МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

з дисципліни

«БАЗИ ДАНИХ ТА СХОВИЩА ДАНИХ»

для студентів денної форми навчання
напряму 126 «Інформаційні системи та технології»

Електронне видання

ЗАТВЕРДЖЕНО
кафедрою ПІТУ
Протокол № ____ від _____

Харків 2018

Методичні рекомендації для виконання лабораторних робіт з дисципліни «БАЗИ ДАНИХ ТА СХОВИЩА ДАНИХ» для студентів денної форми навчання напряму 126 «Інформаційні системи та технології» [Електронне видання] / Упоряд. Сокол В.Є., Шматко О.В., Фонта Н.Г., Іващенко О.В. - Харків: НТУ «ХПІ», 2018. – с.

Упорядники Сокол В.Є., Шматко О.В., Фонта Н.Г., Іващенко О.В.

ЗМІСТ

Вступ

У зв’язку з широким поширенням корпоративних інформаційних систем актульною є підготовка спеціалістів в області розподіленої обробки даних. В методичних рекомендаціях розглядаються теоретичні основи та методи, які використовуються при розробці сучасних розподілених систем обробки інформації. Особлива увага приділяється технології проектування баз даних за допомогою ER-діаграм та способам створення об’єктів бази даних. Також розглядаються питання проектування та створення сховищ даних.

Методичні рекомендації містять систему лабораторних робіт, послідовне виконання котрих забезпечить формування стійких навичків у проектуванні, побудові розподілених баз даних та сховищ даних.

Методичні рекомендації до лабораторних робіт можуть застосовуватися для самостійної роботи при навчання розподіленим базам даних та сховищам даних.

Лабораторна робота № 1. Проектування розподіленої бази даних

Мета лабораторної роботи: Вивчення методики проектування розподілених баз даних

Завдання

1. Для заданої предметної області побудувати ER-модель, виділити сутності, описати атрибути кожної сутності, встановити зв'язки між сутностями.

2. Розробити схему бази даних. При необхідності провести нормалізацію відносин.

3. При побудові фізичної структури бази даних узгодити іншими варіантами назву полів, їх ідентифікаторів і типів даних для всіх сутностей (загальні першій-ліпшій нагоді поля повинні мати однакові ідентифікатори).

Наприклад, позначити таблицю постачальників - **PROVIDER** з атрибутами prov_id, prov_name і т.ін., покупців (споживачів) - **CUSTOMER**, продукція - **PRODUCT** тощо.

4. Реалізувати отриману схему в середовищі Oracle на своїй робочій станції (далі робочі станції будемо позначати **WS1**, **WS2**, ..., **WSn**). При створенні дати ім'я базі даних по імені відповідної робочої станції (наприклад, **WS1**, **WS2** і **WS3**).

5. Заповнити таблиці даними (не менше 10 записів в кожній таблиці бази даних. При цьому дані про постачальників і споживачів не повинні збігатися на різних робочих станціях).

6. Оформити звіт про виконання лабораторної роботи.

Зміст звіту

1. Мета роботи.

2. ER-модель предметної області з зазначенням імен, типів і

обов'язковості кожної зв'язку.

3. Реляційна модель бази даних із зазначенням первинних атрибутів.

4. Структура фізичних таблиць Oracle.

Лабораторна робота № 2. Маніпуляція даними в системах розподілених баз даних

Мета лабораторної роботи: Вивчення методів зв'язування об'єктів і маніпуляцій даними бази даних, розподіленої на декількох комп'ютерах мережі.

Короткі теоретичні відомості

Більшість систем РД складається з декількох баз даних, керованих різними серверами, розташованими в різних місцях. Всі сервери і клієнти Oracle повинні використовувати Net8 - мережеве програмне забезпечення Oracle для взаємодії один з одним по мережі. Кожен сервер бази даних в РБД управляє доступом до своєї локальної базі даних - за керування системою в цілому не відповідає жоден сервер.

Іменування об'єктів

Всі сервіси, доступні в мережі, повинні мати унікальні імена, щоб користувачі і додатки знали, як з ним поводитися. Глобальне ім'я бази даних складається з двох частин:

• основне ім'я бази даних, яка призначається їй при створенні. Ім'я бази даних не повинно містити більше восьми символів.

• мережевий домен бази даних, який показує логічне місцезнаходження бази даних в мережі.

На рис. 1 представлена мережа баз даних гіпотетичної компанії **SALESMENT** (продажу). Мережа **SALESMENT** складається з трьох баз даних **WS1**, **WS2** і **WS3**. Їм відповідають глобальні імена (імена сервісів) **WS1. SALESMENT**, **WS2. SALESMENT**, **WS3. SALESMENT**.

Для того, щоб посилатися на конкретні імена об'єктів схеми бази даних, що не є локальною, потрібно доповнити ім'я об'єкта глобальним ім'ям бази

даних. На рис. 2.1 показано, що в кожній з баз даних **WS1**, **WS2** і **WS3** міститься таблиця **PROVIDER** (виробник / постачальник).

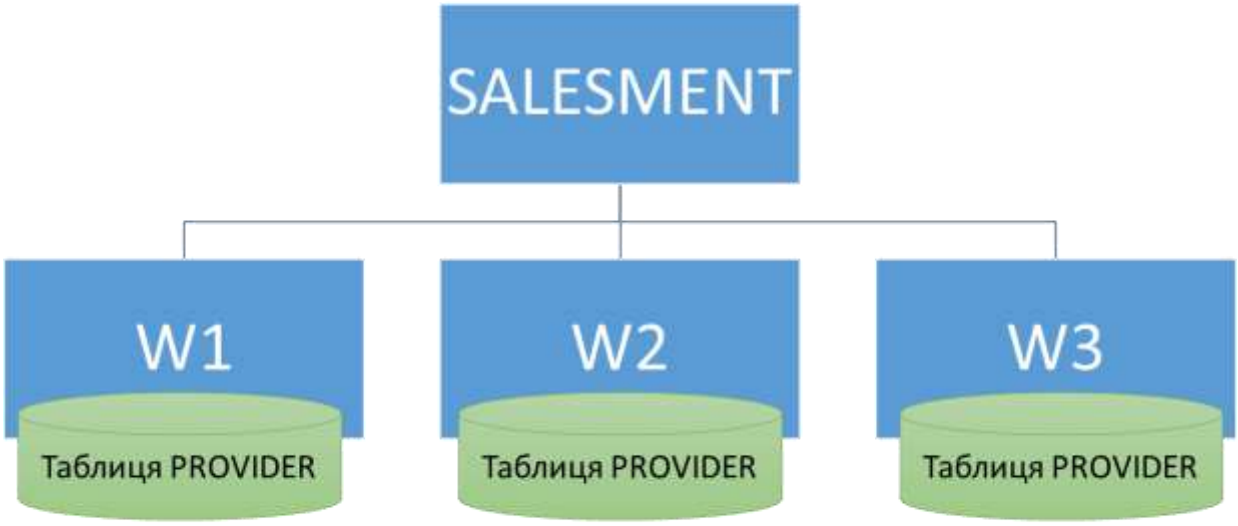

Рисунок 2.1 – Структура мережі баз даних **SALESMENT**

Якщо запустити додаток (наприклад, SQL * Plus) і з'єднатися з базою даних (наприклад, **WS2**), то можна звернутися як до таблиці **PROVIDER** бази даних **WS2**, так і до таблиці **PROVIDER** бази даних **WS1**, ідентифікувавши цей об'єкт за допомогою його складеного імені в розподіленої базі даних:

```
SELECT * FROM provider@ws1.salesment;
```

Виконуючи цей запит, сервер локальної бази **WS2** неявно використовує зв'язок баз даних, що сполучає бази даних **WS1** і **WS2**.

Прозорість розподіленої бази даних

Користуватися зазначеної системою іменування можливо при розробці дуже невеликих баз даних, тому що кожен розробник повинен знати місце свого об'єкту в системі розподіленої бази даних.

У Oracle є засіб, що дозволяє зробити роботу з цими об'єктами прозорою. Це механізм створення синоніма, який приховує фізичне місце зберігання об'єкта в системі розподіленої бази даних:

```
    CREATE PUBLIC SYNONIM prov1 FOR
provider@ws1.salesment;
```

Після створення загального синоніма користувачі локальної бази даних можуть посилатися на віддалену таблицю **PROVIDER** робочої станції **WS1** як на локальну. Oracle автоматично перетворює локальний псевдонім в ім'я віддаленої таблиці і використовує для звернення до неї зв'язок бази даних.

```
    SELECT * FROM prov1;
```

Іншим засобом прозорості можуть служити уявлення. Наприклад, локальне уявлення **PRODUCT** вказує на дані, що містяться в віддаленої таблиці **PRODUCT** робочої станції **WS1**.

```
CREATE VIEW product AS SELECT * FROM
product@ws1.salesment;
```

Встановлення зв'язків баз даних

Oracle пропонує таку підтримку розподілених баз даних, яка дозволяє виробляти віддалені і розподілені запити по каналах зв'язку баз даних.

Для того щоб надати доступ до віддалених баз даних в розподіленій системі, необхідно встановити в локальній баз даних зв'язку баз даних (**database link**). Зв'язок двох баз даних позначає односпрямовану лінію зв'язку між двома базами даних Oracle (рис. 2.2). **WS1. SALESMENT**

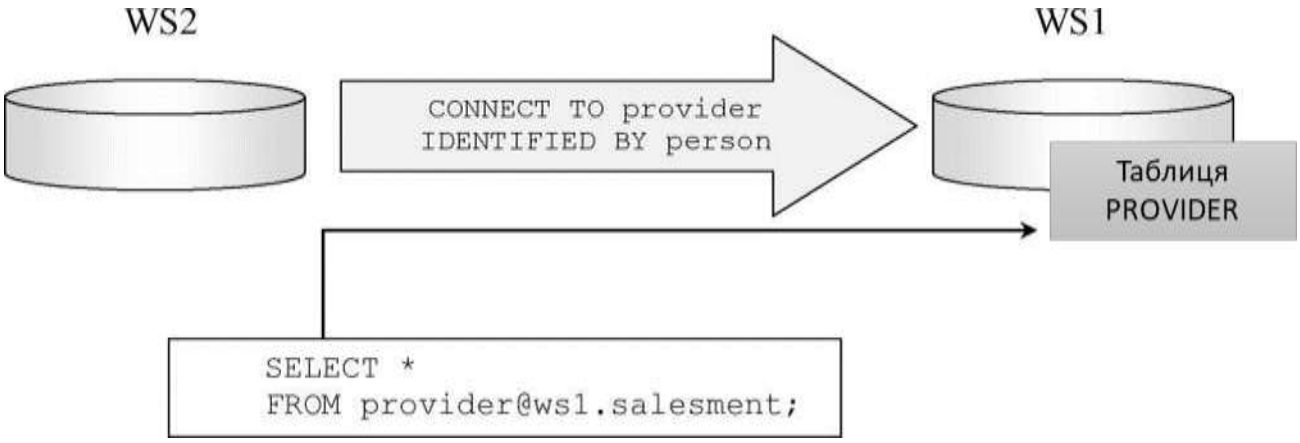

Рисунок 2.2 – Приклад зв`язку в розподіленій базі даних Oracle

Встановлення зв'язків можна використовувати для досягнення двоякої мети - забезпечення прозорості розташування і незалежності розташування. Наприклад, за допомогою наступного оператора в локальній базі **WS2** можна створити зв'язок, який описує шлях до віддаленої бази даних **WS1. SALESMENT**:

```
CREATE DATABASE LINK ws1.salesment;
```

При розробці додатків для роботи в розподіленій базі даних необхідно виконувати різні операції маніпуляцій з даними: віддалені запити, поновлення і т.ін. Розглянемо приклади операторів SQL і PL / SQL, що дозволяють

виконати ці операції.

Встановивши зв'язки між базами даних, програма може маніпулювати даними на будь-якому з вузлів мережі.

Дистанційні запити

Віддалений запит (**remote query**) - це оператор **SELECT**, що зчитує інформацію в декількох віддалених таблицях, які перебувають на одному і тому ж віддаленому вузлі. Наприклад, за допомогою наступного віддаленого запиту інформація зчитується в віддалених таблицях **PROVIDER** і **PRODUCT** бази даних **WS3**.

```
    SELECT prov.name, p.name, p.price FROM
provider@ws3.salesment prov, product@ws3.salesment p;
```

Розподілені запити

Розподілений запит (**distributed query**) - це оператор **SELECT**, що зчитує інформацію з двох або більше баз даних. Наприклад, за допомогою наступного розподіленого запиту інформація зчитується в локальній таблиці **PROVIDER** і віддаленої таблиці **PRODUCT** бази даних **WS3**.

```
    SELECT prov.name, p.name, p.price FROM provider
prov, product@ws3.salesment p;
```

Віддалене оновлення

Віддалене оновлення (**remote update**) - це операція оновлення, за допомогою якої модифікуються дані віддаленої таблиці. Наприклад, за

допомогою наступного оператора **UPDATE** оновлюється рядок в таблиці **PRODUCT** віддаленої бази даних **WS1**.

```
UPDATE product@ws1.salesment SET prod_price = 200 WHERE
prod_id = 1;
```

Розподілене оновлення

Розподілене оновлення (**distributed update**) модифікує дані двох і більше серверів. Єдиний спосіб розподіленого поновлення - створити збережену процедуру, метод об'єктних типів і т.ін. і включити їх до складу двох або більше операції оновлення, кожна з яких оновлює дані в різних базах даних. Наприклад, наступний анонімний блок PL / SQL можна вважати розподіленим оновленням:

```
BEGIN
UPDATE product
SET prod_price = 200 WHERE prod_id = 1;
UPDATE product@ws2.salesment SET prod_price = 200 WHERE prod_id
= 1;
UPDATE product@ws3.salesment SET prod_price = 200 WHERE prod_id
= 1;
END;
```

Дистанційні транзакції

Дистанційна транзакція (**remote transaction**) - це транзакція, яка містить один або більше віддалених операторів, кожен з яких посилається на одну і ту ж віддалену базу даних. Наприклад, наступна віддалена транзакція оновлює інформацію тільки в базі даних **WS1**.

```
UPDATE product@ws1.salesment SET prod_price = 200 WHERE prod_id
= 1;
UPDATE product@ws1.salesment SET prod_price = 300 WHERE prod_id
= 2;
UPDATE product@ws1.salesment SET prod_price = 150 WHERE prod_id
= 3;
COMMIT;
```

Розподілені транзакції

Розподілена транзакція (**distributed transaction**) - це транзакція, що включає один або більше операторів, що оновлюють інформацію в двох і більше різних базах даних. Наприклад, наступна розподілена транзакція оновлює інформацію в декількох базах даних.

```
     UPDATE product
     SET prod_price = 200 WHERE prod_id = 1;
     UPDATE product@ws1.salesment SET prod_price = 200 WHERE
prod_id = 1;
     UPDATE product@ws3.salesment SET prod_price = 150 WHERE
prod_id = 3;
     COMMIT;
```

Двофазна фіксація

При встановленні з'єднань програма може оновлювати дані де завгодно. Однак вона також повинна видати явну команду фіксації в кожен з примірників, в який вона видала DML-команди, інакше в якийсь момент часу різні користувачі можуть бачити різний стан бази даних.

Для транзакції, як єдиного цілого, повинна бути виконано або повне фіксування (операція завершення), або повний відкат. Щоб забезпечити дотримання цього правила для розподілених транзакцій, в Oracle застосовується спеціальний алгоритм двофазного завершення (**two-phase commit mechanism**), який координує управління транзакціями в мережі.

Двофазне завершення необхідно при мережевих і системних збоях, які можуть переривати завершення розподілених транзакцій.

У загальних рисах дія цього механізму виглядає наступним чином: коли видається команда фіксації, один з примірників бази даних, який бере участь в транзакції, приймає на себе роль координатора. На етапі першої фази цей екземпляр вибирає один з примірників в якості точки фіксації і дає всім іншим задіяним екземплярів (серед яких може бути і він сам, якщо він не є точкою фіксації) вказівку приготуватися. Після того як отримано підтвердження на фіксацію від інших примірників, точці фіксації пропонується виконати звичайну фіксацію. Залежно від результату цієї фіксації координатор просить всі інші екземпляри або зафіксувати транзакцію, або виконати її відкат.

Звичайно, і в цьому випадку можливі всякого роду збої. У будь-який момент можуть відмовити як окремі сервери, так і мережеві канали зв'язку, але, незважаючи ні на що, програмне забезпечення повинно гарантувати цілісність даних. В результаті не тільки виникає значний трафік повідомлень, а й створюються потенційно стійкі блокування на рівні блоків.

Механізми двофазного завершення є внутрішніми процесами сервів баз даних Oracle, що беруть участь в транзакції. Користувач повинен лише закінчити розподілену транзакцію оператором **COMMIT**; іншу роботу виконує Oracle.

Закриття каналів зв'язку

Для "популярних" серверів, тобто серверів, які є об'єктом безлічі відкритих каналів зв'язку БД, число одночасно відкритих Oracle-з'єднань може стати таким великим, що викличе надмірне підкачування сторінок пам'яті. Останні версії Oracle не тільки дозволяють адміністратору встановлювати, яку максимальну кількість каналів зв'язку БД може бути відкрито для процесу (через параметр DB_LINKS в файлі init.ora), але і дозволяють сеансу закривати канал зв'язку БД, що займає ресурси на віддаленому сервері. Ось команди для цього прикладу:

```
ALTER SESSION CLOSE DATABASE LINK ws1.salesment;
ALTER SESSION CLOSE DATABASE LINK ws2.salesment;
ALTER SESSION CLOSE DATABASE LINK ws2.salesment;
```

На жаль, для підтримання зв'язку з Oracle необхідні великі витрати часу центрального процесора на серверній стороні каналу, тому при скільки-небудь значній ймовірності того, що цей канал в найближчому майбутньому знадобиться знову, навряд чи ефективно його закривати. "Заморожування" одного-двох мегабайтів пам'яті на віддаленому сервері може виявитися меншим злом, ніж витрати на підключення та відключення, які доведеться понести потім. Якщо ж віддалені сервери мають обмежені ресурси і відомо, що будь-які канали навряд чи ще будуть використовуватися, то їх закриття дозволить збільшити обсяг доступних ресурсів. Однак при цьому також зруйнується прозорість розташування і потрібно більш багате функціональними можливостями додаток, ніж те, яке захочуть реалізувати більшість проектувальників і програмістів.

Варіанти збільшення продуктивності при розподілених з'єднаннях

Розподілені з'єднання можуть створювати проблеми з продуктивністю весь час, поки виконуються операції маніпулювання даними в розподілених базах даних.

Якщо продуктивність незадовільна, можна розглянути наступні варіанти:

• На ранніх стадіях проекту необхідно постійне тестування розподіленого з'єднання на конкретних прикладах.

• Створити з'єднання так, щоб забезпечити посилання на одне або кілька подань і перемістити ключові елементи оптимізації запитів на сервери, де їх можна оптимізувати з більшою ефективністю.

• Виконати з'єднання засобами програми з використанням віддаленого SQL. Ця стратегія може бути ефективною, якщо видалені дані можна отримати за один запит або за дуже мале число таких запитів.

• Змінити розподіл даних, використаних в з'єднанні, перенісши одну або кілька таблиць в іншу базу даних (або будь-яким чином створити репліки цих даних). Це рішення вимагає фундаментальної зміни в структурі, і для пошуку ефективної стратегії розподілу, можливо, буде потрібно перебрати кілька варіантів.

Останнє рішення підводить нас до суті проектування розподілених баз даних:

Стратегія розподілу даних повинна визначатися вимогами до продуктивності і працездатності додатки, тобто дані повинні бути розподілені по мережі таким чином, щоб максимально відповідати запитам інформаційної системи, що використовує ці дані.

Завдання

1. Встановити зв'язки між локальними базами даних **WS1**, **WS2**, **WS3**.

2. Користувачу, обраному в якості адміністратора, створити синоніми для локальних баз даних **WS1**, **WS2**, **WS3**.

3. Створити уявлення на основі локальних і віддалених таблиць.

4. Перевірити працездатність створених уявлень командою **SQL Select**.

5. Виконати віддалений запит, віддалене оновлення, віддалене додавання, віддалене видалення. При виконанні використовуйте детальний контроль доступу (**WHERE**).

6. Виконати розподілений запит, розподілене оновлення, розподілене додавання, розподілене видалення. При виконанні використовуйте детальний контроль доступу (**WHERE**).

7. Виконати віддалену транзакцію.

8. Виконати розподілену транзакцію.

9. Оформити звіт про виконання лабораторної роботи.

Зміст звіту

1. Мета роботи.

2. Оператори **SQL**, що дозволяють створювати синоніми, уявлення, зв'язку в системі розподіленої бази даних.

3. Оператори **SQL**, що дозволяють виконати віддалені і розподілені маніпуляції з даними.

4. Структура фізичних таблиць Oracle.

Лабораторна робота № 3. Фрагментація бази даних

Мета лабораторної роботи: Вивчення методів фрагментації бази даних, розподіленої на декількох комп'ютерах мережі.

Короткі теоретичні відомості

Таблиці бази даних можуть бути фрагментовані, тобто, розбиті на частини, і ці частини можуть зберігатися в різних вузлах розподіленої бази даних. Підставою для фрагментації є підвищення продуктивності. Частини таблиці зберігаються в тих місцях, де до них відбуваються найбільш часті звернення - це зменшує мережевий трафік і скорочує час доступу до даних.

Фрагментація може бути горизонтальної або / і вертикальної.

Горизонтальна фрагментація

Горизонтальна фрагментація - розбиття таблиці по рядках. У цьому випадку значення в деякому стовпці (комбінація значень в стовпцях) розглядається як ключ фрагмента, однозначно визначає фрагмент, в який входить рядок таблиці. Так, наприклад, якщо таблиця **PRODUCT** представляє список товарів, то дані про кожен товар можуть зберігатися в вузлі того відділу, який займається продажем цього товару. Очевидно, що ключем фрагмента в цьому випадку повинен бути код відділу. Оскільки в базі даних вузла фрагмент визначається як таблиця, в визначення таблиці може (і повинно) вводитися обмеження на значення ключа фрагмента.

Вертикальна фрагментація

Вертикальна фрагментація - розбиття таблиці по стовпцях. Так, характеристики сортності товару можуть зберігатися в фрагменті на вузлі відділу якості, стовпці, що визначають вартість - в фрагменті на вузлі відділу продажів і т.ін. Вертикальна фрагментація, по суті, являє собою декомпозицію в схемі даних, коли одна таблиця розбивається на дві або більше таблиць,

пов'язаних один з одним відношенням 1: 1. Декомпозиція повинна бути виконана без втрат, тобто, набір значень первинних ключів у всіх таблицях повинен повністю збігатися.

Незалежність від фрагментації полягає в тому, що жоден з фрагментів не є похідним від інших фрагментів. Відновлення повної таблиці з фрагментів проводиться операціями об'єднання (горизонтальна фрагментація) або / та природного з'єднання (вертикальна фрагментація) таблиць-фрагментів. Повна таблиця може бути описана як уявлення, яке визначається через зазначені операції. При оновленні подання повної таблиці можуть виникати проблеми, типові для поновлення уявлень: необхідність визначення того фізичного фрагмента, до якого відноситься оновлення. Так, у наведеному вище прикладі горизонтальної фрагментації переклад товару з однієї категорії в іншу (зміна значення в стовпці коду категорії) зажадає перенесення запису з одного фрагмента до іншого.

Фрагментація при розподілених запитах

У розподілених запитах оптимізація ще більш важлива, ніж в локальних. Оскільки виконання розподілених запитів включає в себе звернення до віддалених вузлів і пересилання даних між вузлами, мінімізація числа таких звернень і обсягу даних, що пересилаються на багаторазово зменшити час виконання запиту. Виконання розподілених запитів включає в себе етап глобальної оптимізації, на якому оптимізатор вирішує, які дані і з якого вузла на який будуть пересилатися, і локальної оптимізації на кожному бере участь в запиті вузлі.

Наприклад, якщо на вузлі А є таблиця ТА, що містить 100 рядків, а на вузлі В є таблиця ТВ, яка містить 106 рядків, і запит вимагає з'єднання цих таблиць, то очевидно, що вигідніше переслати таблицю ТА на вузол В і виконати з'єднання на вузлі В , а не навпаки.

Інший приклад. Вище ми розглянули приклад таблиці **PRODUCT** (список товарів), фрагментований горизонтально по продукції певного виду.

Якщо в описах її фрагментів вказано обмеження на код продукту для фрагмента, і оптимізатор "знає" про це обмеження, то запит:

```
SELECT * FROM product WHERE prod_cod = 'milk'
```

оптимізатор може локалізувати на єдиному вузлі - на тому, для якого обмеження ключа фрагмента збігається з умовою, заданому в запиті.

Розглянемо приклад, який ілюструє обидва типи фрагментації. Є таблиця **PROVIDER** (*prov_id, prov_name, prov_phone і т.ін.*), в базі даних на вузлі **WS1**. Є така сама таблиця, в базі даних на вузлі в **WS2**. Обидві таблиці зберігають інформацію про постачальників товару. Крім того, в базі даних на вузлі в **WS3** визначена таблиця **PRODUCT** (*prod_id, prod_price*). Тоді запит "отримати інформацію про організації-постачальниках" може бути сформульований так:

```
SELECT *
FROM provider@ws1.salesment, provider@ws2.salesment,
ORDER BY prov_id;
```

У той же час запит "отримати інформацію про вартість поставлених товарів" буде виглядати наступним чином:

```
     SELECT product.prod_id, product.prod_price,
provider.prov_name
     FROM provider@ws1.salesment, provider@ws2.salesment,
product@ws3.salesment ORDER BY prod_id;
```

Завдання (необхідно адаптувати до своєї предметної області)

1. На основі горизонтальної фрагментації:

• створити уявлення, в яке потраплять, наприклад, всі постачальники продукції всіх видів;

• створити уявлення із зазначенням всіх атрибутів, в яке потраплять, наприклад, всі споживачі продукції всіх видів;

• створити уявлення, що містить список, наприклад, всіх продуктів всіх виробників із зазначенням коду (артикулу), найменування та вартості;

2. На основі вертикальної фрагментації:

• виділити стовпці, що відповідають, наприклад, за характеристику товару (наприклад, *сортність*) в окрему таблицю і розмістити її на робочій станції **WS1**;

• виділити стовпці (наприклад, *код*, *найменування*, *вартість*) в окрему таблицю і розмістити її на робочій станції **WS2**;

• з робочої станції **WS3** виконати розподілений запит і отримати уявлення з таблиць, отриманих раніше завдання 2), що містить, наприклад, інформацію про товари 1 сорту, вартість яких відрізняється від середньої не більше, ніж на 10 %;

3. Оформити звіт про виконання лабораторної роботи.

Лабораторна робота №4. Проектування структури сховища даних

Мета лабораторної роботи: Вивчення методів проектування сховищ даних.

Короткі теоретичні відомості

Визначення та типові архітектури сховищ даних

Визначення поняття «сховище даних» першим дав Вільям Інмона в своїй монографії - це «предметно-орієнтована, інтегрована, яка містить історичні дані, неруйнівного сукупність даних, призначена для підтримки прийняття управлінських рішень».

Дані з різних джерел поміщаються в сховище, а їх опис - в репозиторій метаданих. Кінцевий користувач, використовуючи різні інструменти (засоби візуалізації, побудови звітів, статистичної обробки і т.ін.) і вміст сховища аналізує дані в сховищі. Результатом є інформація у вигляді готових звітів, знайдених прихованих закономірностей, будь-яких прогнозів. Так як цілі роботи кінцевого користувача зі сховищем даних можуть бути найрізноманітнішими, то теоретично їх вибір не повинен впливати на структуру сховища і функції його підтримки в актуальному стані. Фізична реалізація даної концептуальної схеми може бути найрізноманітнішою.

Дворівнева архітектура сховища даних передбачає побудову вітрин даних (**data mart**) без створення центрального сховища, при цьому інформація надходить з реєструючих систем і обмежена конкретною предметною областю. При побудові вітрин використовуються основні принципи побудови сховищ даних, тому їх можна вважати сховищами даних в мініатюрі. Плюси: простота і низька ціна реалізації; висока продуктивність за рахунок фізичного поділу реєструють і аналітичних систем, виділення завантаження і

трансформації даних в окремий процес, оптимізованої під аналіз структурою зберігання даних; підтримка історії; можливість додавання метаданих.

Побудова повноцінного корпоративного сховища даних зазвичай виконується в трирівневії архітектурі. На першому рівні розташовані різноманітні джерела даних - внутрішні реєструючі системи, довідкові системи, зовнішні джерела (дані інформаційних агентств, макроекономічні показники). Другий рівень містить центральне сховище, куди стікається інформація від всіх джерел з першого рівня, і, можливо, оперативний склад даних, який не містить історичних даних і виконує дві основні функції. По-перше, він є джерелом аналітичної інформації для оперативного управління і, по-друге, тут підготовляються дані для подальшого завантаження в центральне сховище. Під підготовкою даних розуміють їх перетворення і проведення певних перевірок. Наявність оперативного складу даних просто необхідно при різному регламенті надходження інформації з джерел. Третій рівень являє собою набір предметно-орієнтованих вітрин даних, джерелом інформації для яких є центральне сховище даних. Саме з вітринами даних і працює більшість кінцевих користувачів.

Види моделей сховища даних

Сховища будуються на основі багатовимірної моделі даних, що має на увазі виділення окремих вимірювань (час, географія, клієнт, рахунок) і фактів (обсяг продажів, дохід, кількість товару) з їх аналізом за обраними вимірам. Багатовимірна модель даних фізично може бути реалізована як в багатовимірних, так і в реляційних СУБД. В останньому випадку вона виконується за схемою «зірка» або «сніжинка». Кожна таблиця фактів містить детальні дані і зовнішні ключі на таблиці вимірювань.

Теорія побудови багатовимірної моделі даних і її втілення в реляційної структурі відома, проте інформації з проблеми поданням ієрархій дуже мало. Як приклад вимірювання, широко застосовується при аналізі діяльності підприємства і має ієрархічну структуру, можна привести довідник статей

витрат (рис. 4.1).

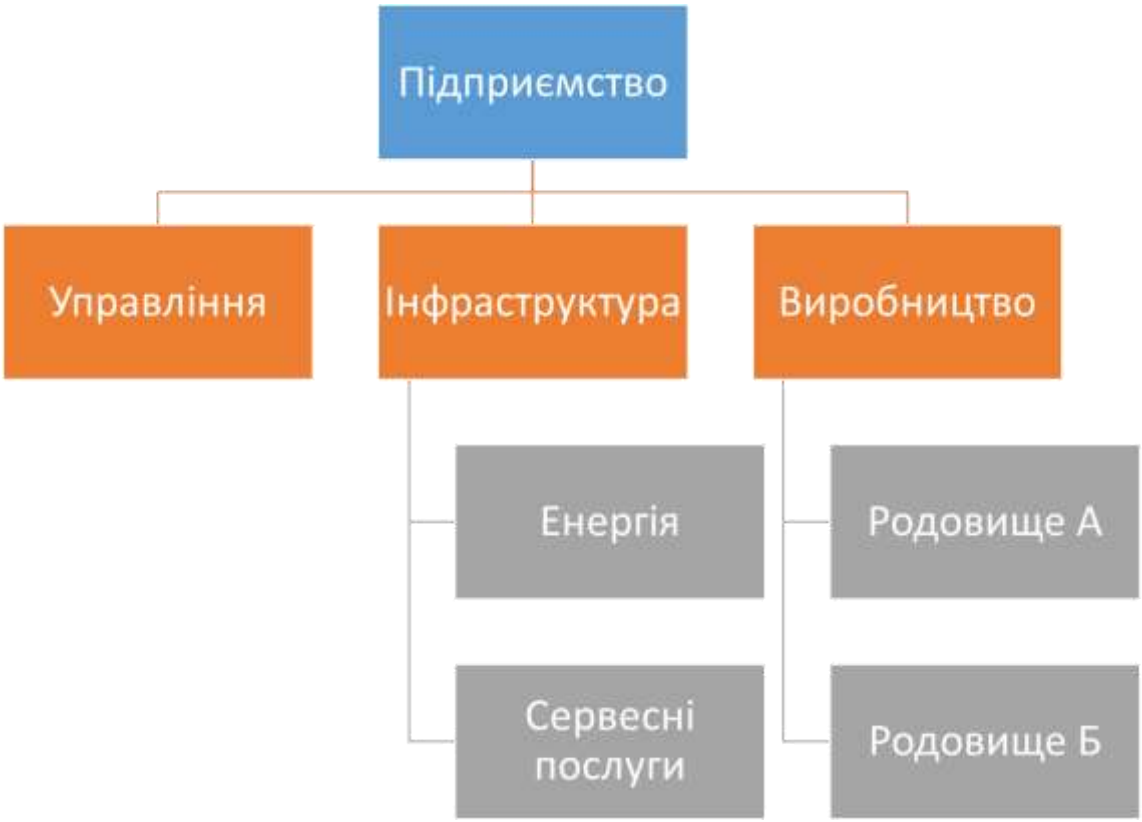

Рисунок 4.1 - Модель ієрархічного довідника

Класична проблема представлення ієрархій вирішується за допомогою рекурсивного зв'язку, що дозволяє поміщати в одній таблиці дерево будь-якої глибини і розмірності. У нашому випадку отримані дані представлені у вигляді таблиці 4.1.

Таблиця 4.1 - Представлення ієрархій за допомогою рекурсивної зв'язку

ID	NAME	PARENT ID
1	Підприємство	
2	Управління	1
3	Інфраструктура	1
4	Виробництво	1
5	Енергія	3
6	Сервісні послуги	3

7	Родовище А	4
8	Родовище Б	4

Однак в простоті цього рішення ховається і основний його недолік: стандарт SQL не підтримує рекурсивні покажчики, тому для подання дерев в сховище даних використовують інші методи.

Метод, запропонований Джо Селко, заснований на теорії множин - всі вузли дерева проходяться в прямому порядку і для кожного вузла заповнюються два значення (рис. 4.2).

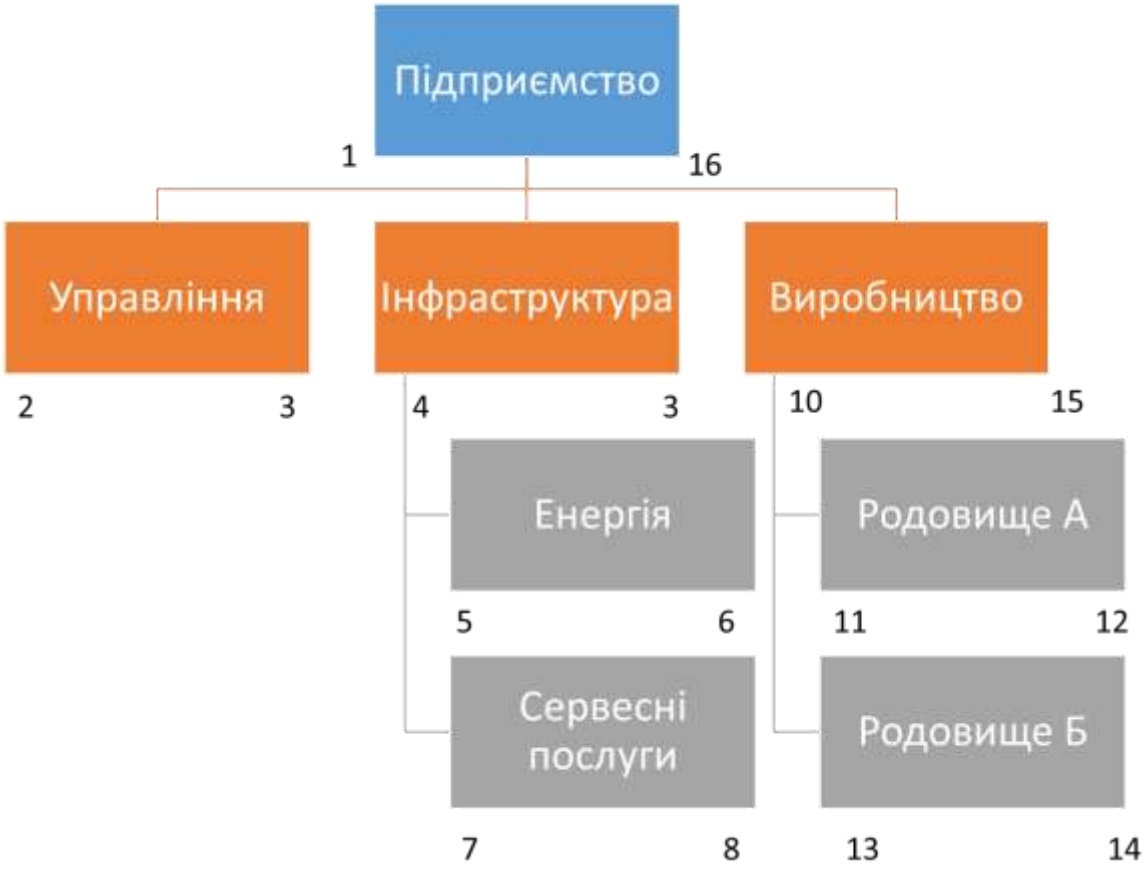

Рисунок 4.2 – Нумерація лівої і правої меж вузлів дерева

Спочатку заповнюється ліва межа і лише потім права - при русі назад від нащадків до батьків. При такій нумерації вузлів кожен батько містить нащадків, ліва і права межа яких лежить в інтервалі між лівою і правою кордоном батька. Аналогічно всі батьки нащадка мають ліву межу, яка менше

лівої межі нащадка і праву, велику правої межі нащадка. Отже, суму витрат для конкретного місця виникнення витрат і всіх його складових можна отримати одним запитом. Наприклад, для отримання витрат по інфраструктурі можна виконати наступний SQL-запит:

```
    select sum(fact_table.cost)
from fact_table, dimension_table D1, dimension_table D2
where fact_table.dimension_id = D2.id
and D2.left >= D1.left
and D2.right <= D1.right
and D1.name = «Інфраструктура»
```

Для простоти роботи з таким довідником крім полів *left*, *right* варто додати ще два поля: «*Level*» - рівень вузла в дереві, «*Is_leaf*» - прапор, який показує чи є вузол листом в дереві чи ні. Таким чином, ми отримуємо таблицю «*dimension_table*» (таблиця 4.2), яка дозволяє зберігати дерево будь-якої глибини укладення і розмірності і вибирати нащадків і батьків за допомогою одного запиту.

Таблиця 4.2 - Представлення ієрархій за допомогою лівої і правої меж

ID	NAME	LEFT	RIGHT	LEVEL	IS-LEAF
1	Підприємство	1	16	1	N
2	Управління	2	3	2	Y
3	Інфраструктура	4	9	2	N
4	Виробництво	10	15	2	N
5	Енергія	5	6	3	Y
6	Сервісні послуги	7	8	3	Y

7	Родовище А	11	12	3	Y
8	Родовище Б	13	14	3	Y

Інший спосіб, описаний Ральфом Кімболом, заснований на введенні допоміжної таблиці («*helper-table*»), через яку здійснюється зв'язок таблиці фактів з таблицею вимірювання. Ця допоміжна таблиця відображає ієрархічну структуру вимірювання і підпорядковується наступному закону: допоміжна таблиця містить весь набір пар «*батько-нащадок*», причому нащадок може не бути безпосереднім нащадком батька. Структура такої таблиці і її вміст показано в таблиці 4.3.

Таблиця 4.3 - Структура і зміст допоміжної таблиці

PARENT ID	CHILD ID	DISTANCE	IS LEAF
1	1	0	N
1	2	1	N
1	3	1	N
1	4	1	N
1	5	2	N
1	6	2	N
1	7	2	N
1	8	2	N
2	2	0	Y
3	3	0	N
3	5	1	N
3	6	1	N
4	4	0	N
4	7	1	N
4	8	1	N
5	5	0	Y
6	6	0	Y
7	7	0	Y
8	8	0	Y

Пов'язуючи таблицю фактів (рис. 4.3) з ідентифікатором нащадка в

допоміжній таблиці, а таблицю вимірювань з ідентифікатором батька, ми можемо обчислювати суму витрат для кожного місця виникнення витрат і всіх його складових одним запитом, як і в попередньому випадку. При цьому, додаючи обмеження на поля «*Distance*» і «*Is Leaf*», ми можемо легко рахувати витрати для будь-якого рівня в ієрархії.

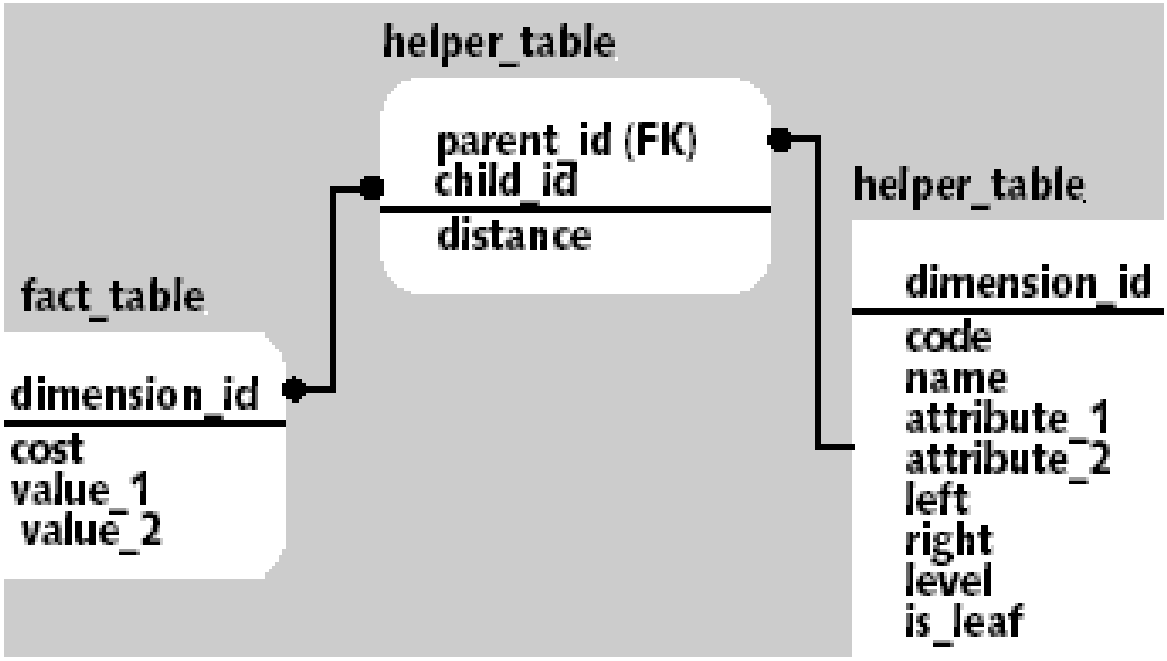

Рисунок 4.3 - Модель ієрархічного довідника з допоміжної таблицею

Наприклад, для того, щоб порахувати суму витрат, що виникають в місцях, що знаходяться за ієрархією на один рівень нижче «Інфраструктури», необхідно виконати наступний SQL-запит:

```
    select sum(fact_table.cost)
from fact_table, dimension_table, helper_table
where fact_table.dimension_id = helper_table.child_id
and dimension_table.dimension_id =
helper_table.parent_id
and dimension_table.name = «Інфраструктура»
and helper_table.distance = 1
```

Проблема проектування ієрархічних довідників ще більш ускладнюється коли вимір може мати кілька альтернативних ієрархій і стає зовсім важкою при необхідності підтримувати історію зміни таблиці вимірювання.

Важливий момент, з яким часто доводиться стикатися розробнику сховища даних, пов'язаний з агрегатними значеннями. Цей клас завдань умовно можна розділити на два підкласу. Перший розглядає завдання створення і підтримки агрегатів за наявними детальним даними і широко висвітлений в літературі. Другий пов'язаний з тим, що джерела даних для сховища надають собою не детальні значення, а вже деякий набір агрегованих даних. Така ситуація є типовою при створенні сховищ для керуючих компаній і державних контролюючих органів, які збирають безліч звітних форм.

Крайнім випадком такого підходу є модель, яку умовно можна назвати «показник-значення». Суть її полягає в тому, що збирається великий набір показників, що характеризують фінансово-господарську діяльність підприємства. Ці показники можуть бути як пов'язаними між собою функціонально, так і немає, можуть відображати одні і ті ж величини, але з різним ступенем деталізації і т.ін. При спробі уявити такі дані у вигляді багатовимірної моделі розробник стикається зі значними проблемами і дуже часто йде по шляху створіння не сховища даних, а сховища форм. Типове сховище форм будується на основі трьох вимірів - економічні показники, час, звітні форми; таблиці фактів - значення економічних показників і допоміжних таблиць, що описують, як показники і їх значення розташовані в звітних формах. При аналізі таких даних аналітик буде відчувати значні труднощі, пов'язані головним чином з тим, що показники різних форм можна порівнювати між собою. Єдине, що йому залишається - це відстеження змін показників однієї форми в часі.

Багатомірне моделювання і зіркоподібні схеми

Вище ми побіжно згадали про те, що сховища даних можна моделювати

методами багатовимірного моделювання з використанням зіркоподібній схеми. Повторимо, що традиційні методи моделювання даних і проектування баз даних, як правило, не дозволяють отримати структури даних, оптимізовані для масових запитів. Тому доведеться відкинути деякі традиційні ідеї про проектування і вивчити ряд нових прийомів. В цій лабораторній роботі частково зачіпаються питання аналізу, тому що це важливо для розуміння того, як визначаються ці моделі.

Як і більшість понять в комп'ютерному світі, зіркоподібна схема має кілька синонімів: *багатовимірна схема, куб даних і з'єднання за схемою "зірка"*. Причина, по якій цю схему називають зіркоподібній, полягає в тому, що її графічне представлення нагадує зірку.

Зіркоподібна модель складається з центральної таблиці фактів (*fact table*), яка оточена декількома таблицями вимірювань (*dimension table*). Фізично таблиця фактів часто є кілька секціонованих таблиць. Термін "секціонування" дуже широко використовується фахівцями з баз даних і в загальному випадку означає формування підмножин даних.

Відносини між таблицею фактів і таблицями вимірювань повинні бути простими, щоб існував тільки один можливий шлях з'єднання будь-яких двох таблиць і щоб сенс цього з'єднання був очевидний і добре зрозумілий. Перевага простоти відносин полягає в тому, що це допомагає підвищити продуктивність.

Все, що аналітик повинен зробити при багатовимірному моделюванні, - це виявити факти і їх вимірювання. Факти зазвичай являють собою основні види бізнес-діяльності організації і фактори, що впливають на даний бізнес або його сектор. Що можна сказати про таблиці вимірювань? У широкому сенсі слова - це елементи, які можуть чинити певний вплив або породжувати різні тенденції в розвитку фактів. Виходячи з цього, вимірювання можна розбити на наступні категорії: *люди, місця, речі і час* (рис. 4.4).

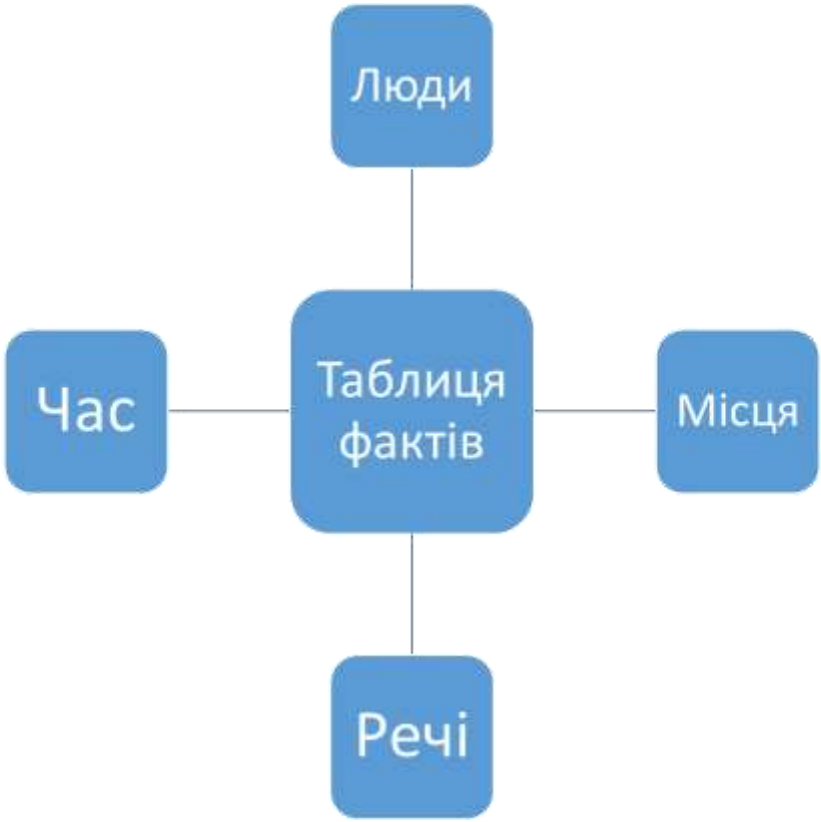

Рисунок 4.4 – Класифікація вимірювань: люди, місця, речі і час

Розглядайте таблицю фактів у світлі реляційної теорії: у неї є зовнішній ключ до кожної відповідної таблиці вимірювання. Вона об'єднує вимірювання в щось, що представляє собою значуща подія, і містить якісну і кількісну інформацію про цю подію. У сценарії продажів таблиця фактів може містити дані про те, скільки товару реалізовано і на яку суму, а також зовнішні ключі до таблиць вимірів, що характеризують операцію продажу (який товар, коли він був проданий, хто продав його і який спосіб платежу був використаний).

Часовий вимір в зіркоподібною схемою має низку цікавих властивостей. Факт, швидше за все, буде пов'язаний не з одним моментом часу, а з якимось часовим проміжком. Отже, часовий вимір зазвичай є узагальнюючим. Якщо ви вибрали одиницю зміни часу доводиться вибирати між зберіганням величезного числа фактів в сховище (які в більшості запитів необхідно буде узагальнювати) та відсутністю даних про сезонні тенденції внаслідок того, що при узагальненні обраний занадто великий період часу. Часовий вимір також часто виступає в якості основи для секціонування таблиці фактів. У багатьох

випадках секціонування необхідно для зменшення обсягу таблиці фактів і полегшення управління нею. Час же використовується тому, що при цьому, як правило, отримують приблизно рівні за обсягом кванти, і воно часто є найбільш логічним для секціонування виміром. Крім того, час - найбільш ймовірна основа для усічення таблиці фактів (якщо коли-небудь вдасться змусити підприємство погодитися на видалення даних з його сховища).

Вибір ступеня деталізації часового виміру (тиждень, день, місяць і т.ін.) може мати виняткову важливість. Якщо, наприклад, вибрати місяць, то засіб аналізу, можливо, зможе узагальнити дані в ще більшому ступені і показати результати по роках. Потім за допомогою різних методів деталізації можна уявити дані по кварталах потрібного року, а потім знову по місяцях. Очевидно, що в цьому випадку не можна виконати більш детальну тимчасову деталізацію, оскільки для цього немає даних. Вибір ступеня деталізації за часом має велике значення, тому що дозволяє врівноважити два фактора - необхідність в більш детальної розбивки і обсяг сховища даних. Таким чином, при проектуванні сховища даних необхідно приділяти належну увагу тимчасовим вимірам і в кожному конкретному випадку приймати рішення про необхідну для узагальнення ступеня деталізації.

Завдання на лабораторну роботу

На підставі БД, створеної в ході попередньої лабораторної роботи, спроектуйте структуру сховища даних. Для цього:

1. Виділіть вимірювання і факти сховища даних.
2. Визначте структури вимірювань і фактів сховища даних.
3. Перевірте таблиці на предмет множинності зв'язків між ними.
4. Перевірте, чи всі вимірювання впливають на таблицю фактів.

Лабораторна робота №5. Реалізація сховища даних під керуванням MICROSOFT SQL SERVER ANALYSIS SERVICES

Мета лабораторної роботи: Вивчення реалізації сховищ даних засобами MICROSOFT SQL SERVER ANALYSIS SERVICES.

Короткі теоретичні відомості

1. Вимірювання (Dimension)

У службах Microsoft SQL Server Analysis Services (SSAS) вимірювання є основними компонентами куба. У вимірах дані прив'язані до певної предметної області, наприклад замовники, магазини або службовці. У службах Analysis Services вимірювання містять атрибути, які відповідають стовпцям в таблицях вимірювання. Ці атрибути виглядають у вигляді ієрархій атрибутів і можуть бути організовані в багаторівневі ієрархічні структури. Ці ієрархії застосовуються для організації заходів, які містяться в кубі.

Всі вимірювання засновані на таблицях або уявленнях в поданні джерела даних. Вимірювання існують незалежно від куба, можуть використовуватися в декількох кубах і можуть використовуватися багаторазово в одному кубі. Вимірювання, що існує незалежно від куба, називається виміром бази даних, а вимір, що використовується в кубі, називається виміром куба.

Структура вимірювання в основному визначається структурою таблиці або таблиць базового вимірювання. Найпростіша структура називається схемою "зірка", в якій кожний вимір засноване на одній таблиці вимірювання, яка безпосередньо пов'язана з таблицею фактів зв'язком первинного та зовнішнього ключів.

Найпростішим методом визначення вимірювань бази даних і куба є використання майстра кубів для створення їх одночасно з визначенням куба. Майстер кубів створить вимірювання на основі таблиць вимірів, знайдених ним або вказаних користувачем з уявлення джерела даних, що

використовується для куба. Після цього майстер створює виміру бази даних і додає їх до нового кубу, створюючи вимірювання куба.

При створенні куба також можна додати будь-які вимірювання, які вже існують в базі даних. Ці вимірювання могли бути раніше визначені для іншого куба або майстром вимірювань. Після створення виміру бази даних його можна відредагувати в конструкторі вимірювань.

2. Міра (Mesuare)

Як правило, міра є стовпець, що містить кількісні, статистично обчислюються (зазвичай числові) дані, відповідні колонки в таблиці фактів. Вираз заходи також може використовуватися для визначення значення заходи на основі стовпця в таблиці фактів, зміненої багатовимірним виразом. Вираз заходи дозволяє виконати зважування значення заходи, наприклад, для перетворення валют при зважуванні заходи продажів за валютним курсом.

Для визначення заходів можна використовувати стовпці атрибутів таблиць вимірів, але такі заходи є, як правило, полуаддітівнимі або неаддитивними в залежності від режиму статистичного обчислення

Крім того, можна визначити обчислюється міру за допомогою багатовимірних виразів, щоб отримати обчислюється значення заходи на основі інших заходів в кубі. Такі заходи, звані обчислюються, додають кубу в службах Analysis Services гнучкі функціональні можливості аналізу. У кубі заходи згруповані по таблиці фактів в групу заходів. Групи заходів використовують для зв'язку вимірювань з заходами. Групи заходів також використовують для заходів з окремим лічильником як режим статистичного обчислення, що дозволяє виконувати оптимальне статистичне обчислення.

Перетин окремої заходи і окремого елемента всіх вимірювань, включених в групу заходів, представлено в службах Analysis Services у вигляді окремого осередку куба.

Заходи та групи заходів мають властивості, що дозволяють визначати та контролювати, як заходи і групи заходів функціонують і відображаються для користувачів.

Властивості групи заходів

Властивості групи заходів визначають характеристики всієї групи заходів і встановлюють характеристики за замовчуванням для певних властивостей заходів в групі заходів.

Властивості заходів

Заходи успадковують певні властивості у групи заходів, елементами яких вони є, якщо тільки ці властивості не перевизначені на рівні заходів. Властивості заходів визначають статистичне обчислення заходів, їх тип даних, папку для відображення заходів, рядок форматування, будь-які вирази заходів, їх зрозумілі імена, базові вихідні стовпчики і видимість для користувачів

Куб (cube)

У службах Microsoft SQL Server Analysis Services куб розробляється на основі таблиць і уявлень, що моделюються в поданні джерела даних. Куб являє собою набір заходів, які є фактами, і вимірювань, які є важливими областями, такими як, наприклад час, продукт і замовник. Куб доповнюється обчисленнями, ключовими індикаторами продуктивності, діями, секціями, перспективами і перетвореннями. Куб, по суті, є синонімом уніфікованої багатовимірної моделі. Заходи куба засновані на шпальтах з однієї або декількох таблиць фактів, а елементи вимірювань куба засновані на шпальтах з однієї або декількох таблиць вимірів. Куб також може бути розроблений без базового реляційного джерела даних. У цьому випадку може бути сформована базова реляційна структура для підтримки куба. Факти в кубі статистично обчислюються на основі ієрархій вимірювань.

Властивості кубів

Куби мають ряд властивостей, змінюючи параметри яких можна впливати на поведінку всього куба, і кілька незмінних властивостей.

Реалізація

Analysis Manager – программа, работающая под управлением Microsoft® Management Console (MMC). Она представляет собой утилиту, входящую в состав Analyses Services и предназначенную главным образом для администраторов баз данных OLAP.

Як запускати Analysis Manager

Клацнути кнопку **Start**, пункт **Програми->Microsoft SQL Server->Analyses Services** і потім клацніть **Analysis Manager**.

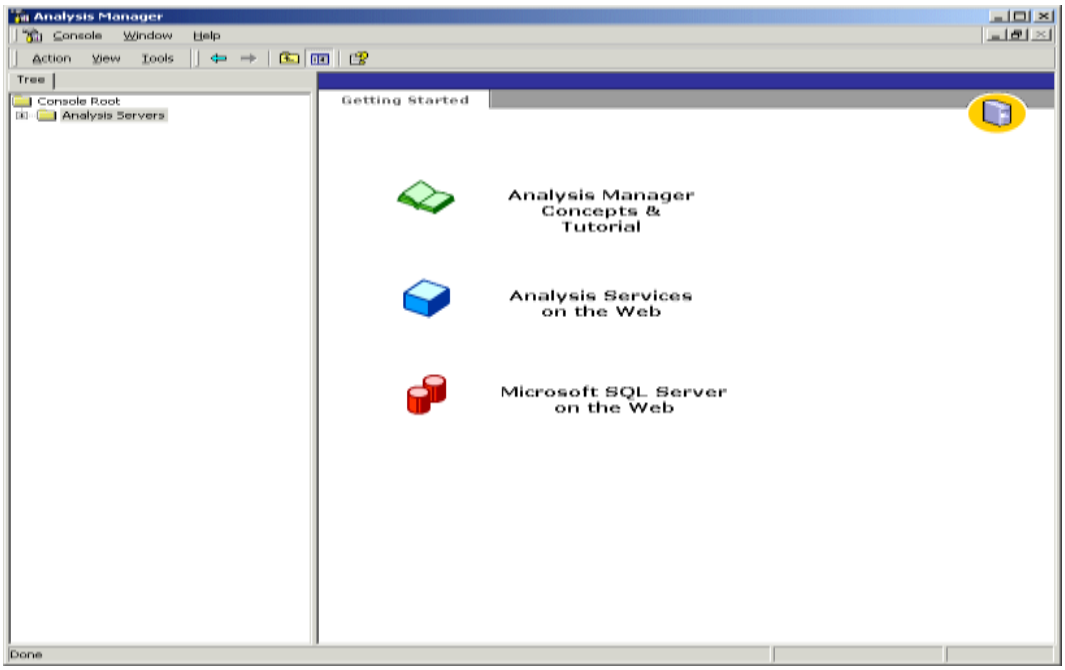

Рисунок 5.1 – Вікно програми Analysis Manager

Як налаштовувати структуру бази даних

1. У **Analysis Manager** в дереві розкрийте Сервери Аналізу.

2. Клацнути назву вашого сервера. Підключення з сервером аналізу буде встановлено.

3. Клацнути правою кнопкою миші на назві вашого сервера, і потім клацніть **New Database**.

4. У діалоговому вікні **Database**, в блоці імені Бази даних, вводите **Tutorial**, і натисніть кнопку **OK**.

5. В області вікна дерева **Analysis Manager**, розкрийте сервер, і потім розширите базу даних **Tutorial**, яку Ви тільки що створювали.

Ваша нова база даних **Tutorial** містить наступні елементи:

• Джерела Даних

• Куби

• Загальнодоступні Вимірювання

• Видобувні Моделі

• Ролі Бази даних

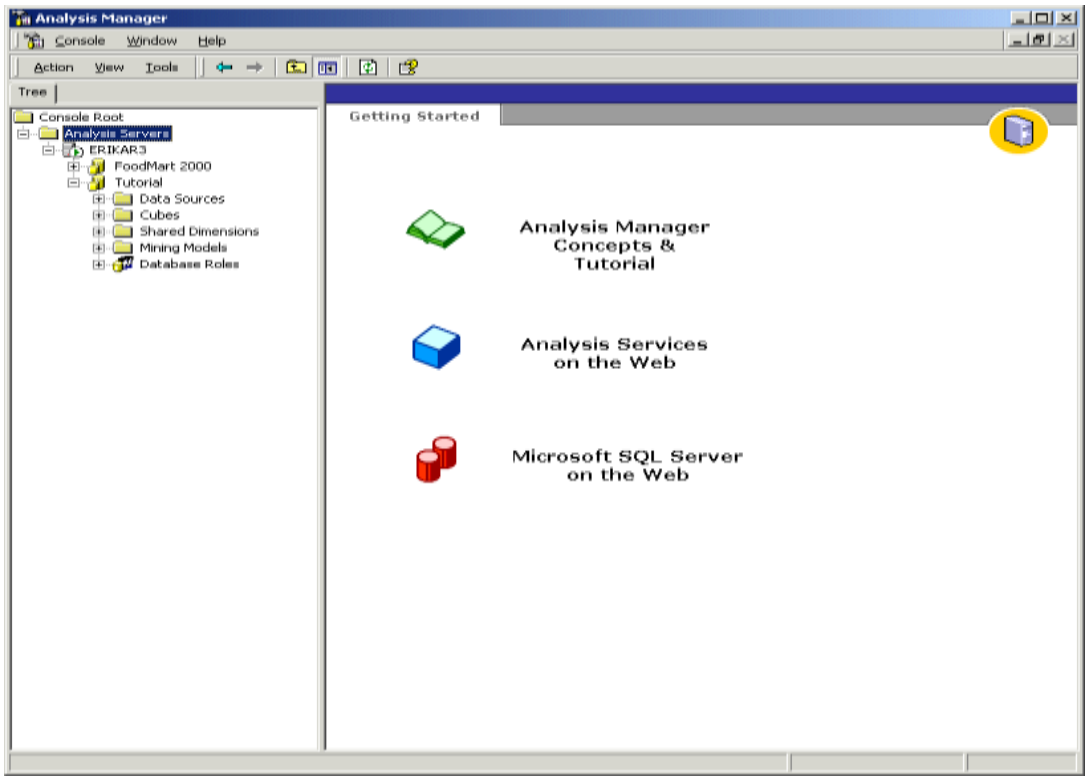

Рисунок 5.2 – База даних Tutorial

Потім встановіть підключення до типових даних в джерелі даних **Tutorial**. Цей приклад буде використовуватися для всіх вправ в цій навчальній програмі.

Установка джерела даних в Менеджері Аналізу підключає вашу базу даних до системного імені джерела даних (DSN), яке встановлюється в ODBC Адміністратора Джерела Даних.

Як налаштовувати джерело даних

1. У **Analysis Manager** в області дерева клацніть правою кнопкою миші на папці Джерел Даних під базою даних **Tutorial**, і потім клацніть **New Data Source**.

2. У діалоговому вікні **Data Link Properties**, клацніть вкладку **Provider**, і потім клацніть **Microsoft OLE DB Provider for ODBC Drivers**.

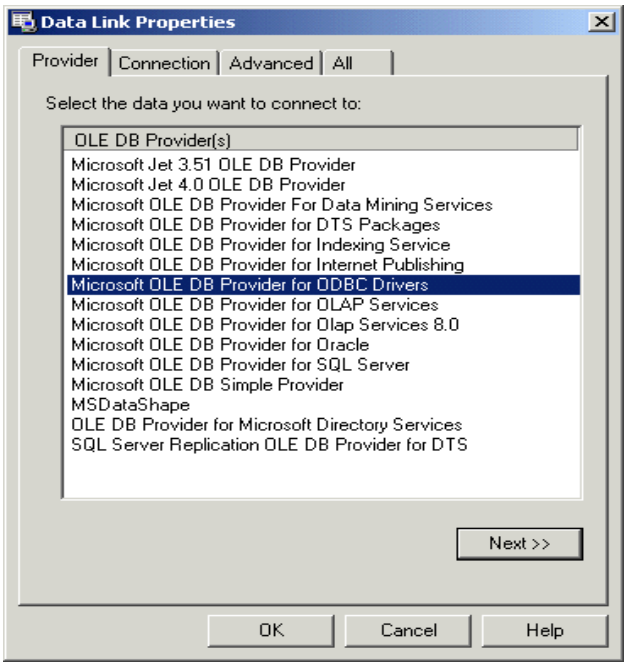

Рисунок 5.3 – Обрання провайдеру

3. Клацнути вкладку **Connection**, і потім в списку імен джерел даних клацніть **Tutorial** (рис.5.3)

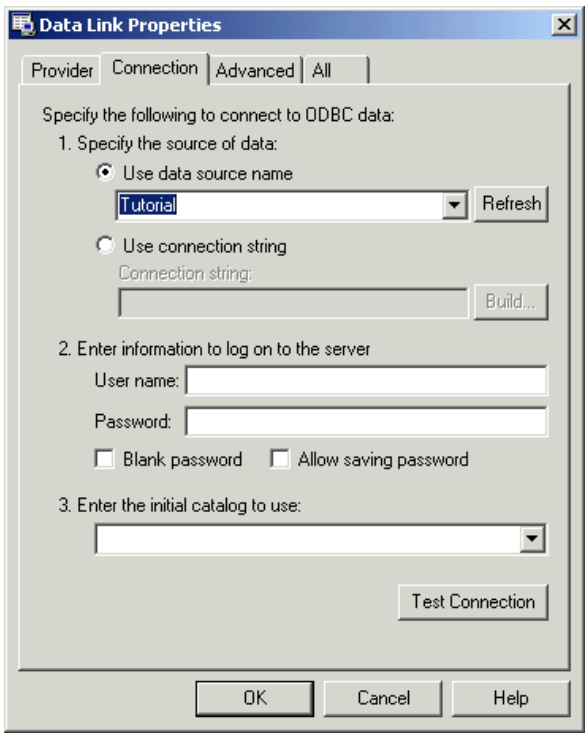

Рисунок 5.3 – Обрання підключення

4. Клацнути **Test Connection**, щоб переконатися, що все працює.

5. Клацнути **OK**, щоб закрити діалогове вікно **Data Link Properties.**

Побудова куба

Куб - багатовимірна структура даних. Куби визначені набором вимірювань і заходів.

Як відкрити Майстер Куба

У Менеджері Аналізу в області вікна дерева, під базою даних **Tutorial**, клацнете правою кнопкою миші на папці кубів, виберіть новий куб і потім клацнете **Wizard**.

Як додавати виміри до кубу

Виміри - кількісні значення в базі даних, яку Ви хочете аналізувати. Зазвичай-використовувані виміри комерційні, вартість, і дані бюджету. Виміри проаналізовані проти різних категорій виміру куба.

1. На етапі **Welcome** Майстри Куба, клацніть **Next**.

2. У виборі таблиці факту на кроці джерела даних, розкрийте джерело

даних **Tutorial**, і потім клацніть *sales_fact_1998*.

3. Ви можете розглядати дані в таблиці *sales_fact_1998*, клацаючи дані **Browse**.

4. Щоб визначити заходи для вашого куба, під таблицею Факту, двічі клацнете на стовпці *store_sales*. Повторіть цю процедуру для стовпців *store_cost і unit_sales,* а потім натисніть **Next**.

Як побудувати вимір часу

1. На етапі **Select the dimensions for your cube** майстра **Cube**, клацніть **New Dimension**, щоб викликати Майстер Вимірювання (рис.5.4).

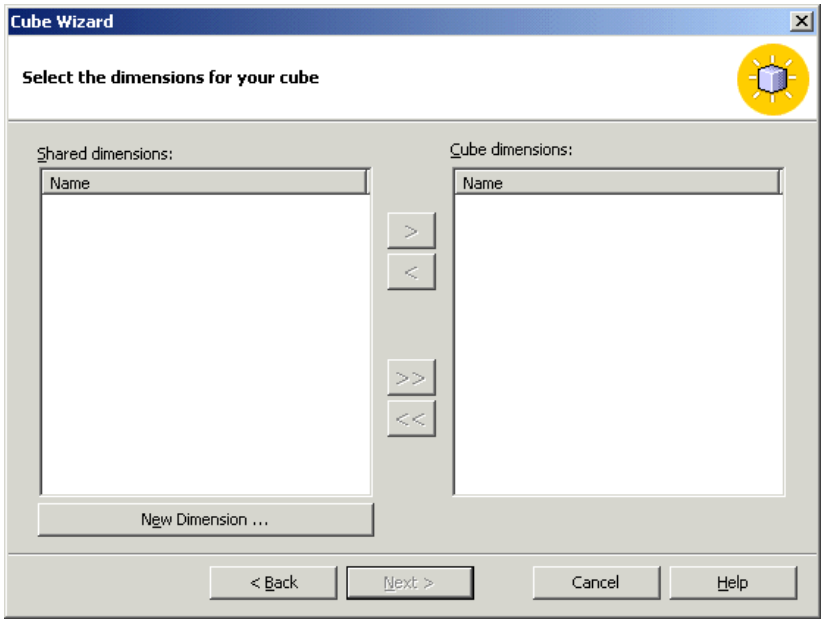

Рисунок 5.4 – Майстер побудови куба

2. На кроці **Welcome** клацніть **Next**.

3. Виберіть **Star Schema** для одиночної таблиці вимірювання і потім клацнете **Next**.

4. На кроці вибору таблиці вимірювання, клацніть *time_by_day*. Ви можете переглядати дані, що містяться *в time_by_day* таблиці, клацаючи **Browse Data**.

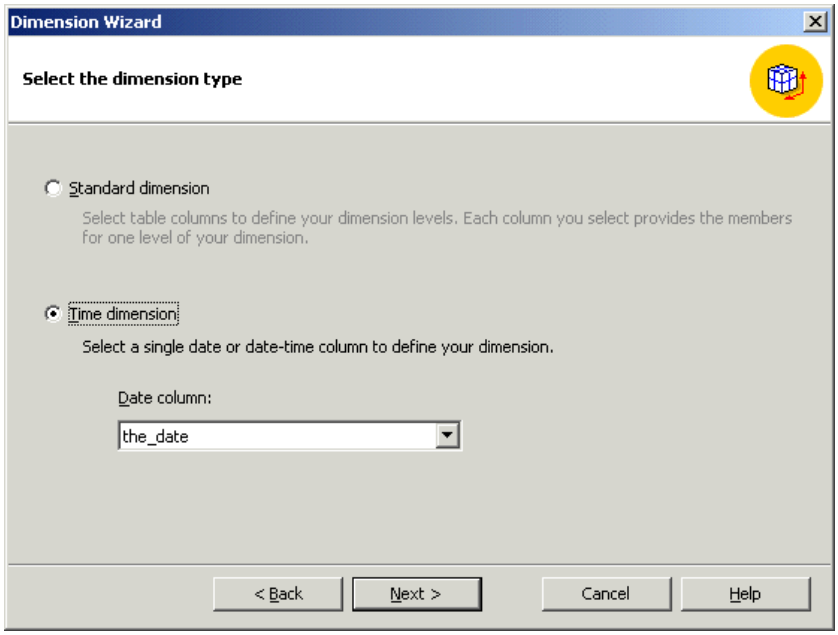

Рисунок 5.4 – Вікно вибору типу вимірювання

5. На етапі вибору типу вимірювання, виберіть **Time Dimension** (рис.5.4), а потім натисніть **Next**.

6. Потім визначте рівні для вашого вимірювання. На наступному кроці клацніть **Select time level**, виберіть *Year*, *Quarter*, *Month*, а потім натисніть **Next**.

7. На кроці додаткових параметрів вибору натисніть **Next**.

8. На останньому кроці майстра введіть ім'я вашого нового виміру.

Зверніть увагу: Ви можете визначити, чи буде цей вимір розділений або можливо тільки приватного використання ресурсів цього виміру, за допомогою перемикача, який розташований в нижньому лівому кутку екрана.

9. Клацніть **Finish**, щоб повернутися до Майстра Куба.

10. У Майстрі Куба тепер відбивається вимір часу в списку вимірювань Куба.

Як побудувати вимір product

1. Клацнути **New dimension** знову. На кроці **Welcome** Майстри Вимірювання клацніть **Next**.

2. Виберіть **Snowflake Schema** і потім клікнути **Next**.

3. На кроці вибору таблиць вимірювання двічі клацніть *product і product_class*, щоб додати їх до обраних таблиць. Клацніть **Next**.

4. Ці дві таблиці, які ви вибрали на попередньому і поточному кроці з'єднуються між собою (рис.5.5). Клацніть **Next**.

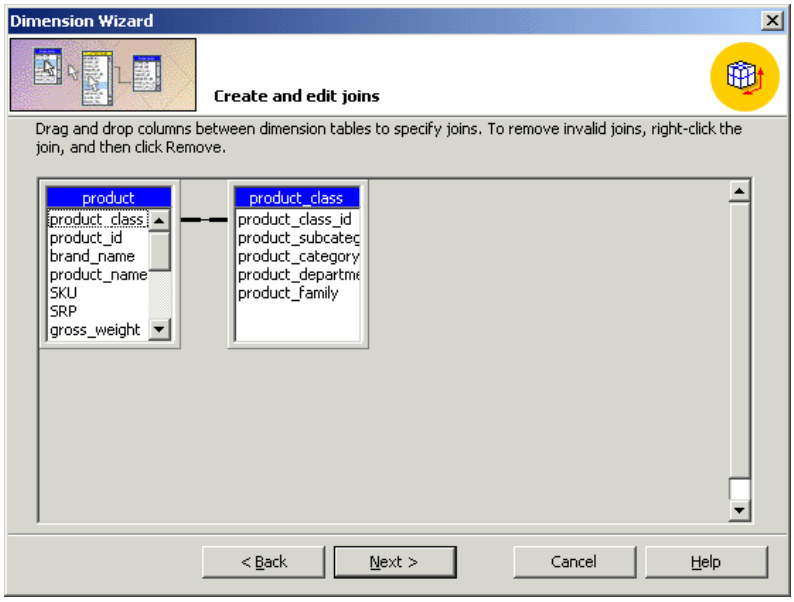

Рисунок 5.5 – Вікно створення та редагування з`єднання

5. Щоб визначити рівні для вимірювання, під стовпцями **Available**, двічі клацнете стовпці: *product_category, product_subcategory, і brand_name*. Після того, як ви двічі клацаєте кожен стовпець, його ім'я з'являється під рівнями вимірювань. Клацніть **Next** після того, як вибрали всі три стовпці.

6. На кроці визначення ключів стовпців клацніть **Next**.

7. На кроці додаткових параметрів вибору клацніть **Next**.

8. На останньому кроці майстра введіть *product* в блок назви вимірювання. Клацніть **Finish**.

9. Ви повинні побачити вимір *product* в списку вимірювань куба.

Як побудувати вимір Customer

1. Клацнути **New dimension**.

2. На кроці **Welcome** Майстри Вимірювання клацніть **Next**.

3. Виберіть **Star Schema** а потім натисніть **Next**.

4. На кроці вибору таблиці вимірювання, клацніть *Customer*, а потім натисніть **Next**.

5. На етапі вибору типу вимірювання клацніть **Next**.

6. Щоб визначити рівні для вашого вимірювання, під стовпцями **Available** двічі клацніть стовпці: *Country, State_Province, City, і lname*. Після того, як ви двічі клацаєте кожен стовпець, його ім'я з'являється під рівнями вимірювань. Після того, як ви вибрали всі чотири стовпці, а потім натисніть **Next**.

7. На кроці визначення ключів стовпців клацніть **Next**.

8. На кроці додаткових параметрів вибору клацніть **Next**.

9. На останньому кроці майстра введіть *Customer* в блок назви вимірювання. Клацніть **Finish**.

10. Ви повинні побачити вимір *Customer* в списку вимірювань куба.

Як побудувати вимір store

1. Клацнути **New dimension**.

2. На кроці **Welcome** Майстра Вимірювання клацніть **Next**.

3. Виберіть **Star Schema** а потім натисніть **Next**.

4. На кроці вибору таблиці вимірювання, клацніть *store*, а потім натисніть **Next**.

5. На етапі вибору типу вимірювання клацніть **Next**.

6. Щоб визначити рівні для вашого вимірювання, під стовпцями **Available** двічі клацніть стовпці: *store_country, store_state, store_city,* і *store_name*. Після того, як ви двічі клацаєте кожен стовпець, його ім'я з'являється під рівнями вимірювань. Після того, як ви вибрали всі чотири стовпці, а потім натисніть **Next**.

7. На кроці визначення ключів стовпців клацніть **Next**.

8. На кроці додаткових параметрів вибору клацніть **Next**.

9. На останньому кроці майстра введіть *store* в блок назви вимірювання.

Клацніть **Finish**.

10. Ви повинні побачити вимір *store* в списку вимірювань куба.

Як закінчити побудову куба

1. У майстра куба клацніть **Next**.

2. Клацнути **Yes** у вікні повідомлення (рис.5.6).

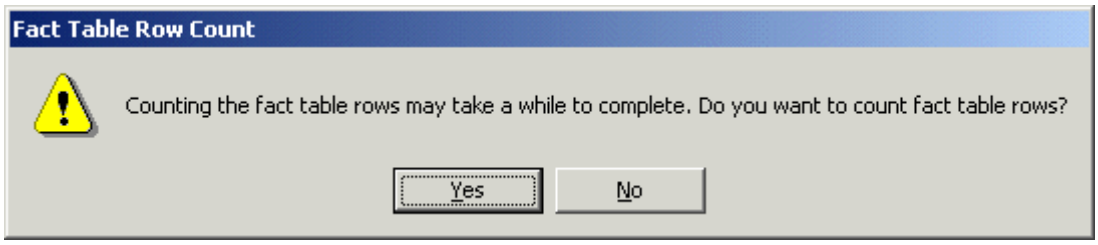

Рисунок 5.6 – Вікно Fact Table Row Count

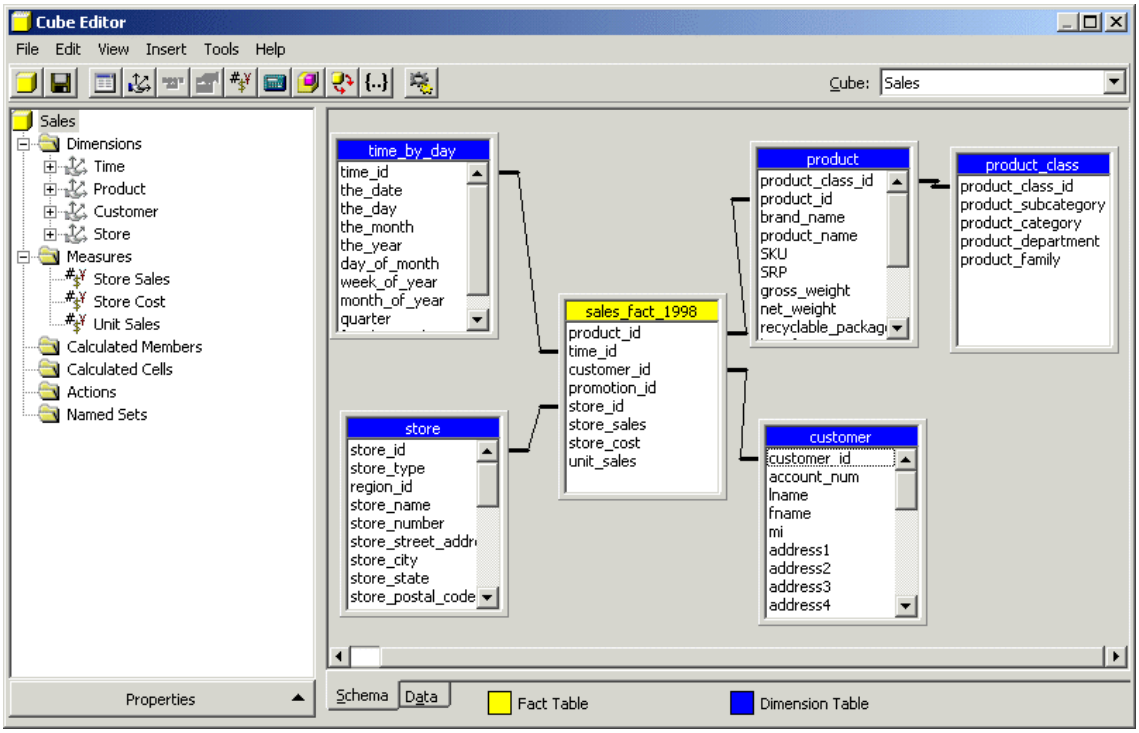

Рисунок 5.7 – Вікно редагування кубу

3. На останньому кроці майстра куба назвіть ваш куб **Sales** і потім клацніть **Finish**.

4. Майстер закривається і потім запускається **Редактор Куба** (рис. 5.7), який містить новий куб. Натискаючи по синім або жовтим областям заголовка,

Упорядкуйте таблиці так, щоб вони відповідали наступній ілюстрації (рис. 5.7). Зверніть увагу: ви не повинні закривати **Редактор Куба**, тому що будете редагувати куб в наступному розділі навчальної програми.

Редагування куба

Ви можете використовувати два методи для доступу до редактора куба:

• в менеджері аналізу в області дерева клацніть правою кнопкою миші на існуючому кубі і потім клацніть **Edit**;

• можна створити новий куб, використовуючи редактора куба безпосередньо. Цей метод не рекомендується, якщо ви не досвідчений користувач.

Якщо ви продовжуєте від попереднього розділу, ви повинні вже бути в редакторі куба.

В області вікна схеми редактора куба, ви можете бачити таблицю фактів (з жовтої областю заголовка) і з'єднаними таблицями вимірювань (сині області заголовка). В області вікна дерева редактора куба можливий попередній перегляд структури куба в ієрархічному дереві. Ви можете редагувати властивості куба, клацаючи кнопку **Properties** внизу лівої області вікна.

Зберігання проекту і обробка куба

Ви можете проектувати опції зберігання для даних у вашому кубі. Перш, ніж ви можете використовувати або переглядати дані в ваших кубах, ви повинні обробити їх.

Як проектувати зберігання, використовуючи майстер проекту зберігання

1. В область дерева менеджера аналізу розкрийте папку кубів, клацніть правою кнопкою миші на **Commerse Cube** і потім клацніть **Design Storage**.

2. На кроці **Welcome** клацніть **Next**.

3. Виберіть **MOLAP** в якості вашого типу зберігання даних і потім клацніть **Next**.

4. Під опціями набору натисніть **Performance gain reaches**. У блоці введіть **40%**. Це налаштовує **Analyses Services** на підвищення продуктивності

40%, незалежно від того, скільки дискового простору це вимагає. Адміністратори можуть використовувати цю здатність настройки для збалансування потреба в продуктивності запитів і дискового простору, необхідного, щоб зберігати дані агрегатів.

5. Клацніть **Start**.

6. Ви можете спостерігати виконання в правій частині майстра (рис.5.8). Після закінчення роботи натисніть **Next**.

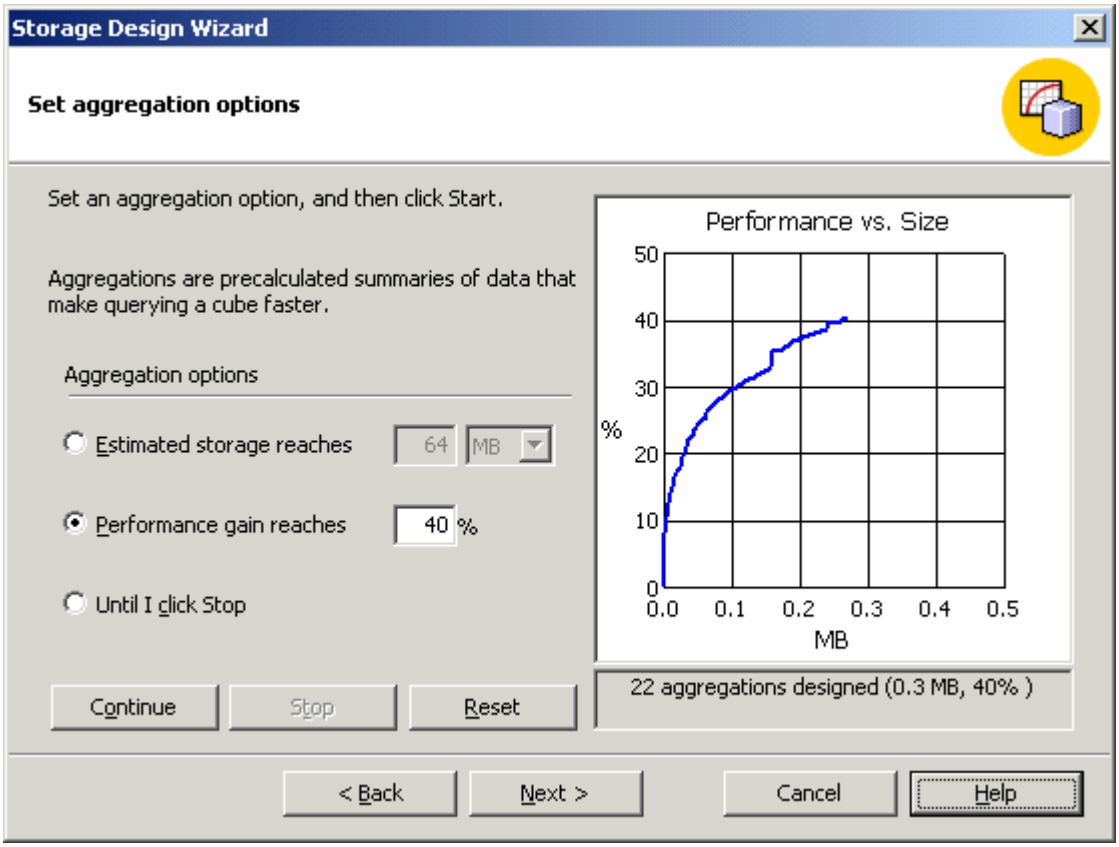

Рисунок 5.8 – Вікно майстра проекту зберігання

7. На наступному кроці виберіть **Process** і потім клацніть **Finish**.

Зверніть увагу: обробка агрегатів може займати деякий час.

8. У вікні ви можете спостерігати ваш куб, в той час як відбувається обробка. Коли обробка закінчена, видається повідомлення, підтверджуючи, що обробка була закінчена успішно.

9. Необхідно клацнути **Close** для повернення до менеджера аналізу. Тепер ви готові переглянути дані в кубі.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Базова

1.Watson, R. T. (2016). Data management, databases and organizations (6. ed.). John Wiley & Sons.

2. Elmasri, R., & Navathe, S.B. (2016). Fundamentals of Database Systems, 7th edition. New York: Pearson Education.

3. Provost, F. & Fawcett, T. (2013). Data Science for Business. Sebastopol: O'Reilly Media

4. Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. Big Data & Society, 1(1), 2053951714528481.

5. Bob Bryla, Kevin Loney (2013) Oracle Database 12c The Complete Reference. Oracle Press

6. Carlos Coronel, Steven Morris (2014) Database Systems: Design, Implementation, & Management

7. David M. Kroenke, David J. Auer, Scott L. Vandenberg, Robert C. Yoder (2017) Database Concepts (8th Edition) 8th Edition

8. GalitShmueli, Peter C. Bruce, InbalYahav, Nitin R. Patel, Kenneth C. Lichtendahl (2017) Data Mining for Business Analytics: Concepts, Techniques, and Applications in R. Wiley

9. Ralph Kimball, Margy Ross (2013) The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling

10. Lawrence Corr and Jim Stagnitto (2011) Agile Data Warehouse Design: Collaborative Dimensional Modeling, from Whiteboard to star schema

11 Matteo Golfarelli and Stefano Rizzi (2009) Data Warehouse design: Modern Principles and Methodology

12 Erik Thomsen (2002) OLAP Solutions: Building Multidimensional Information Systems

Допоміжна література

13. Claudia Imhoff, Nicholas Galemmo, Jonathan G. Geiger: Mastering Data Warehouse Design Relational and Dimensional Techniques, Wiley Publishing, 2003.

14. M. Tamer Özsu, Patrick Valduriez Principles of Distributed Database Systems 3rd ed. 2011 Edition

15. W. H. Inmon Building the Data Warehouse 4th Edition

16. Maarten van Steen, Andrew S. Tanenbaum Distributed Systems 3.01 Edition

ІНФОРМАЦІЙНІ РЕСУРСИ В ІНТЕРНЕТІ

1. Дистанційний курс «Бази даних та сховища даних» [Електронний ресурс]. – Режим доступу: https://www.moodle.hneu.edu.ua/course/view.php?id=28.

Електронне Навчальне видання

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

з дисципліни

«БАЗИ ДАНИХ ТА СХОВИЩА ДАНИХ»

для студентів денної форми навчання
напряму 126 «Інформаційні системи та технології»

Упорядники Сокол В.Є., Шматко О.В., Фонта Н.Г., Іващенко О.В.

Відповідальний випусковий М.Д.Годлевський

Авторська редакція