ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 51.00.00.000 ПЗ

Група ІП-19-1К

Собчишин Ростислав

2022

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: “ Моделі контролю доступу та представлень структурованих та неструктурованих даних ”

(назва згідно з наказом ректора)

спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 51.00.00.000 ПЗ

(позначення)

Консультанти:

Студент

ІП-19-1К /Р.А.Собчишин/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

к.т.н., доц. /В.І.Шекета/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

д.т.н.,проф. /В.І.Шекета/
(посада) (підпис) (дата) (розшифровка підпису)

з питань нормоконтролю

доц., к.т.н /В.Я. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерія програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
проф. В.І. Шекета
“ ____ ” _____2022 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Собчишину Ростиславу Андрійовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Моделі контролю доступу та представлень структурованих та неструктурованих даних”

керівник проекту (роботи) Шекета Василь Іванович, професор

затверджені наказом закладу вищої освіти від “ __ ” __травня__ 2022 р. № ___

2. Строк подання студентом проекту (роботи) 17 червня 2022 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Вступ до проблеми
2. Огляд існуючих концепцій, рішень та сервісів в даній області
3. Побудова моделі або алгоритму власного рішення
4. Документування реалізації власного оригінального рішення вибраними засобами
5. Висновки по роботі

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Архітектура RBAC (рішення NIST) (рис. 1.1)
2. Ієрархічна структура RBAC (рис. 1.2)
3. Поділ обов'язків RBAC (рішення NIST) (рис. 1.3)
4. Елемент RBAC(рішення NIST)(рис. 1.4)
5. Матеріалізована архітектура сайту http://docs.oracle.com (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **25 травня 2022 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Огляд існуючих сервісів	26.05.2022	виконано
2	Побудова моделі або алгоритму власного рішення	29.05.2022	виконано
3	Розробка алгоритмів програмного засобу здійснення опитувань	30.05.2022	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	04.06.2022	виконано
5	Тестування програмного продукту	10.06.2022	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2022	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

В бакалаврській роботі подано інформаційну структуру, яка підвищує безпеку на рівні концепції матеріалізованих представлень. Матеріалізовані представлення можна використовувати з міркувань продуктивності в дуже великих системах, таких як сховища даних або розподілені системи, або для надання відфільтрованого вибору даних із більш загальної бази даних. У цій роботі представлена схема контролю доступу на основі ролей для матеріалізованих представлень, у яких правила авторизації даних пов’язані з ролями та визначені в синтаксисі простих текстових файлів, обмеження рівня стовпців накладається на матеріалізоване представлення на основі призначеної користувачем ролі та визначається стратегія конфлікту ролей, у якій пріоритет надається кожній конфліктній ролі для вирішення конфліктів ролей, якщо користувач одночасно отримує дозвіл на права, пов’язані з конфліктними ролями.

КЛЮЧОВІ СЛОВА: РІВЕНЬ ДОСТУПУ, МАТЕРІАЛІЗОВАНІ ПРЕДСТАВЛЕННЯ, БАЗА ДАНИХ, РОЛЬ КОРИСТУВАЧА, ПРІОРИТЕТ, РОЗПОДІЛЕНА СИСТЕМА.

ANNOTATION

The bachelor thesis presents an information structure that increases security at the level of the concept of materialized representations. Materialized views can be used for performance reasons in very large systems such as data warehouses or distributed systems, or to provide a filtered selection of data from a more general database. This paper presents a role-based access control scheme for materialized views, in which data authorization rules are role-bound and defined in plain text file syntax, column-level restriction is imposed on the materialized view based on the user-assigned role, and a role conflict strategy is defined. in which priority is given to each conflicting role to resolve role conflicts if the user is simultaneously granted rights associated with the conflicting roles.

KEYWORDS: ACCESS LEVEL, MATERIALIZED REPRESENTATIONS, DATABASE, USER ROLE, PRIORITY, DISTRIBUTED SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 9

ВСТУП 10

1 **ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ В ПРЕДМЕТНІЙ ОБЛАСТІ** 12

1.1 Методи контролю доступу до даних 13

1.2 Матеріалізований вигляд 18

1.3 Перегляд авторизації 20

1.4 Переписування запитів 22

1.5 Постановка проблеми 23

1.6 Структура побудованих представлень 24

1.7 Висновки до розділу 25

2 **АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ РІШЕННЯ ДЛЯ ЗАДАЧ ДАНОГО КЛАСУ** 26

2.1 Контроль доступу на основі контексту (CBAC) 26

2.2 Контроль доступу на основі атрибутів (ABAC) 27

2.3 Контроль доступу на основі XML 28

2.4 Авторизація доступу до реляційної бази даних 29

2.5 Детальна політика авторизації 31

2.6 Перегляди авторизації та утримання умовних запитів 30

2.7 Контроль доступу до матеріалізованих представлень 33

3 **РЕАЛІЗАЦІЯ МОДЕЛЕЙ ТА РІШЕННЯ ДЛЯ СТРУКТУРОВАНИХ ТА НЕСТРУКТУРОВАНИХ МАТЕРІАЛЬНИХ ПРЕДСТАВЛЕНЬ** 37

3.1 Пропонувана основа 37

3.2 Контроль доступу до даних до матеріалізованих представлень 37
3.3 Запропонована рольова структура 38
3.4 Реалізація та верифікація 48
3.5 Результати тесту на масштабованість 60
ВИСНОВКИ 62
ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА 59
ДОДАТКИ
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

МП – матеріалізоване представлення

БД – База даних

СУБД - Система управління базами даних

ПЗ - Програмне забезпечення

ВСТУП

Контроль доступу до даних вважався основною проблемою в інформаційно-технологічному співтоваристві. Основною метою дослідника є забезпечення механізму захисту даних та забезпечення доступу до даних на основі ідентичності та атрибутів відомих користувачів або процесу за допомогою еталонного монітора та спеціальних правил авторизації.

Політики авторизації визначені для обмеження доступу до даних на основі атрибутів або ролі користувача. Було запропоновано кілька різних підходів на основі вимог різних доменів. Є деякі дослідницькі роботи, які є загальними і можуть бути застосовані до будь-якої області. Контроль доступу на основі ролей є провідною моделлю контролю доступу завдяки своїй гнучкості та простоті обслуговування.

Зараз кількість даних і їх доступність швидко збільшуються; велика частина даних зберігається у віддалених файлових системах. Матеріалізовані представлення — це ще одне доповнення до систем управління реляційними базами даних. Матеріалізоване представлення використовує інший підхід, у якому результат запиту кешується як конкретна таблиця, яка час від часу може оновлюватися з вихідних базових таблиць. Це забезпечує набагато ефективніший доступ до даних. Це може бути локальна копія даних, розташованих віддалено, або підмножина рядків чи стовпців таблиці чи результат об’єднання, або може бути підсумок, заснований на агрегації даних таблиці.

Існуючі запропоновані методи забезпечують контроль доступу на основі правил для матеріалізованих представлень, але, наскільки нам відомо, адміністрування таких систем займає багато часу та громіздко у великому середовищі, оскільки адміністратори визначають правила для кожного користувача, які може контролювати. доступ до матеріалізованих представлень.

У цьому дослідженні ми визначили, що роль автоматизації є ще однією важливою областю дослідження. Автоматизація ролей потрібна у великих організаціях, де тисячі співробітників виконують свої обов'язки. У таких організаціях створення, призначення та зміна ролі кожного співробітника є трудомістким процесом. Наразі спеціальний відділ, наприклад відділ кадрів, призначає роль кожному співробітнику на основі його/її посадових інструкцій. Весь процес контролюється вручну; як тільки відповідний відділ призначає роль співробітнику, системний адміністратор створює новий запис у таблиці бази даних або списку контролю доступу, в якому зберігаються атрибути профілю співробітника.

У системі контролю доступу на основі ролей ланцюжок команд (ієрархії ролей) повинен бути чітко визначений, де найвища роль є першою в команді, а наступна за командою знаходиться на один рівень нижче і так далі. Кожній ролі в ієрархії ролей призначаються обов'язки, наприклад, керівник несе відповідальність за подання робочого часу своїх підлеглих.

Щоб уникнути таких затримок, нам потрібен механізм, який уповноважує кожну роль передавати свої обов'язки наступній в ієрархії ролей. Передбачається, що перенаправлення відповідальності на іншу роль в ієрархії ролей має бути тимчасовим, і під час процесу передачі має бути призначений часовий інтервал. Після призначеного часу повноваження, пов'язані з даними обов'язками, повинні бути відкликані системою.

Майбутня робота включає впровадження та інтеграцію автоматизації ролей та системи керування робочими процесами з нашою існуючою структурою.

В нашому дослідженні ми представили структуру, яка забезпечує авторизацію на основі призначеної користувачем ролі в локальній матеріалізованій базі даних представлення. Майбутня робота включає розширення фрейморку, яка дозволить користувачеві отримувати доступ до даних з віддалених баз даних у розподіленому середовищі на основі авторизації, визначеної в локальній базі даних (подання локальної авторизації) і віддалених

базах даних (подання глобальної авторизації). Локальна та глобальна авторизація повинна бути проаналізована системою, перш ніж надавати доступ користувачам. Щоб вирішити конфлікти між локальними та глобальними уявленнями про авторизацію, нам потрібно розробити стратегію вирішення таких конфліктів.

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ В ПРЕДМЕТНІЙ ОБЛАСТІ

1.1 Методи контролю доступу до даних

Декілька методів контролю доступу до даних були введені з урахуванням вимог організації та чутливості даних. У цьому розділі ми обговорюємо різні методи контролю доступу до даних.

Обов'язковий контроль доступу (MAC)

Обов'язковий контроль доступу використовує жорстко закодовані правила безпеки. Правила закодовані в програмі або операційній системі. Політика безпеки адмініструється централізовано і може бути перевизначена користувачами, і вона застосовується до різних ресурсів, об'єктів і програм. Класифікація даних політики безпеки MAC починається з конфіденційних, секретних і конфіденційних, а далі – класифікація ресурсів, які будуть робити запити на дані. Концепція MAC використовується в основному у військових та урядових програмах, де потрібен високий рівень безпеки.

Перевага цієї моделі полягає в тому, що правила жорстко закодовані в програмному забезпеченні, тому є дуже менші шанси на адміністративну помилку або соціальну інженерію.

Недоліком цієї моделі є те, що правила жорстко запрограмовані, тому потрібен час, щоб переглянути та змінити правила в міру зміни вимог. MAC найкраще підходить для групи користувачів зі схожими потребами.

Дискреційний контроль доступу (DAC)

Дискреційний контроль доступу може використовуватися як централізована та розподілена модель. Централізована модель DAC керується адміністратором або групою адміністраторів, які відповідають за створення політики безпеки та призначення привілеїв відповідно до політики, але цей підхід займає багато часу, особливо якщо адміністратор відключений або залучений стороннім

виконавцем. При розподіленому підході доступ до даних розподіляється між деякими відповідальними працівниками, такими як менеджери, керівники або команда.

Такий підхід дозволяє уникнути затримок, оскільки адміністрування рахунків розпорошено.

Недоліком цієї моделі є те, що однорідність доступу до даних для кінцевого користувача з однаковими робочими функціями може бути зменшена, оскільки доступ до даних розподіляється на розсуд власника.

Контроль доступу на основі ролей (RBAC)

У цьому конкурентному середовищі ризик втрати інформації більше для провідної організації. MAC і DAC захищають дані, але вони мають обмеження. Для подолання їх недоліків було запропоновано RBAC [24]. Оскільки засоби керування доступом на основі ролей існували протягом останніх 20 років, особливо в середовищах UNIX і мейнфреймів, але їм бракує деяких стандартів, оскільки кожна система використовує свої власні елементи. Виникла потреба розробити таку систему, яка була б стандартизованою, масштабованою, логічною за проектом і незалежною від системи.

Національний інститут стандартів і технологій (NIST) розпочав проект уніфікованих стандартів RBAC шляхом інтеграції існуючої моделі.

У 1992 році Девідом Феррайоло і Ріком Куном була представлена модель, яка намагалася задовольнити вимоги області застосування і створила повноцінне рішення RBAC.

RBAC0 є першим запропонованим методом у цій серії, ця модель складається з поділу обов'язків і надання мінімальних привілеїв кожній ролі. Він не має механізму ієрархії, тому дозволи призначалися безпосередньо користувачам у межах певної ролі чи функції.

Враховуючи потребу в ієрархії, яка існує в будь-якій організації, як-от менеджер, керівник та члени команди, RBAC1 вводиться на основі RBAC0. Він забезпечує природний розподіл обов'язків в організації, який зазвичай

поділяється на старші та молодші ролі. Цей метод багатошарового поширення безпеки підходить для великих середовищ.

Обмеження введені в RBAC2, які пропонують більший контроль над будь-якою мережею у великих середовищах. Обмеження допомагають застосовувати політику, не маючи ієрархії. Обмеження працюють як обмежувачі та забезпечують дотримання політики. Наприклад, якщо організація хоче надати адміністративні права одному користувачеві або ролі, обмеження гарантують, що лише один користувач має права адміністрування системи.

Іншою метою обмежень є регулювання доступу шляхом забезпечення відповідності певним критеріям. Наприклад, щоб отримати дозволи на роль старшого аналітика, потрібно мати членство в ролі молодшого аналітика.

Нарешті, обмеження служать цілям поділу обов'язків, обмежуючи користувачів у певному домені.

RBAC3 є завершеною моделлю RBAC, і вона складається як з ієрархії, так і з обмежень.

На рисунку 1.1 представлена архітектура RBAC.

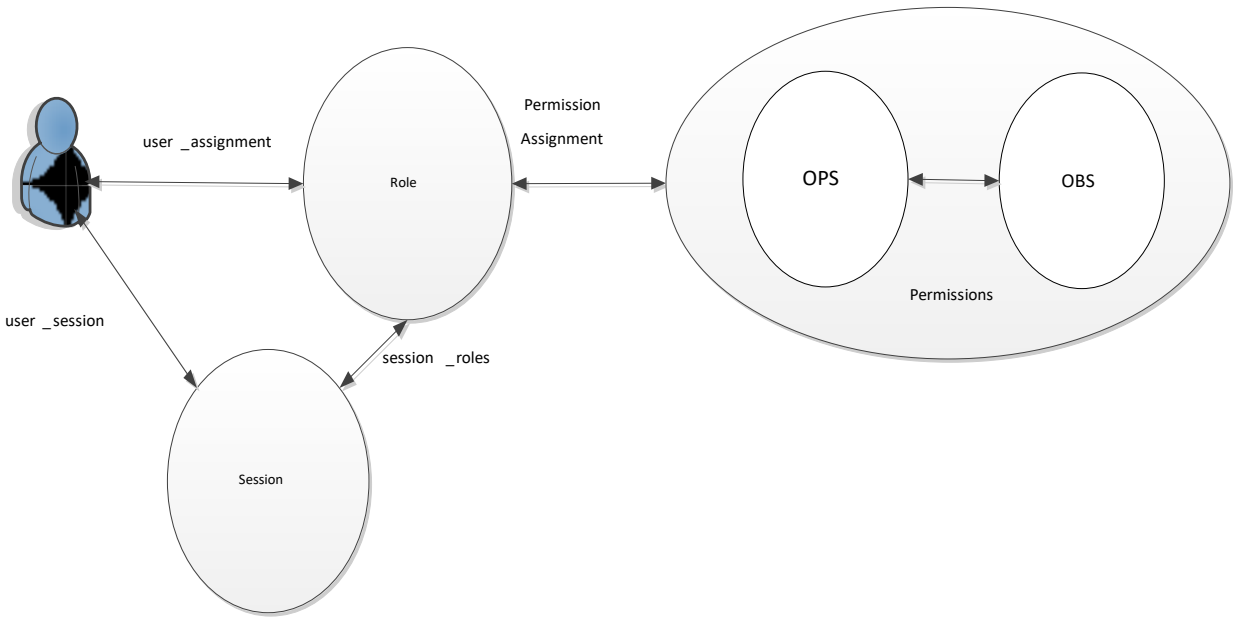


Рисунок 1.1 – Архітектура RBAC (рішення NIST)

У RBAC3 обмеження використовуються для регулювання доступу до ієрархічної структури. Наприклад, роль програміста пов’язана з роллю старшого програміста, і є кілька старших програмістів, але роль програміста пов’язана лише з певним старшим програмістом. Оскільки RBAC3 підтримує ієрархічний дизайн, він забезпечує простоту адміністрування, дозволяючи правам перетікати вниз до підпорядкованих об’єктів. На рисунку 1.2 представлена ієрархічна структура RBAC.

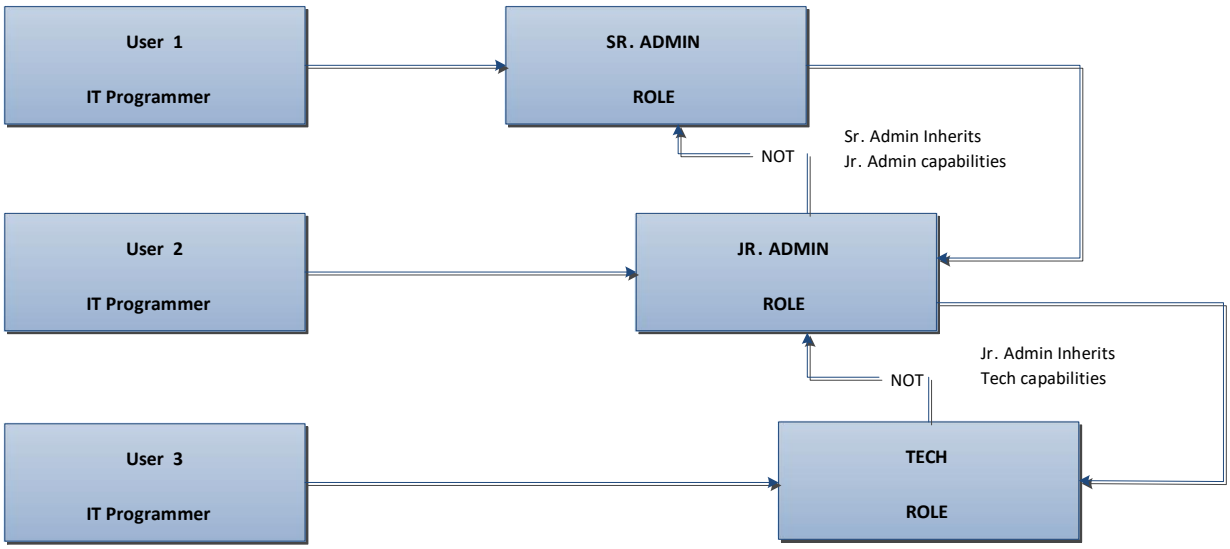


Рисунок 1.2 – Ієрархічна структура RBAC (рішення NIST)

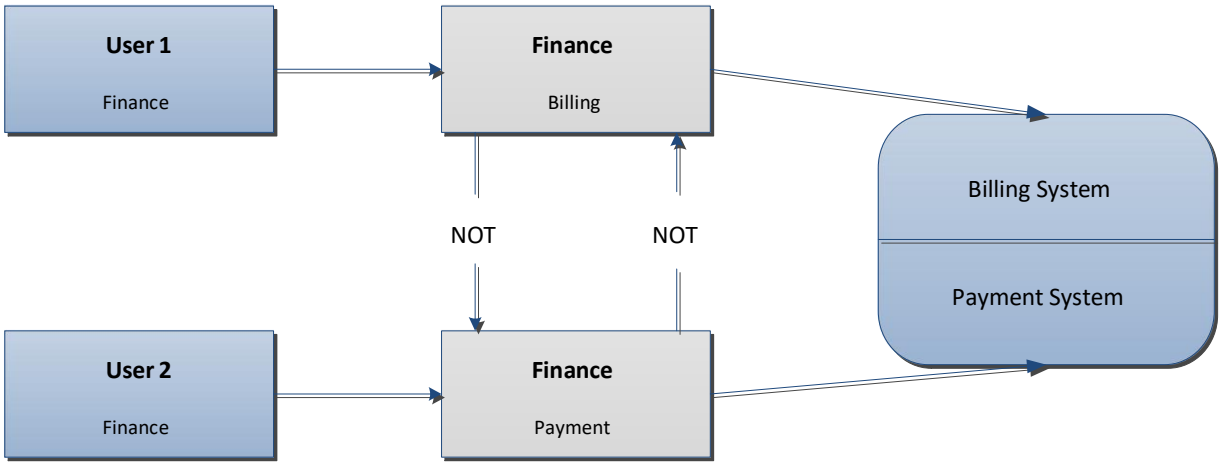


Рисунок 1.3 – Поділ обов'язків RBAC (рішення NIST)

Іншою перевагою є безліч ролей, які пов’язані одна з одною і забезпечують кращу функціональність для користувачів. Обмеження RBAC3 забезпечують поділ обов'язків; це обмежує користувачів виконувати інші завдання, які не входять до його службової функції. На рисунку 1.3 представлено розділення обов'язків RBAC (SoD).

На рисунку 1.3 «НЕ» означає, що обидві ролі не можуть бути активовані одним користувачем одночасно.

RBAC3 є складним методом, але він підходить для великих середовищ, де впровадження та обслуговування інших моделей безпеки займають багато часу та високі витрати.

RBAC3 забезпечує всі функціональні можливості, які потрібні великій організації, але реалізація є склад

ною через свою структуру. Перед впровадженням RBAC3 організаційна структура або ролі повинні бути добре задокументовані, інакше це може перетворитися на кошмар для організації.

RBAC3 має п'ять елементів, як показано на рисунку 1.4, вони пов'язані один з одним, щоб створити рівень дозволів.

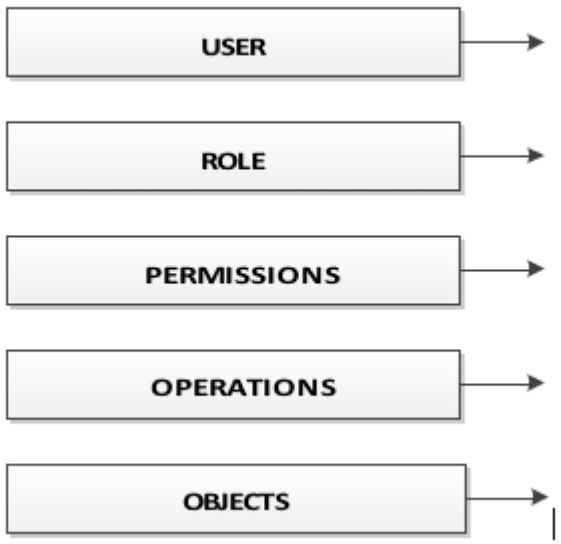


Рисунок 1.4 – Елементи RBAC (рішення NIST)

1.2 Матеріалізоване представлення

Матеріалізоване представлення — це копія таблиці або таблиць, яка створюється в розподіленому середовищі, де основна таблиця розташована в основній базі даних. Матеріалізовані представлення створюються та розгортаються у віддалених місцях у розподіленому середовищі, щоб полегшити навантаження на мережу та забезпечити безперебійне вилучення даних, оскільки для цього не потрібне виділене мережеве з'єднання з основним сервером бази даних.

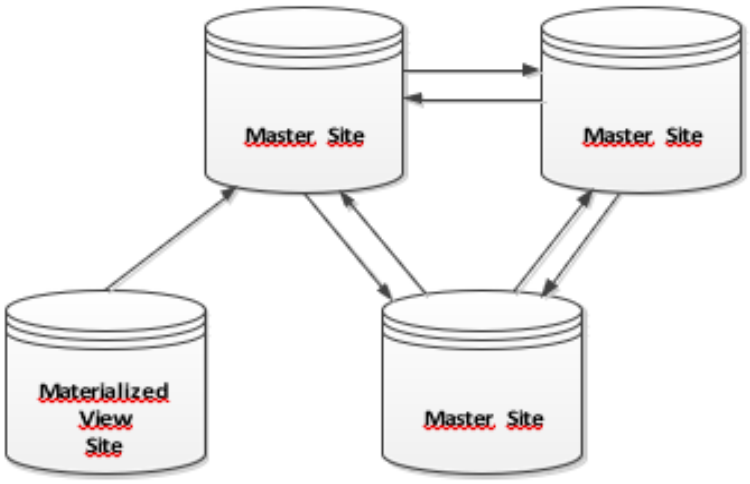


Рисунок 1.5 – Матеріалізована архітектура сайту http://docs.oracle.com

Матеріалізовані представлення можуть підвищити продуктивність виконання запиту такими способами:

• Для мінімізації дорогих обчислень під час виконання запиту можна створити попередньо обчислені агрегації.

• Таблиці можна попередньо об'єднати, а отримані дані можна зберегти у матеріалізованому поданні.

• Комбінації об'єднань і агрегацій можна зберігати з метою аналізу.

Програма, яка отримує переваги від створення/розгортання матеріалізованих представлень:

• Системи підтримки прийняття рішень

• Витрини даних

• Сховища даних

• Онлайн-аналітична обробка (OLAP)

• Робочі навантаження Data Mining

Подання — це тимчасова таблиця, яка створюється під час виконання та відкидається, коли сеанс закривається. Матеріалізоване представлення — це фізична таблиця, яка зберігається в базі даних, наприклад у традиційних таблицях бази даних. Процес створення матеріалізованого представлення відрізняється в кожній системі управління базами даних (СУБД).

Oracle підтримує матеріалізовані представлення лише для читання, оновлення та запису [27]. Користувачі не можуть маніпулювати матеріалізованими представленнями лише для читання, але вони можуть виконувати дії з матеріалізованими представленнями, які можна записувати та оновлювати.

Oracle пропонує кілька різних типів матеріалізованих уявлень, щоб відповідати різним ситуаціям реплікації.

а) Матеріалізовані погляди первинного ключа

b) Матеріалізовані погляди об'єкта

в) ROWID матеріалізовані погляди

г) Складні матеріалізовані погляди

а. Матеріалізовані погляди первинного ключа

Матеріалізовані подання первинного ключа — це тип матеріалізованих подання за замовчуванням. Вони оновлюються та створюються як частина матеріалізованої групи переглядів. Матеріалізовані представлення, які можна оновлювати, повинні знаходитися в іншій базі даних замість головної групи реплікації. Зміни відбуваються на рівні рядка, що визначається значенням первинного ключа рядка.

б. Матеріалізовані представлення об'єктів

Матеріалізоване представлення об’єкта базується на таблиці об’єктів і створюється за допомогою пропозиції типу OF. Структура така ж, як і таблиця об’єктів, і складається з об’єктів рядка, а кожен об’єкт рядка ідентифікується стовпцем ідентифікатора об’єкта (OID).

c. Матеріалізовані перегляди ROWID

Матеріалізоване подання ROWID засноване на фізичних ідентифікаторах рядків (rowids) рядків у головній таблиці. Вони використовуються з базою даних Oracle7.

d. Складні матеріалізовані погляди

Існують певні обмеження, які необхідно дотримуватися під час створення матеріалізованого подання, і якщо ці обмеження не дотримуються, то матеріалізоване уявлення вважається складним матеріалізованим уявленням, і його не можна швидко оновити.

У Microsoft SQL Server матеріалізовані подання називаються індексованими, а створення матеріалізованих представлень доступне в навчальних посібниках.

У системах баз даних з відкритим вихідним кодом, таких як MySQL, які самі по собі не забезпечують матеріалізоване представлення. Але легко створювати матеріалізовані уявлення вручну або за допомогою деяких доступних API, таких як flexviews [28], що допомагає створювати матеріалізовані уявлення та оновлювати їх.

1.3 Перегляд авторизації

Подання авторизації — це добре відома техніка бази даних, яка забезпечує детальний контроль доступу [2][3][20]. Він надає спосіб обмежити користувачам перегляд лише авторизованих даних, які вказані в доступних переглядах авторизації для цього конкретного користувача. Подання авторизації — це логічні таблиці, які не містять жодних записів. Структура цих представлень

складається з назви таблиці та кількості стовпців, які згадуються під час процесу створення авторизованого представлення.

Існують різні підходи до визначення перегляду авторизації. Motro [15] запропонував модель, в якій доступ до бази даних задається в термінах переглядів. За словами авторів, «визначається набір переглядів, і кожному користувачеві надається дозвіл на доступ до одного або кількох представлень». Користувачі запитують базу даних, а система бази даних отримує уявлення запиту, які є уявленнями, до яких користувач має дозвіл на доступ, а потім представляє доступні представлення.

Різві та ін. [20] представив параметризовані перегляди авторизації. На думку авторів, створювати та підтримувати перегляди авторизації для кожного користувача недоцільно, коли база даних має тисячі користувачів. Крім того, якщо є незначні зміни в політиці авторизації, це вплине на велику кількість переглядів. Різві та ін. [20] стверджує, що параметризовані уявлення подібні до звичайних уявлень, але є деякі додаткові параметри, такі як ідентифікатор користувача, час і місцезнаходження користувача, що з’являються у його визначенні.

Різві та ін. [20] визначає створення представлення таким чином:

- створити перегляд авторизації Co-studentGrades як вибрати Оцінки.* з Оцінки, Зареєстровано, де Registered.student-id = $user-id і Grades.course-id = Registered.course-id.

В [3] запропоновано підхід, заснований на висновках, щоб контролювати доступ до матеріалізованих представлень. За даними [3], їхній підхід «полегшує адміністрування правил контролю доступу для забезпечення конфіденційності даних на рівні матеріалізованих переглядів». Автори використовують перегляди авторизації для забезпечення детального контролю доступу до матеріалізованих уявлень. У цьому підході автори пропонують використовувати Datalog як формальну основу для вираження правил контролю доступу. Datalog — це декларативна мова програмування, яка синтаксично є підмножиною Prolog. Він

використовується як мова запитів для дедуктивних баз даних. Журнал даних більш виразний, ніж SQL; він може виконувати запити до кількох баз даних із чистішим синтаксисом і полегшує повторне використання фрагментів коду SQL для частих об'єднань і формул. В останні роки Datalog використовується для інтеграції даних, вилучення даних, хмарних обчислень, аналізу даних та безпеки.

Автори [3] припускають існування трьох типів символів: змінних, констант та імен предикатів. p(t1….tn) – це літерал, де p – ім'я предиката з арністю n, і кожен ti для $1 \leq i \leq n$ є або константою, або змінною.

На думку авторів, логічне речення пов'язане з правилом Datalog

P(u) ← q1(u1),…,qn(un) це:

x1.. xn(p(u1) ← q1(u1) ^…^ qn(un)).

Bahloul та ін. [3] стверджує, що відносини, визначені дедуктивними правилами, називаються інтенційними відносинами. Наприклад,

Info-Doc (Id-D, Dname, Dfname, Dspeciality) ← лікар (Id-D, Dname, Dfname, Dadr, Dphone, Dspeciality,Dsalary)

це правило, яке визначає (навмисне) відношення Info-Doc у термінах (розширеного) відношення лікаря.

У наведеному вище прикладі користувачеві дозволено переглядати ідентифікатор лікаря, прізвище, ім'я та спеціальність з доступного матеріалізованого перегляду.

1.4 Переписування запитів

Проблемі відповідей на запити за допомогою представлень останнім часом приділено значну увагу через проблеми керування даними. Для вирішення таких проблем запроваджено алгоритми переписування запитів. Ці алгоритми переписують запит користувача, використовуючи визначені уявлення про авторизацію, які пов'язані з традиційними таблицями бази даних або матеріалізованими представленнями. Запит переписується, щоб якнайкраще

скористатися підсумками, об’єднаннями або агрегаціями базової таблиці, які знаходяться в матеріалізованих представленнях

Існує кілька запропонованих алгоритмів переписування запитів, ми обговоримо алгоритм Bucket [18], щоб зрозуміти процес перезапису запиту в наступному параграфі.

Алгоритм Bucket створює підціль g у вихідному запиті, а потім кожна підціль g містить уявлення, які містять підцілі, на які g можна зіставити під час переписування запиту. На другому кроці алгоритм розглядає кон'юнктивне переписування запитів, кожен з яких складається з однієї кон'юнкти з кожного сегмента. Далі алгоритм перевіряє, чи міститься він або можна зробити так, щоб він містився за допомогою предикатів об’єднання в запиті. Результат роботи алгоритму Bucket є результатом перезапису кон’юнктивного запиту.

приклад:

створити перегляд авторизації MyGrades як вибір * з Оцінок, де student-id = $user-id

Нехай q буде запитом користувача.

q : виберіть середній (оцінка) з Оцінки

Змінений системою запит q′: виберіть avg(grade) з MyGrades

1.5 Постановка проблеми

У розподіленому середовищі матеріалізовані уявлення можна використовувати для реплікації даних на розподілених сайтах. Матеріалізовані представлення надають локальний доступ до даних замість доступу до даних з віддалених сайтів.

У великих організаціях матеріалізовані сайти перегляду створюються у віддалених пунктах призначення, щоб зменшити навантаження на мережу та навантаження від основного сервера бази даних. Запит користувача надсилається до локальної бази даних матеріалізованого представлення замість основного сервера бази даних. Існуючі запропоновані методи забезпечують контроль

доступу на основі правил для матеріалізованих уявлень; однак адміністрування таких систем займає багато часу та громіздко у великому середовищі, де адміністратори визначають правила для кожного користувача для контролю доступу до матеріалізованих представлень.

Щоб контролювати доступ до матеріалізованого представлення, нам потрібна структура, яка забезпечує детальний контроль доступу до матеріалізованого представлення за допомогою перегляду авторизації [3], і забезпечує доступ на основі ролі замість визначення політик для кожного користувача. У нашому дипломному дослідженні ми виявили проблему призначення переглядів авторизації ролям замість визначення для кожного користувача. Ми також помітили, що якщо наша структура дозволяє користувачам активувати кілька ролей в одному сеансі з різними дозволами, то нам потрібно надати стратегію для вирішення конфліктів ролей, які виникають, якщо користувач отримує дозвіл на дозволи, пов’язані з конфліктуючими ролями одночасно. . У нашому дипломному дослідженні ми виявили такі проблеми:

Як ми можемо призначити подання авторизації ролям?

• Як ми можемо накласти обмеження рівня стовпця на матеріалізоване представлення на основі правил, пов'язаних з ролями?

• Як ми можемо уникнути конфліктів ролей, якщо користувач отримує авторизацію на дозволи, пов'язані з конфліктними ролями?

1.6 Структура побудованих представлень

• Детальна структура контролю доступу для запобігання несанкціонованого доступу до матеріалізованих представлень

• Користувачам призначаються ролі, а користувачі отримують дозволи через перегляди авторизації

• Забезпечує обмеження рівня стовпця на основі призначеної ролі

• Накладає обмеження, щоб уникнути конфліктів між ролями

• Запропонована структура може бути реалізована для матеріалізованих сайтів перегляду в розподілених середовищах і для сховищ даних

1.7 Висновки до розділу

Отже, у даному розділі було досліджено та проаналізовано декілька методів контролю доступу до даних та були введені дані з урахуванням вимог організації та чутливості даних. У цьому розділі ми розглянули різні методи контролю доступу до даних.

2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ РІШЕННЯ ДЛЯ ЗАДАЧ ДАНОГО КЛАСУ

2.1 Контроль доступу на основі контексту (CBAC)

RBAC надає доступ на основі ролі незалежно від контексту запиту. У CBAC запит надається шляхом перевірки контексту запиту.

В [22] пропонують «семантичну модель політики контролю доступу з урахуванням контексту». Автори стверджують, що вони використовують ресурсоцентрований підхід до моделювання контексту; контексти пов'язані з ресурсами, які підлягають контролю, і представляють лише ті умови, які забезпечують доступ до ресурсів. На думку авторів, політики контролю доступу визначають для кожного контексту, як працювати з пов'язаним ресурсом. Автори описують, що контекст складається з характеризаційної інформації, яка вважається важливою для контролю доступу, такої як навантаження, операція сутності з ресурсом, ролі, ідентифікаційні дані та облікові дані безпеки, а також умови навколишнього середовища, такі як час, місце розташування та інші доступні ресурси. .

В [13] запроваджують модель RBAC (CRBAC) з урахуванням контексту, яка є розширенням RBAC. За словами авторів, у моделі NIST RBAC ролі призначаються адміністратором, і в таких системах ролі мають тривалий термін служби. У моделі CRBAC ролі визначаються як частина дизайну програми, а ролі виникають лише тоді, коли програма розгорнута та виконана. Контекст людини визначається з точки зору його/її поточного фізичного місцезнаходження, пристроїв, що використовуються, мережі, до якої пристрої підключені, і діяльності, якою зараз займається користувач.

В [8] пропонують TCAC «Модель контролю доступу на основі довіри та контексту для відкритих і розподілених систем». За словами авторів, призначення ролей ґрунтується на надійності та інформації про контекст запитувачів. Автори стверджують, що якщо значення довіри не менше

встановленого порогового значення, визначеного системними політиками, і інформація про контекст користувача задовольняє контекстним обмеженням, користувачу призначається роль і він може виконувати операції, пов'язані з призначеною роллю. Автори стверджують, що контекстні обмеження, які враховуються в їх моделі, такі як час і місце перебування користувача.

2.2 Контроль доступу на основі атрибутів (ABAC)

У ABAC доступ до даних надається на основі трьох атрибутів: атрибути суб'єкта, атрибути ресурсу та атрибути середовища.

В [5] пропонують «Систему контролю доступу на основі ролей і атрибутів», розширюючи модель керування доступом на основі ролей (RBAC) для динамічного зв'язування ролей з користувачами. Автори стверджують, що в їхній структурі пов'язані з ресурсами привілеї призначаються залежно від значень атрибутів ресурсів, значення атрибутів, пов'язані з користувачами, визначають зв'язок користувачів із привілеями, а функція відображення розташування між фізичним і логічним розташуванням дозволяє вмикати/ відключити ролі залежно від логічного розташування користувачів і таким чином зберегти конфіденційність розташування.

Автори в [7] представити модель ROWLBAC «Представляючи рольовий контроль доступу в OWL». Автори стверджують, що їхня робота визначає взаємозв'язок між мовою веб-онтології (OWL) і контролем доступу на основі ролей (RBAC). Крім того, вони додатково вивчають та оцінюють придатність OWL для вирішення двох інших проблем контролю доступу: підтримки контролю доступу на основі атрибутів і виконання аналізу безпеки в рамках системи управління довірою. За словами авторів, у їхньому підході ієрархії ролей представлені ієрархіями класів OWL, члени отримують більше привілеїв у міру просування вгору по ієрархії, тоді як в ієрархіях класів класи отримують більше атрибутів, коли користувач рухається вниз. Автори стверджують, що обмеження

доступу базуються на загальних атрибутах дії, включаючи обмеження щодо її суб'єкта та об'єкта. Автори далі стверджують, що це забезпечує загальну підтримку більш загальної моделі контролю доступу на основі атрибутів.

2.3 Контроль доступу на основі XML

EXtensible Access Control Markup Language (XACML) є одним із добре відомих форматів XML для контролю доступу. XACML — це мова політики керування доступом загального призначення, і він надає синтаксис для застосування політик контролю доступу, які допомагають керувати рішеннями щодо авторизації. Політики, визначені XACML, вирішують, чи дозволяти доступ до даних і в якому розмірі. Крім того, він надає архітектуру, яка підтримує послуги за допомогою двох об'єктів або модулів примусового виконання: точки виконання політики (PEP) і точки прийняття рішення про політику (PDP).

Архітектура XACML складається з чотирьох компонентів: точки адміністрування політики (PAP), точки ухвалення політики (PDP), точки виконання політики (PEP) та точки інформації про політику (PIP). PAP створює політики, оцінені PDP. PDP оцінює політику щодо вхідних запитів і надсилає результати до PEP. PEP здійснює контроль доступу на основі рішення про авторизацію, наданого PDP. Нарешті, PIP надає значення атрибутів, надані PDP під час процесу оцінки політики. Компонент, який отримує та передає всю інформацію між цими компонентами, називається Context Handler, який виконує роль посередника. На рисунку 2.1 представлений потік даних XACML.

В [10] пропонують фреймворк X-RBAC «мову специфікації політики контролю доступу на основі XML», який розширює NIST RBAC, і забезпечує структуру для визначення політик посередництва в багатодоменному середовищі та розширює RBAC тимчасовими обмеженнями, атрибутами ролі, контекстуальні умови, поняття рольових станів та передумови переходів станів.

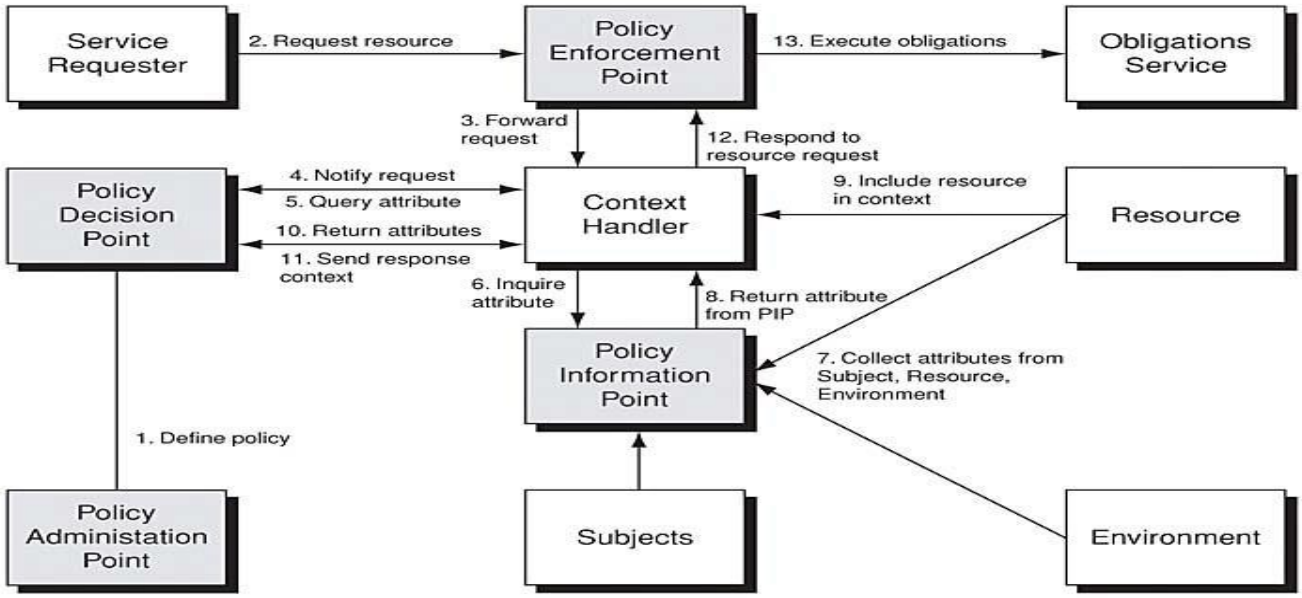


Рисунок 2.1 – Потік даних XACML

За словами авторів, X-RBAC забезпечує широкий спектр деталей захисту для захищених даних і підтримує відображення політик у багатодоменних середовищах. На рисунку 2.2 представлена модель X-RBAC.

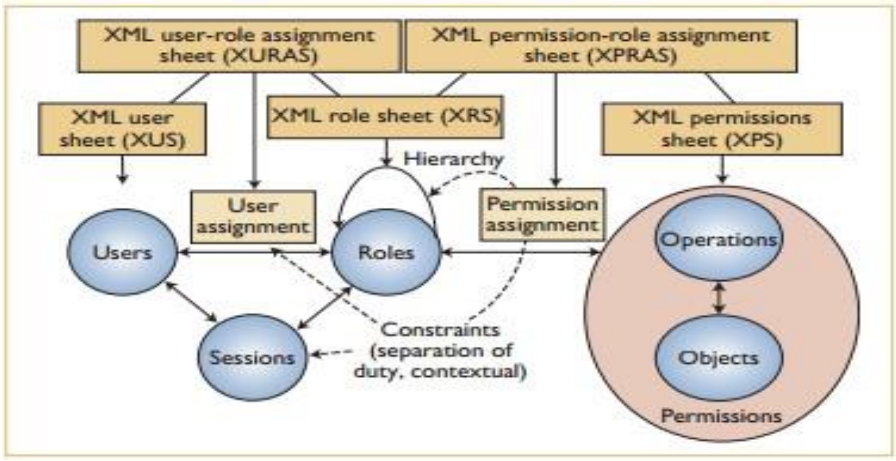


Рисунок 2.2 – Модель RBAC та компоненти політики X-RBAC

2.4 Авторизація доступу до реляційної бази даних

В [15] запропонував модель реляційної бази даних. За словами автора, «у цій моделі дозволи доступу є формою знання бази даних, з якої виводяться

дозволи доступу, які застосовуються до конкретних запитів». Існує три основні принципи цієї моделі:

1) Доступ до бази даних визначається в термінах переглядів: визначено набір переглядів авторизації для контролю доступу до даних, і кожному користувачеві надаються дозволи на основі доступних представлень.

2) Запити користувача надсилаються до фактичної бази даних, а не до будь-якого окремого представлення.

3) Коли запит надсилається в систему бази даних, система перевіряє доступні представлення запиту, які можуть бути представленнями переглядів, до яких користувач має права доступу.

За словами автора, модель являє собою визначення поглядів у спеціальних “метавідношеннях” і розширює до цих відносин алгебраїчні оператори.

Приклад процесу авторизації запропонованої моделі,

Припустимо, що користувач надсилає запит на отримання імен і спонсорів великих проектів:

отримати(НОМЕР ПРОЕКТУ, ПРОЕКТ.СПОНЗОР), де ПРОЕКТ.БЮДЖЕТ ≥ 250 000

Реалізація наведеного вище запиту з такою послідовністю алгебраїчних операцій:

1. A ← σ(БЮДЖЕТ ≥ 250 000)(ПРОЕКТ)

2. A ← ΠNUMBER,СПОНЗОР(A)

ПРОЕКТ’ включає лише кортежі представлень, до яких Браун має право доступу, як визначено у такому відношенні:

ПРОЕКТ

ПЕРЕГЛЯД	NUMBER	СПОНСОР	БЮДЖЕТ
ПСА	*	акме*	*

Тепер ті самі операції, які застосовуються до відношень бази даних, застосовуються до їх мета-відношень:

1. A′ ← σ(БЮДЖЕТ ≥ 250 000)(ПРОЕКТ′)

2. A′ ← ПНОМЕР, СПОНСОР(A′)

Вибір зберігає лише ті кортежи перегляду, які не змінені, а остаточна проекція виглядає так: A'

NUMBER	СПОНСОР
*	акме*

Наведена вище маска вказує, що користувач обмежений проектами, спонсорованими Acme, і наступне визначення подання повідомить користувача, що дозвіл існує лише для SPONSOR = Acme.

дозвіл (НОМЕР, СПОНСОР), де СПОНСОР = Acme.

2.5 Детальна політика авторизації

В [20] запропонував детальну модель контролю доступу, засновану на уявленнях про авторизацію, які дозволяють виконувати «прозорі для авторизації» запити, у яких запити можуть бути записані у відношенні бази даних без посилання на перегляди авторизації. За словами авторів, у їхньому підході запити користувачів можуть бути записані в термінах зв'язку з базою даних, і запит є дійсним, лише якщо на нього можна відповісти за допомогою доступних переглядів авторизації. Автори стверджують, що вони ввели нове поняття валідності та умовної перевірки достовірності в запропоновану ними структуру.

За словами авторів, їхня модель базується на шести ключових характеристиках, а саме:

1. Контроль доступу визначається за допомогою переглядів авторизації. Подання може бути традиційним реляційним або параметризованим поданням. За словами авторів, параметризоване подання авторизації схоже на звичайне представлення, але є деякі додаткові параметри, такі як ідентифікатор користувача, час і місцезнаходження користувача, що з'являються у його визначенні:

- створити авторизацію, переглянути MyGrades as

- Виберіть * з оцінок, де student-id = $user-id

2. Запити можуть бути написані прозорим для авторизації способом у зв'язку з базою даних без посилань на перегляди авторизації.

3. Запит q є безумовно дійсним, якщо існує еквівалентний запит q′ і обидва запити дають однаковий результат для всіх станів бази даних.

4. На певні запити можна відповісти за допомогою доступного набору переглядів авторизації, навіть якщо їх неможливо переписати за допомогою представлень.

5. Умовно дійсні запити, на які можна відповісти, використовуючи інформацію, що міститься в наборі переглядів авторизації в заданому стані бази даних.

6. Набір потужних правил висновку, які перевіряють безумовну та умовну дійсність запитів.

2.6 Представлення авторизації та утримання умовних запитів

В [25] запропонував алгоритм, який перевіряє умовне утримання кон'юнктивних запитів щодо набору матеріалізованих кон'юнктивних уявлень. За словами авторів, вони

визначили проблему .Автори стверджують, що на основі свого алгоритму,

вони перевіряють, чи є запит умовно авторизованим з урахуванням набору матеріалізованих переглядів авторизації.

За словами авторів, вони приймають визначення Rizvi et al. [20]: запит умовно дійсний для набору представлень V і набору матеріалізованих представлень MV для всіх станів бази даних, де значення представлень V узгоджуються з MV, q узгоджується з q′.

Автори визначають умовний запит, умовно порожній запит і умовно авторизований запит у таких визначеннях:

Визначення 1. Для будь-яких двох запитів Q1 і Q2, Кажуть, що Q1 умовно міститься в Q2 щодо V і MV, позначається Q1 V, MV Q2, якщо для кожного d в D, Q1(d) Q2(d). Q1 називається умовно еквівалентним Q2 відносно V і MV, що позначається Q1≡V, MV Q2, якщо Q1 V, MV Q2 і Q2 V, MV Q1.

Визначення 2. Запит Q умовно порожній щодо V і MV, якщо Q(d) ми порожнім для кожного d у D.

В роботі [25] стверджує, що їхній підхід спочатку ініціює параметризовані представлення, витягуючи значення параметрів, пов'язані з користувачем і сеансом, перш ніж оголошувати, чи слід умовно авторизувати запит.

Визначення 3. Запит Q є умовно авторизованим щодо переглядів авторизації V та матеріалізованих представлень MV, якщо є запит Qr, який записується з використанням лише представлень у V, і який умовно еквівалентний Q.

2.7 Контроль доступу до матеріалізованих представлень

У попередніх розділах ми обговорювали деякі методи контролю доступу до даних, які використовуються організаціями на основі організаційних вимог. В [3] запропонував підхід до контролю доступу матеріалізованих уявлень. Автори виявили проблему автоматичного створення правил контролю доступу для матеріалізованих уявлень на основі правил контролю доступу, визначених над базовими відносинами. Автори використовують техніку авторизаційних переглядів для контролю доступу до матеріалізованих уявлень.

Автори пропонують використовувати синтаксис Datalog для визначення правил контролю доступу. Автори припускають існування трьох типів символів: змінних, констант та імен предикатів. p(t1….tn) – це літерал, де p – ім'я предиката з арністю n, і кожен ti для 1≤ i ≤ n є або константою, або змінною.

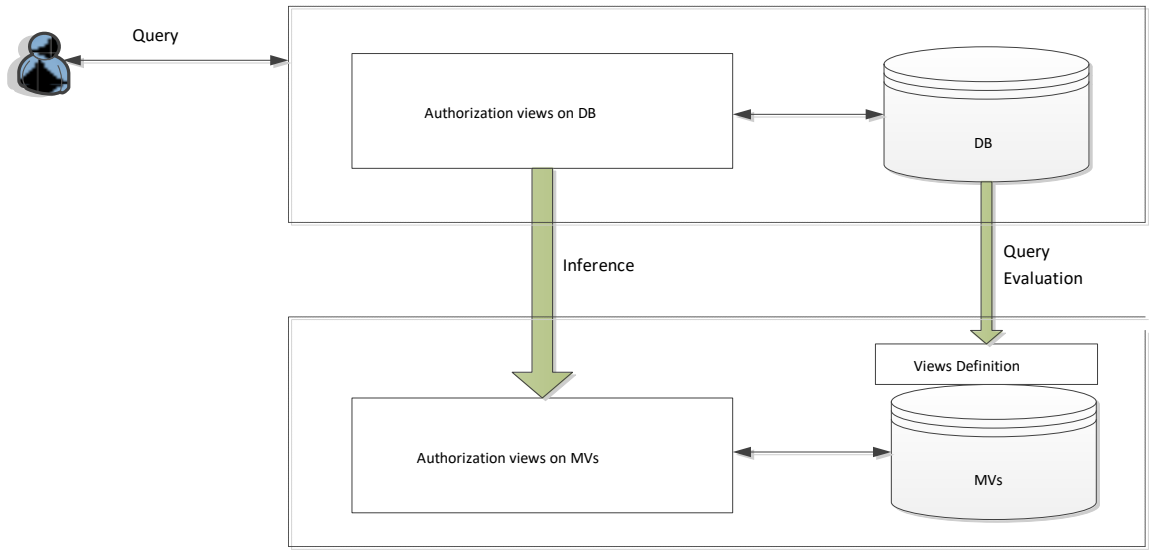


Рисунок 2.3 – Політика авторизації для матеріалізованих переглядів

На думку авторів, логічне речення пов'язане з правилом Datalog

$$P(u) \leftarrow q1(u1),\ldots,qn(un) \text{ це:}$$

$$x1..xn(p(u1) \leftarrow q1(u1) \wedge \ldots \wedge qn(un)).$$

В [3] стверджує, що відносини, визначені дедуктивними правилами, називаються інтенційними відносинами. Наприклад, Info-Doc (Id-D, Dname, Dfname, Dspeciality) ← лікар (Id-D, Dname, Dfname, Dadr, Dphone, Dspeciality,Dsalary) це правило, яке визначає (навмисне) відношення Info-Doc у термінах (розширеного) відношення лікаря.

У наступному прикладі визначено перегляди авторизації для контролю доступу до бази даних лікарень.

База даних лікарні: лікар (IdD, Dname, Dfname, Dadr, Dphone, Dspecialty, Dsalary) медсестра (IdI, Snum, Nname, Nfname, Nadr, Nphone, Nsalary).

Перегляди авторизації:

av1 (IdD, Dname, Dfname, Dspecialty) ← лікар (IdD, Dname, Dfname, Dadr, Dphone, Dspecialty, Dsalary) av2 (IdI, Snum, Nname, Nfname) ← медсестра (IdI, Snum, Nname, Nfname, Nadr, Nphone, Nsalary)

Щоб визначити, які дані доступні для кожного навмисного відношення mv у MV, першим кроком є вибір перегляду авторизації. За словами авторів, вони використовують техніку переписування запитів, побудовану на алгоритмі Bucket, який переписує запит користувача на основі доступних переглядів авторизації.

Автори стверджують, що вони модифікували оригінальний алгоритм [18], так як якщо вони використовують оригінальний алгоритм, то перегляд авторизації буде вважатися нерелевантним. Наприклад, (1), показано нижче, є матеріалізованим визначенням подання. Подання авторизації (2) визначає кортежи x, y, до яких користувач має право доступу.

$$mv(x,y,z) \leftarrow r(x,y,z)\ (1)\ av(x,y) \leftarrow r(x,y,z)\ (2)$$

У оригінальному алгоритмі Bucket атрибут «z» з'являється в підцілі запиту, а також у заголовку запиту, тоді він також має бути в заголовку подання. На думку авторів, він занадто суворий, оскільки можна спроектувати «z», створивши відповідне представлення авторизації на mv.

Таблиця 2.1 – Порівняння різноманітних дослідницьких робіт

Рік	Автор	Пропонована робота	Підхід до контролю доступу	Специфікація правила щодо мови	Домен доступних даних	Contributio n	Реалізація
2004	Шарік Різві	Розширення запиту	На основі правил	SQL	Загальний	Надає потужний набір правил висновку для перевірки запиту	Не адресовано
2010	Альфредо	Ефективно і	На основі правил	Журнал даних	Загальний	Представляє синтаксис на основі журналу даних для вираження	Не адресовано

2011	Cuzzocrea	Ефективно	На основі правил	Журнал даних	Загальний	правила, і	
2012	Сара Найт Балул	Вибір доступу	На основі правил	Журнал даних	Загальний	Алгоритм VSP-Bucket для запиту	Не адресовано
2013	Хасаан Юсафі	На основі ролі Управління доступом Схема для Матеріалізований Перегляди	На основі ролей + правил	Журнал даних	Загальний	переписування	

У 2012 році їх автори пропонують використовувати алгоритм MiniCon [18] для переписування запитів із попереднім підходом. На думку авторів, цей алгоритм більш ефективний з точки зору узгодження кортежів із набором доступних переглядів авторизації.

3 РЕАЛІЗАЦІЯ МОДЕЛЕЙ ТА РІШЕННЯ ДЛЯ СТРУКТУРОВАНИХ ТА НЕСТРУКТУРОВАНИХ МАТЕРІАЛЬНИХ ПРЕДСТАВЛЕНЬ

3.1 Пропонувана основа

У цьому розділі ми представляємо детальну інформацію про запропоновану нами структуру «Схема контролю доступу на основі ролей для матеріалізованих уявлень». Ми представляємо архітектуру, яка дозволяє організаціям визначати та керувати авторизаціями контролю доступу до даних для матеріалізованих уявлень на основі ролей. У цій роботі представлена схема контролю доступу на основі ролей для матеріалізованих уявлень, у яких правила асоціюються з ролями, обмеження на рівні стовпців накладається на матеріалізоване представлення на основі призначеної користувачем ролі, а стратегія конфлікту між ролями визначається, якщо користувач отримує перевагу. авторизація дозволів, пов'язаних із конфліктними ролями. У цій запропонованій структурі ми зосереджуємось на системах управління базами даних з відкритим кодом, таким як MySQL і PostgreSQL, оскільки бази даних з відкритим вихідним кодом не забезпечують жодного автоматизованого процесу для забезпечення авторизації на матеріалізованих представленнях. Запропонована структура також може бути застосована до інших реляційних баз даних, оскільки перегляди авторизації визначаються, зберігаючи в уявленні структуру матеріалізованих таблиць уявлень, визначених у реляційних базах даних.

3.2 Контроль доступу до даних до матеріалізованих представлень

Матеріалізоване представлення записує результати запиту (або просто подання) у фізичну таблицю, яка може бути збережена, і користувач може потім запитувати матеріалізоване представлення таким же чином, як і запити до бази даних. Матеріалізовані уявлення можна використовувати з міркувань продуктивності в дуже великих системах, таких як сховища даних або

розподілені системи, або для надання відфільтрованого вибору даних із більш загальної бази даних. Існуючі запропоновані методи забезпечують контроль доступу на основі правил для матеріалізованого перегляду, але, наскільки нам відомо, адміністрування таких систем займає багато часу та громіздко у великому середовищі, оскільки адміністратори визначають правила для кожного користувача, щоб контролювати доступ до матеріалізованих уявлень.

У цій роботі ми розширюємо попередню роботу, запропоновану Bahloul et al. [3], який визначає політику авторизації в Datalog як формальну структуру для вираження правил контролю доступу (Abiteboul et al. [1]). У цьому дослідженні ми визначаємо політику авторизації для ролей, а не визначаємо ці політики для окремих осіб. Ми обговорюємо деталі пов'язаних робіт, які визначають політику авторизації даних на рівні бази даних, наприклад параметризовані перегляди авторизації (Rizvi et al. [20]), а також ми обговорили нещодавню роботу Bahloul et al. [3], що визначає перегляди авторизації в синтаксисі Datalog.

3.3 Запропонована рольова структура

Запропонована архітектура представляє новий підхід до визначення авторизацій контролю доступу до даних для матеріалізованих представлень. Ми визначили адміністративні проблеми визначення та зміни правил контролю доступу до даних для матеріалізованих представлень у великих організаціях, де тисячі співробітників виконують свої обов’язки. Існуючі запропоновані методи забезпечують контроль доступу на основі правил для матеріалізованих уявлень; однак адміністрування таких систем займає багато часу та громіздко у великому середовищі, де адміністратори визначають правила для кожного користувача для контролю доступу до матеріалізованих представлень.

У нашому підході ми використовуємо існуючу модель керування доступом на основі ролей (RBAC) [24], яку запропонував NIST. Модель RBAC на даний

момент є широко використовуваною моделлю серед усіх інших існуючих моделей. завдяки своїй динамічності та простоті адміністрування.

У RBAC доступ до організаційних ресурсів надається на основі ролі, призначеної користувачеві. Коли користувачу призначається роль, користувачу також надаються всі пов’язані з цією роллю привілеї. У першому розділі ми детально обговорили RBAC.

У запропонованій нами структурі ми визначаємо подання авторизації для кожної ролі. Авторизації визначаються в синтаксисі Datalog. Подання авторизації, визначені для матеріалізованих представлень, надають обмеження на рівні стовпців.

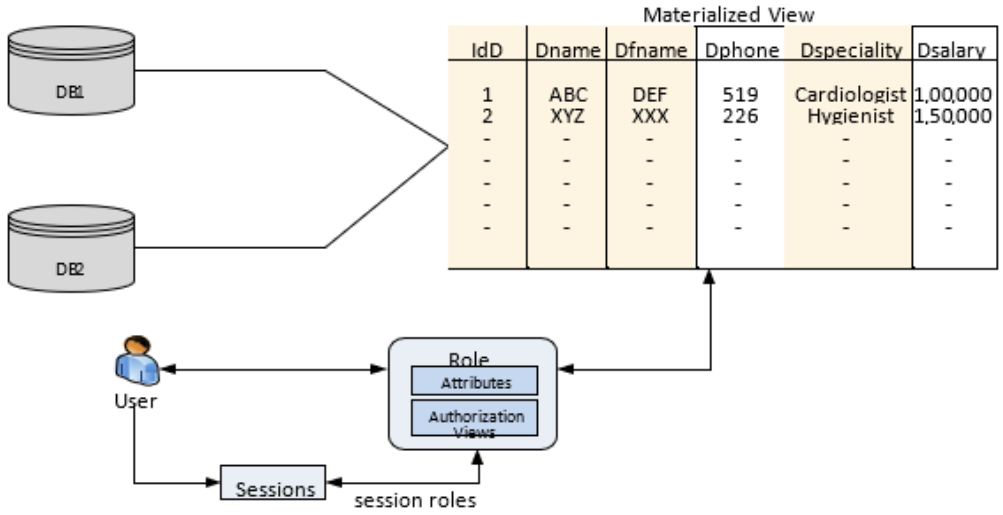


Рисунок 3.1 – Запропонована архітектура RBAC для матеріалізованих уявлень

На рисунку 3.1 матеріалізоване представлення створюється шляхом об’єднання двох таблиць із двох різних баз даних, а подання розгортається в локальній базі даних на розподіленому сайті. Як показує запропонована архітектура, перегляди авторизації пов’язані з призначеною користувачем роллю, а атрибути профілю користувача витягуються під час процесу призначення ролі. Ці атрибути витягуються з таблиці бази даних, де зберігаються атрибути профілю користувача, такі як ім’я користувача, назва відділу, роль

користувача та інші атрибути. Атрибути використовуються для аутентифікації та призначення ролі.

У запропонованій нами структурі ми також підтримуємо session_roles, оскільки ми помітили під час нашого дослідження, що користувачеві можна призначити кілька ролей, наприклад, менеджер проекту може одночасно працювати як програміст. Таким чином, враховуючи призначення кількох ролей, ми дозволяємо користувачеві відкривати кілька session_roles. Ми надаємо детальну інформацію про session_roles.

Оскільки ми дозволяємо користувачам відкривати кілька облікових записів під час одного сеансу, нам також потрібно відстежувати конфлікт ролей, який виникає через конфлікт інтересів між двома ролями. Користувач повинен деактивувати один сеанс, щоб активувати інший конфліктуючий session_role. Ми докладно обговорюємо рольові конфлікти.

Призначення ролі

Процес призначення ролі починається після процесу аутентифікації, якщо облікові дані користувача, такі як ім’я користувача та пароль, дійсні, процес продовжується далі.

На рисунку 3.2 представлено робочий процес процесу призначення ролей.

Після підтвердження облікових даних користувача процес перевіряє вже активні сеанси.

Активний сеанс – це таблиця, в якій зберігається вся інформація про активні сеанси користувача.

Ми описуємо наведений вище робочий процес у наступних шести кроках, як показано на рисунку:

1. Процес перевіряє, чи інформація про користувача вже існує в таблиці active_session. Якщо інформація користувача не існує в таблиці active_session, то інформація користувача (тобто ім’я користувача, відділ і роль), яка витягується з бази даних на основі облікових даних користувача (тобто ім’я користувача, пароль), надсилається до модуля призначення ролі, де роль призначені

користувачеві. В іншому випадку, якщо інформація про сеанс користувача існує, атрибути користувача збігаються з уже існуючими атрибутами сеансу користувача. Якщо атрибути відповідають одному з уже активних сеансів, то вважається, що користувач уже є активним користувачем з такою ж роллю.

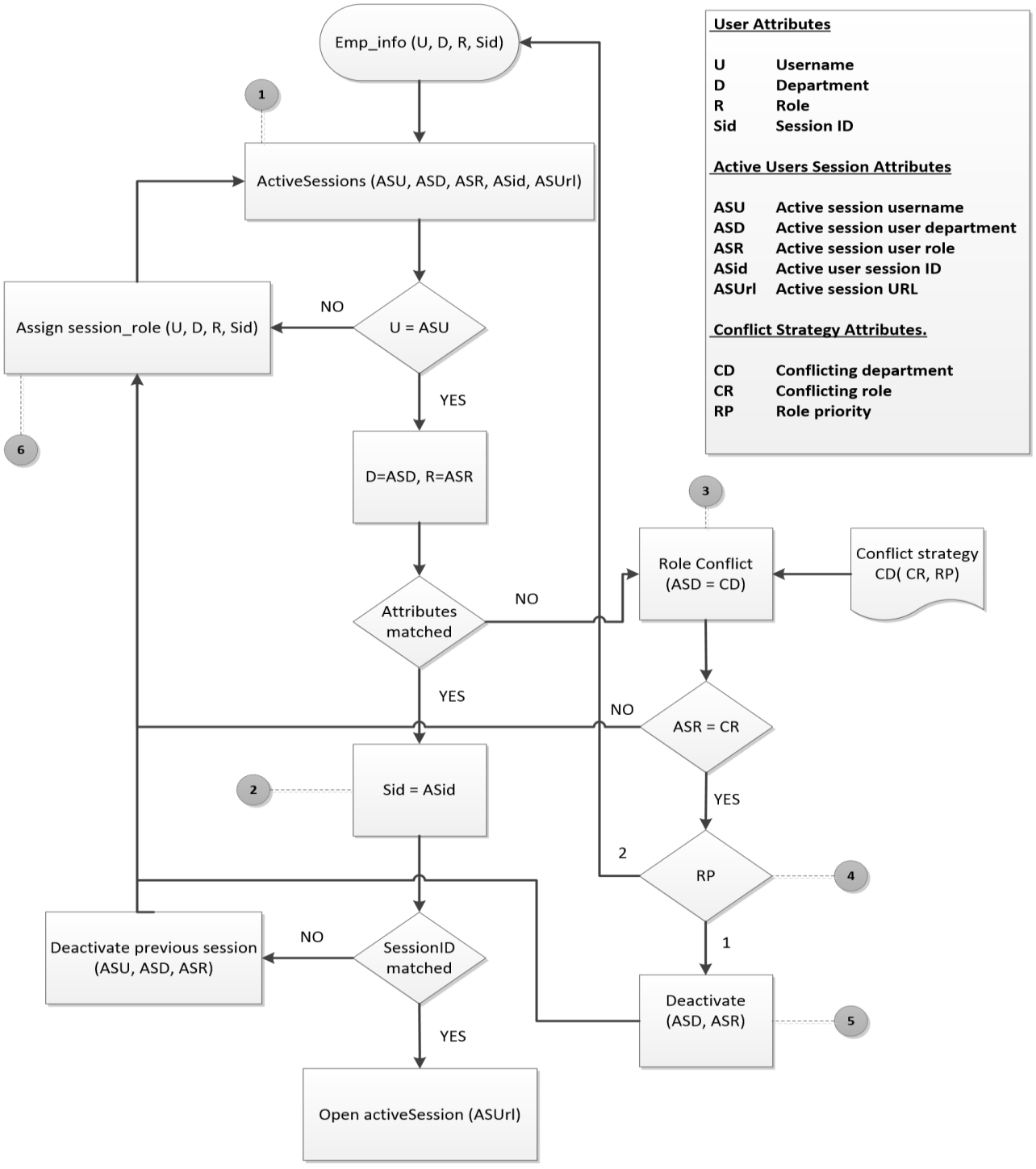


Рисунок 3.2 – Робочий процес процесу призначення ролі

2. Якщо користувач уже є активним користувачем, то перевіряється ідентифікатор сеансу користувача, якщо ідентифікатор сеансу збігається з ідентифікатором сеансу активної ролі, тоді користувач перенаправляється на той самий сеанс, інакше, якщо ідентифікатор сеансу інший, процес деактивує вже активний сеанс і переходить до процесу призначення ролі.

3. Якщо атрибути сеансу користувача не збігаються з уже активним сеансом користувача, крім імені користувача, як описано в кроці 1, вважається, що користувач є активним користувачем з іншою роллю. У цьому випадку конфлікт ролей перевіряється шляхом вилучення атрибутів, визначених у стратегії конфлікту ролей, з атрибутами вже активних сеансів користувача.

4. Якщо рольовий конфлікт існує після аналізу стратегії рольового конфлікту, далі перевіряється пріоритет ролі. У конфліктних стратегіях визначаються рольові пріоритети. Якщо пріоритет дорівнює «1», то вже існуючий конфліктуючий сеанс деактивується без повідомлення користувача, а новий користувач переходить до процесу призначення ролі. Якщо пріоритет дорівнює «2», користувачу надсилається сповіщення з інформацією про конфліктну роль, що існує конфлікт ролей, і щоб активувати нову session_role, користувач повинен деактивувати конфліктну роль session_role.

5. Після дезактивації конфліктної session_role користувач може перейти до процесу призначення ролі, як показано на рисунку.

6. Якщо конфлікт ролей не існує, як описано на кроці 4, користувачу призначається роль, яка ґрунтується на облікових даних користувача.

Авторизації на основі ролей

У запропонованій нами структурі авторизація доступу до матеріалізованого представлення базується на ролі користувача. На рисунку представлена архітектура авторизації на основі ролей.

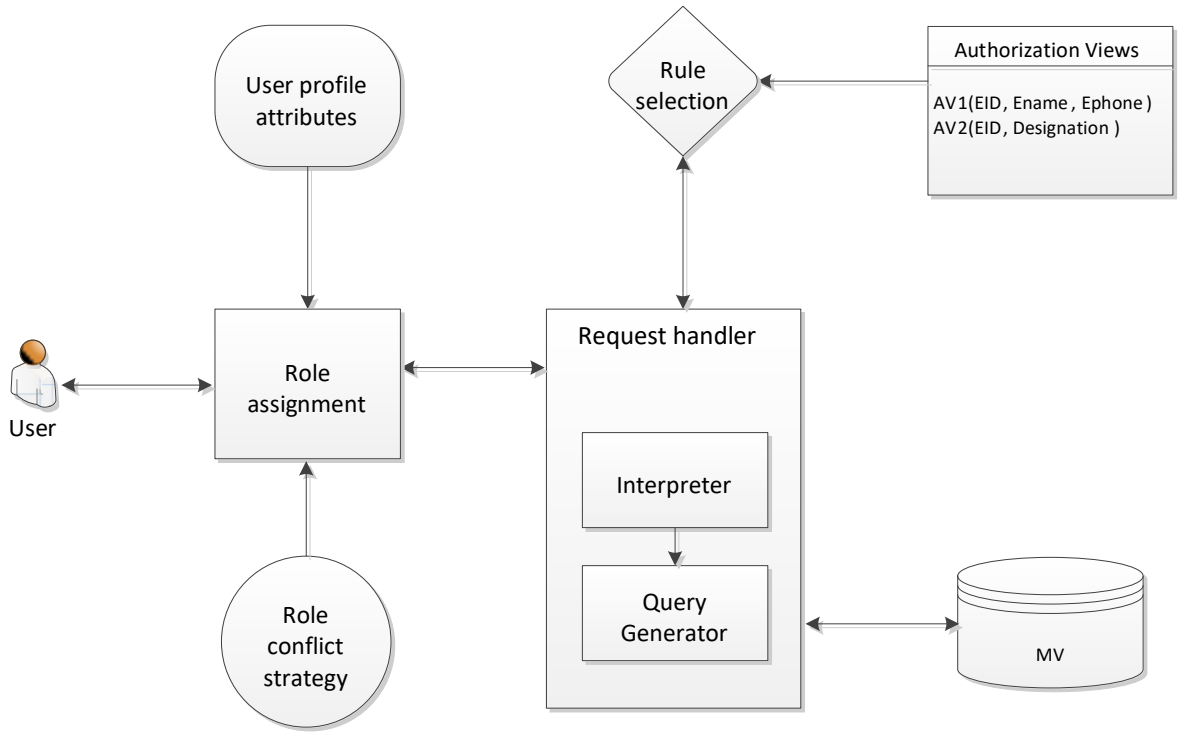


Рисунок 3.3 – Архітектура авторизації на основі ролей

Процес починається з призначення ролі, атрибути користувача витягуються на основі облікових даних користувача, які користувач надає під час процесу входу. Наступний приклад описує вищезгадану архітектуру.

Приклад:

Після того, як користувачу буде призначено роль після проходження процесу призначення ролі, як обговорювалося в попередньому розділі, запит користувача надсилається до обробника запитів.

Обробник запитів надсилає матеріалізоване ім’я представлення, як зазначено в запиті користувача, до модуля вибору правила.

Модуль «Вибір правила» шукає вказане ім’я представлення матеріалізації у пов’язаному файлі переглядів авторизації ролі, призначеної користувачу. Якщо ім’я матеріалізованого представлення знайдено в списку авторизованих представлень, процес вибору правила витягує інформацію авторизованого представлення та повертає її обробнику запитів.

Обробник запитів надсилає інтерпретатору авторизоване представлення.

Інтерпретатор витягує назви стовпців із даного авторизованого представлення та надсилає його до генератора запитів.

Модуль «Генератор запитів» генерує запит на основі наданих імен стовпців та матеріалізованого подання та повертає його обробнику запитів.

Обробник запитів далі надсилає запит до системи бази даних і повертає результати користувачеві.

Ми використовуємо існуючий підхід для визначення уявлень про авторизацію, запропонований Bahloul et al. [3]. У наступному прикладі автори визначають перегляди авторизації для контролю доступу до бази даних лікарень.

приклад:

База даних лікарень:

лікар (IdD, Dname, Dfname, Dadr, Dphone, Dspecialty, Dsalary).

медсестра (IdI, Snum, Nname, Nfname, Nadr, Nphone, Nsalary).

Перегляди авторизації:

av1 (IdD, Dname, Dfname, Dspecialty) ← лікар (IdD, Dname, Dfname, Dadr, Dphone, Dspecialty, Dsalary)

av2 (IdI, Snum, Nname, Nfname) ← медсестра (IdI, Snum, Nname, Nfname, Nadr, Nphone, Nsalary)

У наведеному вище прикладі є дві частини перегляду авторизації. Ліва сторона визначає тіло авторизаційного подання, а права — повне визначення матеріалізованого подання, з якого виходить подання авторизації. Тіло перегляду авторизації використовується для визначення авторизацій для конкретного користувача. У запропонованій нами структурі ми визначаємо подання авторизації для ролей, як показано на рисунку.

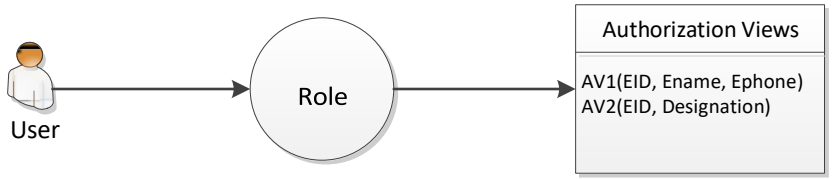


Рисунок 3.4 – Представлення ролі та авторизації

У запропонованій нами структурі ми використовуємо модель керування доступом на основі ролей (RBAC), запропоновану NIST. RBAC використовується більшістю організацій через його динамічний характер і простоту адміністрування. Для RBAC визначено три основні правила:

1. Призначення ролі: користувач може отримати доступ до ресурсів, лише якщо він/вона вибрав роль або йому було призначено.

2. Авторизація ролі: активна роль користувача має бути авторизована для користувача.

3. Авторизація дозволу: користувач може отримати доступ до ресурсів, лише якщо дозвіл авторизовано для активної ролі користувача.

Специфікації RBAC:

• Призначення користувача = UA USER X ROLES, зіставлення користувача багато до багатьох

відношення призначення ролі.

• Призначення дозволів = PA PREMS X ROLES, відображення багато до багатьох

дозвіл на відношення призначення ролі.

• session_users (s: SESSIONS) → USERS, відображення сеансів s на відповідного користувача.

РОЛІ

• session_roles (s: SESSIONS) → 2 , відображення сеансів s на набір ролей.

Користувач може відкривати кілька одночасних сеансів з різними ролями та дозволами. У запропонованій нами структурі ми дотримуємося наведених вище правил, а також дозволяємо користувачеві відкривати кілька одночасних сеансів. Користувач може відкрити кілька сеансів з різними ролями, і кожен одночасний сеанс надає користувачеві доступ до представлень на основі призначеної ролі. Кожен сеанс витягує атрибути користувача (тобто ім’я користувача, відділ і роль), які передаються через загальну URL-адресу до обробника запитів після призначення ролі. Коли користувач надсилає запит,

обробник запитів витягує атрибути та дозволяє користувачеві переглядати записи запитаного матеріалізованого представлення на основі доступного перегляду авторизації, який визначено у файлі авторизованих переглядів, пов'язаному з роллю користувача в певному відділі.

Рольові конфлікти

У попередньому розділі ми обговорювали декілька одночасних сеансів, які дозволяють користувачам відкривати кілька сеансів з різними ролями та дозволами. Але ми також повинні обмежити користувачів, щоб уникнути конфліктів ролей. Рольовий конфлікт визначає, що жодна особа не може взяти на себе владу двох або більше конфліктуючих ролей одночасно. Рольові конфлікти виникають, коли у людей є різні суперечливі обов'язки

RBAC вводить розділення обов'язків (SoD), яке обмежує користувачів виконувати обов'язки в конфліктних ролях одночасно, дотримуючись обмежень.

Ми візьмемо приклад фінансової системи, в якій користувач має доступ до білінгової системи через роль «Фінансові розрахунки», а також має доступ до платіжної системи через роль «Фінансові платежі». У цьому випадку користувач не може активувати обидві ролі одночасно. Користувач повинен деактивувати один сеанс, щоб активувати інший сеанс.

У запропонованій нами структурі ми вводимо стратегії конфліктів, які пов'язані з роллю користувача так само, як і перегляди авторизації, щоб уникнути конфлікту ролей. У запропонованій нами стратегії конфлікту ми вводимо пріоритет ролей.

Якщо під час процесу призначення ролі виникає конфлікт ролей, система перевіряє пріоритет ролі, визначений у стратегії конфлікту адміністратором. Ми опишемо стратегію конфлікту так.

Після визначення рольового конфлікту між двома ролями стратегія конфлікту визначається наступним чином.

Відділ (роль, пріоритет)

Стратегія конфлікту визначається в текстовому файлі, пов’язаному з кожною роллю. Під час процесу призначення ролі, як ми обговорювали в попередньому розділі. Обробник запитів перевіряє вже активовані session_roles того самого користувача з різними ролями. Він витягує атрибути активних session_roles один за одним і збігається з атрибутами, визначеними в стратегії конфлікту (тобто відділ, роль). Якщо атрибути збігаються, це вважається рольовим конфліктом. У цьому випадку обробник запитів перевіряє пріоритет ролі, визначений у стратегії конфлікту. Якщо пріоритет дорівнює «1», система активує нову session_role і деактивує попередній сеанс. Якщо пріоритет дорівнює «2», система надсилає сповіщення з конфліктуючими атрибутами ролі, і користувач може активувати нову session_role після дезактивації вже активної session_role.

У наведеному нижче прикладі ми визначаємо стратегії конфлікту для ролей Finance Billing і Finance Paying, які належать фінансовому відділу.

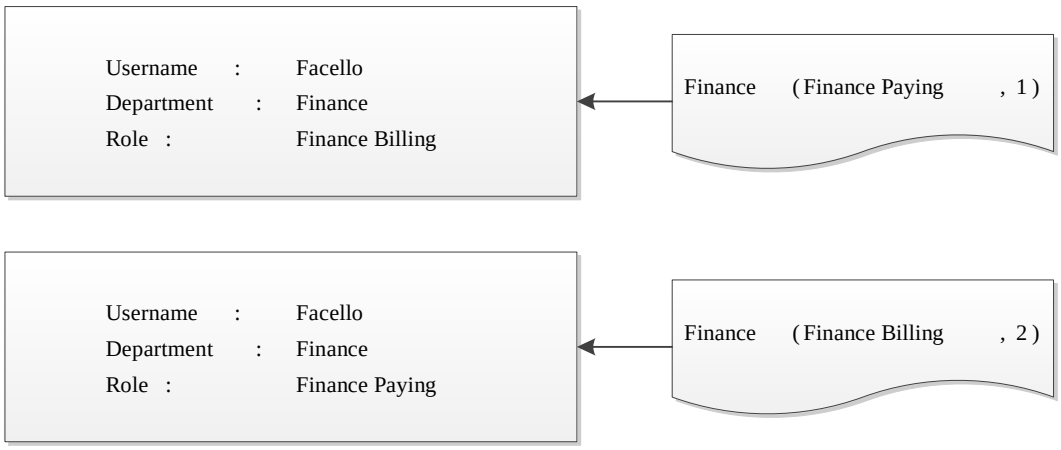


Рисунок 3.5 – Стратегії рольових конфліктів

На рисунку 3.5 користувачу Facello призначено дві ролі, але він не може активувати обидві ролі одночасно. Пріоритети ролей визначаються в стратегії конфлікту, пов’язаної з обома ролями. У наведеному вище сценарії ролі фінансових рахунків призначається перший пріоритет, а ролі фінансових платежів — другий пріоритет.

3.4 Реалізація та верифікація

Основна мета цього дипломного дослідження полягає в тому, щоб забезпечити архітектуру для контролю доступу до матеріалізованих представлень на основі ролей користувачів. Це дослідження побудовано на основі досліджень щодо створення автоматизованих інструментів для проведення опитувань у сфері охорони здоров'я, системи підтримки прийняття рішень та реальних система управління даними часу. У цій дослідницькій роботі ми виявили потребу в забезпеченні конфіденційних даних про охорону здоров'я. У запропонованій нами структурі ми представляємо архітектуру контролю доступу на основі ролей, в якій авторизації пов'язані з ролями користувачів. Крім того, ми вводимо механізм вирішення конфліктів між ролями, визначаючи пріоритети ролей у стратегіях конфліктів ролей, які також пов'язані з роллю користувача так само, як і перегляди авторизації.

У попередньому розділі ми обговорюємо кроки, пов'язані з автентифікацією, призначенням ролей та авторизацією користувачів у запропонованій нами структурі. У цьому розділі ми надаємо подробиці впровадження запропонованої нами рамки.

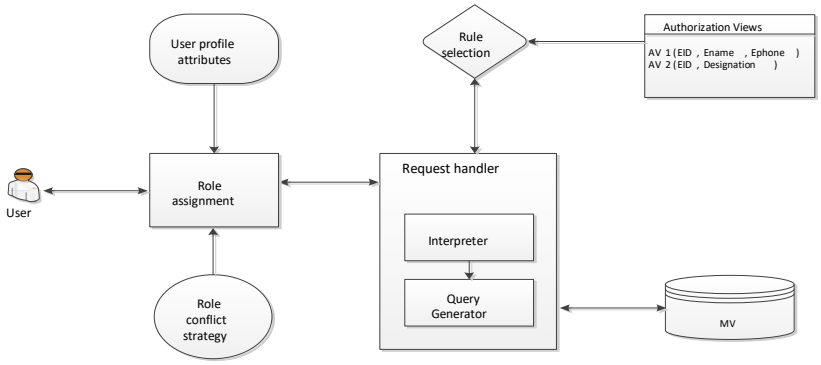


Рисунок 3.6 – Пропонована архітектура

У запропонованій структурі ми зосереджуємось на системах управління базами даних з відкритим вихідним кодом (СУБД), оскільки СУБД з відкритим кодом не забезпечує жодного автоматизованого процесу для регулювання

доступу до матеріалізованих представлень. У реалізації запропонованого нами фреймворка ми використовуємо базу даних MySQL, внутрішня структура нашого додатка розроблена на PHP, а інтерфейс користувача (UI) розроблено на HTML і CSS.

Модуль призначення ролей

Призначення ролі — це процес призначення ролі користувачеві на основі облікових даних, які користувач надає під час процесу аутентифікації. Перш ніж призначити роль, користувач повинен пройти кілька перевірок, щоб уникнути будь-яких конфліктів ролей і дублювання session_roles. У попередньому розділі ми пояснюємо повний робочий процес модуля призначення ролей. У цьому розділі ми надаємо псевдокод цього модуля.

```
CHECK if the user is already an active user
IF user exists in Actives_Sessions

EXTRACT user Active_Sessions attributes

IF user attributes match with any Active_Sessions
        CHECK for SessionID
            IF SessionID match with current user SessionID
                REDIRECT user to same Session_Role
              ELSE
                Deactivate ActiveSession role
                Activate new Session_Role
    BREAK

CHECK Role Conflict with existing Session_Roles in ActiveSessions

EXTRACT attributes of ActiveSessions
EXTRACT Conflict_Strategy of current role
          IF any of ActiveSessions role match with role defined in
Conflict_Strategy for current user
      CHECK for the role priority defined in Conflict_Strategy
              IF Priority is 1
                  Deactivate ActiveSession role
                   Activate current role
              IF Priority is 2
                 Send an alert with conflicting role information
    BREAK
```

Рисунок 3.7 – Псевдокод модуля призначення ролі

У таблиці Active Sessions ми зберігаємо інформацію про користувача (тобто ім’я користувача, роль, відділ, ідентифікатор сеансу та URL-адресу сеансу) щоразу, коли користувач входить у систему. Інформація, доступна в цій таблиці, витягується під час процесу призначення ролі, щоб узгодити активні ролі користувача та поточні атрибути ролі, щоб уникнути конфлікту ролей та дублювання ролей session_roles. Ідентифікатор сеансу зберігається, щоб запобігти активації користувачами кількох сеансів для однієї і тієї ж ролі session_role з іншої системи. Користувач не може відкрити session_role з двох різних ідентифікаторів сеансу одночасно. Якщо користувач намагається активувати session_role, яка вже є в списку активних сеансів, система деактивує наявну session_role і створює новий запис у активних сеансах з новим ідентифікатором сеансу. Таким чином ми можемо відстежувати діяльність користувачів.

Ми витягуємо інформацію про активний сеанс користувача, викликаючи функцію activeSession.

Active Sessions

```
function activeSession (username) {
EXTRACT username, role, department, session_id, url from active_sessions
        WHERE username == username
    IF found
      Store user ActiveSessions attributes in an ARRAY
          Return ARRAY
    ELSE
          Return NULL
}
```

Рисунок 3.8 – Псевдокод Active Sessions

Обробник запитів обробляє запити всіх користувачів, коли користувачу призначається роль. Він отримує запит користувачів і надсилає його до модуля

Вибір перегляду. Перед обробкою запиту користувача обробник запитів перевіряє, що користувач все ще існує в активних сеансах. Якщо користувач все ще є активним користувачем, обробник запитів підтверджує, що ідентифікатор сеансу користувача збігається з наявним ідентифікатором сеансу в Active Sessions. Якщо це так, то він обробляє запит користувача, інакше він деактивує користувача session_role і перенаправляє користувача на сторінку входу. Обробник запитів працює за такими кроками:

1. Обробник запитів отримує запит користувача, витягує запитане ім’я матеріалізованого представлення із запиту та надсилає його до модуля вибору перегляду.

2. Модуль Вибір перегляду шукає запитане ім’я матеріалізованого представлення у файлі Authorization Views, яке пов’язане з роллю користувача.

3. Модуль вибору перегляду повертає інформацію про перегляд авторизації до обробника запитів.

4. Обробник запитів надсилає Інтерпретатору інформацію про авторизований перегляд.

5. Інтерпретатор витягує авторизовані назви стовпців із перегляду авторизації та надсилає їх обробнику запитів.

6. Обробник запитів надсилає ім'я подання та назву авторизованих стовпців до генератора запитів.

7. Генератор запитів генерує запит на SQL на основі інформації, наданої обробником запитів, і надсилає запит до бази даних.

8. Обробник запитів отримує записи від генератора запитів і повертає їх користувачеві.

Вибір перегляду

Вибір перегляду — це процес пошуку та вибору перегляду авторизації на основі вибраного користувачем матеріалізованого перегляду. Процес вибору перегляду починається, коли запит надсилається до модуля вибору перегляду з обробника запитів, який містить запитане ім’я матеріалізованого

представлення. Щоб зв’язати файл Authorization Views, пов’язаний з кожною роллю, модулі вибору перегляду отримують атрибути користувача (тобто назву відділу та роль) разом із ім’ям матеріалізованого представлення від обробника запитів і генерують шлях для зв’язку з авторизацією. Переглядає файл ролі, призначеної користувачу.

View Selection

```
function ViewSelection (MaterializedView Name, Department Name, Role) {
        Path to user Auhtorization Views file = "AuthorizationViews/department_name/role"
        LOOP Search for MaterializedView in defined Authorization Views
                IF found
                        Store AuthorizationView body information in an ARRAY
                        Return AuthorizationView
                ELSE
                        Return NULL
}
```

Рисунок 3.9 – Псевдокод Active Sessions

Інтерпретатор перекладає подання авторизації в синтаксис SQL. Обробник запитів надсилає інформацію про перегляд авторизації, яка повертається модулем вибору перегляду, до інтерпретатора. Подання авторизації отримується в такому форматі.

av1(emp_no, ім'я, прізвище)

Інтерпретатор видаляє ім’я подання та дужки, які містять авторизовані назви стовпців, із даного представлення авторизації та повертає назви стовпців до обробника запитів, щоб надіслати їх до модуля генерування запитів.

emp_no, ім'я, прізвище

Генерування запитів

Генерування запитів — це процес створення запиту в синтаксисі мови послідовного запиту (SQL) і відправлення його обробнику запитів, щоб

надіслати запит до бази даних. Наступна архітектура описує процес побудови запиту.

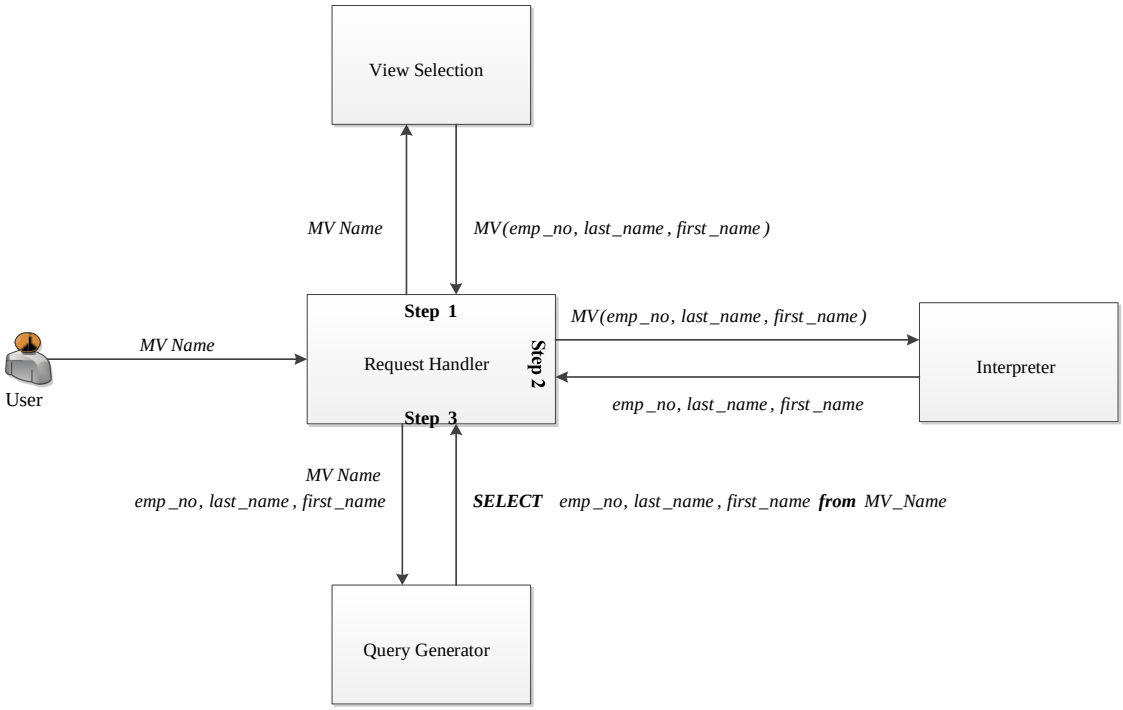


Рисунок 3.10 – Побудова запиту

На рисунку 3.10 модуль генерування запитів отримує матеріалізоване ім’я подання та імена стовпців від обробника запитів. Матеріалізоване ім'я представлення та імена стовпців призначаються динамічним змінним, які оголошуються в обробнику запитів. Обробник запитів передає змінні генератору запитів, щоб створити повний запит. Речення SQL SELECT і FROM статично визначено в синтаксисі запиту, а динамічні змінні розміщуються для назв стовпців та назви перегляду. Існують деякі інші речення, які використовуються в операторі SQL SELECT, наприклад WHERE, GROUP BY, ORDER BY і LIMIT, які можуть бути визначені користувачем за допомогою параметрів, доступних в інтерфейсі користувача. Значення цих додаткових пропозицій SELECT також призначаються динамічним змінним, які оголошуються в обробнику запитів, і обробник запитів передає ці значення генератору запитів разом з іменами MV та іменами стовпців.

Щоб перевірити запропоновану нами структуру, ми впровадили додаток на основі запропонованої архітектури. Програма розроблена відповідно до специфікації, описаної вище. Ми обговоримо частину реалізації в попередньому розділі, де ми визначаємо кроки, які виконує кожен модуль. У цьому розділі ми розглянемо нашу програму, яку ми розробили для перевірки та перевірки запропонованого підходу.

Основні вимоги

Перевірка основних вимог полягає в тестуванні основних елементів програми. У нашому дослідженні ми зосереджені на створенні архітектури, яка дозволить користувачам отримати доступ до даних на основі їхньої ролі. Процес призначення ролі починається після процесу аутентифікації. Користувачу необхідно вказати дійсне ім’я користувача та пароль для аутентифікації.

На рисунку 3.11 представлена сторінка входу.



Рисунок 3.11 – Сторінка входу

Після процесу аутентифікації користувач перенаправляється на домашню сторінку. Наступне зображення представляє домашню сторінку.

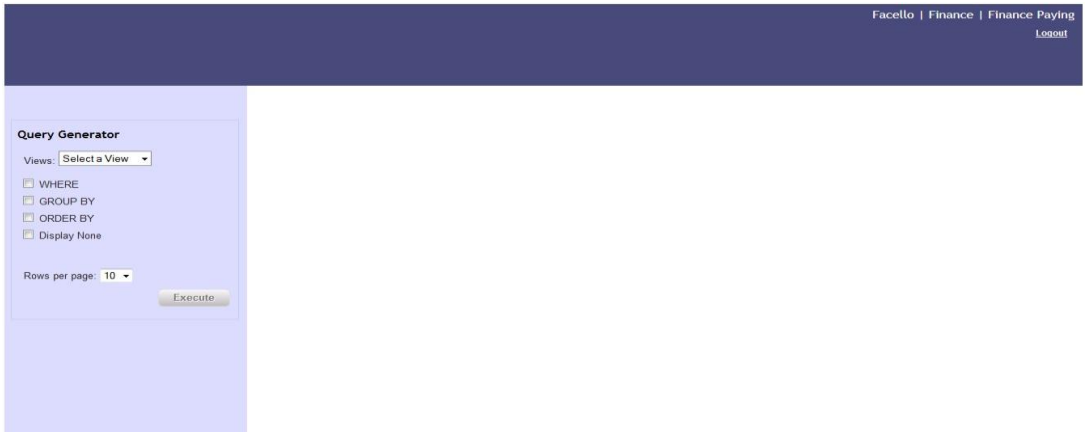


Рисунок 3.12 – Домашня сторінка

Ім’я користувача, відділ та роль користувача відображаються у верхньому правому куті розділу заголовка, як на рисунку 3.12. Доступна кнопка виходу, яка деактивує сеанс користувача та перенаправляє користувача на сторінку входу. Ліва панель до заголовка використовується для розміщення генератора запитів. Користувач може створити запит за допомогою функцій, доступних у Генераторі запитів. Для вибору матеріалізованого перегляду доступний спадний список. Цей спадний список містить лише ті авторизовані перегляди, які визначені у файлі перегляду авторизації, пов’язаному з призначеною користувачем роллю. Є деякі прапорці, які використовуються для ввімкнення та вимкнення додаткових операторів SQL, а функція «Не відображати» використовується для цілей аналізу.

Основна вимога цієї фреймворку полягає в тому, щоб обмежити користувача переглядом лише тих стовпців матеріалізованого представлення, які користувач має авторизацію. Наступне зображення представляє просте виконання запиту без згадки будь-яких додаткових пунктів.

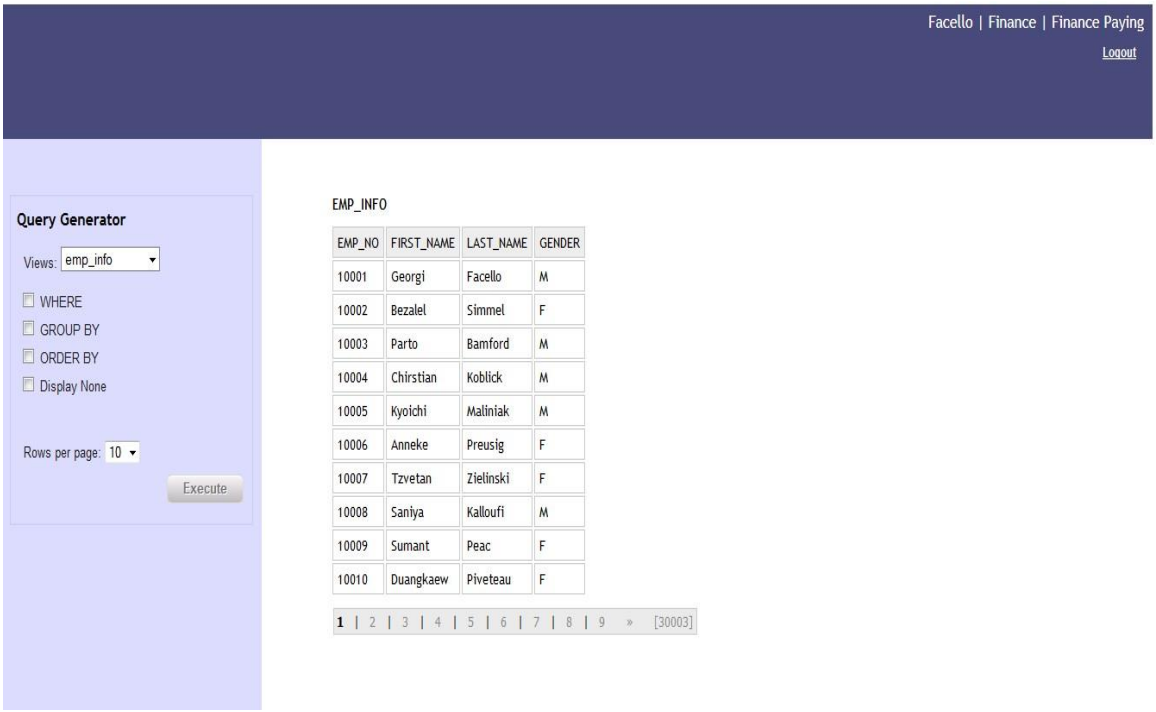


Рисунок 3.13 – Просте виконання запиту та результати

На рисунку 3.13 користувач Facello вибирає представлення emp_info зі спадного списку представлень і виконує запит без додавання будь-яких додаткових пунктів. Система повертає лише інформацію про ті стовпці, які користувач уповноважив переглядати, як визначено для призначеної користувачам ролі, у пов’язаному файлі переглядів авторизації, як показано нижче.

```
1  departments(dept_no, dept_name) <- department(dept_no, dept_name)
2  dept_emp(emp_no,dept_no) <- dept_emp(emp_no, dept_no, from_date, to_date)
3  emp_info(emp_no,first_name,last_name,gender) <- emp_info(emp_no,birth_date,first_name,last_name,gender,hire_date)
```

Рисунок 3.14 – Перегляди авторизації

На рисунку 3.14 повне визначення представлення визначено в правій частині, яке містить шість стовпців, але користувач не має права переглядати інформацію двох стовпців (тобто дата_народження та дата_найму), як визначено зліва. На Рисунок 3.15 система повертає результати на основі визначеного перегляду авторизації.

У запропонованій нами структурі ми вводимо стратегії рольових конфліктів. Користувач може активувати декілька ролей session_roles одночасно, оскільки користувач може працювати в кількох ролях в одній організації. Ми визначаємо стратегію конфлікту, щоб уникнути конфлікту інтересів між двома ролями, оскільки конфліктні ролі не можуть бути активовані одним користувачем одночасно. У запропонованій нами структурі ми призначаємо пріоритети конфліктним ролям. Стратегія конфлікту пов'язана з ролями, такими ж, як і перегляди авторизації.

Ми детально обговорюємо рольові конфлікти на прикладі. Щоб підтвердити цю вимогу, ми визначаємо стратегію конфлікту ролей для ролей «Фінансові платежі» та «Фінансові рахунки» у фінансовому відділі, і ми призначаємо обидві ролі одному користувачеві. Стратегії визначаються наступним чином.



Рисунок 3.15 – Стратегія конфлікту

На рисунку 3.16 стратегія конфлікту визначена для ролей «Фінансові рахунки» та «Фінансові платежі» у відповідних файлах ролей, оскільки обидві ролі не можна активувати одночасно. Роль «Фінансові рахунки» має перший пріоритет, а роль «Фінансові платежі» — другий пріоритет. Користувач активує роль «Фінансові рахунки», і в той же час користувач хоче активувати роль «Оплата фінансів». Оскільки ролі «Фінансові рахунки» надається першочерговим, користувач отримує таке повідомлення під час процесу призначення ролі та перенаправляється на сторінку входу, як показано на рисунку.



Рисунок 3.16 – Виявлено конфлікт ролей

Якщо користувач спочатку активує роль «Фінансові платежі», а в той же час користувач активує роль «Фінансові рахунки». У цьому випадку система деактивує роль «Фінансові платежі», не повідомляючи користувача, оскільки ролі «Фінансові рахунки» надається першочерговим.

Додаткові функції

У нашому додатку є деякі додаткові функції. Ці функції дозволяють користувачам фільтрувати результати, що надходять з бази даних. Додаткові функції включають додаткові пропозиції SQL, такі як WHERE, GROUP BY і ORDER BY. Ці функції доступні в генераторі запитів. На наступному рисунку представлені додаткові положення, доступні в інтерфейсі користувача.

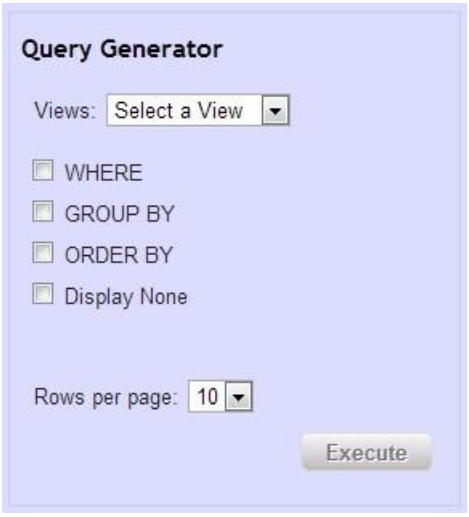


Рисунок 3.17 – Додаткові умови оператора SQL

На рисунку 3.17 є додаткові умови оператора SQL, які користувач може увімкнути для фільтрації результатів. Користувач може ввімкнути їх, натиснувши прапорці, доступні в кожному пункті. На наступному рисунку представлено генерацію запиту з використанням додаткових пропозицій.

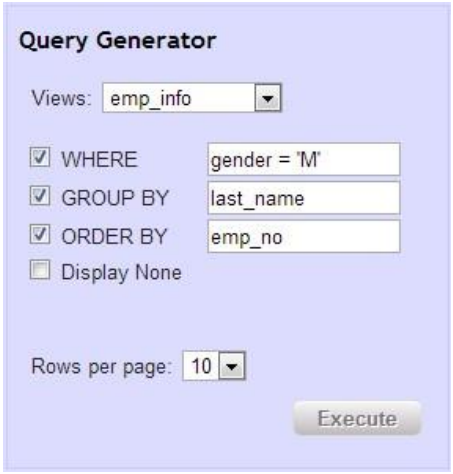


Рисунок 3.18 – Генерування запиту з використанням операторів SQL

На рисунку 3.18 представлено генерування запитів з використанням додаткових операторів SQL, користувач виконує запит, а обробник запитів повертає результати на основі перегляду авторизації, визначеного для призначеної користувачем ролі, як показано на рисунку 3.18.

EMP_INFO

EMP_NO	FIRST_NAME	LAST_NAME	GENDER
10001	Georgi	Facello	M
10003	Parto	Bamford	M
10004	Chirstian	Koblick	M
10005	Kyoichi	Maliniak	M
10008	Saniya	Kalloufi	M
10012	Patricio	Bridgland	M
10013	Eberhardt	Terkki	M
10014	Berni	Genin	M
10015	Guoxiang	Nooteboom	M
10016	Kazuhito	Cappelletti	M

Рисунок 3.19 – Результати запиту з додатковими реченнями SQL

Окрім додаткових положень SQL, є ще одна додаткова функція «Не відображати», яка додається в інтерфейс користувача з метою аналізу даних. Обробник запитів витягує кількість записів для запитуваного запиту, і якщо кількість записів менша за значення, визначене в полі «Не відображати», тоді обробник запиту повертає загальну кількість рядків.

На рисунку 3.20 увімкнено функцію «Не відображати», а в полі «рядки менше ніж» вказано значення «2». У цьому випадку обробник запитів не повертає жодних результатів, якщо кількість записів менше двох. На наступному рисунку представлені результати після виконання запиту, як зазначено на Рисунок 3.19.

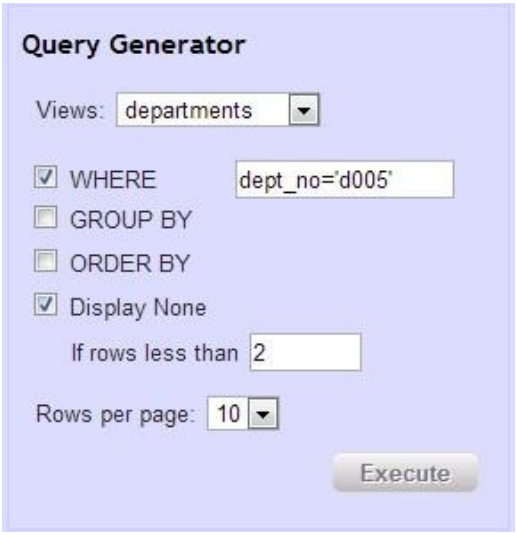


Рисунок 3.20 – Генерування запиту за допомогою функції Display None



Рисунок 3.21 – Результати запиту за допомогою функції Display None

3.5 Результати тесту на масштабованість

NU = кількість користувачів

SARO = кількість ролей сеансу

SARU = Кількість правил, пов'язаних з ролями

Таблиця 3.1 – Результати тесту на масштабованість

NU	**SARO**	**SARU**	Час RoleAssg (сек.)	**Час вибору правила (сек.)**	**Час виконання запиту (сек.)**	**Час відповіді на запит (сек.)**
15	21	234	0.95854	0.00126	1.59740	0.00050
30	40	468	0.98607	0.00163	1.59740	0.00036
45	57	520	1.07722	0.00140	1.50204	0.00035

У цьому розділі ми детально обговорили реалізацію запропонованої нами структури. Ми також підтвердили свій підхід і представили результати. Запропонована структура не є предметною. Його може прийняти будь-яка організація, яка має величезну кількість співробітників, які працюють у різних віддалених місцях у розподіленому середовищі. Організації створюють базу даних Materialized Views і розгортають такі бази даних на віддалених сайтах, щоб полегшити навантаження на мережу, а також зменшити навантаження на основний сервер бази даних. Немає такого автоматизованого механізму, доступного в базах даних з відкритим кодом, який можна було б використовувати для контролю доступу до матеріалізованих представлень локально на віддалених сайтах. Запропонована структура може бути використана для локальних сайтів Materialized View у розподіленому середовищі.

ВИСНОВКИ

В бакалаврській рооті представлено основу для контролю доступу до матеріалізованих представлень на основі ролі користувача. Авторизації пов’язані з роллю користувача, і ці авторизації визначаються в синтаксисі Datalog. Визначають подання авторизації для окремих користувачів, щоб контролювати доступ до матеріалізованих представлень, але в нашому підході ми визначаємо подання авторизації для ролей.

У запропонованій нами структурі ми використовуємо специфікацію підходу до контролю доступу на основі ролей, і ми дозволяємо користувачеві переглядати дані на основі призначеної ролі. Ми надаємо авторизацію на основі стовпців у запитаному матеріалізованому представленні.

В описаній структурі ми дозволяємо користувачам активувати кілька ролей одночасно. У нашому дослідженні ми виявили, що якщо дві ролі мають конфлікт інтересів, то їх не можна активувати одночасно. Щоб уникнути конфліктів ролей, ми вводимо стратегії конфлікту ролей, ці стратегії пов’язані з роллю користувача так само, як і перегляди авторизації. У стратегії рольового конфлікту атрибути визначаються для забезпечення поділу обов’язків, а конфліктним ролям надається пріоритет для вирішення рольових конфліктів.

Щоб перевірити запропоновану архітектуру, ми розробили програму з використанням технологій з відкритим кодом (тобто MySQL і PHP). Реалізація виконується відповідно до основної вимоги запропонованої нами архітектури.

Результати показують, що програма відповідає основним вимогам запропонованого нами фреймворку, які включають визначення переглядів авторизації для ролей, надання авторизацій на основі стовпців, активацію кількох ролей одночасно, виявлення конфліктних ролей та встановлення пріоритетів для вирішення конфліктів ролей. Запропонована нами структура не є специфічною для домену; це є загальна структура і може бути використана в будь-якій області.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. S. Abiteboul, R. Hull, and V. Vianu, Foundations of Databases. AddisonWesley, 1995.

2. A. Aboulnaga and K. Salem, “Report: 4th international workshop on self-managing database systems (SMDB 2009),” IEEE Data Engineering Bulletin,vol. 32, no. 4, pp. 2–5, 2009.

3. B. Adelberg, H. Garcia-Molina, and B. Kao, “Applying update streams in a soft real-time database system,” in Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 245–256, San Jose,California, USA, May 1995.

4. M. E. Adiba and B. G. Lindsay, “Database snapshots,” in Proceedings of the1980 International Conference on Very Large Data Bases, pp. 86–91, Montreal, Quebec, Canada, October 1980.

5. F. Afrati and R. Chirkova, “Selecting and using views to compute aggregatequeries,” in Proceedings of the 2005 International Conference on Database Theory, pp. 383–397, Edinburgh, UK, January 2005.

6. F. Afrati and R. Chirkova, “Selecting and using views to compute aggregate queries,” Journal of Computer and System Sciences, vol. 77, no. 6, pp. 1079– 1107, 2011.

7. F. Afrati, C. Li, and J. Ullman, “Generating efficient plans for queries using views,” in Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, pp. 319–330, Santa Barbara, California, USA, June 2001.

8. F. N. Afrati, R. Chirkova, M. Gergatsoulis, B. Kimelfeld, V. Pavlaki, and Y. Sagiv, “On rewriting XPath queries using views,” in Proceedings of the 2009 87 Full text available at: http://dx.doi.org/10.1561/1900000020 88 References International Conference on Extending Database Technology, pp. 168–179,Saint Petersburg, Russia, March 2009.

9. F. N. Afrati, R. Chirkova, M. Gergatsoulis, and V. Pavlaki, “View selectionfor real conjunctive queries,” Acta Informatica, vol. 44, no. 5, pp. 289–321, 2007.

10. F. N. Afrati, M. Damigos, and M. Gergatsoulis, “Union rewritings for XPath fragments,” in Proceedings of the 2011 International Database Engineering and Applications Symposium, pp. 43–51, Lisbon, Portugal, September 2011.

11 F. N. Afrati, C. Li, and P. Mitra, “Answering queries using views with arithmetic comparisons,” in Proceedings of the 2002 ACM Symposium on Principlesof Database Systems, pp. 209–220, Madison, Wisconsin, USA, June 2002.

12. F. N. Afrati, C. Li, and J. D. Ullman, “Using views to generate efficient evaluation plans for queries,” Journal of Computer and System Sciences, vol. 73,no. 5, pp. 703–724, 2007.

13. F. N. Afrati and V. Pavlaki, “Rewriting queries using views with negation,”AI Communications, vol. 19, no. 3, pp. 229–237, 2006.

14. P. K. Agarwal, J. Xie, J. Yang, and H. Yu, “Scalable continuous query processing by tracking hotspots,” in Proceedings of the 2006 International Conferenceon Very Large Data Bases, pp. 31–42, Seoul, Korea, September 2006.

15. C. C. Aggarwal, ed., Data Streams: Models and Algorithms. Springer, 1st ed.,Novermber 2006.

16. C. C. Aggarwal and P. S. Yu, “A survey of synopsis construction in data streams,” in Aggarwal [15], pp. 169–207.

17. D. Agrawal, A. E. Abbadi, A. Most´efaoui, M. Raynal, and M. Roy, “The lord of the rings: Efficient maintenance of views at data warehouses,” in Proceedings of the 2002 International Symposium on Distributed Computing, pp. 33–47, Toulouse, France, October 2002.

18. D. Agrawal, A. E. Abbadi, A. K. Singh, and T. Yurek, “Efficient view maintenance at data warehouses,” in Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, pp. 417–427, Tucson, Arizona, USA, May 1997.

19. P. Agrawal, A. Silberstein, B. F. Cooper, U. Srivastava, and R. Ramakrishnan, “Asynchronous view maintenance for VLSD databases,” in

Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 179–192, Providence, Rhode Island, USA, June 2009.

20. S. Agrawal, N. Bruno, S. Chaudhuri, and V. R. Narasayya, “AutoAdmin: Self-tuning database systems technology,” IEEE Data Engineering Bulletin, vol. 29, no. 3, pp. 7–15, 2006.

21. S. Agrawal, S. Chaudhuri, L. Koll´ar, A. P. Marathe, V. R. Narasayya, and M. Syamala, “Database tuning advisor for Microsoft SQL Server 2005,” in Proceedings of the 2004 International Conference on Very Large Data Bases, pp. 1110–1121, Toronto, Canada, August 2004.

22. S. Agrawal, S. Chaudhuri, L. Koll´ar, A. P. Marathe, V. R. Narasayya, and M. Syamala, “Database tuning advisor for Microsoft SQL Server 2005: demo,” in Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 930–932, Baltimore, Maryland, USA, June 2005.

Full text available at: http://dx.doi.org/10.1561/1900000020 References 89

23. S. Agrawal, S. Chaudhuri, and V. R. Narasayya, “Automated selection of materialized views and indexes in SQL databases,” in Proceedings of the 2000 International Conference on Very Large Data Bases, pp. 496–505, Cairo, Egypt, September 2000.

24. S. Agrawal, S. Chaudhuri, and V. R. Narasayya, “Materialized view and index selection tool for Microsoft SQL Server 2000,” in Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, p. 608, Santa Barbara, California, USA, June 2001.

25. M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, “Matching events in a content-based subscription system,” in Proceedings of the 1999 ACM Symposium on Principles of Distributed Computing, pp. 53–61, Atlanta, Georgia, USA, May 1999.

26. Y. Ahmad, O. Kennedy, C. Koch, and M. Nikolic, “DBToaster: Higherorder delta processing for dynamic, frequently fresh views,” Proceedings of the VLDB Endowment, vol. 5, no. 10, pp. 968–979, 2012.

27. A. Ailamaki, S. Babu, P. Furtado, S. Lightstone, G. M. Lohman, P. Martin, V. R. Narasayya, G. Pauley, K. Salem, K.-U. Sattler, and G. Weikum, “Report: 3rd International Workshop on Self-Managing Database Systems (SMDB 2008),” IEEE Data Engineering Bulletin, vol. 31, no. 4, pp. 2–5, 2008.

28. A. Ailamaki, S. Chaudhuri, S. Lightstone, G. M. Lohman, P. Martin, K. Salem, and G. Weikum, “Report on the Second International Workshop on SelfManaging Database Systems (SMDB 2007),” IEEE Data Engineering Bulletin, vol. 30, no. 2, pp. 2–4, 2007.

29. M. O. Akinde, O. G. Jensen, and M. H. B¨ohlen, “Minimizing detail data in data warehouses,” in Proceedings of the 1998 International Conference on Extending Database Technology, pp. 293–307, Valencia, Spain, March 1998.

30. M. Arenas, P. Barcel´o, L. Libkin, and F. Murlak, Relational and XML Data Exchange. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

31. A. Arion, V. Benzaken, I. Manolescu, and Y. Papakonstantinou, “Structured materialized views for XML queries,” in Proceedings of the 2007 International Conference on Very Large Data Bases, pp. 87–98, Vienna, Austria, September 2007.

32. Z. Asgharzadeh Talebi, R. Chirkova, and Y. Fathi, “Exact and inexact methods for solving the problem of view selection for aggregate queries,” International Journal o f Business Intelligence and Data Mining, vol. 4, no. 3/4, pp. 391–415, 2009.

33. Z. Asgharzadeh Talebi, R. Chirkova, Y. Fathi, and M. Stallmann, “Exact and inexact methods for selecting views and indexes for OLAP performance improvement,” in Proceedings of the 2008 International Conference on Extending Database Technology, pp. 311–322, Nantes, France, March 2008.

34. B. Babcock, M. Datar, and R. Motwani, “Load shedding in data stream systems,” in Aggarwal [15], pp. 127–147.

35. S. Babu, K. Munagala, J. Widom, and R. Motwani, “Adaptive caching forcontinuous queries,” in Proceedings of the 2005 International Conference onData

Engineering, Tokyo, Japan, April 2005.Full text available at: http://dx.doi.org/10.1561/190000002090 References

36. S. Babu and K.-U. Sattler, “Report: 5th international workshop on selfmanaging database systems (SMDB 2010),” IEEE Data Engineering Bulletin,vol. 33, no. 3, pp. 4–7, 2010.

37. S. Babu, U. Srivastava, and J. Widom, “Exploiting k-constraints toreduce memory overhead in continuous queries over data streams,” ACMTransactions on Database Systems, vol. 29, no. 3, pp. 545–580, 2004.

38. A. Balmin, F. Ozcan, K. S. Beyer, R. Cochrane, and H. Pirahesh, “A ¨framework for using materialized XPath views in XML query processing,” in Proceedings of the 2004 International Conference on Very Large Data Bases, pp. 60–71, Toronto, Canada, August 2004.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи бакалавра: " Моделі контролю доступу та представлень структурованих та неструктурованих даних"

Об’єм пояснювальної записки: 67 аркушів

Дата закінчення дипломної роботи 15 червня 2022р.

Підпис студента _____