تعريف مي‌شود كه مجوزهاي دسترسي به زير شاخه‌ها را انتشار مي‌دهد. هنگامي كه دسترسي لغو مي‌گردد، نتايج ارزيابي فرزندان نيز DENY مي‌شود. به علت قانون سازگاري رو به پايين عدم پذيرش هنگامي كه مسير دستيابي شده توسط هيچ قانوني پوشانده نمي‌شود مقدار UNDECIDED برگردانده مي‌شود. به صورت پيش فرض اگر كنترل دسترسي اضافي تعريف شده باشد، دسترسي رد مي‌شود.

* جداول نگاشت

هنگامي که سيستم درخواستي را دريافت مي‌کند با استفاده از جدول نگاشت مشخص مي‌کند کدام تابع دستور براي ارزيابي دسترسي بايد انجام شود. بر حسب اينکه سيستم از مدل ORF يا SRF استفاده کند جداول نگاشت زير را خواهيم داشت:

* + جدول نگاشت ORF:

اين جدول براي هر مسير XPath مشخص مي‌کند کدام يك از توابع ORF بايد اجرا گردند. البته براي مشخص کردن توابع ذکر نام بسته و کلاسي که تابع در آن قرار دارد الزامي است.

* + جدول نگاشت SRF:

اين جدول براي هر موضوع مشخص مي‌کند که کدام يك از توابع SRF بايد اجرا گردند. در اينجا نيز براي مشخص کردن توابع ذکر نام بسته و کلاسي که تابع در آن قرار دارد الزامي است.

* الگوريتم ارزيابي

بر پايه اجزاء تعريف شده (توابع دستور، نتايج ارزيابي و جداول نگاشت)، مدل مبتني بر تابع يك نتيجه تصميم از يك درخواست دسترسي را با استفاده از الگوريتم‌هاي زير محاسبه مي‌كند. در اين رويكرد نياز داريم كه خط‌مشئ كنترل دسترسي ثابت باشد چنانكه تمامي اجداد يك فرزند با يك امتياز قابليت دسترسي مشخص شده، بايد در دسترس باشند.

* + الگوريتمORF

مسير و موضوع يك درخواست، با ورودي متناظر در جدول نگاشت ORF جستجو مي‌شود. اگر ورودي متناظر وجود داشته باشد، آنگاه تابع ORF پيدا شده با پارامتر موضوع فراخواني و اجرا مي‌شود و خروجي تابع نتيجه ارزيابي مي‌باشد.

در ارزيابي قابليت دسترسي، ابتدا يك تابع دستور عمومي GRF اجرا مي‌گردد. اگر GRF مقدار DENY را برگرداند، نتيجه تصميم قطعي شده است و ديگر روش ORF اجرا نمي‌شود. در غير اينصورت تابع ORF فراخواني و براي نتيجه تصميم اجرا مي‌گردد. براي مقدار UNDECIDEC بر حسب سياست سيستم مقدار "اعطاء" يا "رد" براي حق دسترسي موضوع به شئ در نظر گرفته مي‌شود.

* + الگوريتمSRF

الگوريتم SRF خيلي ساده‌تر از الگوريتم ORFاست. مسير و موضوع درخواست داده‌شده، با ورودي متناظر در جدول نگاشت SRF جستجو مي‌شود. اگر ورودي متناظر وجود داشته باشد، تابع SRF با آن نام پيدا شده در جدول احضار و اجرا مي‌شود. در غير اينصورت نتيجه تصميم بصورت پيش فرض DENY مي‌باشد.

* مقايسه مدل مبتني بر تابع با مدل‌هاي ليست كنترل دسترسي و قابليت

مدل كنترل دسترسي مبتني برتابع از سه جنبه با مدل‌هاي ليست كنترل دسترسي (ACL) و قابليت تفاوت دارد. اولاً در اين مدل مفهوم مولفه جديد GRF را داريم. تابع GRF براي اداره قواعدي كه نمي‌توانند به يك ORF خاص و يا يك SRF خاص محدود شوند، معرفي شده‌ است. فرض كنيد ORF(objk) نشان مي‌دهد ORF را براي قواعد اشتراكي objk و SRF(subj) نشان مي‌دهد SRF را براي قواعد تعريف شده جهت subj ، هنگاميكه 1 ≤ k ≤ m و 1 ≤ j ≤ s باشد يك خط مشئ كنترل دسترسي مي‌تواند به صورت زير تبديل شود:

Policy := {GRF, ORF(obj1), … , ORF(objk), … , ORF(objm)}

:= {GRF, SRF(sub1), … , SRF(subj), … , SRF(subs)}

در مدل خط‌مشئ دسترسي پذيرفته‌ايم كه موضوعات خاص هستند و بنابراين GRF در مدل مبتني بر SRF مورد نياز نمي‌باشد. اما تابع GRF در يك برنامه كاربرد بايد دسترسي‌ها از طريق يك آدرس را تشخيص دهد و قواعد كنترل دسترسي جهت دامنة خاص را ايجاد نمايد.

ثانياً مدل‌هاي سنتي ACL و قابليت بر روي اجراي كنترل دسترسي در مسيرهاي ساده تمركز دارند. در مقابل مدل‌هاي مبتني بر ORF و SRF توانائي ادارة كنترل‌هاي دسترسي درگير با // و گزاره‌ها را دارند كه معناي سيستم كنترل دسترسي را توسعه مي‌دهد.

ثالثاً بر خلاف مجوز مثبت در مدل‌هاي ACL و قابليت، توابع ORF و SRF هم مجوزهاي مثبت و هم منفي را پشتيباني مي‌كنند. در طول كنترل دسترسي، نتايج ارزيابي توسط توابع دستور كه براي محاسبه نتايج تصميم تركيب شده‌اند، برگردانده مي‌شوند. اگر يك مجوز باعث بروز اشكال شود، در حين تركيب اين مشكل حل مي‌شود.

اين سه نقطه قوت مشخص مي‌كنند كه مدل مبتني بر تابع مدل‌هاي سنتي ACL و قابليت را در جهت توسعه معني بهبود بخشيده‌اند.

* توليد تابع دستور

قواعد كنترل دسترسي قابل تبديل به گروهي از توابع ORF يا گروهي از توابع SRF هستند. خط‌مشئ كنترل دسترسي بر اساس قواعد اصلي ايجاد مي‌شود. به عنوان مثال (+r)، (+R)، (+R, +r) و(-R, +R, +r) كه (-R, +r) و (-R, +R) مي‌توانند بوسيله (-R, +R, +r) پوشانده شوند. از آنجائيكه براي يك موضوع چندين قاعده وجود دارد لذا نياز به رفع ابهام خواهد بود. با توجه به اصل اولويت گرفتن عدم پذيرش و مكانيزم انتشار، ترتيب اجراي قواعد تبديل شده در SRF مستقيماً نتايج قابليت دسترسي را تعيين مي‌كند.

فرض كنيم مسير P شاملK گره از n1 تا nk بوده و n1 گره ريشه مي‌باشد.

1) (+r): همه قواعد با+r مشخص شده‌اند. از آنجائيكه قابليت دسترسي وابستگي را تضمين مي‌كند، لذا فقط لازم است كه /n1/n2/…/nk چك شوند كه آيا با +r مشخص شده‌اند يا نه. در اينجا نحوه تبديل تصادفي مي‌باشد.

2) (+R): همه قواعد با +R مشخص شده‌اند. واضح است كه اگر يك يا چندتا از اجداد با +R مشخص شده باشند آنگاه دسترسي به فرزندان واگذار شده است. بنابراين ترتيب چك كردن اينكه آيا يك شئ جد مسيري كه درخواست شده‌است مي‌باشد يا خير، بصورت تصادفي مي‌باشد.

3) (+R, +r): قواعد يا با +R و يا با +r مشخص شده‌اند. فرض كنيد گره ni با +R و گره nj با +r مشخص شده‌اند و i < j < kهر دو گرة ni و njجد nk مي‌باشند. انتشار رو به پايين از ni باعث مي‌شود كه nk بدون ملاحظه ارتباط بين ni و nj داراي امتياز گردد. هنگاميكه i=k يا j=k باشد، واضح است كه nk امتياز را از ni گرفته است. بنابر اين هنگاميكه هم +R و هم +r در خط مشئ وجود داشته باشند، كد برنامه بايد اولين بررسي را براي +R انجام دهد. اعم از اينكه مسير درخواستي جد يكي از اين اشياء باشد يا خير. سپس كدي كه براي +r بررسي مي‌كند ادامه مي‌دهد. سپس از آن با تطبيق مراحل 1 و 2 ترتيب‌ها در هر گروه از +R يا +r مي‌توانند تصادفي باشند.

4) (-R, +R, +r): قواعد با -R ، +R و يا +r مشخص شده‌اند. بواسطه اصل اولويت گرفتن عدم پذيرش، ابتدا كدها براي -R مي‌آيند كه بررسي مي‌كنند آيا مسير درخواستي جد اين اشياء مي‌باشد يا خير. سپس بر پايه قسمت 3 با قوانين +R و سپس +r كار ادامه مي‌يابد.

نتيجه اينكه قواعد به كدهاي مجزا به ترتيب -R ، +R و سپس +r تبديل مي‌شوند.

* خط‌مشئ با مسير ساده

در مدل مبتني بر ORF، خط‌مشئ كنترل دسترسي در ابتدا بر اساس اشياء مرتب شده‌است. براي هر شئي و زير مجموعه قواعد متناظر با آن شئ، يك تابع ORF شامل كدي كه آن قواعد را پياده‌سازي مي‌كند، توليد شده است. در درون تابع ORF قوانين توسط موضوع متمايز شده‌اند و نتيجه ارزيابي مربوطه هنگاميكه موضوع با درخواست كننده تطبيق داشته باشد، در مقدار برگشتي قرار مي‌گيرد. به عنوان مثال، اگر زير مجموعه‌اي از خط‌مشئ كنترل دسترسي به نام P1 شامل Rule(subk, +r, /a) و Rule(subp, +R, /a) باشد، آنگاه قطعه كد جاوا از ORFبراي P1 به صورت نشان داده شده در شكل 2-9 مي‌باشد.

1 static public Integer ORF(String sub) {

2 if (sub.equals(“subk”)) // for (1)

3 return GRANT\_ON\_NODE;

4 else if (sub.equals(“subp”)) // for (2)

5 return GRANT\_ON\_SUBTREE;

6 else return UNDECIDED;

7 }

مثالي از ORF براي خط‌مشئ P1

مدل مبتني بر SRF متفاوت از مدل مبتني بر ORF مي‌باشد. در مدل مبتني بر SRF قواعد كنترل دسترسي بوسيله موضوع مرتب مي‌شود. يك SRF مرتبط با زير مجموعه‌اي از قواعد هر موضوع توليد مي‌شود. در درون تابع SRF، قواعد بوسيله شئ متمايز شده‌اند. نتيجه ارزيابي مربوطه زماني در مقدار برگشتي كد مي‌شود كه مسير دستيابي شده با وضعيت شئ مطابقت داشته باشد. شكل 2-10 تابع SRF را براي قواعد 1 و 2 نشان مي‌دهد.

// the SRF for subject subk

1 static public Integer SRF0(String path) {

2 if (path.equals(“/a”)) // for (1)

3 return GRANT\_ON\_NODE;

4 else return UNDECIDED;

5 }

// the SRF for subject subp

6 static public Integer SRF1(String path) {

7 if (path.startsWith(“/a”)) // for (2)

8 return GRANT\_ON\_SUBTREE;

9 else return UNDECIDED;

10 }

مثالي از SRF براي خط‌مشئ P1

* خط‌مشئ با //

يك عبارت مسير شامل // گره‌هايي را انتخاب مي‌كند كه موقعيتشان در ساختار داده دقيقاً مشخص و معلوم نمي‌باشد، و بنابراين ممكن است مسير به چندين مسير مشخص نگاشت شود. از آنجائيكه ORF بوسيله شئ شاخص‌بندي مي‌شود، واضح است كه ORF نمي‌تواند قواعد دربرگيرنده // را پشتيباني نمايد. در عوض از GRF براي پشتيباني قواعدي كه نمي‌توانند به يك ORF خاص نگاشت شوند استفاده مي‌كنيم.

تابعGRF به عنوان يك تابع دستور كلي از يك كلاس توليد مي‌شود. هنگامي از تابع GRF استفاده مي‌شود كه مسير درخواستي در مقابل قواعد حاوي // تطبيق داده شوند. معمولاً تطبيق با استفاده از عبارات با قاعده انجام مي‌شود. به عنوان نمونه، سومين قاعده در p1 بصورت   
Rule(subp, -R, /a//d) مي‌باشد. تابع GRF توليد شده در شكل 2-11 نشان داده شده است.

1 static Pattern p1 = Pattern.compile("/a/.\*/d|/a/.\*/d/.\*");

2 static public Integer GRF(String sub, String path) {

3 Matcher m1 = p1.matcher(path);

4 if (sub.equals(“subp”)) { // for (3)

5 if (m1.matches())

6 return DENY;

7 }

8 return UNDECIDED;

9 }

مثالي از GRF كه // را پشتيباني مي‌كند

براي مدل مبتني بر SRF، از آنجائيكه موضوع در مدل خاص مي‌باشد، هر قاعده مي‌تواند به يك SRF خاص محدود شود. بنابراين قواعد دربرگيرنده // در SRF براي موضوع پشتيباني مي‌شوند. شكل 2-12، SRF مربوط به قواعد 2 و 3 را نشان مي‌دهد. كد مربوط به قاعده 3 قبل از 2 اجرا مي‌شود. يك SRF با يك ORF در داخل بدنه تابع دستور نيز متفاوت مي‌باشد. اين تفاوت در اين است كه در داخل هر تابع دستور در SRF، ابتدا قواعد عدم پذيرش ليست شده و سپس بوسيله قواعد اعطاء كار دنبال مي‌شود.

1 static Pattern p1 = Pattern.compile("/a/.\*/d|/a/.\*/d/.\*");

2 static public Integer SRF1(String path) {

3 Matcher m1 = p1.matcher(path);

4 if (m1.matches()) // for (3)

5 return DENY;

6 else if (path.startsWith(“/a”)) // for (2)

7 return GRANT\_ON\_SUBTREE;

8 return UNDECIDED;

9 }

مثالي از SRF كه // را پشتيباني مي‌كند

* خط مشئ با گزاره‌ها

يك شئ شامل يك گزاره مي‌باشد كه ابتدا بوسيله جداسازي گزاره pred و مسير p پيش پردازش مي‌شود. در مدل مبتني بر ORF كد جاوا كه pred را نشان مي‌دهد در داخل ORF(p) ايجاد شده‌است. در مقابل، در مدل مبتني بر SRF هم pred و هم p در تابع دستور براي موضوع برنامه‌نويسي شده‌اند.

يك گزاره يك وضعيت مقايسه‌اي داده XML يا يك حرف ربط چنين وضعيت‌هايي مي‌باشد. مقايسه با عملگرهاي رابطه‌اي رياضي مانند =، <، >، ≥ و ≤ انجام مي‌پذيرند. در هر دو مدل ORF و SRF، عملگرهاي رياضي براي اجراي ارزيابي گزاره به كد جاوا ترجمه شده‌اند. از آنجائيكه ارزيابي گزاره نيازمند ذخيره داده XML در بانك اطلاعات XML مي‌باشد، لذا داده ارجاع شده در گزاره نياز دارد كه قبل از ارزيابي، بازيابي شود. بنابراين يك رابط API بنام retrieveData ، ايجاد شده است كه داده‌هاي مورد نياز را از DBMS بازيابي مي‌كند. رابط retrieveData دو پارامتر دارد: گره درخواست شده بنام n و عبارت مسير p از گره‌اي كه بوسيله گزاره تحميل شده است. نظر به اينكه در زمان اجرا DBMS اطلاعات مكاني گره درخواستي را دارد (pos)، مكان n مي‌تواند بوسيلة پيمايش از pos تاp پيدا شود.

براي مثال، ممكن است يك ركورد كه شامل جزئيات سفارش است داشته باشيم. هر سفارش با يك شماره منحصر بفرد مشتري (المان CustKey) مشخص شده‌است. براي تضمين اينكه يك مشتري فقط مي‌تواند جزئيات سفارش خودش را ببيند، خط‌مشي كنترل دسترسي P2 مي‌تواند بصورت زير نوشته شود:

Rule (role: customer, +r, /Orders)

Rule (role: customer, +R, /Orders/Order[CustKey = $custID])‌

در P2، $custID يك متغير سيستمي است كه بصورت خودكار زماني كه يك مشتري وارد سيستم مي‌شود، مقدار مي‌گيرد. مقدار $custID بوسيله obtainSystemData ( كه يك رابط API ديگر براي مقادير متغيرهاي سراسري سيستم مي‌باشد) به‌دست مي‌آيد. از آنجائيكه شئ DataCenter هر دو رابط API را پياده‌سازي مي‌كند، قطعه كد جاوا براي بررسي ارزيابي گزاره در شكل 2-13 نشان داده شده است. به عبارت ديگر، اگر قاعده‌اي شامل چندين گزاره باشد، پيوستگي گزاره‌ها در تابع دستور مخفي شده است.

String value0 = DataCenter.retrieveData(“CustKey”, “/Orders/Order”);

String value1 = DataCenter.obtainSystemData(“custID”);

if (value0.equals(value1))

return GRANT\_ON\_SUBTREE;

else

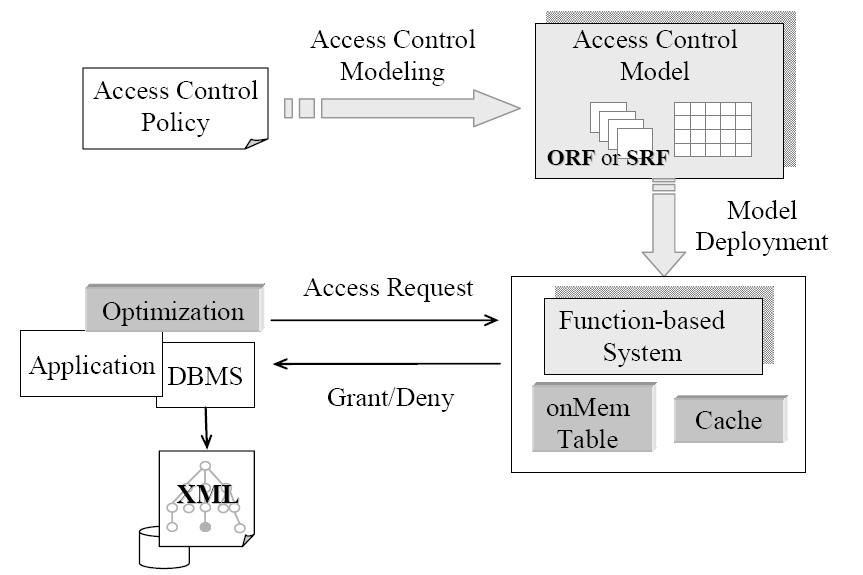
return UNDECIDED;

قطعه كدي جهت گزاره CustKey = $custID

* معماري سيستم كنترل دسترسي مبتني بر تابع

معماري سيستم كنترل دسترسي مبتني بر تابع در شكل 2-14 نشان داده شده است [9]. اين سيستم با مدل‌سازي كنترل دسترسي و سپس اجراي پروسه گسترش مدل [[32]](#footnote-32) ايجاد مي‌شود. در اين سيستم، كنترل دسترسي از موتور بانك اطلاعات جدا شده است. اين كار اجازه مي‌دهد كه براي هر XML DBMS، كنترل دسترسي مقياس‌پذير فراهم گردد.

در بخش مدل‌سازي كنترل دسترسي، خط‌مشئ كنترل دسترسي به كلاس‌هاي جاوا تبديل مي‌شود كه هر كلاس قواعد كنترل دسترسي براي تعداد خاصي از موضوعات يا اشياء را بيان مي‌كند. در طي اين مدل‌سازي، جدول نگاشت مربوطه توليد مي‌گردد. بعلاوه قطعاتي از كد جاوا به كدهاي باينري قابل اجرا كامپايل مي‌شود.



سيستم كنترل دسترسي مبتني بر تابع

هنگامي که سيستم شروع به کار مي‌کند پروسه‌اي به نام توسعه مدل اجرا شده و جدول نگاشت را داخل حافظه اصلي بارگذاري مي‌کند. سپس يك جدول سراسري خالي به نام onMemTable آماده مي‌گردد كه در حافظه اصلي نمونه‌هاي شئ جاوا را ذخيره مي‌كند. از آنجائيكه هر گروه تابع يك نام منحصر بفرد دارد، جدول onMemTable نام گروه را به عنوان كليد استفاده مي‌كند و آنرا با نمونه شئ متناظر مرتبط مي‌كند.

يك درخواست دسترسي شامل يك موضوع و يك عبارت مسير ساده را در نظر بگيريد. سيستم كنترل دسترسي در زمان اجرا درخواست دسترسي را ارزيابي مي‌كند. اگر ارزيابي دسترسي مقدار DENY را برگرداند، بدون اينكه داده‌اي از بانك اطلاعات XML بازيابي شود پاسخ "دسترسي غيرممكن" به كاربر برگشت داده مي‌شود. در غير اينصورت، بعد از اينكه داده‌ها از بانك اطلاعات بازيابي شدند، خروجي توليد و به كاربر باز مي‌گردد. حافظه نهان مي‌تواند در هر دو سيستم كنترل دسترسي و DBMS براي كاهش هزينه‌هاي ارزيابي قابليت دسترسي بكارگرفته شود.

در صورت استفاده از حافظه نهان، هنگامي که موتور کنترل دسترسي درخواستي را دريافت مي‌کند ابتدا به جستجو در حافظه نهان مي‌پردازد. اگر نتيجه ارزيابي براي درخواست در حافظه نهان وجود داشته باشد آنرا به عنوان نتيجه برمي‌گرداند. در غير اين صورت با استفاده از موضوع يا شي مربوط به درخواست، توابع دستور مرتبط با درخواست را در جدول نگاشت يافته و پس از اجراي توابع و بدست آوردن نتيجه، آنرا در حافظه نهان ذخيره و به فراخواننده بر مي‌گرداند.

* + 1. مدل كنترل دسترسي مبتني بر نگاشت بيتي

مدل كنترل دسترسي مبتني بر نگاشت بيتي كنترل دسترسي را در سطوح مختلف بانك اطلاعات XML برقرار مي‌كند [8]. اين مدل كه توسط آقاي يون و همكارانش مطرح گرديد از يك رويكرد نگاشت بيتي سه بُعدي براي كنترل دسترسي به اطلاعات مستندات XML استفاده مي‌كند [8]. قبل از اينكه به مدل كنترل دسترسي مبتني بر نگاشت بيتي بپردازيم چند مفهوم اساسي در اين رابطه را مختصراً توصيف مي‌كنيم.

* مسير المان‌ها[[33]](#footnote-33)

يك سند XML دنباله‌اي از مسيرها مي‌باشد كه با محتواي المان مرتبط شده‌است. مسير المان دنباله‌اي از المان‌هاي تودرتو مي‌باشد كه آخرين المان داخلي، المان با محتوي ساده مي‌باشد.

* نگاشت بيتي

يك نگاشت بيتي [19]، يك جدول دو بعدي است كه ارتباط بين دو موجوديت مثل مسير‌ المان‌ها و مستندات را نشان مي‌دهد (شكل 2-15). براي مستندات XML، نمايش مسير المان‌ها در مستندات مي‌تواند با استفاده از نگاشت بيتي بيان گردد. اگر يك سند، مسير المان داشته باشد آنگاه در خانة متناظر با آن مقدار 1 و در غير اينصورت همة خانه‌ها مقدار صفر قرار مي‌گيرد.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 |
| d1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| d2 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| d3 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

يك شاخص نگاشت بيتي براي مستندات XML

* ديدهاي محدود[[34]](#footnote-34)

يك سند XML ساده مي‌تواند به عنوان يك درخت ديده شود كه المان‌هاي آن گره‌هاي درخت مي‌باشند. اين ساختار پاية XML مي‌تواند حوزة دسترسي يك كاربر را به برخي از مستندات يا بخشي از مستندات محدود كند. اين روش قراردادن محدوديت براي دسترسي كاربر مي‌تواند به‌طور وسيع به سه دسته زير تقسيم‌بندي شود:

الف) محدوديت ديدن سند: محدوديت روي تعدادي از مستندات كه مي‌توانند ديده شوند.

ب) محدوديت ديدن گره: در يك سند منفرد، محدوديت مي‌تواند روي بخش بخصوصي از گره‌ها اعمال شود. مثلاً كاربر نتواند گره‌هاي خاصي را از سند XML ببيند.

د) محدوديت ديدن تركيب شده: محدوديت توسط تركيب دو ديد بالا شكل گرفته است كه كاربر مي‌تواند فقط برخي از مستندات و در آن‌ها فقط بخشي از سند را مشاهده نمايد.

در شكل 2-16 يك سند با محدوديت ديد روي بخشي از گره‌هاي آن نشان داده شده است. محدوديت روي يك گره بوسيله محدوديت روي مسير Software.Author و Software.Design انجام شده است.

|  |  |
| --- | --- |
| <Software>  <Title>XML editor </Title>  <Price>1500$</Price>  <Author>ABC</Author>  <Design>3-dimensional structures  </Design>  <Languages>java</Languages>  <Year>1999</Year>  </Software> | <Software>  <Title>XML editor </Title>  <Price>1500$</Price>  <Languages>java</Languages>  <Year>1999</Year>  </Software> |
| سند اصلي | ديد محدود شده |

مثالي از يك ديد محدود شده

* كنترل دسترسي و نگاشت بيتي

يكي از ويژگي‌هاي كليدي در امنيت اين است كه تصميم گرفته شود كه چه كسي در يك مجموعه معين از كاربران مفروض، ادارة داده‌ها را در دست گيرد؛ و به اين ترتيب به افراد اجازه داده نشود كه با قصد عناد روي اطلاعات ديگران خرابكاري كنند. به اين جهت است كه روي مكانيزمي كه كنترل دسترسي ناميده مي‌شود تكيه مي‌كنيم. كنترل دسترسي مكانيزمي است كه دسترسي به داده يا اجراي عمليات يك برنامه را اعطا و يا لغو مي‌نمايد. در قالب مستندات XML، كنترل دسترسي در چهار سطح مختلف مي‌تواند تعريف شود [8]:

* + سطح شما يا DTD
  + سطح سند
  + سطح مسير المان
  + سطح محتوي

اين سطوح، دسترسي كاربران را به مستندات يا به بخشي از مستنداتي كه براي آنها تعيين شده است، محدود مي‌كنند. آقاي يون و همكارانش نشان داده‌اند كه اين ساختار سلسله مراتبي از سطوح امنيت، مي‌تواند يك نگاشت بيتي را استفاده كند. يك نگاشت بيتي {dx , dy}، يك نمايش دو بعدي است كه dx و dy خوانده مي‌شود. در ادامه مثال‌هايي از نمايش نگاشت بيتي از انواع مختلف كنترل دسترسي نشان داده مي‌شود.

مثال: دسترسي در سطح DTD (در تعريف كنترل دسترسي همة مستندات وابسته به DTD مقادير 1 را مي‌گيرند)

نگاشت بيتي نمايش داده شده در جدول 2-3 كنترل دسترسي را براي 4 كاربر نشان مي‌دهد. وقوع عدد 1 نشان مي‌دهد كه كاربر صلاحيت ديدن مسير المان متناظر در همة مستنداتي كه به DTD خاصي وابسته هستند را دارد. بنابراين مطابق با نگاشت بيتي، كاربر U1 فقط مي‌تواند مسير P1 در همة مستندات را ببيند، كاربر U2 مي‌تواند مسير P1 و مسير P3 در همة مستندات را ببيند، كاربر U3 هيچ يك از مسيرها در مستندات را نمي‌تواند ببيند و كاربر U4 همة مسيرها در همة مستندات را مي‌تواند ببيند.

نمايش نگاشت بيتي از كنترل دسترسي در سطح DTD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 |
| U1 | 1 | 0 | 0 | 0 |
| U2 | 1 | 0 | 1 | 0 |
| U3 | 0 | 0 | 0 | 0 |
| U4 | 1 | 1 | 1 | 1 |

مثال: دسترسي در سطح سند

نگاشت بيتي در جدول 2-4 كنترل دسترسي را براي 4 كاربر نشان مي‌دهد. وقوع عدد 1 نشان مي‌دهد كه كاربر صلاحيت ديدن سند مشخصي را دارد. بنابراين مطابق با نگاشت بيتي، كاربر U1 فقط مي‌تواند سند D2 و D4 را ببيند، كاربر U2 مي‌تواند فقط سند D3 ، كاربر U3 فقط سند D1 و كاربر U4 همة سندها را مي‌تواند ببيند.

نمايش نگاشت بيتي از كنترل دسترسي در سطح سند

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | D1 | D2 | D3 | D4 |
| U1 | 0 | 1 | 0 | 1 |
| U2 | 0 | 0 | 1 | 0 |
| U3 | 1 | 0 | 0 | 0 |
| U4 | 1 | 1 | 1 | 1 |

مثال: دسترسي در سطح مسير المان

نگاشت بيتي در جدول 2-5 دسترسي اطلاعات براي يك كاربر در سطح مسير را نشان مي‌دهد. اگر كاربر اجازه ديدن تمام سند را داشته باشد، آنگاه سطر مربوط به آن سند براي همة مسيرها مقدار 1 را مي‌گيرد. اگر كاربر قابليت ديدن فقط تعدادي از مسيرهاي يك سند را داشته باشد، آنگاه نگاشت بيتي در مكان‌هاي مربوطه مقدار 1 را مي‌گيرد. با توجه به نگاشت بيتي بالا كاربر اجازه ديدن تمام سند D4 و فقط مسيرهاي P1 و P3 در سند D2 را دارد.

نمايش نگاشت بيتي از كنترل دسترسي در سطح مسير المان

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 |
| D1 | 0 | 0 | 0 | 0 |
| D2 | 1 | 0 | 1 | 0 |
| D3 | 0 | 0 | 0 | 0 |
| D4 | 1 | 0 | 0 | 1 |

مثال: دسترسي در سطح محتوا

نگاشت بيتي نشان داده شده در جدول 2-6 دسترسي اطلاعات براي يك كاربر در سطح محتوا را نشان مي‌دهد. محتوا (كه در جدول 2-6 با W نشان داده شده است) در جملات XML اگر با مسيري كه موجود است بيان شود مي‌بايست بيشتر مشخص گردد. بنابراين، زوج مرتب (محتوا ، مسير) را براي نمايش محتواي حاضر در يك سند استفاده مي‌كنند. اگر كاربر اجازه ديدن تمام سند را داشته باشد آنگاه سطر مربوط به آن سند براي همة زوج‌هاي مرتب (محتوا ، مسير) مقدار 1 را خواهد داشت. چنانچه كاربر قابليت ديدن فقط بعضي از زوج‌هاي مرتب (محتوا ، مسير) را داشته باشد، در اين‌صورت نگاشت بيتي در مكان مخصوص به آن مقدار 1 را مي‌گيرد. با توجه به نگاشت بيتي مثال قبل، كاربر اجازه ديدن تمام سند D4 ، محتوي (P1, W2) در سند D2 و محتواي (P1, W1)، (P1, W3)، (P2, W3) در سند D2 را دارد. بنابراين، نمايش نگاشت بيتي كنترل دسترسي براي يك كاربر خاص مقدار ({ePath, content}, document) را به عنوان ابعاد جدول دارد.

نمايش نگاشت بيتي از كنترل دسترسي در سطح محتوا

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | (P1,W1) | (P1,W2) | … | (P2,W1) | … | (P4,W4) |
| D1 | 0 | 1 |  | 0 |  | 0 |
| D2 | 1 | 0 |  | 0 |  | 0 |
| D3 | 0 | 0 |  | 0 |  | 0 |
| D4 | 1 | 1 |  | 1 |  | 1 |

* قانون استنتاج[[35]](#footnote-35)

يك نگاشت بيتي براي كنترل دسترسي {dx , dz}، مي‌تواند از هر دو نمايش نگاشت بيتي كنترل دسترسي {dx , dy} و {dy , dz} به شرط آنكه dxi = {dz1 , dz2 , … , dzk} و x≠y≠z نتيجه شود. در اين صورت به شكل زير عمل مي‌شود:

1. براي هر ورودي dyb اگر  آنگاه ورودي  در نگاشت بيتي   
   {dx , dz} براي هر k برابر صفر خواهد بود.
2. در غير اينصورت، در نگاشت بيتي {dx , dz}، فقط براي k اگر {dyb , dzc}=1 آنگاه  ، و گرنه 

با استفاده از اين قانون استنتاج، مي‌توان نمايش نگاشت بيتي كنترل دسترسي در سطح مسير را از نگاشت بيتي درسطح سند و DTD براي كاربر بدست آورد. همچنين با داشتن قانون استنتاج يك مجموعه كمينه از نمايش‌هاي نگاشت بيتي را مي‌توان بدست آورد.

* حساسيت سند

يكي از جنبه‌هاي مبحث كنترل دسترسي حساسيت سند مي‌باشند. به عبارت ديگر اگر يك سند مفروض حساسيت بيشتري داشته باشد، بيشتر محافظت مي‌شود. در مدل مبتني بر نگاشت بيتي حساسيت يك سند را نسبت تعداد بيت‌هاي 1 به كل بيت‌ها تعريف مي‌كنند [8]. اين حسساسيت براي يك سند مفروض با توجه به بردارهاي بيتي كنترل دسترسي همة كاربران به آن سند، محاسبه مي‌شود. هرچه تعداد بيت‌هاي 1 مربوط به يك سند كمتر باشد آن سند حساس‌تر مي‌باشد. به طور ذهني يك آستانه حساسيت را تعريف مي‌كنند كه اگر حساسيت يك سند كمتر از آن باشد در اين‌صورت مي‌گويند كه آن سند حساس مي‌باشد.

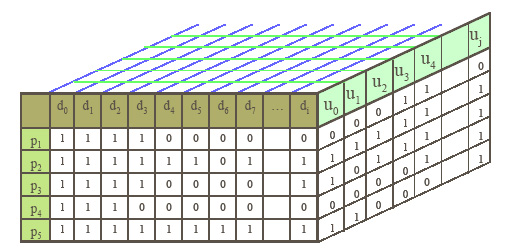
(تعداد كل بيت‌ها) / (تعداد بيت‌هاي 1) = حساسيت سند

به عنوان مثال با توجه به جدول 2-4، حساسيت سند D2 برابر 0.5 خواهد بود. اگر فرض كنيم كه آستانه حساسيت 0.6 مي‌باشد، لذا اين سند يك سند حساس است.

* مكعب امنيت

مكعب امنيت يك ساختار داده سه بعدي طراحي شده بر روي نگاشت بيتي است كه هدف آن نشان دادن كنترل دسترسي مؤثر براي مستندات XML مي‌باشد [8]. دغدغه اصلي آقاي يون و همكارانش اين بوده است كه يك راه حل براي نشان دادن كنترل دسترسي جهت چندين كاربر تعيين كنند، لذا يك ساختار داده ساده‌اي كه همچون يك نگاشت بيتي توسعه مي‌يابد و آنرا مكعب امنيت مي‌نامد، پيشنهاد مي‌كنند.

نگاشت بيتي مي‌تواند به‌طور مؤثر براي تهيه كنترل دسترسي جهت چندين كاربر روي مستندات متعدد با مسيرهاي بسيار زياد استفاده شود. در قسمت قبل نشان داده شد كه چگونه نگاشت بيتي مي‌تواند به‌طور مؤثر براي نمايش كنترل دسترسي چندين كاربر هم در سطح DTD و هم در سطح سند استفاده شود. امّا ديديم كه در سطح مسير نمايش كنترل دسترسي فقط براي يك كاربر انجام شد. لذا آقاي يون جهت نمايش كنترل دسترسي در سطح مسير براي چندين كاربر، مكعب امنيت را استفاده مي‌كند. اين مكعب مجموعه‌اي از نگاشت‌هاي بيتي مي‌باشد كه هر نگاشت بيتي كنترل دسترسي براي يك كاربر را نشان مي‌دهد. شكل 2-17 نمونه‌اي از اين مكعب را نشان مي‌دهد.



مكعب امنيت

ويژگي‌هاي مكعب امنيت را مي‌توان به صورت زير عنوان نمود:

الف) ساختار سه بعدي جهت نمايش كنترل دسترسي در سطح مسير

مكعب امنيت يك ساختار داده با مقادير 0 و 1 است كه مي‌تواند كنترل دسترسي را نمايش دهد. سه وجه مكعب امنيت را اسناد، مسيرها و كاربران تشكيل مي‌دهند. ‌مكعب امنيت براي نمايش كنترل دسترسي در پايين‌ترين سطح مسير يعني محتواي هر مسير قابل استفاده مي‌باشد. از آنجائيكه مكعب امنيت در پايين‌ترين سطح تعريف مي‌شود، لذا مي‌تواند يك مسير بسيار امن را شبيه‌سازي نمايد. جدول 2-7 فرمول محاسبه اندازة مكعب امنيت (فضاي حافظه مورد نياز) را نشان مي‌دهد.

روش محاسبه اندازة مكعب امنيت

|  |
| --- |
| Size of SecurityCube = (D × P × U) / 8 bytes.  D: تعداد اسناد , P: تعداد مسيرها , U: تعداد كاربران |

ب) دسترسي سريع

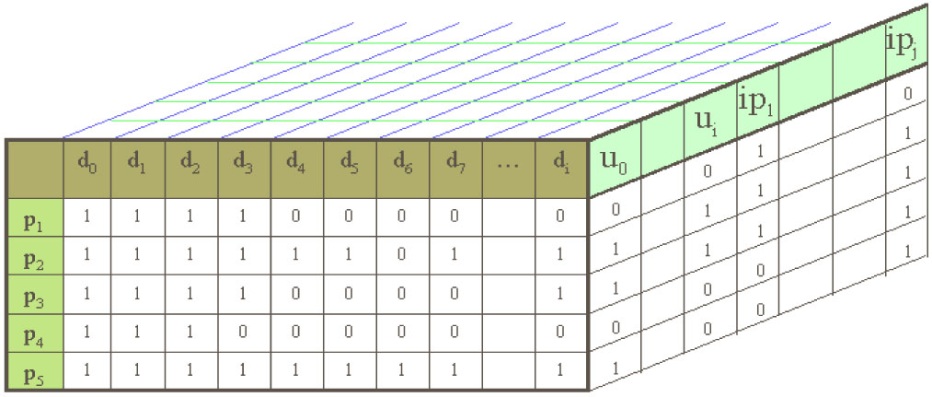
مكعب امنيتي كه آقاي يون و همكارانش ارائه كردند بر پايه مكعب بيتي[[36]](#footnote-36) [20] مي‌باشد. ساختار مكعب بيتي قابليت دسترسي سريع را فراهم مي‌نمايد. همچنين انعطاف‌پذيري براي نگهداري و به‌روزرساني را دارد [20]. براي مثال به‌روزرساني و تغيير در كنترل دسترسي مجوزها مي‌تواند با تغيير يك بيت انجام پذيرد. جدول 2-8 الگوريتم به‌روزرساني نگاشت بيتي كنترل دسترسي براي يك كاربر در مكعب امنيت را نشان مي‌دهد.

الگوريتم به‌روزرساني مكعب امنيت براي كاربر/مجوز جديد

|  |
| --- |
| UpdateSecurityCube (Document d, ePath p, User u, Permission Grant/Deny)  D: مجموعه اسناد , P: مجموعه مسير المان‌ها , U: مجموعه كاربران  If (u ⊄U) ⇒ create a new Bitmap (D × P) for u with permissions Grant/Deny  And append it to the SecurityCube  Else If (Permission = Grant) ⇒ SecurityCube (d, p, u) = 1  Else SecurityCube (d, p, u) = 0; |

پ) قابل توسعه به موضوعاتي از قبيل آدرس‌هاي IP و نام‌هاي سمبليك

كنترل دسترسي براي موضوعي كه به يك شئ خاص دسترسي دارد (منظور از شئ در اينجا اسناد و يا مسير المان‌ها است)، تعريف شده است. موضوعات مي‌توانند كاربران، گروه‌هاي كاربري، كاربران اينترنتي و يا كاربراني كه با نام خاصي به اشياء دسترسي دارند، باشند. اين شرايط خاص براي كنترل دسترسي در مكعب امنيت قابل نمايش هستند. مثلاً كاربر U1 از IPx به شئ خاصي دسترسي پيدا كند امّا از IPy اجازه دسترسي نداشته باشد. شكل 2-18 نمونه‌اي از مكعب امنيت با موضوعاتي از قبيل IP را نشان مي‌دهد.



نمونه‌اي از مكعب امنيت با موضوعاتي از قبيل كاربران، آدرس‌هاي IP و نام‌هاي سمبليك

ت) قابل توسعه جهت نوشتن و اضافه كردن مجوزها

مكعب امنيت تعريف شده توسط آقاي يون و همكارانش فقط مجوز خواندن را دارد. امّا اگر بخواهيم مجوزهاي ديگري از قبيل نوشتن، افزودن، حذف كردن و ... را داشته باشيم مي‌بايست به اين مكعب يك وجه ديگر به عنوان وجه چهارم بيافزاييم، كه در اين وجه مجوزها قرار مي‌گيرد. بنابراين ابعاد ساختار جديد بصورت (اسناد، مسير المان‌ها، كاربران، مجوزها) خواهد بود. جدول 2-9 انواع مختلف مكعب‌هاي امنيت را در ساختاري كه آقاي يون و همكارانش تعريف نموده‌اند نشان مي‌دهد.

انواع مختلف مكعب‌هاي امنيت

|  |  |
| --- | --- |
| Dimensions | Security Cube |
| Documents, ePaths, User | 1- Simple |
| Documents, ePaths, User + IP | 2- Extended with IP or Symbolic name |
| Documents, ePaths, User +  Symbolic name |
| Documents, ePaths, User + IP +  Symbolic name |
| Documents, ePaths, User, Privileges | 3- Extended with other Privileges (Four–dimensional structure) |

ث) قابل توسعه جهت برقراري كنترل دسترسي در سطح محتوي

مكعب امنيت كنترل دسترسي را در سطح مسير المان توصيف مي‌كند. اين مكعب مي‌تواند جهت بيان كنترل دسترسي در سطح محتوا نيز استفاده گردد. براي نائل آمدن به اين هدف كافيست بعد مسير المان را به زوج مرتب (ePath, Content) توسعه دهيم. بنابراين ابعاد مكعب امنيت جهت استفاده به صورت (Document, {ePath,Content}, Users) خواهد بود.

ج) مقياس پذيري

آقاي يون و همكارانش رويكرد نگاشت بيتي را براي تعريف كنترل دسترسي بر روي مستندات XML در سطوح مختلف امنيت توصيف كرده‌اند. به جز سطح مسير المان و سطح محتوي بقية سطوح امنيت كنترل دسترسي در يك نگاشت بيتي دو بعدي قابل نمايش هستند. مقياس‌پذيري در اينجا به اين معناست كه افزودن كاربر يا سند جديد را پشتيباني مي‌كند. در نگاشت بيتي، توسعه ابعاد نيز مي‌تواند انجام پذيرد.

چ) ايجاد و توصيف حساسيت اسناد

مكعب امنيت مي‌تواند تراكم و يا پراكندگي مربوط به حساسيت اسناد را مشخص كند. براي هر سند يك نگاشت بيتي وجود دارد كه ليست كاربراني را مشخص مي‌كند كه مي‌توانند به مسير المان‌هاي آن سند دسترسي داشته باشند. چگالي يك سند تعداد يك‌هاي نمايش داده شده در نگاشت بيتي مرتبط با آن سند تعريف مي‌شود. بنابراين حساسيت سند با چگالي آن ارتباط عكس دارد. هرچه سند حساس‌تر باشد چگالي آن سند كمتر مي‌باشد. اگر يك سند خيلي حساس باشد بنابراين كاربران كمتري به آن دسترسي دارند. لذا آن سند از چگالي كمتري برخوردار خواهد بود. در مكعب امنيت اسناد با حساسيت پايين را مي‌توان در انتهاي مكعب قرار داد. اين سازماندهي باعث دسترسي آسانتر به اسناد با حساسيت‌هاي مختلف مي‌گردد. البته استفاده از اين روش نيازمند سازماندهي مجدد در مكعب امنيت مي‌باشد كه ممكن است در اسناد با حجم بسيار زياد باعث سربار در سيستم گردد.

* 1. نتيجه‌گيري

‌در نتيجه‌گيري آخر فصل، با توجه به بررسي انجام‌شده بر روي مراجع تحقيق، افق‌هاي تازه‌اي كه پيش‌روست يا خلأهاي تحقيقاتي بيان مي‌شود

1. روش تحقيق
   1. مقدمه

با توجه به مطالب ارائه شده در فصل قبل، مشاهده مي‌شود كه يك روش خوب و جامع براي كنترل دسترسي در بانك اطلاعات XML بايد حداقل داراي خصيصه‌هاي زير باشد:

* سادگي و قابل فهم بودن روش
* كارائي بالا
* قابليت توسعه
* سرعت بالا در دستيابي به سياست‌ها و داده‌ها
* ايجاد كنترل دسترسي در پايين‌ترين سطوح
* پشتيباني از پرس‌وجوهايي كه شامل گزاره‌هاي // و \* هستند
* امكان پياده‌سازي و بكارگيري در بانك‌هاي اطلاعات بزرگ

در اين فصل به ارائه مدل پيشنهادي جهت برقراري كنترل دسترسي در بانك اطلاعات XML مي‌پردازيم. مدل پيشنهادي بر پايه مدل مبتني بر نگاشت بيتي مي‌باشد. اين مدل تمامي خصيصه‌هاي يك مدل جامع را دارا مي‌باشد. در ادامه به شرح مدل پيشنهادي مي‌پردازيم و وجود خصيصه‌هاي فوق را اثبات مي‌كنيم. قبل از شروع ابتدا مفاهيم و اصطلاحاتي را كه در اين مدل استفاده خواهيم كرد را توصيف مي‌نماييم.

* 1. مفاهيم و اصطلاحات
* المان بانك: در يك بانك اطلاعات XML، المان ريشه را المان بانك مي‌ناميم. علت اين نام‌گذاري به جهت قرارگيري نام بانك اطلاعات و يا نام سند XML در آن مي‌باشد. به عنوان مثال در شكل 3-1 كه نمونه‌اي از DTD يك بانك اطلاعات سيستم آموزشي ارائه شده است، المان edu يك المان بانك مي‌باشد.

|  |
| --- |
| <!ELEMENT edu ((prof | stud| crs| sec| clg)+)>  <!ELEMENT prof (pname, office, esp, degree, clg\_id)>  <!ATTLIST prof prof\_id ID #REQUIRED >  <!ELEMENT pname (#PCDATA)>  <!ELEMENT office (#PCDATA)>  <!ELEMENT esp (#PCDATA)>  <!ELEMENT degree (#PCDATA)>  <!ELEMENT clg\_id (#PCDATA)>    <!ELEMENT stud (sname, father, fieldstudy, level, avg, address)>  <!ATTLIST stud s\_id ID #REQUIRED clg\_id IDREF #REQUIRED>  <!ELEMENT sname (firstname, lastname)>  <!ELEMENT address (city, addr, zip)>  <!ELEMENT firstname (#PCDATA)>  <!ELEMENT lastname (#PCDATA)>  <!ELEMENT father (#PCDATA)>  <!ELEMENT avg (#PCDATA)>  <!ELEMENT city (#PCDATA)>  <!ELEMENT addr (#PCDATA)>  <!ELEMENT zip (#PCDATA)>  <!ELEMENT fieldstudy (#PCDATA)>  <!ELEMENT level (#PCDATA)>  <!ELEMENT clg (clgname, city)>  <!ATTLIST clg clg\_id ID #REQUIRED head IDREF #REQUIRED>  <!ELEMENT clgname (#PCDATA)>  <!ELEMENT city (#PCDATA)>  <!ELEMENT crs (cname, unit)>  <!ATTLIST crs crs\_id ID #REQUIRED provider-clg IDREFS #REQUIRED>  <!ELEMENT cname (#PCDATA)>  <!ELEMENT unit (#PCDATA)>  <!ELEMENT sec (crs\_id, s\_id, prof\_id, term, score)>  <!ELEMENT crs\_id (#PCDATA)>  <!ELEMENT s\_id (#PCDATA)>  <!ELEMENT prof\_id (#PCDATA)>  <!ELEMENT term (#PCDATA)>  <!ELEMENT score (#PCDATA)> |

نمونه‌اي از يك DTD سيستم آموزش

* المان سند: المان‌هايي كه معرف مفهوم سند در بانك اطلاعات XML مي‌باشند. اين المان‌ها بلافاصله بعد از المان بانك در DTD تعريف مي‌شوند. در واقع اين المان‌ها را مي‌توانيم متناظر با جداول در بانك‌هاي اطلاعات رابطه‌اي در نظر بگيريم. در شكل 3-1 المان‌هاي prof، stud و crs المان‌هاي سند مي‌باشند. هر المان بانك مي‌تواند تعداد يك يا چند المان سند داشته باشد.
* المان شئ : المان‌هايي كه معرف اشياء در بانك اطلاعات XML مي‌باشند. در واقع اين المان‌ها را مي‌توانيم متناظر با مفهوم ركورد در بانك‌هاي اطلاعات رابطه‌اي در نظر بگيريم. تفاوت المان سند با المان شئ در اين است كه المان سند در DTD تعريف مي‌شود و مفهوم انتزاعي دارد ولي المان شئ در فايل اصلي بانك اطلاعات XML بوده و مفهوم واقعي از يك شئ را در بر مي‌گيرد. در واقع هر المان سند داراي صفر يا چند المان شئ مي‌باشد.
* المان فيلد: المان‌هايي كه معرف اجزاء اشياء مي‌باشند. اين المان‌ها مشابه فيلدهاي جداول در مدل رابطه‌اي مي‌باشند. توجه شود كه صفات يك المان نيز جزء المان فيلد محسوب مي‌گردند. به عنوان مثال المان pname، office و esp در شكل 3-1 المان فيلد مي‌باشند.
  1. بهينه‌سازي مكعب امنيت

به دليل اينكه در اكثر بانك‌هاي اطلاعات XML هر المان سند، المان فيلد مختص خود را دارد؛ لذا در نگاشت بيتي كنترل دسترسي در سطح مسير المان‌ها، مسيرهايي وجود خواهند داشت كه ارتباطي به يك سند خاص ندارند. اين مسيرها باعث بوجود آمدن خانه‌هايي در نگاشت بيتي مي‌شوند كه بلااستفاده بوده و هميشه مقدا صفر را خواهند داشت. بنابر اين در مكعب امنيت نيز به دليل قرارگيري مسير المان‌ها در يك وجه مكعب، خانه‌هايي بوجود خواهند آمد كه بلااستفاده مي‌باشند. حال اگر بانك اطلاعات داراي داده‌هاي بسيار زيادي باشد، مكعب امنيت مربوط به آن داراي فضاي بلااستفاده بسيار زيادي خواهد بود.

به عنوان مثال يك بانك اطلاعات XML با DTD ذكر شده در شكل 3-1 را در نظر بگيريد. اگر بخواهيم نگاشت بيتي كنترل دسترسي در سطح مسير المان را براي المان prof و المان stud ترسيم كنيم، با خانه‌هايي در نگاشت بيتي مواجه مي‌شويم كه مقدار صفر را دارند. زيرا المان prof و المان stud هيچگونه مسير مشابه‌اي با هم ندارند. بنابراين اگر بخواهيم كنترل دسترسي چندين كاربر را به اطلاعات اساتيد و دانشجويان مشخص نماييم، در مكعب امنيت ايجاد شده با فضايي مواجه مي‌شويم كه داراي مقدار صفر مي‌باشد و هرگز به آنها رجوع نمي‌شود.

بنابراين ملاحظه مي‌كنيد كه مكعب امنيت ارائه شده توسط آقاي يون و همكارانش در مدل كنترل دسترسي مبتني بر نگاشت بيتي جهت استفاده براي بانك‌هاي اطلاعات XML بسيار حجيم با مشكل فضا مواجه خواهد بود. اين ادعا را با مثالي واضح و عملي بيان مي‌كنيم. بانك اطلاعات XML با شماي ذكر شده در شكل 3-1 را درنظر بگيريد. اين بانك اطلاعات مربوط به بخشي از يك سيستم آموزش واقعي مي‌باشد. در پياده‌سازي و ارزيابي مدل پيشنهادي نيز از اين بانك اطلاعات استفاده خواهيم نمود. همانگونه كه مشاهده مي‌شود در اين بانك پنج المان سند وجود دارد. همانطور كه در بخش تعاريف ذكر شد هر المان سند شامل چندين المان شئ، و هر المان شئ شامل چندين المان فيلد مي‌باشد. جدول 3-1 تعداد المان‌هاي موجود در اين بانك را نشان مي‌دهد.

تعداد اقلام اطلاعاتي در يك سيستم آموزش نمونه

|  |  |  |  |
| --- | --- | --- | --- |
| نام المان سند | تعداد المان شئ | تعداد المان فيلد | تعداد مسيرهاي مرتبط |
| prof | 760 | 6 | 4560 |
| stud | 4352 | 11 | 47872 |
| crs | 2383 | 4 | 9532 |
| sec | 48231 | 5 | 241155 |
| clg | 12 | 4 | 48 |

تعداد مسيرهاي مرتبط با هر المان سند از فرمول زير محاسبه مي‌گردد:

تعداد المان شئ مرتبط با المان سند \* تعداد المان فيلد مرتبط با المان سند = تعداد مسيرمرتبط با هر المان سند

با توجه به جدول 3-1 و اينكه در اين سيستم 5137 كاربر شامل كارمندان آموزش، اساتيد و دانشجويان وجود دارد، حجم مكعب امنيت به شرح جدول 3-2 محاسبه مي‌گردد.

حجم مكعب امنيت براي سيستم آموزش نمونه

|  |
| --- |
| Size of SecurityCube = (D × P × U) / 8 bytes.  ⇒ S = (5 × 303167 × 5137) / 8 = 973355549 bytes ≈ 973 MB  D: تعداد اسناد , P: تعداد مسيرها , U: تعداد كاربران |

ملاحظه مي‌گردد كه جهت تهيه مكعب امنيت به فضايي معادل 973 مگابايت نيازمند خواهيم بود. البته اگر بخواهيم به عنوان مثال مجوزهاي نوشتن، حذف و ويرايش را علاوه بر مجوز خواندن درنظر بگيريم مي‌بايست به مكعب امنيت يك وجه ديگر اضافه نماييم. اضافه نمودن اين وجه با عث مي‌شود كه فضاي لازم جهت پياده‌سازي مكعب امنيت برابر با 89/3 گيگا بايت شود.

با توجه به اين مسأله، متوجه مي‌شويم كه در عمل روش مبتني بر نگاشت بيتي با دارا بودن مزاياي بسيار خوب، قابل استفاده براي بانك‌هاي اطلاعات بزرگ و واقعي نخواهد بود. در واقع فضاي نگهداري مكعب امنيت و سپس دسترسي به آن يكي از مشكلات عمده روش كنترل دسترسي مبتني بر نگاشت بيتي و مكعب امنيت مي‌باشد.

جهت رفع اين مشكل، مكعب امنيت بهينه را پيشنهاد مي‌كنيم. مكعب امنيت بهينه با حذف يك بعد از مكعب امنيت آقاي يون و همكارانش بدست مي‌آيد. در مكعب امنيت بهينه بُعد سند را حذف و به جاي آن از بُعد مجوز‌ها استفاده مي‌كنيم. در واقع مكعب امنيت بهينه با در نظر گرفتن بعد مجوزها داراي سه بعد خواهد بود. اين ابعاد عبارتند از: {ePaths, Users, Privileges}. به جاي بعد سند كه از مكعب امنيت حذف گرديد، مكعب امنيت بهينه را براي هر المان سند در بانك اطلاعات XML جداگانه ايجاد مي‌كنيم. در واقع براي هر المان سند فقط از نگاشت بيتي كنترل دسترسي در سطح مسير المان استفاده مي‌كنيم.

همانگونه كه در بخش 2-4-6-2 ذكر شد نگاشت بيتي كنترل دسترسي در سطح مسير المان مي‌تواند از دو نگاشت بيتي كنترل دسترسي در سطح سند و DTD استنتاج شود. از طرف ديگر نگاشت بيتي در سطح مسير المان، كنترل دسترسي را در سطحي پايين‌تر مطرح مي‌كند. بنابراين حذف بعد سند از مكعب امنيت خللي در كنترل دسترسي تعريف شده ايجاد نمي‌كند. لذا مكعب امنيت بهينه همان كنترل دسترسي تعريف شده در مكعب امنيت آقاي يون و همكارانش را حفظ مي‌كند. همچنين مكعب امنيت بهينه به دليل بنا شدن بر پايه نگاشت بيتي، كليه ويژگي‌هاي ذكر شده براي مكعب امنيت را نيز حفظ مي‌كند. به اين ترتيب با توجه به كاهش ابعاد مكعب امنيت، فضاي مورد نياز نيز كاهش مي‌يابد. اين كاهش فضا به سرعت دسترسي نيز كمك نموده و كارائي سيستم را افزايش مي‌دهد.

ارزيابي مكعب امنيت بهينه

جهت ارزيابي مكعب امنيت بهينه، حجم آنرا براي بانك اطلاعات XML سيستم آموزش كه در شكل 3-1 آورده شده محاسبه و با مكعب امنيت آقاي يون مقايسه مي‌كنيم. محاسبات انجام شده جهت مكعب امنيت بهينه در جدول 3-3 آورده شده است. در اينجا با توجه به اينكه پنج المان سند وجود دارد لذا پنج مكعب امنيت بهينه متناظر با هر يك از المان‌هاي اسناد ايجاد مي‌گردد. با توجه به جدول 3-3 ملاحظه مي‌گردد كه جهت تهيه مكعب امنيت بهينه در كل به فضايي معادل 778 مگابايت نياز خواهيم داشت. اين درحاليست كه مجوزهاي خواندن، نوشتن، حذف و ويرايش را درنظر گرفته‌ايم. با مقايسه اين فضا و فضاي مورد نياز مكعب امنيت آقاي يون در مي‌يابيم كه تقريباً 80% در فضا صرفه‌جويي شده است. مكعب امنيت پيشنهادي در عمل نيز از سرعت دستيابي بهتري برخوردار مي‌باشد. زيرا به ازاء هر درخواست دسترسي به اطلاعات، جستجو فقط در مكعب امنيت بهينه مربوط به همان سند انجام مي‌پذيرد.

حجم مكعب امنيت بهينه براي سيستم آموزش

|  |
| --- |
| Size of OptimizedSecurityCube = (V × P × U) / 8 bytes.  V: تعداد مجوزها , P: تعداد مسيرها , U: تعداد كاربران |
| ⇒ Sprof = (4 × 4560 × 5137) / 8 = 11712360 bytes ≈ 11.7 MB |
| ⇒ Sstud = (4 × 47872 × 5137) / 8 = 122959232 bytes ≈ 122.9 MB |
| ⇒ Scrs = (4 × 9532 × 5137) / 8 = 24482942 bytes ≈ 24.4 MB |
| ⇒ Ssec = (4 × 241155 × 5137) / 8 = 619406618 bytes ≈ 619 MB |
| ⇒ Sclg = (4 × 48 × 5137) / 8 = 123288 bytes ≈ 123 KB |

* 1. مدل پيشنهادي

مدل پيشنهادي بر پايه نگاشت بيتي و مكعب امنيت بهينه بنا شده است. مدل پيشنهادي داراي بخش‌هاي زير مي‌باشد:

* بخش مديريت سياست‌ها
* يك تابع دستور جهت بررسي قوانين و مجوزهاي نقش‌ها و ايجاد نگاشت بيتي دسترسي در سطح سند و مسير المان از روي قوانين،
* يك تابع به‌روزرسان مكعب امنيت بهينه،
* يك جدول نگاشت بيتي دسترسي در سطح سند مربوط به المان سندهاي بانك‌اطلاعات XML ،
* يك تابع ارزيابي دسترسي جهت بررسي مجوز دسترسي به اسناد و اطلاعات،
* يك تابع واكشي اطلاعات از بانك اطلاعات XML،
* يك يا چند مكعب امنيت بهينه مرتبط با المان سندهاي بانك اطلاعات XML

در مدل پيشنهادي تمامي توابع در DBMS پياده‌سازي مي‌شوند. در اين مدل جهت تعريف و به‌روزرساني خط‌مشي‌هاي كنترل دسترسي از مدل گراف نقش و براي اعمال خط‌مشي‌هاي كنترل دسترسي در بانك اطلاعات XML، از مكعب امنيت بهينه استفاده مي‌كنيم. دليل استفاده از اين روش‌ها مديريت آسان واگذاري مجوزها به كاربران و يا لغوآنها مي‌باشد. مدل گراف نقش، مدلي رسا و قابل فهم بوده كه مي‌تواند در زمان تعريف و يا تغيير سياست‌ها بسيار كارا باشد. از طرف ديگر مكعب امنيت بهينه كنترل دسترسي مبتني بر نگاشت بيتي را فراهم مي‌آورد كه در زمان اجرا از كارائي و سرعت بسيار بالايي برخوردار است. خط‌مشي‌هاي كنترل دسترسي در كنار بانك اطلاعات XML و در سندي جدا نگهداري مي‌شوند. بخش مديريت سياست‌ها وظيفه تعريف و به‌روزرساني سياست‌ها و مجوزهاي كاربران را به عهده دارد. هنگامي كه سياست‌ها تغيير و يا تعريف مي‌شوند اين بخش وظيفه فعال‌سازي تابع دستور را نيز دارد.

تابع دستور در مدل پيشنهادي دو كار انجام مي‌دهد. اولاً با استفاده از قوانين و سياست‌هاي تعريف شده جدول نگاشت بيتي دسترسي در سطح سند را ايجاد و به‌روزرساني مي‌كند. ثانياً تغييرات مربوط به نقش‌ها و سياست‌ها را به تابع به‌روزرسان اطلاع مي‌دهد. مكعب‌هاي امنيت بهينه با تغيير سياست‌هاي كنترل دسترسي و يا با كم و زياد شدن اطلاعات مستقر در بانك اطلاعات XML توسط تابع به‌روزرسان، به‌روز رساني مي‌شوند.

تابع ارزيابي دسترسي وظيفه بررسي درخواست‌هاي رسيده از كاربران و يا سيستم‌هاي ديگر را دارد. اين تابع با استفاده از جدول نگاشت بيتي دسترسي در سطح سند، دسترسي به المان اسناد را در بانك اطلاعات XML بررسي مي‌نمايد. در صورتيكه دسترسي به المان سند بلامانع باشد، دسترسي‌هاي مجاز را از مكعب امنيت بهينه مربوط به المان سند محاسبه و به تابع واكشي اطلاعات ارسال مي‌كند. تابع واكشي اطلاعات وظيفه اجراي درخواست‌ها را دارد. اين تابع با توجه به دسترسي‌هاي مجاز، پرس‌وجوي دريافت شده را اجرا و نتيجه آنرا در اختيار تابع ارزيابي قرار مي‌دهد.

* + 1. خط‌مشي كنترل دسترسي

همانطور كه گفته شد در مدل پيشنهادي براي تعريف خط‌مشي‌هاي كنترل دسترسي از مدل گراف نقش استفاده مي‌كنيم. اما جهت اعمال كنترل دسترسي، خط‌مشي‌هاي كنترل دسترسي را در مكعب امنيت بهينه پياده‌سازي مي‌كنيم. قوانين کنترل دسترسي در مدل پيشنهادي به صورت شش تايي­هاي مرتب به فرم زير هستند:

Rule = <Subject, Object, Action, Sign, Propagation, Condition>

در اينجا Subject فقط مي­تواند نقش باشد چون همانطور که گفتيم در گراف نقش مجوزها فقط به نقش­ها داده مي­شوند. Object يا شي يک نوع گره در يک فايل XML مي­باشد. بنابراين هر شي يک عبارت Xpath مربوط به شماي يک فايل XML مي­باشد. مسير Xpath مربوط به شي بايد به صورت ساده باشد و نمي‌تواند شامل // ، \* و يا گزاره باشد. Action مشخص کننده نوع مجوز مي‌باشد که در پياده‌سازي انجام گرفته در اين پايان‌نامه يکي از مقادير read، write/create، update و delete را دارد که به ترتيب مشخص کننده اعمال خواندن، ايجاد/نوشتن، تغيير و حذف گره مي­باشد. عبارت Sign مي­تواند مقدار + و – را به ترتيب براي اعطاي دسترسي (GRANT) و رد دسترسي (DENY) داشته باشد. در اين قانون دسترسي Propagation مقدار L (Local) يا R (Recursive) را دارد که به ترتيب مشخص مي­کند آيا علامت دسترسي Sign فقط به اين گره اعمال شود يا به گره­هاي فرزند آن نيز منتشر شود. عبارت Condition شرط دسترسي را براي قانوني با علامت + بيان مي‌کند، اگر مقدار آن خالي باشد براي همه گره‌ها با مسير Object قابل اعمال است. اما اگر بخواهيم براي بعضي از آنها استفاده کنيم آنرا مقداردهي مي­کنيم. در اين پايان­نامه براي سادگي از مقدار Condition براي قانون با علامت "-" صرف نظر مي­کنيم.

بنابراين يک قانون مشخص کننده اعطا يا رد انجام عمل Action روي شي Object براي فرآيند Subject مي‌باشد و اين مجوز يا فقط براي دسترسي به آن گره است يا به گره­هاي فرزند آن نيز منتشر مي‌شود. براي درک بهتر موضوع مثال زير را مشاهده کنيد:

<Student, /students/student/name, read, +, recursive, >

اين قانون مشخص مي­کند که نقش Student مي­تواند گره /students/student/name را بخواند همچنين مي­تواند گره‌هاي فرزند /students/student/name را نيز بخواند مگر اينکه براي گره­هاي فرزند آن قوانين ديگري نيز مشخص شده باشند. بنابراين اگر گره name داراي فرزندان Firstname و Lastname باشد آنگاه نقش Student مي­تواند آن­ها را نيز بخواند.

* رفع ابهام

فرض کنيم چند قانون مختلف به صورت <role, object, action, sign, propagation, condition> در سياست­هاي سيستم تعريف شده است که مقادير role، action و object همه آنها يکسان است. به عبارت ديگر در مجموعه قوانين سيستم چند قانون براي يک نقش و انجام يک عمليات روي يک شي يا مسير Xpath تعريف شده باشد. در اين حالت ابهام برخورد پيش مي­آيد. براي رفع ابهام بايد با ترکيبي از قانون­ها، قانوني را انتخاب و اعمال نماييم. در اينجا براي انتخاب قانون از اصل اولويت با عدم پذيرش استفاده مي­کنيم، بدين صورت که از قوانين موجود قانوني انتخاب مي­شود که دسترسي را محدودتر مي­کند. اين روش را به صورت زير تعريف مي­کنيم.

* + اگر علامت يکي از قوانين "-" (رد) باشد آن قانون انتخاب مي­شود.
  + اگر علامت همه قوانين يکسان باشد و قانوني وجود داشته باشد که انتشار آن به صورت local باشد آنگاه انتشار قانون انتخاب شده نيز بايد local باشد.
  + اگر علامت همه قوانين يکسان باشد آنگاه شرط قانون انتخاب شده ترکيب عطفي شرط قانون­ها خواهد بود.

رفع ابهام هنگام ايجاد و يا به‌روزرساني مكعب امنيت بهينه اعمال مي­گردد. براي رفع ابهام كافيست قانون انتخاب شده از فرآيند بالا را با مجوزهاي مربوطه كاربران در مكعب امنيت بهينه با عملگر AND تركيب نماييم.

* انتشار مجوزها

انتشار مجوزها ممکن است به صورت local يا recursive باشد. در حالتي که انتشار local مي‌باشد آن قانون فقط براي مسير المان مورد نظر تعريف مي‌شود و در حالتي که انتشار recursive مي‌باشد آن قانون به تمامي مسير المان‌هاي فرزند نيز منتشر مي­شود. وجود انتشار به صورت recursive باعث مي­شود که بتوانيم، سياست­هاي سيستم را با حجم کمتري از قانون­ها ايجاد کنيم و نيز کار تعريف سياست­ها آسانتر گردد. فرض کنيد نقش دانشجو مي­تواند مشخصات کامل خود را در پايگاه داده دانشگاه بخواند. اگر فقط از قوانين با انتشار local بخواهيم استفاده کنيم بايد قوانيني كه در شكل 3-2 نشان داده شده است را براي نقش دانشجو در سيستم تعريف نماييم.

|  |
| --- |
| 1. < student, /university/students/student, read, +, local, @studID=$studID >  2. < student, /university/students/student/sName, read, +, local, >  3. < student, /university/students/student/sName/sFName, read, +, local, >  4. < student, /university/students/student/sName/sLName, read, +, local, >  5. < student, /university/students/student/clg\_ID, read, +, local, >  6. < student, /university/students/student/idNo, read, +, local, >  7. < student, /university/students/student/parent, read, +, local, >  8. < student, /university/students/student/address, read, +, local, >  9. < student, /university/students/student/address/state, read, +, local, >  10. < student, /university/students/student/address/city, read, +, local, >  11. < student, /university/students/student/address/dAddress, read, +, local, > |

قوانين مربوط به نقش دانشجو براي مجوز خواندن بدون انتشار مجوز

همانطور که مشاهده مي‌کنيم در صورتي که فقط از قوانين با انتشار local استفاده کنيم، يازده قانون لازم است تا دسترسي دانشجو به تمام اطلاعات خودش را تضمين کنيم. ولي اگر از انتشار recursive استفاده کنيم تنها با يک قانون، دسترسي دانشجو به اطلاعات کامل خودش را تضمين خواهيم كرد. اين قانون به صورت شكل 3-3 است.

|  |
| --- |
| <student, /university/students/student, read, +, recursive, @studID=$studID > |

قوانين مربوط به نقش دانشجو براي مجوز خواندن با انتشار مجوز

بعد از اينکه قوانين سيستم خوانده شدند و نگاشت بيتي مربوط به آنها ساخته شد، بايد علامت­ها منتشر شوند تا جدول نگاشت بيتي معتبر گردد. قاعده منتشر کردن علامت­ها به صورت زير است:

1. اگر علامت "-" باشد و انتشار آن به صورت recursive باشد آنگاه براي همان مسير المان و همه فرزندان آن مسير مقدار صفر را در نگاشت بيتي مربوط به آن مجوز درج مي‌كنيم.
2. اگر علامت "-" باشد و انتشار آن به صورت local باشد آنگاه براي همان مسير المان و همه فرزندان آن مسير مقدار صفر را در نگاشت بيتي مربوط به آن مجوز درج مي‌كنيم.
3. اگر علامت "+" باشد و انتشار آن به صورت local باشد آنگاه فقط براي همان مسير المان در نگاشت بيتي مربوط به آن مجوز مقدار يك را درج مي‌كنيم.
4. اگر علامت "+" باشد و انتشار آن به صورت recursive باشد آنگاه براي همان مسير المان مقدار يك را در نگاشت بيتي مربوط به آن مجوز درج مي‌كنيم و براي فرزندان آن مسير المان با توجه به ديگر قوانين و قانون رفع ابهام عمل مي‌كنيم.

همانطور که ديده مي­شود علامت "+" با انتشار recursive به فرزنداني منتشر مي­شود که فاقد قانون هستند. وقتي علامت يک مسير المان به صورت "-" باشد دسترسي به فرزندان آن مسير المان نيز غير مجاز است. بنابراين علامت "-" به فرزندان منتشر مي­شود.

* + 1. كنترل دسترسي

كنترل دسترسي در مدل پيشنهادي براساس نگاشت بيتي دسترسي در سطح سند و مكعب امنيت بهينه صورت مي‌گيرد. ابتدا با استفاده از نگاشت بيتي دسترسي در سطح سند، دسترسي به اسناد و المان سند كنترل مي‌گردد. در صورتيكه نتيجه بررسي مقدار DENY باشد، دسترسي به المان سند مجاز نبوده و مقدار DENY به كاربر ارسال مي‌گردد. در غير اينصورت مكعب امنيت بهينه مربوط به المان سند، فراخواني مي‌گردد. با استفاده از اين مكعب امنيت، دسترسي‌هاي مجاز به مسير المان‌ها بررسي و پاسخ مناسب تهيه و به كاربر ارسال مي‌گردد. چنانچه پرس‌وجوي كاربر از نوع UPDATE و يا DELETE باشد، در مكعب امنيت بهينه مجوز تغيير و يا حذف در ارتباط با مسير المان بررسي مي‌گردد. چنانچه مقدار عددي خانه مربوطه در مكعب امنيت بهينه عدد يك باشد، اجراي پرس‌وجو بلامانع بوده و پرس‌وجو اجرا مي‌گردد.

در اين مدل با دراختيار داشتن قوانين و سياست‌ها كه در آن نقش‌ها و مجوزها مشخص مي‌باشند، نگاشت بيتي دسترسي در سطح سند و مكعب امنيت بهينه براي هر المان سند بانك اطلاعات XML ساخته مي‌شود. نگاشت بيتي يك زوج مرتب (D,U) مي‌باشد كه در آن D نشان‌دهنده المان سند و U نشان‌دهنده كاربر مي‌باشد. فرآيند ايجاد نگاشت بيتي دسترسي در سطح سند به اين‌صورت است كه ابتدا قوانين و سياست‌ها ليست مي‌گردند. سپس با توجه به كاربران و نقش‌هاي آنها، دسترسي كاربران به المان‌هاي اسناد بانك اطلاعات XML مشخص مي‌گردد. براساس اين دسترسي جدول نگاشت بيتي تهيه و در سيستم ذخيره مي‌گردد. در اينجا المان سندها سطرهاي جدول و كاربران ستون‌هاي جدول را تشكيل مي‌دهند. به عنوان مثال جدول 3-4 نمونه‌اي از جدول نگاشت بيتي در سيستم آموزش را نشان مي‌دهد.

نمونه‌اي از جدول نگاشت بيتي دسترسي در سطح سند در سيستم آموزش

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | U1 | U2 | U3 | U4 | U5 | … | S1 | S2 | S3 | S4 | S5 | … | P1 | P2 | P3 |
| stud | 1 | 1 | 1 | 1 | 1 | … | 1 | 1 | 1 | 1 | 1 | … | 1 | 1 | 1 |
| prof | 1 | 1 | 1 | 1 | 1 | … | 0 | 0 | 0 | 0 | 0 | … | 1 | 1 | 1 |
| clg | 1 | 1 | 1 | 1 | 1 | … | 1 | 1 | 1 | 1 | 1 | … | 0 | 0 | 0 |
| crs | 1 | 1 | 1 | 1 | 1 | … | 1 | 1 | 1 | 1 | 1 | … | 0 | 0 | 0 |
| sec | 1 | 1 | 1 | 1 | 1 | … | 1 | 1 | 1 | 1 | 1 | … | 1 | 1 | 1 |

با توجه به گراف نقش و سياست‌هاي تعريف شده مكعب‌هاي امنيت بهينه نيز به سادگي قابل پياده‌سازي مي‌باشند. فرآيند ايجاد مكعب‌هاي امنيت بهينه به دو‌صورت قابل انجام است.

روش اول اين است كه ابتدا مكعبي مي‌سازيم كه سه وجه آنرا كاربران، مسير المان‌ها و مجوزها تشكيل مي‌دهند. در شروع كار مقادير همه خانه‌هاي مكعب امنيت بهينه صفر مي‌باشد. سپس قوانين يكي يكي خوانده شده و بر روي مكعب امنيت بهينه اعمال مي‌گردند. قوانين رفع ابهام و انتشار را نيز در همين جا اعمال مي‌كنيم. روش دوم به اين صورت است كه در حين ساخت مكعب امنيت كليه قوانين بررسي و بر روي مكعب امنيت بهينه اعمال مي‌شوند.

براي بالا بردن كارائي سيستم جدول نگاشت بيتي دسترسي در سطح سند و همچنين مكعب‌هاي امنيت بهينه ايجاد شده را در سيستم در حافظه جانبي ذخيره مي‌كنيم. اين عمل باعث مي‌شود كه در مراجعات بعدي كنترل دسترسي به سرعت انجام پذيرد.

جهت اعمال تغييرات در جدول نگاشت بيتي و مكعب‌هاي امنيت بهينه از يك تابع دستور استفاده مي‌كنيم. لازم به ذكر است كه مفهوم اين تابع دستور با توابع دستور استفاده شده در مدل مبتني بر تابع كاملاً متفاوت مي‌باشد. اين تابع دستور، قوانين و سياست‌هاي تعريف شده را بررسي مي‌كند و تغييرات لازم را در جدول نگاشت بيتي و مكعب‌هاي امنيت بهينه مي‌دهد. علت نام‌گذاري اين تابع به نام تابع دستور نيز همين امر مي‌باشد.

لازم به ذكر است كه جهت با لا بردن كارائي سيستم، مكعب امنيت در سطح شِما ساخته و استفاده مي‌گردد. در صورتيكه مكعب امنيت در سطح شِماي بانك اطلاعات XML ساخته گردد به دليل كم حجم بودن آن در حافظه اصلي قابل ذخيره بوده و لذا سرعت و كارائي سيستم را افزايش مي‌دهد.

* + 1. معماري مدل پيشنهادي

معماري مدل پيشنهادي در شكل 3-4 نشان داده شده است.



معماري مدل پيشنهادي

در اين مدل خط‌مشي‌ها و قوانين دسترسي توسط بخش مديريت سياست‌ها تعريف و در فايل Policies نگهداري مي‌شود. اين فايل از ساختار XML طبعيت مي‌كند. هنگامي كه سياستي جديد تعريف مي‌شود و يا مجوزهاي دسترسي تغيير مي‌كنند، مديريت سياست‌ها بلافاصله پس از ثبت آنها تابع دستور را فراخواني نموده و تغييرات را به آن اعلام مي‌كند. تابع دستور قوانين جديد را فراخوانده و تغييرات و يا سياست‌هاي جديد را به‌صورت نگاشت بيتي تبديل مي‌كند. پس از تبديل، اگر تغييرات مربوط به نگاشت بيتي كنترل دسترسي در سطح سند باشد، جدول نگاشت بيتي توسط خود تابع دستور به‌روز رساني مي‌شود. چنانچه تغييرات مربوط به مسير المان‌ها و يا مجوزهاي كاربران باشد كه نيازمند اصلاح مكعب‌هاي امنيت بهينه است، تغييرات به تابع به‌روزرسان ارسال مي‌شود. تابع به‌روزرسان با دريافت تغييرات از تابع دستور مكعب‌هاي امنيت بهينه را به‌روزرساني مي‌كند.

چنانچه داده‌اي جديد نيز در بانك اطلاعات XML ذخيره و يا داده‌اي حذف گردد، مكعب امنيت بهينه مربوط به المان سند آن داده مي‌بايست به روزرساني شود. در اين حالت DBMS تغييرات را به تابع به‌روزرسان اعلام مي‌كند. تابع بروزرسان اطلاعات را از بانك اطلاعات XML فراخوانده و به تابع دستور جهت ايجاد نگاشت بيتي ارسال مي‌كند. پس از تشكيل نگاشت بيتي مربوط به داده‌هاي جديد توسط تابع دستور، اطلاعات جديد به تابع بروزرسان ارسال و در مكعب‌هاي امنيت بهينه ثبت مي‌گردند. الگوريتم به‌روزرساني مكعب امنيت بهينه در مدل پيشنهادي در جدول 3-5 آمده است.

الگوريتم به‌روزرساني مكعب امنيت بهينه در مدل پيشنهادي

|  |
| --- |
| UpdateOptimizedSecurityCube (OptSecurityCube S, ePath p, User u, Permission Grant/Deny)  S: مكعب امنيت بهينه , P: مجموعه مسير المان‌هاي مربوط به المان سند , U: مجموعه كاربران  M: مجموعه مجوزها  If (u ⊄U) ⇒ create a new Bitmap (P × M) for u with permissions Grant/Deny  And append it to the OptimizedSecurityCube  Else If (Permission = Grant) ⇒ OptimizedSecurityCube (p, m, u) = 1  Else OptimizedSecurityCube (p, m, u) = 0; |

در مدل پيشنهادي پرس‌وجوي كاربر به تابع ارزيابي دسترسي فرستاده مي‌شود. تابع ارزيابي دسترسي ابتدا درخواست را با استفاده از جدول نگاشت بيتي درسطح سند، ارزيابي مي‌كند. چنانچه نتيجه اين ارزيابي مقدار DENY باشد، همين مقدار به كاربر ارسال مي‌گردد. در غير اينصورت تابع ارزيابي، مكعب امنيت بهينه مربوطه را فراخواني نموده و دسترسي‌هاي مجاز را به همراه پرس‌وجوي كاربر به تابع واكشي اطلاعات ارسال مي‌كند. تابع واكشي اطلاعات با دراختيار داشتن دسترسي‌هاي مجاز و پرس‌وجوي كاربر، داده‌هاي مجاز را از بانك اطلاعات XML واكشي مي‌كند. نتيجه پرس‌وجو به تابع ارزيابي دسترسي برگردانده شده و به درخواست‌كننده ارسال مي‌گردد. در تابع ارزيابي دسترسي اعتبارسنجي درخواست نيز انجام مي‌گيرد. به اين معني كه اگر درخواست كننده پرس‌وجويي را ارسال نمايد كه معتبر نباشد درخواست ملغي شده و مقدار DENY به درخواست كننده برگردانده مي‌شود.

همانطور كه ذكر گرديد براي بالا بردن كارائي سيستم جدول نگاشت بيتي دسترسي در سطح سند و همچنين مكعب‌هاي امنيت بهينه ايجاد شده را در سيستم در حافظه جانبي و در كنار بانك اطلاعات XML نگهداري مي‌كنيم. اين عمل باعث مي‌شود كه در مراجعات بعدي كنترل دسترسي به سرعت انجام پذيرد.

1. نتايج و تفسير آنها
   1. ارزيابي مدل پيشنهادي

مدل پيشنهادي را پس از پياده‌سازي با بانك اطلاعات XML سيستم آموزش (نمونه ارائه شده در شكل 3-1) ارزيابي نموده‌ايم. در اين ارزيابي پنج نمونه از بانك اطلاعات XML با حجم‌هاي مختلف (10MB، 20MB، 50MB، 100MB و 150MB) را استفاده كرده‌ايم. ارزيابي سرعت دسترسي به اطلاعات را براي پرس‌وجوهايي كه شامل مسيرهاي ساده و مسيرهايي كه شامل // و \* هستند در XPath آزموده‌ايم. در اين آزمون حدود 40 پرس‌وجوي متفاوت اجرا و ميانگين نتايج آنها اندازه‌گيري شد. نمونه‌اي از اين پرس‌وجوها در شكل 4-1 بيان شده است. محيط عملياتي جهت اجرا نيز يك عدد كامپيوتر شخصي با پردازنده Intel Centrino Core 2 Duo 2.0 GHz و حافظه 1GB بوده است.

به دليل اينكه هر يك از روش‌هاي كنترل دسترسي مشكلات عديده‌اي در پياده‌سازي داشتند لذا اين روش را با فقط با روش آقاي يون مقايسه كرده‌ايم. نتايج ارزيابي سرعت دسترسي به اطلاعات در نمودارهاي شكل 4-2 و 4-3 نشان داده شده است.

|  |
| --- |
| /edu/stud[Aug > 17] /sname  /edu/clg/clg\_id (@ head)  /edu/stud[city = 'tehran']  /edu/sec[(term = '86') and (prof\_id = 1184)]  /edu/sec[term = '87']/score  /edu//sname  /edu/stud/\*  /edu/stud/avg | //sname |

نمونه پرس‌وجوهاي اجرا شده جهت ارزيابي سرعت دسترسي به اطلاعات



ارزيابي سرعت دستيابي به اطلاعات با مسيرهاي ساده



ارزيابي سرعت دستيابي به اطلاعات با مسيرهاي داراي // و \*

همچنين جهت به‌روزرساني بانك اطلاعات و ارزيابي سرعت آن از پرس‌وجوهايي كه در شكل 4-4 آورده شده است استفاده نموده‌ايم. در اين قسمت از دستورات XQuery جهت به‌روز‌رساني [37] استفاده شده است. اين دستورات شامل دستورات حذف و جايگزيني اطلاعات مي‌باشند. ارزيابي اجراي اين دستورات در شكل 4-5 آورده شده است.

|  |
| --- |
| delete node fn:doc("university.xml")/crs[cname = $cname]  delete node fn:doc("university.xml")/sec[(s\_id = $s\_id) and (term = $term)]  delete node fn:doc("university.xml")/stud[@s\_id = $s\_id]  replace value of node fn:doc("university.xml")/crs[@crs\_id = $crs\_id]/cname  with $cname  replace value of node fn:doc("university.xml")/sec[(crs\_id = @crs\_id) and  (term = $term)]/prof\_id  with fn:doc("university.xml")/prof[pnme = $pname] @prof\_id |

نمونه پرس‌وجوهاي اجرا شده جهت ارزيابي سرعت به‌روزرساني اطلاعات



ارزيابي سرعت به‌روزرساني اطلاعات با دستورات XQuery در مدل پيشنهادي



ارزيابي سرعت به‌روزرساني اطلاعات با دستورات XQuery در مدل آقاي يون

همان‌طور كه ذكر گرديد در مدل پيشنهادي از مكعب امنيت بهينه استفاده مي‌گردد. اين مكعب نسبت به مكعب امنيت آقاي يون از فضا و حجم حافظه كمتري استفاده مي‌كند. در شكل 4-7 نمودار فضاي استفاده شده در روش آقاي يون براي بانك‌هاي اطلاعات XML استفاده شده در آزمون‌هاي قبل ارائه گرديده است. به دليل اينكه شِماي بانك اطلاعات تغيير نمي‌كند لذا حجم مكعب امنيت در روش پيشنهادي تغيير نمي‌نمايد. به عبارت ديگر حجم مكعب امنيت بهينه به ساختار شماي بانك اطلاعات XML بستگي دارد. در اين آزمون فضاي مكعب امنيت بهينه به ميزان 398 KB بود كه در مقايسه با روش آقاي يون بسيار حجم پاييني مي‌باشد.



حجم مكعب امنيت در مدل آقاي يون

* 1. مقايسه مدل‌هاي كنترل دسترسي

با توجه به مطالب ذكر شده در فصل‌‌هاي قبل ملاحظه مي‌گردد كه هر يك از روش‌‌هاي كنترل دسترسي، جهت استفاده در بانك‌هاي اطلاعات XML داراي مزايا و معايبي هستند. جدول 4-1 به طور خلاصه مقايسه‌اي اجمالي بين روش‌هاي ذكر شده و روش پيشنهادي را نشان مي‌دهد.

مقايسه مدل‌هاي كنترل دسترسي

| مدل كنترل دسترسي | مزايا | معايب |
| --- | --- | --- |
| مدل ليست كنترل دسترسي | - سادگي مدل  - بررسي سريع مجوزها براي يك سند و يا يك المان يا زيرالمان | - براي به دست آوردن همه مجوزهاي يك كاربر، بايد تمام ليست‌هاي كنترل دسترسي بررسي شود  - كارايي پس‌گرفتن مجوزها بر اساس حذف كاربر يا شئ ضعيف مي‌باشد.  - بزرگ شدن ليست در بانك‌هاي اطلاعات حجيم  - عدم پشتيباني از مسيرهاي شامل // ، \* و گزاره‌ها  - در عمل از كارايي پاييني برخوردار مي‌باشد  - در بانك‌هاي XML نامتمركز به راحتي قابل استفاده نمي‌باشد |
| مدل مبتني بر نقش | - سادگي و قابل فهم بودن مدل  - اعمال محدوديت براي كاربران و نقش‌ها  - امكان استفاده از نقش‌هاي پارامتريك  - بررسي سريع مجوزهاي يك كاربر | - پيچيدگي گراف نقش براي بانك‌هاي اطلاعات با داده‌هاي زياد  - نياز به حافظه زياد جهت محاسبات  - امكان ايجاد پيچيدگي در مدل توسط محدوديت‌ها  - داشتن مشكل با مسيرهاي شامل // ، \* و گزاره‌ها  - در بانك‌هاي XML نامتمركز به راحتي قابل استفاده نمي‌باشد |
| مدل مبتني بر تابع | - پياده‌سازي آسان در نرم‌افزارهاي كاربردي  - سرعت بالا در بررسي مجوزها  - قابليت توسعه مدل  - پشتيباني از مسيرهاي شامل // ، \* و گزاره‌ها  - دارا بودن معماري باز جهت توسعه  - ايجاد كنترل دسترسي در پايين‌ترين سطح | - پياده‌سازي سياست‌ها در توابع  - وابستگي به زبان برنامه‌نويسي  - عدم قابليت حمل  - انعطاف‌پذيري پايين  - استفاده مستقيم از حافظه اصلي  - عدم استفاده در بانك‌هاي XML نامتمركز |
| مدل مبتني بر نگاشت بيتي | - كارايي بالا در پس‌گرفتن مجوزها بر اساس حذف كاربر، سند، المان و يا زير المان  - سرعت بالا در بررسي مجوزها  - قابليت توسعه  - پشتيباني از مسيرهاي شامل // ، \* و گزاره‌ها  - ايجاد كنترل دسترسي در پايين‌ترين سطح  - قابل استفاده در بانك‌هاي XML نامتمركز | - حجيم شدن مكعب دسترسي در بانك‌هاي اطلاعات با داده‌هاي بسيار زياد |
| مدل پيشنهادي | - سادگي و قابل فهم بودن مدل  - سرعت بالا در بررسي مجوزها  - كارايي بالا در پس‌گرفتن مجوزها بر اساس حذف كاربر، سند، المان و يا زير المان  - قابليت توسعه  - پشتيباني از مسيرهاي شامل // ، \* و گزاره‌ها  - ايجاد كنترل دسترسي در پايين‌ترين سطح |  |

1. جمع‌بندي و پيشنهادها

امنيت در بانك‌هاي اطلاعات و به‌تبع آن در بانك اطلاعات XML ، براساس مدل‌هاي كنترل دسترسي ايجاد مي‌گردد. در اين پايان‌نامه، ابتدا به بررسي اجمالي امنيت و چند مدل كنترل دسترسي مهم در بانك اطلاعات XML پرداختيم. مدل كنترل دسترسي مبتني بر نقش با وجود سادگي و قابل فهم بودن آن معايب و مشكلاتي با بانك‌هاي اطلاعات XML دارد. بخصوص اينكه اگر حجم بانك اطلاعات XML زياد شود. مدل كنترل دسترسي مبتني بر تابع براي فراهم كردن كنترل دسترسي مقياس‌پذير و بامعني جهت بانك‌هاي اطلاعات XML بررسي گرديد. اين مدل داراي معايبي است كه باعث مي‌شود در بانك‌هاي XML بزرگ با تراكنش‌هاي زياد قابل استفاده نباشد. مدل كنترل دسترسي مبتني برتابع بيشتر براي انباره‌هاي داده XML مناسب مي‌باشند كه در آنها تغييرات و تراكنش‌ها به ندرت انجام مي‌گيرد. مدل كنترل دسترسي مبتني بر نگاشت بيتي نيز بررسي گرديد. اين مدل با وجود قابليت‌هاي بسيار، در بانك‌هاي اطلاعات XML بزرگ با مشكل مواجه مي‌شود. در اين پايان‌نامه با نشان دادن مشكل اصلي مدل مبتني بر نگاشت بيتي راه حلي جهت بهبود اين مشكل نيز ارائه گرديد. در اين راه حل ما از مفهوم مكعب امنيت بهينه استفاده نموديم.

در ادامه مدلي پيشنهاد گرديد كه بر پاية نگاشت بيتي و مكعب امنيت بهينه پيشنهاد شده بنا گرديده است. در مدل پيشنهادي همچنين از يك تابع دستور جهت بررسي سياست‌هاي دسترسي و ايجاد مكعب امنيت بهينه استفاده مي‌كنيم. مدل پيشنهادي قابليت به‌روزرساني مجوزها در مكعب امنيت بهينه را دارا مي‌باشد. در مدل پيشنهادي سياست‌هاي كنترل دسترسي در قالب مدل گراف نقش به سيستم معرفي و نگهداري مي‌شود. مدل پيشنهادي به دليل بنا شدن بر پايه مكعب بيتي [20، 21] از سرعت و كارائي بالا در بررسي مجوزها و اعمال كنترل دسترسي برخوردار است. همچنين مدلي قابل توسعه و مقياس‌پذير بوده و به دليل پياده‌سازي در DBMS وابستگي به زبان برنامه‌نويسي ندارد. مدل پيشنهادي از معماري كاملاً مستقل برخوردار است و تقريباً تمامي مشكلات كنترل دسترسي موجود در بانك اطلاعات XML را برطرف مي‌نمايد.

در انتها نيز مقايسه‌اي بين مدل‌هاي مختلف كنترل دسترسي و مدل پيشنهادي انجام گرفت كه اين مقايسه در يك جدول نشان داده شد. در اين جدول به‌طور خلاصه معايب و مزاياي هر مدل بيان گرديد.

بهينه‌سازي توابع به‌روزرسان و تابع دستور استفاده شده در اين مدل براي كارهاي آتي پيشنهاد مي‌گردد. همچنين مي‌توان روي حساسيت اسناد تمركز نمود و روش‌هايي جهت بارگذاري مكعب‌هاي امنيت بهينه در حافظه اصلي سيستم به منظور افزايش سرعت دسترسي ارائه كرد. به عبارت ديگر يكي از راه‌هايي كه مي‌توان جهت بهينه‌سازي و بالا بردن سرعت كنترل دسترسي انجام داد استفاده از قابليت حساسيت اسناد در مدل مي‌باشد. به اين منظور مي‌توان براي هر يك از مكعب‌هاي امنيت بهينه توابعي درنظر گرفته شود كه وظيفه كنترل حساسيت اسناد را داشته و مكعب‌هاي امنيت بهينه را با توجه به حساسيت سند بروزرساني نمايند. از ديگر كارهايي كه پيشنهاد مي‌گردد، توسعه و بكارگيري اين مدل براي محيط‌هاي توزيع‌شده مي‌باشد. چرا كه بروزرساني كنترل دسترسي در دوره‌هاي مختلف در محيط‌هاي توزيع شده با چالش‌هايي مواجه مي‌باشد. همچنين استفاده از اين روش براي بانك‌هاي اطلاعات XML كه داراي شِماي مشخصي نمي‌باشند نيز توصيه مي‌گردد.

مراجع

مراجع

1. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler. "*Extensible Markup Language (XML) 1.0*". World Wide Web Consortium (W3C). http://www.w3c.org/TR/REC-xml. Fourth Edition, 29 September 2006. (Visited on 2008-02-09).
2. R.S. Sandhu, E. J. Coyne, H.L. Feinstein, and C.E. Youman. "*Role-Based Access Control Models*". IEEE Computer, Volume 29, No 2, pp.38-47, February 1996.
3. L. Gong. "*A Secure Identity-Based Capability System*". Proc. IEEE Symposium on Security and Privacy, pp.56-65, 1989.
4. J. Wang and S. L. Osborn, "*A Role Based Approach to Access Control for XML Databases*", SACMAT’04 of ACM, page 70-77, June 2004.
5. E. Bertino, S. Castano, E. Ferrari, M. Mesiti, "*Specifying and Enforcing Access Control Policies for XML Document Sources*", World Wide Web, pp. 139-151, Vol. 3, 2000.
6. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "*Controlling Access to XML Documents*", In IEEE Internet Computing, pp. 18-28, Vol. 5, No. 6, 2001.
7. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati "*A Fine-Grained Access Control System for XML Documents*". ACM TISSEC, pp.169-202, 2002.
8. A. Gummadi, J. P. Yoon, B. Shah, V. Raghavan, "*A bitmap-based access control for restricted views of XML documents*", ACM Workshop on XML Security, pp. 60-68, October 2003.
9. M. Kudo, J. Myllymaki, H. Pirahesh and N. Qi, "*A Function-Based Access Control Model for XML Databases*", CIKM’05 of ACM, page 115-122, November 2005.
10. E. Funderburk, et al., “*XTABLES: Bridging Relational Technology and XML*”, IBM system journal, vol. 41, number 4, pp: 616-641, 2002.
11. مصطفي حق جو، علي‌اصغر صفائي. *"بانك اطلاعات علمي- كاربردي"*، جلد دوم، انتشارات دانشگاه علم و صنعت ايران، 1387
12. F. Rabitti, E. Bertino, W. Kim, and D. Woelk. "*A model of authorization for next-generation database systems*". ACM Trans Database Syst, 16(1):88–131, 1991.
13. C. M. Ionita and S. L. Osborn. "*Privilege administration for the role graph model*". In Research Directions in Data and Applications Security, Proc. IFIP WG11.3 Working Conference on Database Security, Kluwer Academic Publishers, pages 15–25, 2003.
14. S. Osborn and Y. Guo. "*Modeling users in role-based access control*". In Fifth ACM Workshop on Role-Based Access Control, pages 31–38, July 2000.
15. D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. "*Proposed NIST standard for role-based access control*". ACM TISSEC, 4(3):224–275, 2001.
16. E. Bertino and E. Ferrari. "*Secure and selective dissemination of XML documents*". ACM TISSEC, 5(3):290–331, 2002.
17. T. Yu, D. Srivastava, L.V.S. Lakshmanan, and H.V. Jagadish, "*Compressed Accessibility Map: Efficient Access Control for XML*". VLDB, pp.478-489, 2002.
18. M. Murata, A. Tozawa, M. Kudo and H. Satoshi, "*XML Access Control Using Static Analysis*". ACM CCS, 2003.
19. C. Chan, Y. Ioannidis. “*Bitmap Index Design and Evaluation*”, In Proceedings of the International ACM SIGMOD Conference, pp. 355-366, 1998.
20. J. Yoon, V. Raghavan, V. Chakilam, L. Kerschberg, “*BitCube: A Three-Dimensional Bitmap Indexing for XML Documents*”, In Journal of Intelligent Information Systems, pp. 241-254, Vol. 17, 2001.
21. J. Yoon, "*High Speed Access Control for XML Documents*" IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 7, pp. 971-987, July 2006.
22. V. Parmar, S. Hongchi, S Chen, “*XML Access Control for Semantically Related XML Documents*”, In Proceedings of the 36th Annual Hawaii International Conference, pp. 288–297, 2003.
23. S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, J. Siméon, "*XQuery 1.0: An XML Query Language*", World Wide Web Consortium (W3C), http://www.w3.org/TR/xquery, January 2007. (Visited on 2008-03-17).
24. A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, J. Siméon, "*XML Path language (XPath) 2.0*", World Wide Web Consortium (W3C), http://www.w3.org/TR/xpath20 , January 2007. (Visited on 2008-03-17).
25. T. Fiebig, S. Helmer, C. Kanne, G. Moerkotte, J. Nemnann, R. Schiele, T. Westmann, "*Anatomy of a native XML base management system*" , VLDB Journal, Vol. 11, Issue 4, Pages 292-314, Dec 2002.
26. E.R. Harold, "*Managing XML data: Native XML databases*", IBM DeveloperWorks XML, http://www-128.ibm.com/developerworks , Jun 2005.
27. X. Meng, D. Luo, J. Ou: "*An Extended Role Based Access Control Method for XML Documents*",Wuhan University Journal of Natural Sciences, Vol. 9, No. 5, pp. 740-744, September 2004.
28. M. Tamer Özsu, Patrik Valduriez, "*Principles of Distributed Database Systems*", Prentice-Hall International, Upper Saddle River, N.J., 1999.
29. R. Sandhu, E. Coyne, H. Feinstein, C. Youman. "*Role-Based Access Control Models*". IEEE Computer, 29:38–47, Feb. 1996.
30. R. Sandhu, V. Bhamidipati, Q. Munawer. "*The ARBAC97 model for role-based administration of roles*". ACM Transaction on Information and Systems Security, 2(1):105–135, Feb. 1999.
31. S. Osborn, R. Sandhu, Q. Munawer. "*Configuring role-based access control to enforce mandatory and discretionary access control policies*". ACM Trans. Information and System Security, 3(2):1–23, 2000.
32. A. Gabillon, E. Bruno, "*Regulating Access to XML Documents*". Working Conference on Database and Application Security, pp.219-314, 2001.
33. S. Cho, S. Amer-Yahia, L.V.S. Lakshmanan, D. Srivastava, "*Optimizing the secure evaluation of twig queries*". VLDB, pp.490-501, 2000.
34. M. Kudo and S. Hada, "*XML Document Security based on Provisional Authorization*". ACM CCS, pp.87-96, 2000.
35. A. Gabillon, M. Munier, JJ. Bascou, L. Gallon, E. Bruno. "*An Access Control Model for Tree Data Structure*". Proceedings of the 5th International Conference on Information Security, pp.117-135, October 2002.
36. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "*Securing XML Documents*", Proceedings of the 7th International Conference on Extending Database Technology (EDBT2000), pp.121-135, March 2000.
37. D. Chamberlin, D. Florescu, J. Melton, J. Robie, J. Siméon, "*XQuery Update Facility 1.0*", World Wide Web Consortium (W3C), http://www.w3.org/TR/2008/CR-xquery-update-10-20080314, March 2008. (Visited on 2008-06-10).

پيوست‌ها

بخش‌هايي از كد پياده‌سازي شده

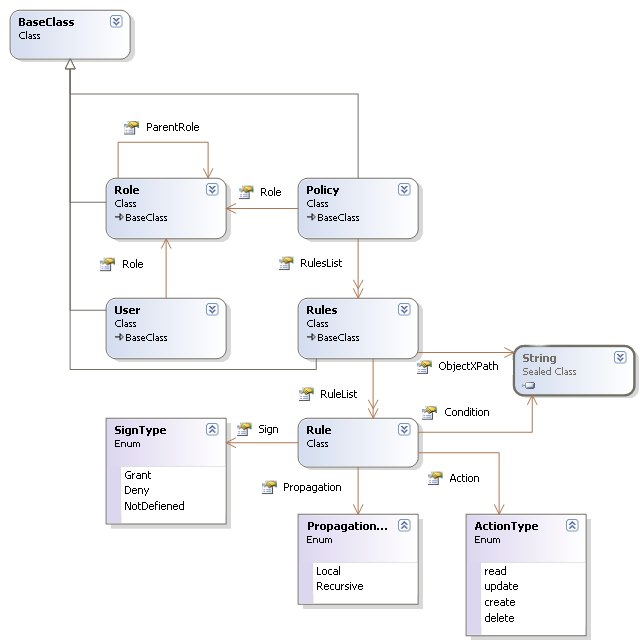
در اينجا بخش‌هايي به صورت نمونه از كد پياده‌سازي شده ذكر مي‌گردد. اين كدها در محيط Microsoft Visual Studio 2005 و با استفاده از زبان C# پياده­سازي شده‌اند. از XPath به عنوان زبان پرس‌وجو روي پايگاه داده استفاده مي­شود. قسمت‌هاي مختلف سيستم به صورت DLL هايي مي‌باشد که در برنامه‌هاي کاربردي ديگر که از پايگاه داده‌هاي XML استفاده مي‌کنند به راحتي قابل استفاده است. به عنوان نمونه يکي از بخش‌هاي سيستم برنامه AccessAdminsitration مي‌باشد که با معرفي فايل‌هاي XML به آن، مدير سيستم مي‌تواند نقش‌ها و سياست‌هاي سيستم را تعريف و مديريت نمايد.

* لايه PolicyManagement

کار لايه PolicyManagement ارتباط با فايل‌هاي کاربران، نقش­ها و سياست­­هاي سيستم است. در واقع وظيفه اصلي اين لايه مديريت نقش‌ها و سياست‌هاي سيستم است. بازيابي ليست کاربران، نقش‌ها و سياست‌ها و همچنين تغيير، حذف و اضافه کردن سياست‌ها در اختيار اين لايه مي‌باشد. لايه­هاي بالاتر براي دسترسي به سياست­هاي سيستم نمي­توانند مستقيماً فايل‌هاي XML مربوط به سياست­هاي سيستم را بخوانند. کلاس دياگرام لايه PolicyManagement به صورت شکل 1 است.

در اينجا سه تا کلاس قابل شمارش[[37]](#footnote-37) وجود دارد که مربوط به خواص قانون هستند توضيحات و مقادير آنها در جدول زير آمده است.

| نام شمارنده | توصيح شمارنده | مقدار | توضيح مقدار |
| --- | --- | --- | --- |
| Sign | علامت دسترسي که مشخص مي­کند که حق دسترسي داده مي­شود يا نه. | GRANT | دسترسي به گره داده مي­شود که معادل "+" مي­باشد. |
| DENY | دسترسي به گره داده نمي­شود که معادل "-" مي­باشد. |
| NotDefiened | اگر براي گره­اي در درخت سياست قانون تعريف نشده باشد با اين علامت برچسب مي­خورد. |
| Propagation | نوع انتشار که مشخص مي­کند آيا علامت دسترسي به گره­هاي فرزند گره موجود در قانون منتشر شوند يا نه. | Local | علامتي که به گره داده شده است به گره فرزند منتشر نشود. |
| Recursive | علامتي که به گره داده شده است به گره فرزند منتشر شود. |
| ActionType | نوع عملي که قانون براي آن تعريف مي­شود. | Read | عمل خواند گره |
| Update | عمل به روزرساني گره |
| Create | عمل ايجاد گره |
| Delete | عمل حذف گره |



شكل 1- کلاس دياگرام لايه PolicyManagement

کلاس Rule نشان دهنده قانون است که داراي يک علامت (Sign)، انتشار (Propagation)، عمل دسترسي (Action) و شرط دسترسي (Condition) مي­باشد اگر براي قانون شرط تعريف نشده باشد فيلد Condition آن قانون خالي مي­باشد.

هر شي Rules قوانيني را که براي يک objectXPath (مسير Xpath که نشان دهنده يک گره يا تعدادي گره از يک نوع است) تعريف شده­اند را در خود دارد.

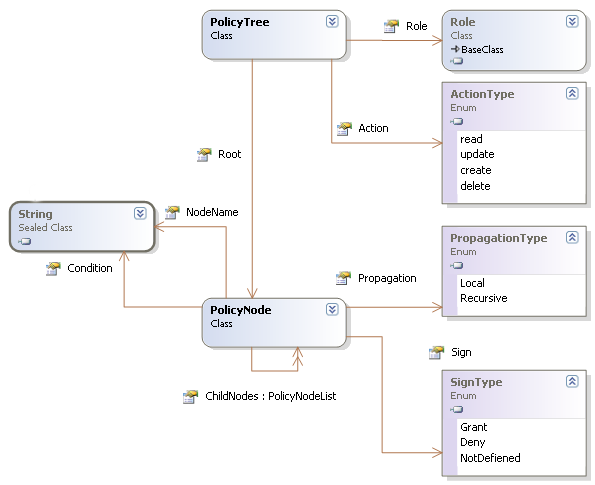
در مرحله بالاتر هر شي Policy مجموعه قوانيني را که براي يک نقش Role تعريف شده اند در خود دارد. بنابراين قوانين موجود در سيستم دسته­بندي مي­شوند. اين دسته­بندي باعث مي­شود که قوانيني که مربوط به يک نقش و گره هستند کنار هم قرار گيرند و هنگام خواندن داده­ها با فايل policies.xml داده­ها به صورت محلي مورد دسترسي قرار گيرند.

کلاس Role نشان دهنده نقش مي­باشد از آنجا که رابطه نقش­ها با يکديگر به صورت درختي است پس کافيست که در هر شي Role اشاره به نقش پدر آن يعني ParentRole را نگهداري کنيم. چون ريشه درخت نقش­ها نقش پدر ندارد بنابراين مقدار ParentRole مربوط به آن null مي­باشد. کلاس User نشان دهنده کاربر مي­باشد و هر شي User يک نقش Role دارد چون همانطور که گفتيم در سيستمي که ما پياده‌سازي کرده­ايم به هر کاربر يک نقش را مي­دهيم. اگر مي­خواستيم به هر کاربر چندين نقش را بدهيم در اينصورت بايد کلاس User داراي آرايه­اي از اشياء کلاس Role باشد. چهار تا کلاس Policy، Role، User و Rules از کلاس پايه BaseClass مشتق مي­شوند که در اين کلاس متدهاي پايه تعريف شده­اند.

اين لايه نمايي شي­گرا از سياست­هاي سيستم را بوجود مي­آورد به زباني ديگر در اين لايه کلاس‌هايي وجود دارد که نشان­دهنده کاربران، نقش­ها و سياست­هاي سيستم هستند که در فايل­هاي XML ذخيره شده­اند. بعد از کامپايل اين لايه به صورت يک DLL مي­باشد و از آن مي‌توان در پروژه­هاي ديگر استفاده نمود.

* لايه AccessDecision

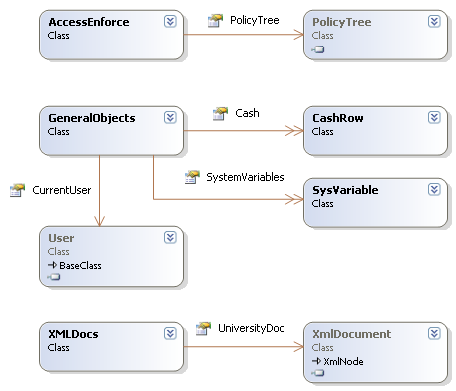
اين لايه کار ارزيابي دسترسي را انجام مي­دهد و مهمترين سرويسي که مي­دهد اين است که درخواست را گرفته و بررسي مي­کند آيا دسترسي مي تواند انجام شود يا نه، اين سرويس­ها توسط لايه AccessEnforcement مورد استفاده قرار مي­گيرند. اين لايه کار بررسي دسترسي را با استفاده از لايه PolicyManagement انجام مي­دهد در واقع ليست سياست­هاي مرتبط با درخواست را از لايه PolicyManagement گرفته و طبق مکانيزم مكعب امنيت بهينه که در اين لايه پياده­سازي شده است کار ارزيابي دسترسي را انجام مي­دهد. کلاس دياگرام اين لايه به صورت شکل 2 مي­باشد.



شكل 2- كلاس دياگرام لايه AccessDecision

* لايه AccessEnforcement

براي دسترسي به داده‌هاي پايگاه داده درخواست­ها به اين لايه تحويل داده مي­شوند اين لايه نيز با استفاده از لايه AccessDecision بررسي مي­کند که آيا درخواست قابل انجام است و با توجه با آن تصميم مي­گيرد عمليات درخواست روي پايگاه داده اجرا شود يا نه و اگر اجرا شود روي چه داده‌هايي اجرا شود. بنابراين لايه‌هاي بالاتر نرم افزار خودشان به پايگاه داده وصل نمي­شوند و درخواست­هايشان را روي آن اجرا نمي‌کنند بلکه درخواست­هاي خود را تسليم اين لايه مي­کنند و اين لايه بعد از کنترل دسترسي درخواست را روي پايگاه داده اجرا مي­کند. لايه AccessDecision کار بررسي دسترسي را با استفاده از مكعب امنيت بهينه انجام مي­دهد اگر اندازه اين مكعب امنيت خيلي بزرگ باشد بررسي يک دسترسي مستلزم هزينه زماني خواهد بود. براي کاهش اين هزينه زماني، قسمتي از نتايج بررسي دسترسي در اين لايه در حافظه نگهداري مي­شود. به اين ترتيب براي بررسي دسترسي درخواست ابتدا به کش به مراجعه مي­شود اگر نتيجه بررسي دسترسي در کش وجود نداشت بررسي دسترسي را با استفاده از لايه AccessDecision انجام داده و نتيجه را نيز در کش ذخيره مي­کند. کلاس دياگرام اين لايه به صورت شکل 3 مي­باشد.



شكل 3- كلاس دياگرام لايه AccessEnforcement

در ادامه بخش كوچكي از كد منبع سيستم ذكر مي‌گردد.

public class AccessEnforce

{

ActionType action = ActionType.read;

private string objectURI;

public PolicyTree policyTree;

public int roleID;

public AccessEnforce();

public ActionType Action { get; set; }

public string ObjectURI { get; set; }

public PolicyTree PolicyTree { get; }

public int RoleID { get; set; }

public XmlNodeList GetChilds(XmlNode node, string ObjectXpath);

public void ReloadPolicyTree();

public string ResultAccess(string objectXpath);

}

public string ResultAccess(string objectXpath)

{

try

{

string resultAccess = "";

int index = GeneralObjects.Cash.IndexOf(roleID, objectXpath);

if (index >= 0)

{

resultAccess = GeneralObjects.Cash[index].AccessResult;

}

else

{

resultAccess = this.PolicyTree.AccessTest(objectXpath);

CashRow row = new CashRow();

row.RoleID = roleID;

row.ObjectXPath = objectXpath;

row.AccessResult = resultAccess;

GeneralObjects.Cash.Add(row);

}

foreach (SysVariable item in GeneralObjects.SystemVariables)

{

resultAccess = resultAccess.Replace(item.Name, item.Value);

}

return resultAccess;

}

catch (Exception ex)

{

throw ex;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

public void PropagateSign()

{

try

{

if (this.sign == SignType.NotDefiened)

this.sign = SignType.رد;

if (this.Sign == SignType.Grant)

if (this.Propagation == PropagationType.Recursive)

{

foreach (PolicyNode node in this.ChildNodes)

{

if (node.Sign == SignType.NotDefiened)

{

node.Sign = SignType.Grant;

node.propagation = PropagationType.Recursive;

}

}

}

if (this.sign == SignType.رد)

{

foreach (PolicyNode node in this.ChildNodes)

{

node.Sign = SignType.رد;

node.Propagation = PropagationType.Recursive;

}

}

foreach(PolicyNode item in this.ChildNodes)

{

item.PropagateSign();

}

}

catch (Exception ex)

{

throw ex;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

public class PolicyNode

{

private string nodeName;

private string condition = "";

private SignType sign = SignType.NotDefiened;

private PropagationType propagation = PropagationType.Local;

private PolicyNodeList childNodes;

public PolicyNode(string nameOfNode);

public PolicyNodeList ChildNodes { get; set; }

public string Condition { get; set; }

public string NodeName { get; set; }

public PropagationType Propagation { get; set; }

public SignType Sign { get; set; }

public PolicyNode Copy();

public PolicyNode FindNode(string name);

public void PropagateSign();

}

**Abstract:**

Nowadays, information on the web is rapidly increasing and a large a mount of this information is in XML format. One of the advantages of XML usage is showing of non-structured data which provides users with many abilities. The XML’s attribute of being non-structured and the flexibility of data made it popular and it is being used in databases too. Hence, providing security in XML documents is necessary. No matter in what format data are saved, they should be protected against possible threats (data stealing, damage, manipulation, hack and so on). To prevent these threats, methods and models have been developed in databases. The most important model is access control. This model is based on several methods that are used in different databases. In this dissertation, after analyzing access control methods, a new bitmap-based and function-based method in order to control access has been proposed. In this method the attempt was to resolve all problems and flaws of the bitmap-based and function-based method.

**Keywords:** XML Database, Access control, Security, Bitmap.



Iran University of Science and Technology

Computer Engineering Department

A new access control model in native XML Databases based on Bitmap and Function

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Computer Engineering

By:

Vahid Alinaghizadeh

Supervisor:

Dr. Mostafa Haghjoo

July 2008

1. Metadata [↑](#footnote-ref-1)
2. Element [↑](#footnote-ref-2)
3. sibling [↑](#footnote-ref-3)
4. ancestor [↑](#footnote-ref-4)
5. Schema [↑](#footnote-ref-5)
6. Well-form [↑](#footnote-ref-6)
7. Grammar-based [↑](#footnote-ref-7)
8. Tree-structure [↑](#footnote-ref-8)
9. Native XML [↑](#footnote-ref-9)
10. XML Enabled [↑](#footnote-ref-10)
11. Data centric [↑](#footnote-ref-11)
12. Document centric [↑](#footnote-ref-12)
13. Path expression [↑](#footnote-ref-13)
14. Secrecy [↑](#footnote-ref-14)
15. Integrity [↑](#footnote-ref-15)
16. Availability [↑](#footnote-ref-16)
17. Discretionary [↑](#footnote-ref-17)
18. Mandatory [↑](#footnote-ref-18)
19. Security classes [↑](#footnote-ref-19)
20. Clearance [↑](#footnote-ref-20)
21. Access Control List (ACL) [↑](#footnote-ref-21)
22. Capability [↑](#footnote-ref-22)
23. RBAC: Role-Based Access Control [↑](#footnote-ref-23)
24. Junior [↑](#footnote-ref-24)
25. User-role [↑](#footnote-ref-25)
26. Group-role [↑](#footnote-ref-26)
27. Granularity [↑](#footnote-ref-27)
28. Contains [↑](#footnote-ref-28)
29. Authorization Association Matrix [↑](#footnote-ref-29)
30. Browsing [↑](#footnote-ref-30)
31. Authoring [↑](#footnote-ref-31)
32. Model Deployment [↑](#footnote-ref-32)
33. Element Path [↑](#footnote-ref-33)
34. Restricted View [↑](#footnote-ref-34)
35. Derivation rule [↑](#footnote-ref-35)
36. Bitcube [↑](#footnote-ref-36)
37. Enumerator [↑](#footnote-ref-37)