**Практична робота №8.Представлення**

**Мета:** Вивчення особливостей методів формування та класифікації представлень даних, які дозволяють не тільки спрощувати структуру запитів, надаючи інформації необхідний вид, але й відіграють одну з ключових ролей у обмежені доступу до інформації для різних груп користувачів при колективному використанні БД Набуття практичних навичок в створенні представлень.

***Методичні рекомендації***

*При вивченні теми слід звернути увагу на створення та збереження структури бази даних(БД), редагування структури, визначення типів полів, зв’язування таблиць у БД.*

1. Повторити лекції №7, 8, теоретичну частину ЛР № 4-8 та ПР№6, 7. Ознайомитися з теоретичною частиною цієї ПР.
2. За результатами виконання ЛР № 4,5,6 та ПР№6 повинна бути сформована та заповнена реляційна БД з таблицями Викладач, Студент, Групи, Предмети, Розклад, Аудиторії, Предмети, Оцінки. Для цих таблиць виконати завдання:

* Сформувати представлення щодо подання підсумкових оцінок студентів по предметах
* Сформувати представлення щодо предметів, які викладає викладач по групах (за потребою, встановіть додаткові поля, зв’язки та доповніть таблиці даними і відповідні запити додайте до звіту)

Таблиці та функції обрати самостійно.

1. Результати оформити звітом, в якому надаються сформовані SQL-запити, опис дій, що вони виконують, та скріншот виконання на вашій БД.
2. Результати надсилати на електронну адресу викладача [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**DB<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **DB3101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт -"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31, 32 - 24.11.2023**

**КОНТРОЛЬНІ ПИТАННЯ**

1. Яка існує різниця та що є спільного між представленням даних та таблицею, яка визначена за допомогою оператора *CREATE TABLE*?
2. Яким чином відбувається процес розв’язання представлення даних?
3. Якими можливостями володіє оператор *CREATE VIEW* для визначення представлення даних?
4. Які особливості роботи визначального оператора *SELECT* Вам відомі?
5. Коли в операторі *CREATE VIEW* імена стовпців задаються обов'язково?
6. Яка існує різниця між обновлюваним представленням даних та тим, що не оновлюється?
7. Які обмеження накладаються на обновлені представлення даних?
8. Які можливості надає використання пропозиції *WITH CHECK OPTION*?
9. Чим відрізняються відомі Вам способи формування представлень даних?
10. Які обмеження накладаються на створення й використання представлень даних?
11. Які переваги та недоліки використання представлень даних Вам відомі?

**МЕТОДИЧНІ ВКАЗІВКИ**

***Представлення***

Представлення є запитом на вибірку, якому привласнили ім'я і зберегли в базі даних. Іншими словами це віртуальна таблиця, яка реально не містить даних, а всякий раз, коли потрібен її вміст, відбувається виконання запиту.

Коли СУБД зустрічає в інструкції SQL посилання на представлення, вона знаходить його визначення, збережене в базі даних, перетворить призначений для користувача запит, що посилається на представлення, в еквівалентний запит до вихідних таблиць представлення і виконує цей запит. Таким чином, СУБД створює ілюзію існування представлення у вигляді окремої таблиці.

Причини використання представлень:

1. Вони ізолюють реалізацію зберігання даних від користувача. Якщо клієнти працюватимуть з даними через представлення, то у розробника бази даних з'являється можливість міняти запит, що лежить в основі представлення, а клієнт нічого не помічатиме.

2. З їх допомогою можна обмежити доступ до даних, дозволяючи певним користувачам мати права на читання/запис даних у представленнях, але навіть не мати поняття про таблиці, лежачі в основі цих представлень.

3. Вони спрощують доступ до бази даних, показуючи користувачам структуру даних, що зберігаються, в найбільш відповідному для них вигляді.

Представлення можуть грунтуватися на даних з декількох таблиць і інших представлень. Представлення можна використовувати як звичайні таблиці. Оператор SELECT, що створює представлення, може робити те ж, що і звичайний оператор SELECT, окрім включення пропозиції ORDER BY і використання як джерело даних збережених процедур.

Синтаксис оператора створення представлення:

***CREATE VIEW І’мя\_Представлення [(Колонка [, Колонка .])]***

***AS <select> [WITH CHECK OPTION];***

***<select> – оператор SELECT, який вибирає дані що включаються у представлення***

WITH CHECK OPTION – якщо вказати цей необов'язковий параметр, то при вставці або зміні рядка представлення перевірятиметься умова "попадання" цього рядка у представлення. Якщо новий або змінений запис не задовольняє умовам запиту, на якому засноване представлення, то операція буде скасована і виникне помилка.

Приклад створення представлення:

*CREATE VIEW ViewStudent(St\_ID, LastName, FirstName)*

*AS SELECT Pr\_ID, Pr\_LastName, Pr\_FirstName*

*FROM Person, Student*

*WHERE Pr\_ID = St\_ID;*

Щоб змінити яке-небудь представлення, його треба видалити і створити заново. При видаленні представлення необхідно також видалити всі залежні від його об'єкти – тригери, збережені процедури і інші представлення.

Приклад видалення представлення:

*DROP VIEW ViewStudent;*

# *Представлення, що модифікуються*

Для деяких представлень вставку, видалення або оновлення рядка можна перетворити в еквівалентні операції по відношенню до вихідних таблиць цього представлення. Можуть існувати представлення, для яких додавання, видалення або оновлення рядків не має сенсу. Такі представлення можна лише читати.

Вміст представлення може змінюватися командами модифікації, які будуть автоматично перенаправлені до базової таблиці. Щоб представлення було таким, що модифікується, воно має бути створене на основі однієї таблиці (або іншого представлення, що модифікується), а запит, на якому засноване представлення, не повинен містити підзапитів, агрегатних функцій, UDF, процедур, що зберігаються, пропозицій DISTINCT, GROUP BY і HAVING. Якщо виконуються всі ці умови, то представлення автоматично стає таким, що модифікується, тобто для нього можна виконувати запити DELETE, INSERT і UPDATE, які змінюватимуть дані в таблиці-джерелі.

Розглянемо детальніше представлення.

**Визначення, створення та класифікація представлень даних**

Таким чином, *представлення – це динамічний результат однієї або декількох реляційних операцій, виконаних над відношеннями БД для одержання нового відношення*. Представлення створюється динамічно. Воно є віртуальним відношенням, яке реально в БД не існує й створюється в результаті виконання запиту користувача. Стикаючись із посиланням на представлення, система керування знаходить відповідне визначення і перетворює вихідний запит до представлення даних в еквівалентний до відношень, які використовуються у визначенні представлення даних, після чого модифікований запит виконується. Цей процес називається *розв’язанням представлення*.

З погляду користувача, представлення виглядає як таблиця з даними, що складається зі стовпців та рядків. Це підвищує захищеність даних, дозволяє проектувальнику настроювати модель користувача та спрощує багатотабличні запити. Після визначення представлення, в якому об’єднані дані з декількох таблиць, можна буде використовувати до нього прості однотабличні запити, замість запитів з виконанням того ж самого багатотабличного з'єднання.

Послідовність формування записів у представленні даних визначається оператором *SELECT*. Для його створення застосовується така конструкція:

***CREATE VIEW****Ім'я [(стовпець[, стовпець ...])*

***AS****<оператор****SELECT****> [****WITH CHECK OPTION****];*

Тут обов'язково має бути визначене ім'я представлення та оператор *SELECT*. Додатково є можливість указати імена для кожного стовпця представлення – [(*стовпець* [, *стовпець* ...])], а для обновлюваних представлень заборонити введення рядків, що не задовольняють умові його формування в пропозиції *WHERE* оператора *SELECT* – [*WITH CHECK OPTION*].

Cписок імен стовпців має складатися з елементів, кількість яких дорівнює числу стовпців у результуючому відношенні, що формує оператор *SELECT*. Якщо список імен стовпців відсутній, то кожний стовпець представлення буде мати ім'я відповідного стовпця результуючого відношення. Список імен стовпців повинен обов'язково задаватися, якщо в іменах стовпців результуючого відношення має місце неоднозначність. Така ситуація виникає тоді, коли в операторі *SELECT* визначені стовпці, що обчислюються, або коли результуюче відношення створюється за допомогою операції з'єднання та містить атрибути з однаковими іменами.

Заданий оператором *SELECT* запит прийнято називати визначальним. Залежно від його структури представлення може бути обновлюваним або ні. В обновлюваних представленнях можна додавати й вилучати рядки, змінювати значення. Усі ці маніпуляції з даними можуть бути збережені у відношеннях БД. Представлення даних, які не оновлюються, можуть тільки відображати інформацію. СКБД зберігає їх визначення в БД.

Для того, щоб представлення було обновлюваним, СКБД повинна однозначно відобразити будь-який його рядок або стовпець на відповідний рядок або стовпець вихідного відношення. Усі оновлення, виконані у відношенні БД, повинні одразу відображатися у всіх представленнях, що звертаються до нього. Якщо дані були змінені в обновлюваному представленні, то це відразу має бути відображено й у відношенні, на базі якого воно побудоване.

Згідно зі стандартом ISO обновлюваним представлення може бути тільки тоді, коли:

- у його визначенні не використовується пропозиція *DISTINCT*, тобто із результатів визначального запиту не виключаються рядки, що дублюються;

- кожний елемент у списку пропозиції *SELECT* визначального запиту являє собою ім'я стовпця, а не константу, вираз або узагальнюючу функцію; причому ім'я кожного зі стовпців у цьому списку згадується не більше одного разу;

- у пропозиції *FROM* має бути зазначене тільки одне відношення, тобто представлення має бути створене на базі одного відношення, до якого користувач має відповідні права доступу; якщо вихідне відношення саме є представленням, то воно також повинне відповідати зазначеним умовам; дана вимога виключає можливість оновлення будь-яких представлень, що побудовані на базі з'єднання, об'єднання (*UNION*), перетинання (*INTERSECT*) або різниці (*EXEPT*) відношень;

- пропозиція *WHERE* не повинна містити будь-яких вкладених запитів, які посилаються на відношення, що зазначені в пропозиції *FROM*;

- визначальний запит не повинен містити пропозиції *GROUP BY* або *HAVING*.

Будь-який рядок даних, який додається за допомогою представлення, не повинен порушувати вимоги підтримки цілісності даних вихідного відношення. Наприклад, при додаванні через представлення нового рядка в усі стовпці відношення, які відсутні в представленні даних, будуть уведені значення *NULL*. Однак при цьому мають виконуватись усі вимоги щодо пропозиції *NOT NULL*, яка зазначена в описі вихідного відношення.

Основну концепцію обмежень на рядки обновлюваного представлення, можна сформулювати так. В оновлене представлення потрапляють тільки ті рядки, які задовольняють умові *WHERE* у визначальному запиті. Якщо рядок у представленні буде змінений таким чином, що він перестане задовольняти цій умові, то він має бути з нього вилучений. Отже, у представленні будуть з'являтися нові рядки щораз, коли вставка або оновлення даних у представленні приведе до того, що нові рядки будуть задовольняти умові *WHERE*. Рядки, які додаються або вилучаються з представлення, прийнято називати мігруючими.

Фразу *WITH CHECK OPTION* можна наводити тільки для обновлюваних представлень. Вона гарантує, що рядок даних, який не задовольняє умовам пошуку в пропозиції *WHERE*, не буде доданий у відношення, зазначене у пропозиції *FROM* оператора *SELECT*. Взагалі *WITH CHECK OPTION* в операторі *CREATE VIEW* використовується для запобігання міграції рядків з представлення. Якщо до обновлюваного представлення застосувати параметр *WITH CHECK OPTION*, то всі спроби додавання нових або зміни існуючих рядків, які порушують умови пошуку даних пропозиції *WHERE* визначального оператору *SELECT*, будуть відкидатися.

Ця функція може бути настільки корисною, що працювати з представленнями виявиться зручніше, ніж з відношеннями БД. У тому разі, коли оператор *INSERT* або *UPDATE* порушує умови, зазначені в пропозиції *WHERE* визначального запиту, операція відміняється. У результаті з'являється можливість реалізувати в БД додаткові обмеження, які спрямовані на збереження коректності й цілісності даних.

Для вилучення представлення використовується оператор *DROP VIEW* Ім'я.

**Способи й обмеження формування представлень даних**

Хоча усі представлення створюються за допомогою одного методу, на практиці для різних цілей використовуються такі способи формування представлень: вертикальний зріз таблиці, горизонтальний зріз таблиці, вертикально-горизонтальний зріз таблиці, підмножина рядків та стовпців з'єднання різних таблиць.

**Представлення даних у вигляді вертикального зрізу таблиці**

Представлення даних формується як вертикальний зріз таблиці, якщо оператор *SELECT* повертає значення не всіх стовпців вихідної таблиці і невизначене обмеження на кортежі, що увійдуть до представлення даних. Такий спосіб зазвичай використовується у тих випадках, коли дані з відношення обробляються різними групами користувачів. За допомогою вертикального представлення даних у їхнє розпорядження надається віртуальне відношення, що має тільки ті атрибути, які їм необхідні.

ПРИКЛАД 1.

Надати користувачам БД «БІБЛІОТЕКА» можливості перегляду таких відомостей про читачів: № квитка читача, прізвище, ім'я, по батькові, місце роботи та посада (табл. 1).

***CREATE VIEW****ReadersView*

***AS***

***SELECT****ReaderCardNumber, FamilyName, Name, Patronymic, Job, Post*

***FROM****Readers*

| Таблиця 1 Вертикальний зріз таблиці *Readers* за прикладом 1 | | | | | |
| --- | --- | --- | --- | --- | --- |
| **FamilyName** | **Name** | **Patronymic** | **ReaderCardNumber** | **Job** | **Post** |
| Іванов | Петро | Іванович | 317 | НГУ, каф. ЕВТ | Асистент |
| Федорец | Ірина | Олегівна | 28 | НГУ, АХЧ | Вахтер |
| Ільїн | Іван | Петрович | 1345 | НГУ, каф. фізики | Доцент |
| Суренко | Дмитро | Павлович | 543 | НГУ, каф. геофізики | Ст. викладач |
| Коршунова | Наталя | Юріївна | 128 | НГУ, каф. гєоінформатики | Асистент |
| Носенко | Олег | Володимирович | 5672 | НГУ, ІКК | Інженер |
| Брусов | Володимир | Михайлович | 485 | НГУ, каф. геодезії | Лаборант |
| Козирєв | Олексій | Сергійович | 759 | НГУ, каф. кримінології | Професор |
| Левченко | Юлія | Павлівна | 146 | НГУ, каф. політичної теорії | Завідувач кафедри |
| Світла | Тетяна | Іванівна | 2021 | НГУ, каф. перекладу | Ст. викладач |
| Щиглів | Петро | Євгенович | 997 | НГУ, каф. електропостачання | Асистент |
| Кириленко | Віктор | Олександрович | 1010 | НГУ, каф. електропривода | Заступник декана |

**Представлення даних у вигляді горізонтального зрізу таблиці**

Представлення даних формується як горизонтальний зріз таблиці, якщо оператор *SELECT* повертає значення усіх стовпців, і на кортежі, які увійдуть у представлення даних, накладені обмеження.

ПРИКЛАД 2.

Надати користувачам БД «БІБЛІОТЕКА» відомості про книжки, що були видані після 1984 року (табл. 2).

***CREATE VIEW****AllBookAfter1984*

***AS***

***SELECT****\**

***FROM****Books*

***WHERE****IssueYear >= '01.01.1985'*

| Таблиця 2 Горизонтальний зріз таблиці *Books* за прикладом 2 | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Code** | **Name** | **Issue Year** | **Drawing** | **UDK** | **Cipher** | **Note** |
| 1 | Автоматизація виробничих процесів на збагачувальній фабриці | 1985 | «Надра» | null | 622.7-52/Т | null |
| 3 | Асимптотичні методи оптимального управління | 1987 | «Автомат» | null | 681.513.5:/А | null |
| 4 | Синтез оптимальних автоматичних систем | 1984 | «Автомат» | null | 681.513.5:/ДО | null |
| 5 | Методи оптимізації стохастичних систем | 1987 | «Матстат» | null | 681.513.5:/ДО | null |
| 6 | Автоматизовані системи керування технологічним процесом збагачення руди | 1987 | «Автомат» | null | 622.7-52/П | null |
| 7 | C/C++ Програмування мовою високого рівня | 2007 | «Пітер» | null | 681.3.06(075) | null |
| 8 | Комп'ютерні мережі. Принципи, технології, протоколи | 2006 | «Пітер» | null | 004.72(075) | null |
| 9 | Довідник з диференційних рівняннях з частинними похідними першого порядку | 2003 | «ФІЗМАТЛІТ» | null | 517.9 | null |
| 10 | Теорія вірогідності і математична статистика | 2004 | «Пітер» | null | 519.2 | null |
| 11 | C#. Програмування мовою високого рівня | 2009 | «Пітер» | null | 004.43 | null |
| 12 | Теорія вірогідності і математична статистика | 2005 | «Вища школа» | null | 519.2 | null |
| 13 | Теорія вірогідності і математична статистика | 2002 | «ФІЗМАТЛІТ» | null | 519.2 | null |
| 14 | Дискретно-групові методи інтегрування звичайних диференційних рівнянь | 1991 | «ЛИИАН» | null | 517.9-37 | null |

**Представлення даних у вигляді вертикально-горизонтального зрізу таблиці**

Представлення даних формується як вертикально-горизонтальний зріз таблиці, якщо оператор *SELECT* повертає значення не всіх стовпців відношення й на кортежі, які увійдуть у представлення даних, накладені обмеження.

ПРИКЛАД 3.

Надати користувачам БД «БІБЛІОТЕКА» відомості про книжки, що були видані після 1984 року. У представлення даних помістити назву книжки, рік видання, видавництво та шифр (табл. 3).

***CREATE VIEW****Bookafter1984*

***AS***

***SELECT****Name, IssueYear, Drawing, Cipher*

***FROM****Books*

***WHERE****IssueYear >= '01.01.1985'*

| Таблиця 3 Вертикально-горизонтальний зріз таблиці *Books* за прикладом 3 | | | |
| --- | --- | --- | --- |
| **Name** | **IssueYear** | **Drawing** | **Cipher** |
| Автоматизація виробничих процесів на збагачувальній фабриці | 1985 | «Надра» | 622.7-52/Т |
| Асимптотичні методи оптимального управління | 1987 | «Автомат» | 681.513.5:/А |
| Синтез оптимальних автоматичних систем | 1984 | «Автомат» | 681.513.5:/ДО |
| Методи оптимізації стохастичних систем | 1987 | «Матстат» | 681.513.5:/ДО |
| Автоматизовані системи керування технологічним процесом збагачення руди | 1987 | «Автомат» | 622.7-52/П |
| C/C++ Програмування мовою високого рівня | 2007 | «Пітер» | 681.3.06(075) |
| Комп'ютерні мережі. Принципи, технології, протоколи | 2006 | «Пітер» | 004.72(075) |
| Довідник з диференційних рівняннях з частинними похідними першого порядку | 2003 | «ФІЗМАТЛІТ» | 517.9 |
| Теорія вірогідності і математична статистика | 2004 | «Пітер» | 519.2 |
| C#. Програмування мовою високого рівня | 2009 | «Пітер» | 004.43 |
| Теорія вірогідності і математична статистика | 2005 | «Вища школа» | 519.2 |
| Теорія вірогідності і математична статистика | 2002 | «ФІЗМАТЛІТ» | 519.2 |
| Дискретно-групові методи інтегрування звичайних диференційних рівнянь | 1991 | «ЛИИАН» | 517.9-37 |

**Формування представлення даних як підмножини рядків та стовпців**

Представлення даних формується як підмножина рядків та стовпців, якщо в операторі *SELECT* використовується внутрішні або зовнішні з'єднання відношень.

ПРИКЛАД 4.

Представлення даних має містити: номер читацького квитка, прізвище й ім'я читача, назву, інвентарний номер, вартість, дату видачі, повернення й фактичного повернення книжки (табл. 4).

***CREATE VIEW****Debtors*

*(ReaderCardNumber, ReaderFamilyName, ReaderName, BookName, BookInventoryNumber,*

*BookCost, IssueDate, ReturnDate, FactReturnDate)*

***AS***

***SELECT****R.ReaderCardNumber, R.FamilyName, R.Name, B.Name, I.InventoryNumber,* *I.Cost, G.IssueDate, G.ReturnDate, G.FactReturnDate*

***FROM****Readers R, Books B, BookInventoryNumbers I, BookGiveOutRecord G*

***WHERE****G.ReturnDate < G.FactReturnDate****AND***

*G.ReaderCode = R.Code****AND***

*G.InventoryCode = I.Code****AND***

*I.BookCode = B.Code*

| Таблиця 4 Підмножина рядків та стовпців відношень за прикладом 4 | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ReaderCardNumber** | **ReaderFamilyName** | **ReaderName** | **BookName** | **BookInventoryNumber** | **BookCost** | **IssueDate** | **ReturnDate** | **FactReturnDate** |
| 1345 | Ільїн | Іван | Синтез оптимальних автоматичних систем | 4512890 | 12.99 | 02.09.2004 | 16.09.2004 | 11.12.2004 |
| 543 | Суренко | Дмитро | Автоматизовані системи керування технологічним процесом збагачення руди | 4632112 | 10.10 | 30.10.2004 | 13.11.2004 | 10.01.2005 |
| 485 | Брусов | Володимир | C/C++ Програмування мовою високого рівня | 5478956 | 45.10 | 07.03.2009 | 21.03.2009 | 10.04.2009 |

**Обмеження на створення й використання представлень даних**

Обмеження на створення й використання представлень наведені у стандарті ISO. Ось деякі з них.

Якщо стовпець у представленні даних створюється за допомогою агрегатної функції, то він може вказуватися в пропозиціях *SELECT* і *ORDER BY* тих запитів, які звертаються до даного представлення. Однак такий стовпець не може використовуватися в пропозиції *WHERE*, і не може бути аргументом в узагальнюючій функції цих запитів.

Згруповане представлення даних ніколи не повинно з'єднуватися з таблицями БД або з іншими представленнями.

**Переваги та недоліки представлень даних**

**Переваги**

У разі роботи СКБД на персональному комп'ютері, що стоїть окремо, при використанні представлень зазвичай ставиться за мету лише спрощення структури запитів до БД. Однак, коли СКБД обслуговує запити багатьох користувачів у мережі, представлення відіграють ключову роль у визначенні структури БД та організації захисту інформації. Основні переваги використання представлень у подібному середовищі полягають у наступному.

***Незалежність від даних***

Незалежність від даних для користувача полягає в стабілізації структури БД, яка буде залишатися незмінною навіть у разі перегляду структури вихідних відношень. Наприклад, додавання або вилучення стовпців, змінення зв'язків, поділ відношень, їх реструктуризація або перейменування. Якщо у відношення додаються або вилучаються атрибути, які не використовуються у представленні даних, то змінювати його визначення у БД не потрібно. Якщо структура вихідного відношення перевпорядковується або поділяється на частини, то можна буде створити представлення, що дозволяє користувачам працювати з віртуальним відношенням попереднього формату. У разі поділу вихідного відношення на декілька частин (відношень), старий формат можливо віртуально відновити за допомогою представлення, побудованого на основі з'єднання цих відношень – звичайно, якщо структура нових відношень це дозволяє. Досягається це за допомогою переміщення в усі нові відношення первинного ключа попереднього відношення.

***Актуальність***

Актуальність даних забезпечується терміновим відображенням у представленні усіх змін, що відбулись у будь-якому відношенні БД, яке встановлено у визначальному запиті.

***Підвищення захищеності даних***

Підвищення захищеності даних відбувається за рахунок того, що кожному користувачу права доступу до даних можуть бути надана тільки через обмежений набір представлень, що містять ту підмножину даних, з якою йому необхідно працювати. Такий підхід дозволяє суттєво посилити контроль над доступом окремих категорій груп і окремих користувачів до інформації в БД.

***Додаткові зручності й можливості настроювання***

Додаткові зручності й можливості настроювання моделі даних користувача полягають у максимальному її спрощенні. У результаті однакові відношення можуть бути показані різними користувачами залежно від їх прав доступу та привілеїв. Це дозволить працювати тільки з тією частиною даних, яка дійсно їм необхідна. Так досягається максимальне спрощення моделі даних кінцевого користувача.

***Зниження складності***

Зниження складності відображення даних є наслідком спрощення структури запитів, що поєднують дані з декількох відношень у єдине віртуальне відношення. У результаті багатотабличні запити зводяться до простих, які працюють із одним представленням даних.

***Забезпечення цілісності даних***

Цілісність даних забезпечується за рахунок унеможливлення вводу кортежів, що не задовольняють умовам відбору у визначальному запиті. Це досягається за допомогою пропозиції *WITH CHECK OPTION* оператора *CREATE VIEW*.

**Недоліки**

Використання представлень дозволяє досягти істотних переваг, але необхідно звернути увагу на такі недоліки.

***Обмежені можливості оновлення***

Обмежені можливості оновлення полягають у відсутності механізмів корегування даних у представленні, що побудоване з декількох відношень.

***Структурні обмеження***

Структурні обмеження стосуються конфігурації представлення даних на момент його створення. Наприклад, якщо визначальний запит представлений у форматі *SELECT* \* *FROM* …, то символ «\*» посилається на всі стовпці, що існують у вихідному відношенні на момент його створення. Якщо згодом у вихідне відношення БД будуть додані нові атрибути, то вони не з'являться в представленні доти, доки воно не буде вилучене і знову створене.

***Зниження продуктивності***

Зниження продуктивності СКБД пов'язане з використанням додаткових обчислювальних ресурсів для розв’язання представлень. Іноді вплив цього фактора буде абсолютно незначним, тоді як в інших випадках він може бути джерелом істотних проблем. Наприклад, представлення, визначене за допомогою складного багатотабличного запиту, потребує значних витрат часу на обробку, оскільки щоразу виконуються з'єднання відношень, коли знадобиться доступ до представлення.

**Висновок**

Представлення даних – є вікном, яке розробник БД надає кожному користувачу для отримання інформації, що відповідає його вимогам, та на яку він має повноваження. Таким чином модель реляційних даних, яку бачить користувач, не залежить від внутрішньої структури БД. Додатково представлення даних дозволяють спростити структуру SQL-запитів для отримання звітів, які необхідні користувачам.

**ClassWork**

CREATE VIEW Personals (Id\_pers,Name\_Pers) as

SELECT St.ID\_stud ID, St.name NAME FROM Students St

UNION

SELECT ID\_T,Name\_T FROM Teachers

ORDER by Name

select \* from Personals

select \* from Personals order by Id\_pers

drop view Personals

CREATE VIEW StSu (Name\_Stud,Name\_Subj,Grade\_Stud, Avg\_stud) as

SELECT St.name, Su.Sub\_name, G.grade, round(avg(G.grade),1) AVG\_G

FROM Grades G,Students St, Subject Su

WHERE G.CodeStud=St.ID\_stud and G.Sub\_code=Su.Sub\_code

group by G.Sub\_code,G.CodeStud

select \* from StSu

select \* from StSu where Avg\_stud-Grade\_Stud > 0

select \* from StSu --where Avg\_stud=Grade\_Stud> 0

--group by Name\_Stud

order by Name\_Stud

select Name\_Stud,Name\_Subj,sum(Grade\_Stud), Avg\_stud from StSu --where Avg\_stud=Grade\_Stud> 0

group by Name\_Stud

order by Name\_Stud

CREATE VIEW StSuGr (Name\_Stud,Name\_Subj,Sum(Grade\_Stud) SG, Avg\_stud) as

SELECT St.name, Su.Sub\_name, G.grade--, round(avg(G.grade),1) AVG\_G

FROM Grades G,Students St, Subject Su

WHERE G.CodeStud=St.ID\_stud and G.Sub\_code=Su.Sub\_code

group by G.Sub\_code,G.CodeStud

select Name\_Stud,Name\_Subj,sum(Grade\_Stud), Avg\_stud from StSu --where Avg\_stud=Grade\_Stud> 0

group by Name\_Stud

order by Name\_Stud

SELECT St.name, Su.Sub\_name,count(G.grade),sum(G.grade), round(avg(G.grade),1) AVG\_G

FROM Grades G,Students St, Subject Su

WHERE G.CodeStud=St.ID\_stud and G.Sub\_code=Su.Sub\_code

group by G.Sub\_code,G.CodeStud

CREATE VIEW StSuGr (Name\_Stud,Name\_Subj,summa,Avg\_stud) as

select Name\_Stud,Name\_Subj,sum(Grade\_Stud), Avg\_stud from StSu --where Avg\_stud=Grade\_Stud> 0

group by Name\_Stud

order by Name\_Stud

select \* from StSuGr

select T.ID\_T,T.Name\_T,TS.Sub\_code,S.Sub\_name FROM Teachers T,TeachSubj TS,Subject S

where T.ID\_T=TS.ID\_T and TS.Sub\_code=S.Sub\_code

Create VIEW TeSu(T\_Name,Sub\_code,Sub\_name) as

select T.Name\_T,TS.Sub\_code,S.Sub\_name FROM Teachers T,TeachSubj TS,Subject S

where T.ID\_T=TS.ID\_T and TS.Sub\_code=S.Sub\_code

CREATE view GrSU (Name\_gr,Sub\_code) as

select GP.Name\_gr, GR.Sub\_code from Grades GR,Students S,Groups GP

where S.Id\_Gr=GP.Id\_Gr and GR.CodeStud=S.ID\_stud