

Інструктивно-методичні
матеріали для виконання
самостійних робіт
з дисципліни
«Бази даних»
ДЛЯ ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДІВ 1 ТА 2 РІВНЯ
АКРЕДИТАЦІЇ
ЗІ СПЕЦІАЛЬНОСТІ 121
«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Розробила викладач _____ С.С. Ланська

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова ПЦК ІІ: _____ С.С. Ланська

Дніпро

2017

План самостійного вивчення теми № 1

з навчальної дисципліни «Бази даних»

Тема Приклад проектування реляційної бази даних

Мета На практичному прикладі пройти всі етапи проектування реляційної

бази даних

знати:

- вміст понять:
 - нормалізація відносин;
 - функціональні залежності;
 - декомпозиція;
 - перша нормальна форма;
 - друга нормальна форма;
 - третя нормальна форма;
 - четверта нормальна форма;
 - нормальна форма Бойса - Кодда;
 - п'ята нормальна форма.

вміти:

- виділяти об'єкти предметної області та їх атрибути;
- будувати концептуальну модель предметної області;
- проводити логічне проектування;
- будувати ER – діаграму проектованої бази даних.

Час – 4 години

Навчальні питання

1 Приклад проектування реляційної бази даних

1.1 Постановка задачі Виділення об'єктів предметної області.

1.2 Концептуальна схема предметної області

2 Нормалізація таблиць бази даних.

3 Опис структури таблиць

4 ER – діаграма бази даних Organization

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше Таблиця перебуває в другій нормальній формі, якщо вона перебуває в першій нормальній формі й всі атрибути, що не є частиною первинного ключа, повністю функціонально залежні від первинного ключа.

По-друге Для того, щоб таблиця перебувала в третій нормальній формі необхідно, щоб вона перебувала в другій нормальній формі й у ній повинні бути виключені всі транзитивні залежності

По-третє

Література

1 Малыхина М.П., Базыданных: основы, проектирование, использование [Текст]: / Малыхина М.П. – СПб.: «БХВ», 2004, –512с.

2 Дейт, К.Дж., Введение в системы баз данных [Текст]: / Дейт К.Дж., Пер. с англ., 7-е изд. – М.; СПб.; Киев: Вильямс, 2001. 1071с.

3 Карпова, Т.С., Базыданных: модели, разработки, реализации [Текст]: / Карпова Т.С. – СПб.; Питер, 2001. 303с.

Навчальні матеріали до самостійного вивчення теми

1 Приклад проектування реляційної бази даних

1.1 Постановка задачі Виділення об'єктів предметної області.

Розглянемо предметну область, пов'язану з роботою організації по розробці програмного забезпечення. У рамках цієї предметної області й залежно від передбачуваних запитів нам буде необхідна наступна інформація:

1 *Співробітники організації*: прізвище співробітника; посада; кваліфікація; відділ, у якому працює співробітник (номер відділу й найменування відділу).
Ураховувати:

Кваліфікація співробітника являє собою список середовищ і додатків, по яких співробітник має кваліфікаційний сертифікат.

В одному відділі працює кілька співробітників.

Один співробітник організації може працювати тільки в одному відділі.

2 *Клієнти організації*: прізвище; адреса; телефон.

3 *Проекти*: тема проекту, керівник проекту, клієнт, для якого розробляється проект; дата укладання договору; строк виконання проекту; вартість проекту;

співробітники, зайняті в реалізації проекту із вказівкою частки співробітника в загальному гонорарі за проект.

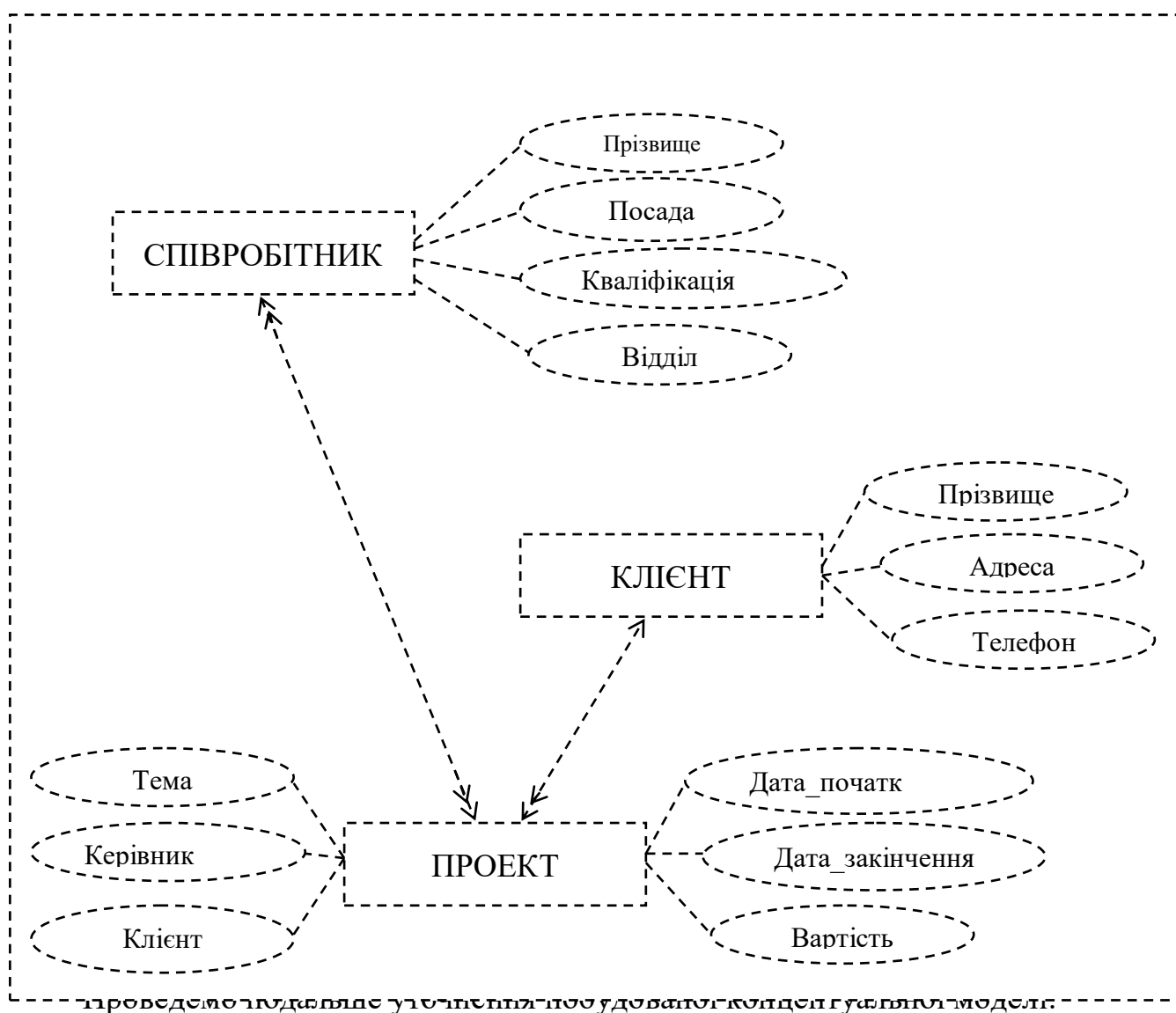
Ураховувати:

Один співробітник може бути керівником декількох проектів, але кожен проект має одного керівника.

Один співробітник може брати участь у розробці декількох проектів й в одному проекті бере участь кілька співробітників.

1.2 Концептуальна схема предметної області

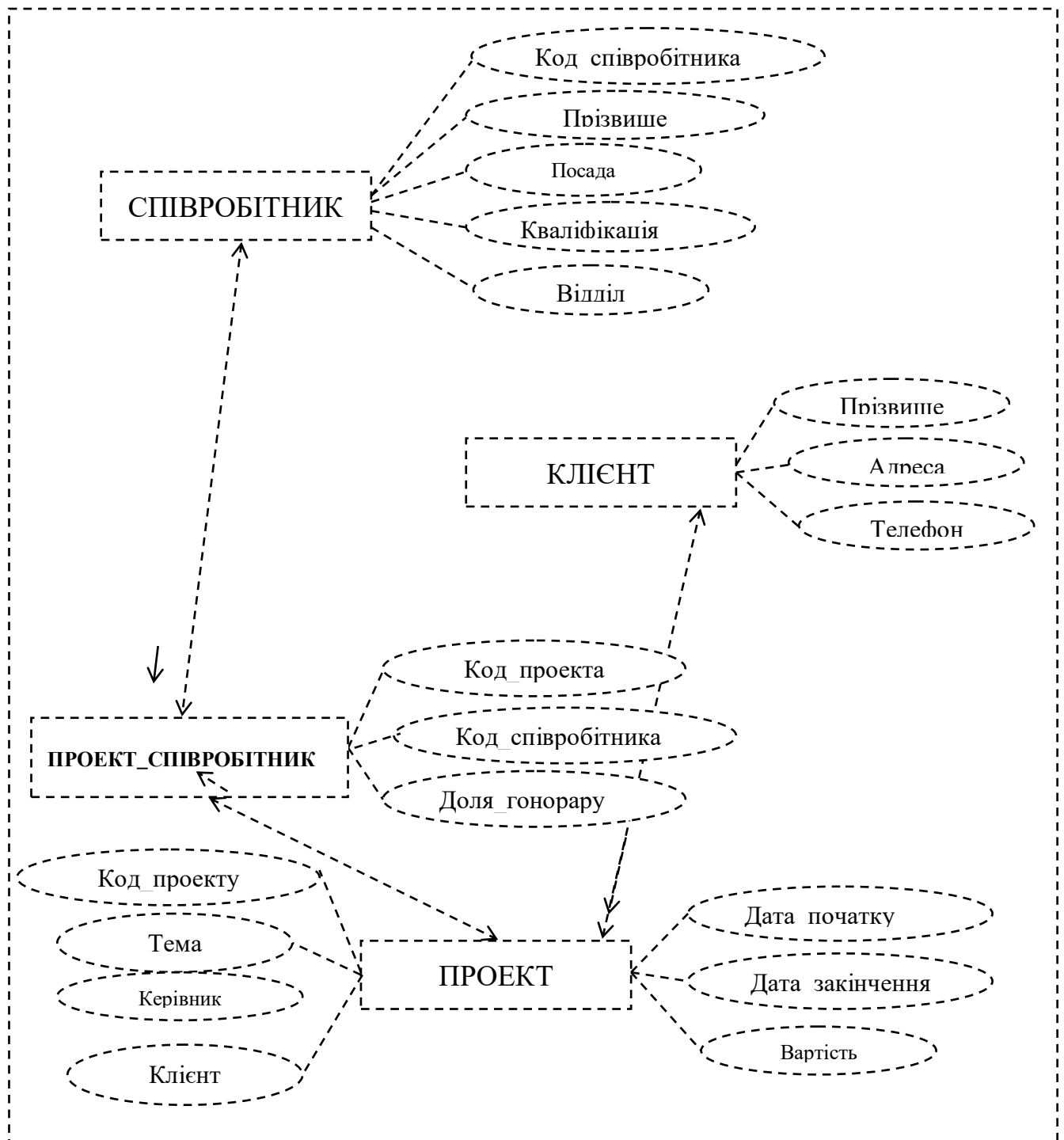
На основі аналізу предметної області будуюмо концептуальну схему розроблюваної інформаційної системи (див. рис. 1.1):



Для однозначної ідентифікації співробітника, клієнта й проекту й для подальшої реалізації зв'язків в об'єкти СПІВРОБІТНИК, КЛІЄНТ і ПРОЕКТ уведемо атрибути Код_співробітника, Код_клієнта, і Код_проекту відповідно.

Розірвемо зв'язок "багато-до-багатьох" між об'єктами СПІВРОБІТНИК і ПРОЕКТ. Для цього введемо проміжний об'єкт ПРО-ЕКТ_СПІВРОБІТНИК з атрибутами Код_проекту, Код_співробітника й До-ля_гонорару.

Остаточна концептуальна схема прийме вид (див. рис. 1.2):



2 Нормалізація таблиць бази даних.

На основі концептуальної схеми створимо таблиці бази даних і проведемо їхню нормалізацію до третьої нормальної форми (3НФ).

Інформацію про співробітників організації помістимо в таблицю **Empl**(Співробітник), що має наступну структуру:

Empl:

e_id	e_name	job	skill	department	
				d_id	d_name
111	Drovko	Administrator DB	MySQL, Linux, Java, UML	12	Projecting department
112	Ivanov	System admin	Windows, Linux, Java, NT	11	Administrative department
113	Konov	Program's	Linux, Java, NT, C++, Peri	14	Testing department
114	Kornienko	Program's	NT, C++, Peri	14	Testing department
121	Koval	Program's	Java, NT, C++, Peri	13	Realization department
122	Palkin	Program's	C++, MySQL, UML	14	Testing department
123	Petrov	Program's	Linux, Java, NT, C++, Peri	13	Realization department
131	Sidorov	Administrator DB	MySQL, Linux, Java, NT	12	Projecting department
132	Turov	System admin	Linux, Java, NT, UML	11	Administrative department
133	Vertko	Program's	Java, C++, Peri, UML	13	Realization department

У цій таблиці:

e_id- кодовий номер співробітника;

e_name- прізвище співробітника;

job- посада;

skill- кваліфікація;

department - відділ (номер відділу - **d_id** і найменування відділу - **d_name**)

Перша нормальна форма (1НФ)

Дана таблиця не відповідає навіть вимогам першої нормальної форми (1НФ).

Значення атрибутів **skill** й **department** не є атомарними. Одне значення атрибута **skill** являє собою список, а одне значення атрибута **department** складається із двох різнотипних величин. Для задоволення вимоги атомарності атрибут **department** замінимо на два окремих атрибути **d_id** й **d_name**, а для атрибута **skill** прийде виділити по одному рядку на кожен елемент кваліфікації співробітника. у результаті одержимо таблицю:

Empl:

e_id	e_name	job	skill	d_id	d_name
111	Drovko	Administrator DB	MySQL	12	Projecting department
111	Drovko	Administrator DB	Linux	12	Projecting department
111	Drovko	Administrator DB	Java	12	Projecting department
111	Drovko	Administrator DB	UML	12	Projecting department
112	Ivanov	System admin	Windows	11	Administrative department
112	Ivanov	System admin	Linux	11	Administrative department

112	Ivanov	System admin	Java	11	Administrative department
112	Ivanov	System admin	NT	11	Administrative department
113	Konov	Program's	Linux	14	Testing department
113	Konov	Program's	Java	14	Testing department
113	Konov	Program's	NT	14	Testing department
113	Konov	Program's	C++	14	Testing department
113	Konov	Program's	Peri	14	Testing department
114	Kornienko	Program's	NT	14	Testing department
114	Kornienko	Program's	C++	14	Testing department
114	Kornienko	Program's	Peri	14	Testing department
121	Koval	Program's	Java	13	Realization department
121	Koval	Program's	NT	13	Realization department
121	Koval	Program's	C++	13	Realization department
121	Koval	Program's	Peri	13	Realization department
122	Palkin	Program's	C++	14	Testing department
122	Palkin	Program's	MySQL	14	Testing department
122	Palkin	Program's	UML	14	Testing department
123	Petrov	Program's	Linux	13	Realization department
123	Petrov	Program's	Java	13	Realization department
123	Petrov	Program's	NT	13	Realization department
123	Petrov	Program's	C++	13	Realization department
123	Petrov	Program's	Peri	13	Realization department
131	Sidorov	Administrator DB	MySQL	12	Projecting department
131	Sidorov	Administrator DB	Linux,	12	Projecting department
131	Sidorov	Administrator DB	Java	12	Projecting department
131	Sidorov	Administrator DB	NT	12	Projecting department
132	Turov	System admin	Linux	11	Administrative department
132	Turov	System admin	Java	11	Administrative department
132	Turov	System admin	NT	11	Administrative department
132	Turov	System admin	UML	11	Administrative department
133	Vertko	Program's	Java	13	Realization department
133	Vertko	Program's	C++	13	Realization department
133	Vertko	Program's	Peri	13	Realization department
133	Vertko	Program's	UML	13	Realization department

Ясно, що таке рішення далеко від ідеального, оскільки воно породжує очевидну надмірність даних: для кожної комбінації співробітник - кваліфікація нам доводиться зберігати всі характеристики працівника.

Друга нормальна форма (2НФ)

Таблиця перебуває в другій нормальній формі, якщо вона перебуває в першій нормальній формі й всі атрибути, що не є частиною первинного ключа, повністю функціонально залежні від первинного ключа.

У перетвореній таблиці **Empl** первинним ключем буде комбінація стовпців **e_id** й **skill**.

Розглянемо можливі функціональні залежності в нашій таблиці:

e_id, skill → e_name, job, d_id, d_name

e_id → e_name, job, d_id, d_name

Інакше кажучи, ми можемо визначити ім'я, посаду й відділ співробітника використовуючи тільки частину первинного ключа - стовпець **e_id**, а значить таблиця не перебуває в другій нормальній формі. Для приведення таблиці до 2НФ розіб'ємо її на такі таблиці, у яких всі не ключові атрибути будуть повністю функціонально залежати від ключа.

Це таблиці **Empl** й **Skills**:

Empl:

e_id	e_name	job	d_id	d_name
111	Drovko	Administrator DB	12	Projecting department
112	Ivanov	System admin	11	Administrative department
113	Konov	Program's	14	Testing department
114	Kornienko	Program's	14	Testing department
121	Koval	Program's	13	Realization department
122	Palkin	Program's	14	Testing department
123	Petrov	Program's	13	Realization department
131	Sidorov	Administrator DB	12	Projecting department
132	Turov	System admin	11	Administrative department
133	Vertko	Program's	13	Realization department

Skills:

e_id	skill
111	MySQL
111	Linux
111	Java
111	UML
112	Windows
112	Linux
112	Java
112	NT
113	Linux
113	Java
113	NT
113	C++
113	Peri
114	NT
114	C++
114	Peri
121	Java
121	NT
121	C++
121	Peri
122	C++
122	MySQL
122	UML
123	Linux
123	Java
123	NT
123	C++
123	Peri
131	MySQL
131	Linux,
131	Java
131	NT
132	Linux
132	Java
132	NT
132	UML
133	Java
133	C++
133	Peri
133	UML

У таблиці **Skills** стовпці **e_id** й **skill** разом формують складений первинний ключ таблиці. Крім того стовпець **e_id** буде виконувати роль зовнішнього ключа, що вказує на таблицю **Empl**.

Таблиця **Skills** перебуває в 3НФ.

Повернемося до таблиці **Empl**.

Третя нормальна форма (3НФ)

Для того, щоб таблиця перебувала в третій нормальній формі необхідно, щоб вона перебувала в другій нормальній формі й у ній повинні бути виключені всі транзитивні залежності. Таблиця **Empl** містить наступні функціональні залежності:

$e_id \rightarrow e_name, job, d_id, d_name$

$d_id \rightarrow d_name$

Первинним ключем є **e_id** і всі атрибути повністю функціонально залежні від нього. Але, помітимо також, що:

$e_id \rightarrow d_id$

$e_id \rightarrow d_name$

й

$d_id \rightarrow d_name$

Це значить, що функціональна залежність **$e_id \rightarrow d_name$** є транзитивною (вона містить проміжний крок - залежність **$d_id \rightarrow d_name$** . Щоб позбутися від цієї транзитивної залежності а, заодно, і зменшити надмірність інформації в таблиці, розбиваємо цю таблицю на дві:

Dep:

d_id	d_name
11	Administrative department
12	Projecting department
13	Realization department
14	Testing department

d_id – первичний ключ таблиці

Empl:

e_id	e_name	job	d_id
111	Drovko	Administrator DB	12
112	Ivanov	System admin	11
113	Konov	Program's	14
114	Kornienko	Program's	14
121	Koval	Program's	13
122	Palkin	Program's	14
123	Petrov	Program's	13
131	Sidorov	Administrator DB	12
132	Turov	System admin	11
133	Vertko	Program's	13

e_id – первичний ключ таблиці

Обидві таблиці перебувають в 3НФ.

З огляду на досвід формування попередніх таблиць, для зберігання інформації про Клієнтів і Проекти організації створюємо наступні таблиці:

Таблиця з інформацією про Клієнтів організації:

Client:

cl_id	f_name	addr	tel
25	Strogof	10945 Brigge Rd. Oacland	0563657298
48	Tvorogenko	578 Blijnde St. Rockville	0456783458
77	Dragov	5768 Seventh Av. Gary	0567234852

Стовпець **cl_id** (код клієнта) є первинним ключем таблиці. Таблиця перебуває в 3НФ.

Формуємо таблицю

Projects:

pr_id	pr_theme	e_id	cl_id	data_start	data_finish	price
234		111	25	12.02.2014	12.02.2015	12 345,67
235		111	77	18.03.2014	27.08.2015	17 345,78
675		132	48	24.05.2014	13.10.2015	20 567,55
897		112	48	20.06.2014	23.03.2015	10 340.99

У цю таблицю ми помістили наступну інформацію:

pr_id - унікальний код проекту (номер укладеного договору на розробку проекту). Буде виконувати роль первинного ключа таблиці;

pr_theme - назва теми проекту;

e_id - код співробітника, що є керівником проекту. Буде виконувати роль зовнішнього ключа, що вказує на таблицю **Empl** для одержання інформації про керівника проекту;

cl_id - код клієнта, що є замовником даного проекту. Буде виконувати роль зовнішнього ключа, що вказує на таблицю **Client** для одержання інформації про те, який клієнт є замовником проекту.

data_start - дата укладання договору на розробку проекту;

data_finish - строк виконання проекту;

price- вартість проекту.

Таблиця перебуває в 3НФ.

У таблицю **Projects** ми не включили інформацію про співробітників, зайняті в реалізації проекту із вказівкою частки співробітника в загальному гонорарі за проект. Вирішуємо цю проблему:

Таблиця Проект - Співробітник

Створимо таблицю **Proj_Emp**, у якій стовпці **pr_id** й **e_id** будуть формувати складений первинний ключ, а окремо кожний з них буде зовнішнім ключем, що вказує на відповідні таблиці **Projects** й **Empl**. Не ключовий стовпець **royalty_share** буде показувати частку співробітника в загальному гонорарі за проект.

Proj_Emp:		
pr_id	e_id	royalty_share
234	111	0,4
234	112	0,2
234	121	0,2
234	133	0,2
235	111	0,4
235	133	0,3
235	123	0,3
678	132	0,4
678	112	0,3
678	122	0,3
897	112	0,4
897	133	0,2
897	121	0,2
897	114	0,2

Таблиця перебуває в 3НФ.

Процес проектування бази даних закінчений.

3 Опис структури таблиць

Дамо базі даних ім'я **Organization** й опишемо структуру сформованих таблиць.

Опис структури таблиці **Empl**:

Ім'я стовбця	Опис	Тип даних	null (Да/Нет)	Ключі
e_id	Унікальний код співробітника	int(3)		первинний
e_name	Прізвище співробітника	varchar(15)		
job	Посада	varchar(15)		
d_id	Номер відділу	int(2)		зовнішній - Dep (d_id)

Опис структури таблиці **Dep**:

Ім'я стовбця	Опис	Тип даних	null (Да/Нет)	Ключі
d_id	Унікальний номер відділу	int(2)		Первинний
d_name	Назва відділу	varchar(30)		

Опис структури таблиці **Skills:**

<i>Ім'я стовбця</i>	<i>Опис</i>	<i>Тип даних</i>	<i>null (Да/Нет)</i>	<i>Ключі</i>
e_id	Код співробітника	int(3)		Первинний, зовнішній - Empl (e_id)
skill	Назва видукваліфікації	varchar(10)		Первинний

Опис структури таблиці **Client:**

<i>Ім'я стовбця</i>	<i>Опис</i>	<i>Тип даних</i>	<i>null (Да/Нет)</i>	<i>Ключі</i>
cl_id	Унікальний код клієнта	int(2)		Первинний
f_name	Прізвище клієнта	varchar(15)		
addr	Адреса клієнта	varchar(50)	Да	
tel	Телефон клієнта	char(10)	Да	

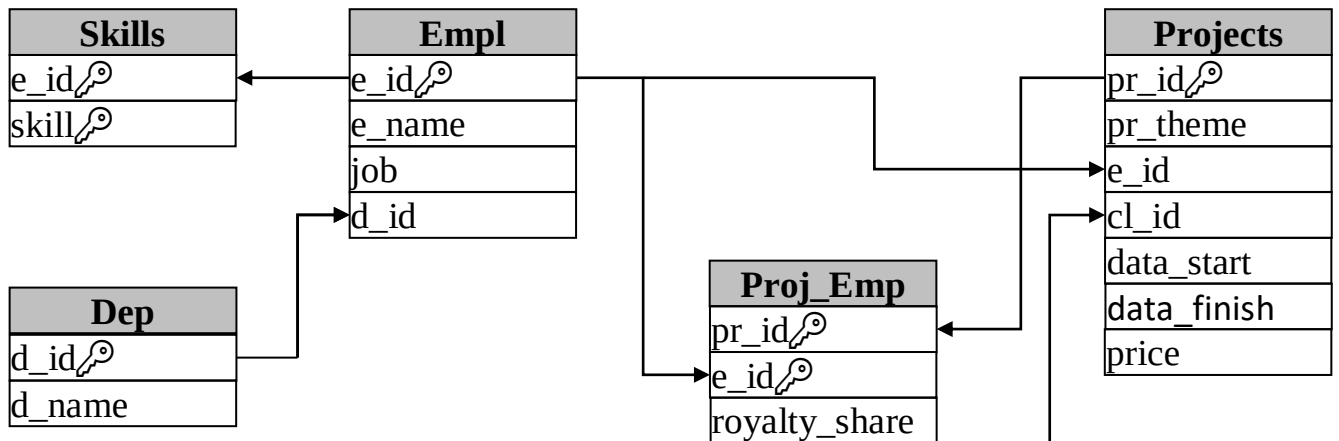
Опис структури таблиці **Projects:**

<i>Ім'я стовбця</i>	<i>Опис</i>	<i>Тип даних</i>	<i>null (Да/Нет)</i>	<i>Ключі</i>
pr_id	Унікальний код проекту	int(3)		Первинний
pr_theme	Назва проекту	varchar(50)		
e_id	Код співробітника – керівника проекту	int(3)		Зовнішній - Empl (e_id)
cl_id	Код клієнта-замовника проекту	int(2)		Зовнішній - Client (cl_id)
data_start	Дата заключення договору на розробку проекту	date		
data_finish	Строк виконання проекту	date	Да	
price	Вартість проекту	Float(7,2)	Да	

Опис структури таблиці **Proj_Emp:**

<i>Ім'я стовбця</i>	<i>Опис</i>	<i>Тип даних</i>	<i>null (Да/Нет)</i>	<i>Ключі</i>
pr_id	Код проекту	int(3)		Первинний, зовнішній - Projects (pr_id)
e_id	Код співробітника	int(3)		Первинний, зовнішній - Empl (e_id)
royalty_share	Частка співробітника в гонорарі за проект	Float(2,1)	Да	

ER – діаграма бази даних Organization



Питання та завдання для самоконтролю знань студентів

1 Поясніть своїми словами зміст термінів:

- надмірність даних;
- аномалії включення, відновлення, видалення;
- сторонній атрибут;
- нормалізація відносин;
- функціональні залежності;
- декомпозиція;
- перша нормальна форма;
- друга нормальна форма;
- третя нормальна форма;
- четверта нормальна форма;
- нормальна форма Бойса - Кодда;
- п'ята нормальна форма.

2 Приведіть приклади відносин, що не перебувають у першій нор-мальній формі.

3 Розгляньте відношення з нав'язаною функціональною залежністю. Продемонструйте алгоритм виявлення функціональної залежності.

4 Що таке транзитивна залежність? До яких негативних наслідків приводить наявність транзитивної залежності? Які міри допомагають позбутися від цих наслідків?

5 Що служить альтернативою нормалізації за допомогою декомпозиції?

6 Які небажані функціональні залежності усуваються за допомогою приведення відносини до нормальної форми Бойса - Кодда?

7 Охарактеризуйте багатозначні залежності. Приведіть приклади відносин, де вони присутні.

8 Освітите процес проектування реляційної бази даних. Які дії припускає фаза логічного проектування реляційної БД?

9 Яким образом можна спростити концептуальну модель даних, щоб полегшити процес переходу від концептуальної моделі до логічної моделі?

10 Викладете правила методики перетворення концептуальних структур даних у реляційні структури.

Розробив: Ланська С.С.

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

План самостійного вивчення теми № 2

з навчальної дисципліни «Бази даних»

Тема	Вбудовані функції MySQL
Мета	Розглянути функції MySQL, які можна застосовувати в запитах, заснованих на використанні оператора SELECT.

знати:

- основні вбудовані функції MySQL;
- оператори мови SQL.

вміти:

- застосовувати функції у виразах SELECT й WHERE;
- використовувати оператори в запитах.

Час – 2 години

Навчальні питання:

- 1 Оператори
- 2 Керуючі функції
- 3 Функції рядків
- 4 Використання LIKE

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

- По-перше* MySQL пропонує широкий вибір вбудованих операторів і функцій. Більшість цих функцій призначено для використання у виразах SELECT й WHERE.
- По-друге* При роботі з операторами порівняння необхідно пам'ятати про те, що, за винятком декількох випадків, що виділяють особливо, порівняння чого-небудь зі значенням NULL дає в результаті NULL
- По-третьє* Самими корисними керуючими функціями є IF й CASE.

Література

- 1Малыхина, М.П. Базыданных: основы, проектирование, использование [Текст] / М.П. Малыхина – СПб.: БХВ– Петербург, 2004, – 512с.: ил.

Навчальні матеріали до самостійного вивчення теми

MySQL пропонує широкий вибір вбудованих операторів і функцій, які можуть виявитися корисними при створенні запитів. Більшість цих функцій призначено для використання у виразах SELECT й WHERE. Існують також деякі спеціальні функції угруповання для використання у вираженні GROUP BY. Ви вже бачили приклади використання основних операторів порівняння, а також функцій count () і max (). Число доступних для використання функцій дуже велико, тому ми розглянемо тільки найбільш корисні з них.

1 Оператори

В MySQL використовуються три головних типи операторів: знаки арифметичних операцій, оператори порівняння й логічних операторів.

Арифметичні операції

В MySQL використовуються звичайні арифметичні операції: додавання (+), вирахування (-), множення (*) і розподіл (/). Розподіл на нульове значення дає безпечний результат NULL.

Оператори порівняння

При роботі з операторами порівняння необхідно пам'ятати про те, що, за винятком декількох випадків, що виділяють особливо, порівняння чого-небудь зі значенням NULL дає в результаті NULL. Це стосується й порівняння значення NULL зі значенням NULL:

```
select NULL=NULL;
```

NULL = NULL
NULL

1 row in set (0.00 sec)

Зрівняєте цей запит з наступним;

```
select NULL IS NULL;
```

```
I NULL IS NULL;
```

NULL IS NULL

1 row in set (0.00 sec)

Іншим моментом, про який також не слід забувати, є той факт, що в MySQL порівняння рядків у більшості випадків нечутливі до регістра символів. Якщо ви хочете, щоб рядки рівнялися з урахуванням регістра, укажіть перед кожною з них ключове слово `BIWARY`. Наприклад, запит:

```
select * from department where name='отдел маркетинга';
```

знайде слова 'отдел маркетинга' незалежно від регістра.

Якщо ж регістр символів важливий, варто вказати ключове слово `binary`:

```
select * from department where name = binary'отдел маркетинга';
```

Озброївшись всією цією інформацією, розглянемо оператори порівняння. Найбільше часто використовувані з них наведені в таблиці 2.1.

Таблиця 2.1 – Оператори порівняння

Оператор	Значення
=	Рівність
!= або <>	Нерівність
<	Менше
<=	Менше або дорівнює
>	Більше
>=	Більше або дорівнює
n BETWEEN min AND max	Перевірка діапазону
n IN (множина)	Приналежність до множини. Як множина може використатися список літеральних значень або виразів або підзапит. Прикладом множини є (яблуко, апельсин, груша)
<=>	Порівняння з урахуванням NULL: повертає 1 (істина) при порівнянні двох значень NULL
n IS NULL	Перевірка на наявність NULL у n
ISNULL(n)	Перевірка на наявність NULL у n

Логічні оператори

MySQL підтримує всі звичайні логічні операції, які можна використати у виразах об'єднання. Логічні вирази в MySQL можуть приймати значення 1 (істина), 0 (хибність) або NULL. Крім того, MySQL інтерпретує будь-яке ненульове значення, відмінне від NULL, як значення "істина".

Деякі з таблиць істинності, що містять значення NULL, небагато відрізняються від того, що можна було б очікувати. Логічні оператори наведені в таблиці 2.2.

Таблиця 2.2 – Логічні оператори.

Оператор	Приклад	Значення
AND або &&	n && m	Логічне « І » істина && істина = істина хибність&&хибність = хибність Всі інші вирази оцінюються як NULL
OR або	n m	Логічне « АБО » істина істина = істина NULL хибність = NULL NULL NULL =NULL хибність хибність = хибність
NOT або !	NOT n	Логічне « НІ » ! істина = хибність ! хибність = істина ! NULL = NULL
XOR	n XOR m	Логічне « виключаюче АБО » істина XOR істина = хибність істина XOR хибність = істина хибністьXOR істина = істина NULLXORn = NULL n XOR NULL + NULL

2 Керуючі функції

Першою множиною функцій, які ми розглянемо, будуть керуючі функції. Самими корисними з них є IF й CASE. Вони працюють так само, як оператори if й switch або case (відповідно) у більшості інших мов програмування.

Функція IF має прототип

IF (e1, e2, e3)

Якщо вираз e1 є істиною, то IF повертає e2, інакше повертається e3. Наприклад, використовуючи базу даних employee, можна виконати наступний запит:

```
select name, if( job='Програміст', "фанат", "нефанат") from employee;
```

Функція CASE має наступні два прототипи:

CASE значення

WHEN [значення-для-порівняння] THEN результат

[WHEN [значення-для-порівняння] THEN результат ...]

[ELSE результат]

END

або

CASE

WHEN [умова] THEN результат

[WHEN [умова] THEN результат ...]

[ELSE результат]

END

Цю функцію можна використати для одержання одного або декількох значень.

Наприклад, розглянемо наступний запит:

```
select workdate, case
when workdate < 2000-01-01 then "архів"
when workdate < 2004-01-01 then "старі"
else "поточні"
end
from assignment;
```

У цьому запиті оцінюються значення `workdate` для кожного завдання з таблиці `assignment`. Завдання минулого століття характеризуються як здані в архів, ті завдання, що мають дату до початку цього року, - як старі, а всі інші - як поточні.

3 Функції рядків

Строкові функції MySQL можна розділити на дві категорії: функції обробки й функції порівняння рядків. Останні виявляються більше корисними, чим перші.

Функції обробки рядків

Список найбільш важливих функцій обробки рядків показаний у таблиці 2.3.

У керівництві ви знайдете значно більше великий список.

Таблиця 2.3 – Функції обробки рядків символів.

Оператор	Призначення
<code>concat (s1, s2, ...)</code>	Конкатенація рядків <code>s1, s2,...</code>
<code>conv(n, исx_базис, новый_базис)</code>	Конвертування числа <code>n</code> із системи числення з основою <code>исx_базис</code> у систему з основою <code>новый_базис</code> . (Може здатися дивним, що ця функція перебуває серед строкових, але деякі основи позначаються буквами, наприклад <code>hexadecimal</code> .)
<code>length(s)</code>	Повертає довжину рядка в символах
<code>load_file(имя_файла)</code>	Повертає вміст файлу <code>имя_файла</code> у вигляді рядка

Оператор	Призначення
locate (<i>игла, стог, старт</i>)	Повертає початкову позицію рядка <i>игла</i> в рядку <i>стог</i> . Пошук починається з позиції <i>старт</i>
lower(s) та upper(s)	Конвертує рядок s у нижній або верхній регістр
quote(s)	Видозмінює рядок s так, щоб віна підходив для уведення в базу даних, тобто містить рядок в одиночні лапки й вставляє зворотну косу рису в необхідних випадках
replace(<i>исходная, поиск, замена</i>)	Повертає рядок, отриманий на основі рядка <i>исходная</i> , шляхом заміни всіх наявних у нього підстрок <i>поиск</i> на рядок <i>замена</i>
suondex(s)	Повертає рядок, схожу по вимові на s. Наприклад, іноді легше порівнювати схожі по вимові імена, чим самі імена
subtribg(s, <i>позиция, число</i>)	Повертає <i>число</i> символів рядка s, починаючи з позиції <i>позиция</i>
trim(s)	Видаляє початкові й кінцеві пробіли з рядка s. (Можна також використати ltrim (s) для видалення пробілів тільки на початку рядка й rtrim(s) для видалення їх тільки наприкінці.)

Функції порівняння рядків

На додаток до оператора рівності для порівняння рядків MySQL пропонує й інші функції порівняння.

- LIKE. Виконує порівняння з використанням групових символів.
- RLIKE. Виконує порівняння регулярних виразів.
- STRCMP. Виконує порівняння рядків, аналогічне strcmp О в С.
- MATCH. Виконує повнотекстовий пошук.

4 Використання LIKE

Використання LIKE для порівняння із груповими символами

Розглянемо приклад використання LIKE:

```
select *\nfrom department\nwhere name like '%проект%';
```

Замість того щоб шукати імена ' проект ', ми тут шукаємо імена, що містять ' проект '. У результаті ми одержимо наступне:

departmentID	name
128	Отдел проектирования

1 row in set (0.04 sec)

Функція LIKE підтримує два види групових символів. Знак відсотка (%), як й у попередньому прикладі, відповідає будь-якому числу символів (включаючи нульове). Таким чином, вираження ' %проект% ' відповідає будь-якому рядку, що містить слово проект. Знову помітимо, що порівняння рядків, загалом кажучи, невідчутно до регістра символів, як уже згадувалося вище.

Другим груповим символом є знак підкреслення (_), що відповідає будь-якому одному символу. Наприклад, '_at' буде відповідати рядкам 'cat', 'mat', 'bat1 і т.д.

Використання RLIKE для порівняння регулярних виразів.

Функцію RLIKE можна використати для порівняння на основі регулярних виражень.

Регулярний вираз - це шаблон, що описує загальну форму рядка. Існує спеціальна нотація для опису особливостей, які ми хотіли б бачити у відповідних рядків. Короткий опис такої нотації приводиться нижче/

Такому рядку відповідає будь-який строковий літерал (тобто строкова константа). Наприклад, за допомогою шаблону 'cat' буде знайдений рядок 'cat'. Але з його допомогою будуть також знайдені 'catacomb' й 'the cat sat on the mat'. Шаблону ' cat ' буде відповідати ' cat ' у будь-якому місці рядку, що перевіряється.

Якщо потрібно знайти тільки слово 'cat ', шаблоном повинне бути l"cat\$1. Знак вставки (") позначає "показчик початку рядка", тобто в цьому випадку рядок повинна починатися зі слова 'cat'. Знак долара (\$) позначає "показчик кінця рядка", тобто в цьому випадку рядок повинна закінчуватися словом ' cat '. Тому шаблону ' "cat\$' буде відповідати тільки слово 'cat' і ніщо інше.

Як і вираження LIKE, регулярні вираження теж можуть містити групові символи, але в цьому випадку припустимий груповий символ іншої й тільки один - це крапка (.), що означає відповідність будь-якому одному символу. Так що тепер '.at' буде відповідати рядкам 'cat1, 'mat1, 'bat' і т.д.

Досить одного групового символу, оскільки є можливість указати, скільки разів символи (включаючи й групові) можуть з'явитися в рядку.

Наприклад, спеціальний знак * після символу означає, що символ може з'явитися або нуль, або більше раз. Так, 'п*' виявить ' ', 'п', 'пп', 'ппп' і т.д. Можна містити символи в дужки, і '(cat) *' виявить '', 'cat', 'catcat', 'catcatcat' і т.д. Можна також використати груповий символ, так що '. *' відповідає будь-якому числу будь-яких символів, тобто практично чому завгодно.

Точно так само знак "плюс" (+) означає, що символ або набір символів, передяким він стоїть, повинен повторюватися один або більше раз, а знак питання (?) означає присутність нуль або один раз. Можна також указати конкретний діапазон для числа повторень, наприклад, '(cat) (2/4)' відповідає 'catcat', 'catcatcat' й 'catcatcatcat'.

Нарешті, є цілий ряд класів символів, що представляють заздалегідь певні безлічі. Наприклад, '[: alnura :]' відповідає будь-якому буквено-цифровому символу.

Якщо ви використаєте мову програмування Perl, вам корисно знати, що MySQL використовує регулярні вираження POSIX, синтаксис яких відмінний від регулярних виражень в Perl.

Питання та завдання для самоконтролю знань студентів

- 1 Який з наступних операторів не годиться для перевірки значення на рівність значенню NULL?
 - а) ISNULL()
 - б) <=>
 - в) IS NULL
 - г) =
- 2 Виклик strcmp ('fred', 'Fred') поверне
 - а) -1
 - б) 0
 - в) 1
 - г) 2
- 3 Яку з наступних функцій варто використати для одержання назви місяця зі значення дати?
 - а) dayname ()

- б) `extract ()`
 - в) `subdate ()`
 - г) `now ()`
- 4 Яку з наступних функцій використовує MySQL для шифрування внутрішніх паролів своїх користувачів?
- а) `password()`
 - б) `encrypt ()`
 - в) `md5 ()`
 - г) `sha()`
- 5 При використанні функцій, що групують, в операторі SELECT без виразу GROUP BY
- а) буде отримане повідомлення про синтаксичну помилку;
 - б) вся таблиця буде розглядатися як єдина група;
 - в) вся результуюча безліч буде розглядатися як єдина група;
 - г) кожен рядок буде вважатися окремою групою.
- 6 Створіть запит, що повертає імена службовців (name) і інформацію про їхній кваліфікації (job), але у випадку, коли кваліфікацією є 'Програміст', замініте її на 'Програміст/Аналітик'.
- 7 Створіть запит, що додає в базу даних новий відділ з назвою "Відділ керування майном". Потім додайте інформацію про нового службовця по імені Фред Смит, що буде працювати в цьому відділі на посаді адміністратора БД. При додаванні номера відділу використайте `last_insert_id()`.
- 8 Запропонуйте запит, що повертає саме останнє завдання з таблиці `assignment`. (Підказка: використайте функцію, що групує, `max ()`.)

Розробив: Ланська С.С.

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

План самостійного вивчення теми № 3

з навчальної дисципліни «Бази даних»

Тема	Тригери: створення, використання, видалення.
Мета	Ознайомитись з поняттям тригера, створенням тригера, прикладами використання тригерів

знати:

- поняття та призначення тригера;
- оператори створення та видалення тригера.

вміти:

- створювати тригер;
- використовувати локальні та контекстні змінні при організації тригера.

Час – 4 години

Навчальні питання:

- 1 Створення тригера.
- 2 Приклади використання тригерів.
- 3 Видалення тригера.
- 4 Перегляд існуючих тригерів.

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

- По-перше* Тригер - це спеціальна збережена процедура, що виконується автоматично при настанні одного з подій: INSERT, UPDATE або DELETE, тобто при додаванні, зміні або видаленні рядків у таблиці. Тригер завжди є прив'язаним до конкретною таблицею.
- По-друге* Тригер створюється за допомогою оператора CREATE TRIGGER.
- По-третє* Для видалення тригера призначений оператор DROP TRIGGER

Література

1Малыхина, М.П. Базыданных: основы, проектирование, использование [Текст] / М.П. Малыхина – СПб.: БХВ– Петербург, 2004, – 512с.: ил.

2Карпова, Т.С., Базыданных: модели, разработки, реализации [Текст]: / Т.С. Карпова – СПб.: Питер, 2001. – 303с.

Навчальні матеріали до самостійного вивчення теми

Тригер - це спеціальна збережена процедура, що виконується автоматично при настанні одного з подій: INSERT, UPDATE або DELETE, тобто при додаванні, зміні або видаленні рядків у таблиці. Тригер завжди є прив'язаним до конкретною таблицею.

1 Створення тригера

Тригер створюється за допомогою оператора CREATE TRIGGER. Синтаксис оператора:

CREATE TRIGGER *ім'я тригера*

Коли Спрацьовує Подія

ON *ім'я таблиці* FOR EACH ROW *Оператори тіла тригера*

Подія:

INSERT - тригер прив'язаний до події вставки нових записів у таблицю.

UPDATE - тригер прив'язаний до події відновлення записів у таблиці.

DELETE - тригер прив'язаний до події видалення записів у таблиці.

Коли Спрацьовує - може мати два значення:

BEFORE - до події

AFTER - після події.

Оператори тіла тригера :

BEGIN... END;

У тілі тригера допускаються всі специфічні для збережених процедур і функцій конструкції:

1 Інші складені оператори BEGIN... END.

2 Оператори керування потоком виконання (IF, CASE, WHILE, REPEAT, LEAVE, ITERATE).

3 Оголошення локальних змінних за допомогою оператора DECLARE і завдання їм значень за допомогою оператора SET.

4 Іменовані умови й оброблювачі помилок.

Крім зазначених операторів у тілі тригера можна використати так називані контекстні змінні, тобто змінні із префіксом NEW й OLD, які дозволяють одержувати доступ, відповідно, до нового й старого значення змінної. Синтаксис:

NEW.Ім'я змінної *OLD.Ім'я змінної*.

Наприклад: NEW.Total - дозволяє одержати доступ до нового значення змінної Total.

OLD.Total - дозволяє одержати доступ до старого значення змінної Total.

NEW не використовується для події DELETE. OLD не використовується для події INSERT.

Для створення тригера потрібна спеціальний привілей
TRIGGER.

Зауваження. У СУБД MySQL тригер не можна прив'язати до каскадного відновлення або видалення записів з таблиці типу InnoDB по зв'язку первинний ключ/зовнішній ключ.

2 Приклади використання тригерів

1 Створимо тригер, що реалізує каскадне видалення для таблиць типу MyISAM. Після кожного видалення рядка в таблиці Otdel виконується автоматичне видалення відповідних рядків з таблиці Sotr. Код тригера має такий вигляд:

```
DELIMITER //
CREATE TRIGGER TRDelOtdel AFTER DELETE OnOtdel
FOR EACH ROW BEGIN
DELETE FROM Sotr WHERE NOtd=OLD.NOtd; END;
//
DELIMITER ;
```

2 Нехай задані таблиці "Заповнення салонів" й "Квитки".

Таблиця "Квитки" має ім'я `bilet` і містить інформацію про продані квитки на різні авіарейси. Таблиця має наступну структуру:

`Nb` - номер квитка, первинний ключ.

`Nr` - номер рейса.

`Data` - дата вильоту.

`FIO`—ПІБ пасажирів.

`Comfort` - рівень комфортності, може приймати одне із двох значень: або `B` - бізнес-клас, або `E` - клас-економ.

Таблиця створюється за допомогою наступного оператора.

```
CREATE TABLE 'bilet' (  
  'Nb' smallint(6) NOT NULL AUTO_INCREMENT,  
  'Nr' smallint(6) NOT NULL,  
  'Data' DATE NOT NULL,  
  'comfort' enum('B','E') NOT NULL,  
  'FIO' varchar(30) NOT NULL, PRIMARY KEY ('Nb')  
) ENGINE=InnoDB AUTO_INCREMENT=1  
DEFAULT CHARSET=utf8 ;
```

Таблиця "Заповнення салонів" містить інформацію про те, скільки квитків кожного виду комфортності вже продано на даний рейс на дану дату. Таблиця називається `Salon` і має наступну структуру:

`Nr` - номер рейса;

`Data` - дата вильоту;

`KolB` - містить кількість проданих квитків бізнес-класу на цей рейс.

`KolE` - містить кількість проданих квитків класу економ на цей рейс.

Таблиця має складений первинний ключ: `Nr`, `Data`.

Таблиця `Reis` створюється за допомогою наступного оператора:

```
CREATE TABLE 'Salon' (  
  'Nr' smallint(6) NOT NULL,  
  'Data' date NOT NULL,  
  'kol' smallint(6) DEFAULT NULL,  
  'Kol' smallint(6) DEFAULT NULL, PRIMARY KEY ('Nr','Data')  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Потрібно написати тригер на додавання записів до таблиці "Квиток". Щораз при покупці квитка пасажиром відбувається додавання нового запису до таблиці "Квиток", що супроводжується автоматичним виконанням наступні:

- Перевіряється, є чи вже інформація про цей рейс у таблиці "Заповнення салонів". Якщо ні, то додається новий запис у таблицю "Заповнення салонів".
- До значень полів KOLB й KOLE таблиці "Заповнення салонів" додається кількість куплених квитків, відповідно, бізнесу-класу й класу економ.

Код тригера має такий вигляд:

```
DELIMITER //

CREATE TRIGGER INSBilet BEFORE INSERT ON Bilet FOR EACH ROW
BEGIN
    DECLARE nomR SMALLINT;
    DECLARE dataR DATE;
    DECLARE kolE1 SMALLINT;
    DECLARE kolB1 SMALLINT;
    SELECT Nr, Data INTO nom, data FROM Salon
    WHERE Nr=NEW.Nr AND data=NEW.data; SET kolB1=0;
    SET kolE1=0;
    IF (NEW.comfort='B') THEN
        SET kolB1=1;
    ELSE
        SET kolE1=1;
    END IF;
    IF (dataR IS NULL) THEN
        INSERT INTO Salon (Nr, data, kolB, kolE)
        VALUES (NEW.Nr, NEW.data, kolB1, kolE1);
    ELSE
        UPDATE Salon SET kolB=kolB+kolB1, kolE=kolE+kolE1
        WHERE Nr=nomR AND data=dataR;
    END IF;
END;

//
```

DELIMITER;

Тригер спрацьовує автоматично при додаванні запису до таблиці Квиток:

```
> INSERT INTO Bilet (Nr, data, FI, comfort)
```

```
VALUES (3337, "2008.11.20", "Іванов І.І", "В");
```

Для перевірки роботи тригера варто переглянути вміст таблиці Salon.

3 Видалення тригера

Для видалення тригера призначений оператор DROP TRIGGER, що має наступний синтаксис:

```
DROP TRIGGER ІМ'Я ТАБЛИЦІ . Ім'я тригера
```

Оператор DROP TRIGGER видаляє існуючий тригер для даної таблиці.

4 Перегляд існуючих тригерів

Одержати список існуючих тригерів можна за допомогою оператора SHOW TRIGGERS, що має наступний синтаксис:

```
SHOW TRIGGERS [FROM ІМЯБАЗИДАНИХ ] [LIKE Шаблон ]
```

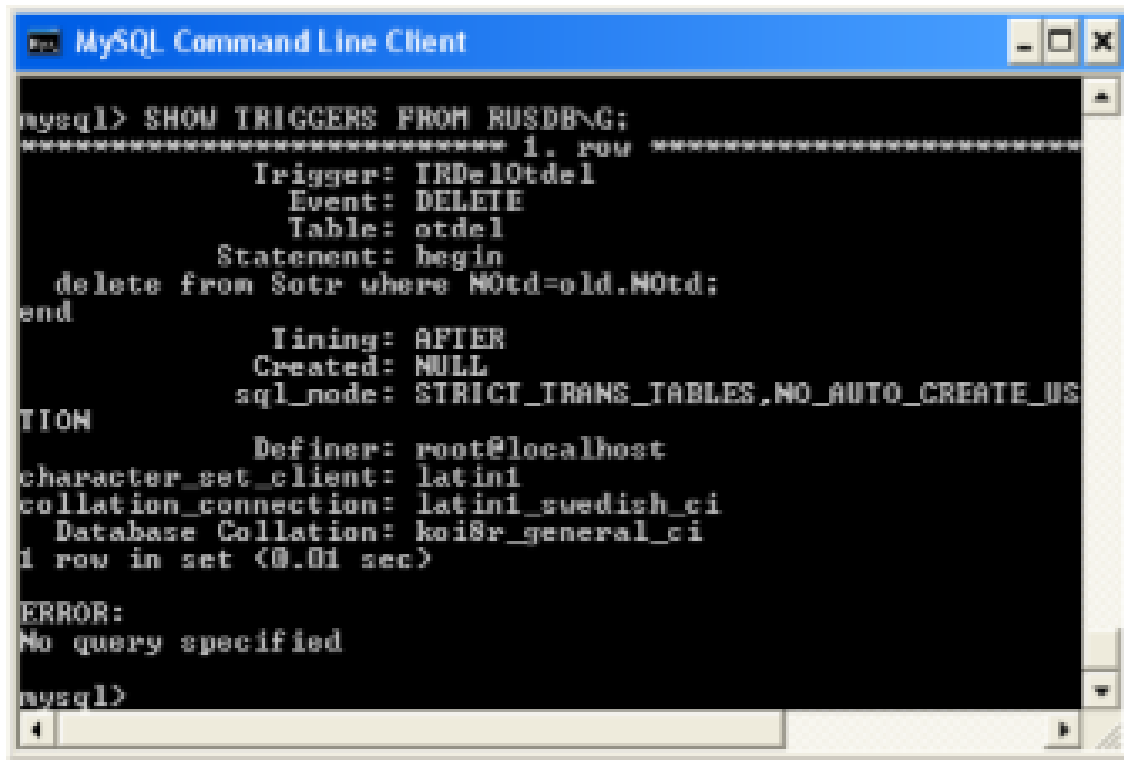
Оператор дозволяє витягти список і характеристики тригера із заданої бази даних. Якщо база даних має велику кількість тригерів, висновок можна обмежити за допомогою ключового слова LIKE, що має той же синтаксис, що й ключове слово LIKE в операторі SELECT. На малюнку 9.1 показане використання команди SHOW TRIGGERS для бази даних RUSDB. Поряд з ім'ям тригера, виводиться подія й таблиця, до якого прив'язаний тригер, а також тіло тригера:

```
BEGIN
```

```
DELETE FROM Sotr WHERE NOtd=OLD.NOtd;
```

```
END
```

На малюнку зазначений момент спрацьовування тригера: AFTER.



```
mysql> SHOW TRIGGERS FROM RUSSDB\G;
+-----+
| Trigger | Trigger | Event | Table | Statement | Timing | Created | sql_mode | Definer | Character Set | Collation Connection | Database Collation |
+-----+
| TRDelOtDel | DELETE | otdel | begin | AFTER | NULL | STRICT_TRANS_TABLES,NO_AUTO_CREATE_US | root@localhost | latin1 | latin1_swedish_ci | koi8r_general_ci |
+-----+
1 row in set (0.01 sec)

ERROR:
No query specified

mysql>
```

Питання та завдання для самоконтролю знань студентів

- 1 Що таке тригер?
- 2 Для чого використовуються контекстні змінні в тілі тригеру?
- 3 Чи можна в тілі тригеру використовувати локальні змінні?
- 4 Чи можна в тілі тригеру використовувати оператори розгалуження та циклічні оператори?
- 5 За допомогою яких команд можна створити тригер?
- 6 Як видалити тригер?
- 7 Як можна переглянути які тригери вже створені в базі даних?

Розробив: Ланська С.С.

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

План самостійного вивчення теми № 4

з навчальної дисципліни «Бази даних»

Тема Рівні привілеїв. Створення облікових записів користувачів за допомогою GRANT й REVOKE

Мета Розглянути створення облікових записів користувачів, різні доступні привілеї й те, як ці привілеї представлені в таблицях MySQL

знати:

- принципи створення облікових записів;
- рівні привілеїв;
- принципи використання таблиць привілеїв..

вміти:

- створювати облікові записи користувачів;
- встановлювати користувальницькі привілеї.

Час – 4 години

Навчальні питання:

- 1 Створення облікових записів користувачів за допомогою GRANT й REVOKE.
- 2 Рівні привілеїв.
- 3 Таблиці привілеїв.

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше Вся інформація про користувачів MySQL і користувальницьких правах в остаточному підсумку зберігається в базі даних MySQL.

По-друге Привілею, які надаються за допомогою оператора GRANT, можна поділити на дві основні категорії: привілею користувача й привілеї адміністратора

По-третє Оператор REVOKE - протилежність оператора GRANT.

Література

1Малыхина, М.П. Базыданных: основы, проектирование, использование [Текст] / М.П. Малыхина – СПб.: БХВ– Петербург, 2004, – 512с.: ил.

2Карпова, Т.С., Базыданных: модели, разработки, реализации [Текст]: / Т.С. Карпова – СПб.: Питер, 2001. – 303с.

Навчальні матеріали до самостійного вивчення теми

1 Створення облікових записів за допомогою GRANT й REVOKE

Користувальницькі привілеї встановлюються оператором GRANT і скасовуються оператором REVOKE. Це - стандартні оператори SQL, які можна виконувати точно так само, як будь-який інший оператор. Вся інформація про користувачів MySQL і користувальницьких правах в остаточному підсумку зберігається в базі даних MySQL, точно так само, як і ваші додатки.

Щоб запустити зазначені оператори, потрібно мати відповідний рівень привілеїв доступу. Якщо ви встановлювали MySQL самостійно, ви будете мати доступ до кореневого облікового запису й, таким чином, мати необхідні привілеї. Якщо ви використовуєте MySQL на машині, контрольованої кимсь іншим (наприклад, на роботі або на машині постачальника послуг Internet), то можете не мати прав доступу, необхідних для виконання відповідних запитів. Якщо ви відповідних прав не маєте, то одержите повідомлення про помилку, подібне наступний:

```
ERROR 1045: Access denied for user: 'laura@127.0.0.1'
```

```
(Using password: YES)
```

Надання привілеїв

Почнемо з розгляду оператора GRANT, що використовується для створення облікових записів користувачів і надання їм прав доступу до баз даних, таблицям і функціям. Спочатку розглянемо наступний простий приклад:

```
grant usage
```

```
on *
```

```
to luke@localhost identified by 'пароль';
```

Цей оператор створює обліковий запис для користувача з ім'ям luke, коли він намагається увійти в систему з вузла localhost. Для нього встановлюється пароль

(замість зазначеного тут пароля пароль, мабуть, варто використати що-небудь більше надійне!). Слово usage указує привілею, який ви наділяєте користувача luke. Воно означає, що з може зареєструватися (тобто увійти в систему) - і нічого іншого. Вираження ON використовується для того, щоб указати, на які об'єкти поширюються надавані права. Через те, що тут ми надаємо тільки право з у системі, вираження ON у цьому випадку не дуже важливо.

Загальна форма оператора GRANT з посібника з MySQL виглядає в такий спосіб:

```
GRANT привілея [ (список_стовбців) ]  
[,привілея [ (список_стовбців) ] ...]  
ON {ім'я_таблиці | * | *.* | ім'я_БД.*}  
TO ім'я_користувача [ IDENTIFIED BY [PASSWORD] 'пароль']  
[, ім'я_користувача [IDENTIFIED BY 'пароль'] ...]  
[REQUIRE  
NONE |  
[{SSL | X509}]  
[CIPHERшифр [AND] ]  
[ISSUER видавець [ AND] ]  
[SUBJECT суб'єкт]]  
[WITH [GRANT OPTION | MAX__QUERIES__PER_HOUR # |  
MAX_UPDATES_PER_HOUR # |  
MAX_CONNECTIONS_FER_HOUR #]]
```

Вираження GRANT містить список надаваних привілеїв. Одні привілеї є глобальними (тобто вони застосовуються до всіх баз даних), а інші ставляться тільки до певних об'єктів (базам даних, таблицям або стовпцям).

У вираженні ON вказуються об'єкти, на доступ до яких надаються права. Це може бути з таблиця або певна база даних з усіма її таблицями (ім'яБД . *). Можна також указати * . *, що позначає всі бази дані й всі таблиці. Якщо вказати *, привілеї будуть надані наобрану в даний момент базу даних. Якщо ніякої бази даних не обрано, привілеї будуть надані так, начебто у вираженні зазначене * . *.

Вираження TO використовується для того, щоб указати користувача, якому надаються привілею. Якщо цей користувач уже має обліковий запис, нові привілеї

будуть додані до неї. Якщо немає - обліковий запис користувача буде створена. У цьому вираженні можна вказати більше одного користувача. Можна також указати імена хостів, з яких ці користувачі можуть зареєструватися, наприклад fred@localhost. Якщо у вас виникають труднощі при реєстрації нового користувача, спробуйте додати в оператор GRANT ім'я хосту, з якого ви реєструєтесь. Ім'я користувача для входу в MySQL може не збігатися з ім'ям користувача в операційній системі. Імена користувачів з містити до 16 символів.

Вираження IDENTIFIED BY установлює пароль для нового користувача або переустановлює пароль, якщо користувач уже існує.

Користувачі можуть змінити свої паролі, виконавши команду

```
set password = password('новий_пароль');
```

Ви можете з пароль користувача за допомогою, наприклад,

```
set password for fred@localhost = password('новий_пароль');
```

Для цього вам потрібно мати право доступу до бази даних з ім'ям mysql.

Вираження WITH GRANT OPTION - це спеціальний привілей, що дозволяє користувачеві надавати привілею. Якщо ви виявили, що не можете надавати ніяких привілеїв користувачам, це значить, що саме ця привілея вам не надана. Точно так само ви не можете надавати іншим користувачам ті привілеї, який не володієте самі.

Вираження WITH з використатися для того, щоб обмежити число запитів, модифікацій або підключень, які користувачеві дозволяється виконувати за одна година. Значенням за замовчуванням для відповідних параметрів є 0, що означає відсутність обмежень.

Вираження REQUIRE дозволяє зажадати, щоб зазначений користувач при вході в систему використав захищене з'єднання. Для цього також необхідно правильно сконфігурувати MySQL.

2 Рівні привілеїв

Привілею, які надаються за допомогою оператора GRANT, можна поділити на дві основні категорії: привілею користувача й привілеї адміністратора.

Привілеї користувача

Привілею користувача наведені в таблиці 4.1.

Таблиця 4.1 – Привілеї користувача

Привілея	Значення
CREATE	Дозволяється створювати таблиці
CREATE TEMPORARY TABLES	Дозволяється створювати тимчасові таблиці
DELETE	Дозволяється видаляти рядки
EXECUTE	Дозволяється виконувати процедури
INDEX	Дозволяється створювати індекси
INSERT	Дозволяється вставляти рядки
LOCK TABLES	Дозволяється блокувати таблиці
SELECT	Дозволяється вибирати рядка
SHOW DATABASES	Дозволяється виконувати команду SHOW DATABASES для одержання списку доступних баз даних
UPDATE	Дозволяється оновлювати рядки
USAGE	Дозволяється тільки вхід у систему

Привілеї адміністратора

Привілеї, надавані тільки адміністраторові, показані в таблиці 4.2. Деякі з них можуть надаватися окремим користувачам на ваш розсуд і із застереженнями, але вуж ніяк не за замовчуванням.

Таблиця 4.2 – Привілеї адміністратора

Привілея	Значення
ALL	Користувач має всі привілеї, крім WITH GRANT OPTION
ALTER	Користувач може змінювати таблиці. Можна дозволити це деяким з найбільш кваліфікованих користувачів, але дотримуйте обережності, оскільки це дає можливість змінювати таблиці привілеїв
DROP	Користувач може видаляти таблиці. Можна дозволити це користувачам, що заслуговують довіри
FILE	Користувач може завантажувати дані з файлу. Можна дозволити це користувачам, що заслуговують довіри. Бережіться користувачів, що намагаються завантажувати файли типу /etc/passwd або щось подібне!
PROCESS	Користувач може виводити повний список процесів, тобто бачити всі процеси, виконувані MySQL
RELOAD	Користувач може використати оператор FLUSH. Цей оператор може виконувати різні завдання.
REPLICATION CLIENT	Користувач може перевірити, де перебувають головні й підлеглі об'єкти
REPLICATION SLAVE	Спеціальний привілей, призначений для користувача реплікації на підлеглому пристрої.
SHUTDOWN	Користувач може викликати mysqladmin shutdown.
SUPER	Користувач може підключитися навіть тоді, коли MySQL має максимальне число з'єднань, і може виконувати команди CHANGE MASTER, KILL (процес), mysql admin debug, PURGE MASTER LOGS й SET GLOBAL
WITH GRANT OPTION	Користувач може передавати іншим будь-які права, які він має

Є ще одна привілея - REFERENCES. Вона зарезервована для майбутнього використання, і хоча ви можете надати її в цей час, реально вона не надає ніяких прав.

Оцінка привілеїв

За допомогою оператора GRANT надаються наступні типи привілеїв.

1 Глобальні привілеї застосовуються у всіх базах даних. Вони вказуються символами *. * в операторі GRANT. Наприклад:

```
grant all on *.* to fred;
```

2 Привілеї рівня бази даних ставляться до однієї конкретної бази даних. Вони надаються за допомогою вказівки ім'яБД. * в операторі GRANT:

```
grant all on employee.* to fred;
```

3 Привілеї рівня таблиці стосуються окремої таблиці. Вони надаються за допомогою вказівки імені конкретної таблиці в операторі GRANT:

```
grant select on department to fred;
```

4 Привілеї рівня стовпця стосуються окремого стовпця. Вони вказуються у вираженні GRANT оператора GRANT. Наприклад:

```
grant select (employee) on employee to fred;
```

При спробі з'ясувати, чи має користувач право на виконання певних дій, MySQL розглядає зв'язану операторами OR комбінацію глобальних привілеїв цього користувача, його привілею рівня бази даних, привілею рівня таблиці й привілеї рівня стовпця.

Використання REVOKE

Оператор REVOKE - протилежність оператора GRANT. Він використовується для того, щоб позбавити користувача привілеїв. Наприклад:

```
revoke all on employee.* from fred;
```

Загальна форма оператора REVOKE виглядає так:

REVOKE *привілея* [(*список_стовпців*)]

[, *привілея* [(*список_стовпців*)] ...]

ON {*ім'я_таблиці* | * | *.* | ім'яБД.*}

FROM *ім'я_користувача* [, *ім'я_користувача* ...]

Як бачите, оператор REVOKE використовує, в основному, ті ж вираження, що й оператор GRANT, але з їхньою допомогою скасовуються відповідні привілеї.

3 Таблиці привілеїв

Дані, які змінюються за допомогою операторів GRANT й REVOKE, зберігаються в базі даних з ім'ям mysql. Замість використання GRANT й REVOKE ви можете змінити таблиці в згій базі даних безпосередньо, якщо, звичайно, ви

знаєте, що робите. Можна також прочитати з них дані, що може виявитися корисним при рішенні проблем прав доступу, які можуть іноді виникати.

Після безпосередньої модифікації цих таблиць необхідно виконати оператор `flush privileges`;

щоб зміни набули чинності ,

У базі даних `mysql` є шість таблиць:

- `user` • `tables_priv`
- `db` • `columns_priv`
- `host` • `func`

Тільки перші п'ять із цих таблиць стосуються привілеїв користувача. (Таблиця `func` зберігає інформацію про обумовлений користувачами функціях).

Перші три таблиці - `user`, `db` й `host` - використовуються для того, щоб з'ясувати, чи дозволене вам з'єднатися з базою даних. Всі п'ять із таблиць привілеїв використовуються для того, щоб з'ясувати, чи маєте ви право виконати запитану команду.

Таблиця user

Таблиця `user` містить інформацію про набір глобальних привілеїв користувача.

Таблиця `user` складається з наступних стовпців.

Стовпці області дії. Вони використовуються для того, щоб визначити, коли застосовна той або інший рядок привілеїв. Ці стовпці наступні.

`Host`. Ім'я хосту, з якого підключається користувач.

`User`. Ім'я користувача.

`Password`. Пароль користувача, шифрований функцією `PASSWORD ()`.

Стовпці привілеїв. Кожний з них відповідає однієї із глобальних привілеїв. Допускаються значення `Y` (якщо користувач наділений соотвествующей глобальним привілеєм) і `N` (якщо користувач такої не має ділової). Нижче наведений список імен цих стовпців.

<code>Select_priv</code>	<code>Shutdown_priv</code>
<code>Insert_priv</code>	<code>Process_priv</code>
<code>Update_priv</code>	<code>File_priv</code>
<code>Delete_priv</code>	<code>Show_db_priv</code>
<code>Index_priv</code>	<code>Super_priv</code>

Alter_priv	Create_tmp_table_priv
Create_priv	Lock_tables_priv
Drop_priv	Execute_jpriv
Grant__priv	Repl_slave_priv
References_priv	Repl_client_priv
Reload_priv	

Стовпці захисту з'єднання. Вони представляють інформацію з виразу REQUIRE оператора GRANT. Ці стовпці наступні.

ssl_type
ssl_cypher
x509_issuer
x509_subject

Стовпці обмежень. Вони містять інформацію про обмеження, які могли бути зазначені наприкінці оператора GRANT й які стосуються використання ресурсів користувачем. Ці стовпці наступні:

max_questions
max_updates
max_connections

Таблиця db

Таблиця db зберігає користувацькі привілеї для конкретних баз даних і складається з наступних стовпців.

Стовпці області дії. MySQL використовує їх для того, щоб визначити, коли застосовний той або інший рядок привілеїв. Ці стовпці наступні:

Host
Db
User

Стовпці привілеїв. Вони вказують, чи буде дана комбінація Host, Db й User наділена відповідним привілеєм. Можуть містити значення Y або N. Ці стовпці наступні:

Select_priv	Create_priv
Insert_priv	Drop_priv
Update_priv	Grantjpriv

Delete_priv	Create_tmp_table_priv
Index_priv	Lock_tables_priv
Alter_priv	

Таблиця host

MySQL звертається до таблиці host, коли потрібно знайти ім'я вільного хосту в таблиці db. Цього не можна зробити за допомогою оператора GRANT, але відповідне значення можна встановити вручну. Таблиця складається з наступних стовпців.

Стовпці області дії. MySQL використовує їх для того, щоб визначити, коли застосовний той або інший рядок привілеїв. Тут кожен рядок містить інформацію про одну базу даних і про ім'я хосту, з якого здійснюється доступ до неї. Стовпці, про які мова йде, наведені нижче.

Host

Db

Стовпці привілеїв. Вони вказують, чи буде дана комбінація Host й Db наділена відповідним привілеєм, і можуть містити значення Y або N. Ці стовпці наступні:

Select_priv	Create_priv
Insert_priv	Drop_priv
Update_priv	Grant_priv
Delete_priv	Create_tmp_table_priv
Index_priv	Lock_tables_priv
Alter_priv	

Таблиця tables_priv

У таблиці tables_priv відображається інформація про користувальницькі привілеї, що стосуються окремих таблиць. Вона складається з наступних стовпців.

Стовпці області дії. Визначають, коли застосовна той або інший рядок привілеїв. Ці стовпці наступні.

Host

Db

User

Table_name,

Column_name

Стовпець Column_priv. Визначає, якими привілеями наділений об'єкт, заданий стовпцями області дії. Може містити наступні значення: Select, Insert, Update й References.

Стовпець Column_priv. Визначає, якими привілеями наділений користувач відносно всіх стовпців даної таблиці. Може містити наступні значення: Select, Insert, Update й References. Якщо яка-небудь із привілеїв тут відсутня, MySQL може звернутися до таблиці columns_priv за більше докладною інформацією про те, що дозволено що не дозволено робити із зазначеними стовпцями таблиці.

Таблиця columns_priv

У таблиці columns_priv відображається інформація про користувальницькі привілеї, що стосуються окремих стовпців. Вона складається з наступних стовпців.

Стовпці області дії. Аналогічні стовпцям таблиць, про які говорилося вище, але є додатковий стовпець Table_name, що вказує ім'я таблиці, до якої ставиться надаваний привілей.

Ці стовпці наступні:

Host

Db

User

Table_name

Стовпці донора. Містять інформацію про те, хто й коли надав привілеї. Ці стовпці наступні.

Grantor

Timestamp

Стовпець Table_priv. Визначає, якими привілеями наділений Host/Db/User відносно таблиці, зазначеної в Table_name, Може містити наступні значення: Select, Insert, Update, Delete, Create, Drop, Grant, References, Index й Alter.

Стовпець Timestamp. Повідомляє про те, коли була надана привілея.

Питання та завдання для самоконтролю знань студентів

1 Привілей GRANT OPTION дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) оновляти привілею.

2 Привілей USAGE дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) оновляти привілею.

3 Привілей RELOAD дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) оновляти привілею.

4 Привілей FILE дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) оновляти привілею.

5 Значення стовпця tables_priv, table_priv

- а) містить список привілеїв користувача для даної таблиці;
- б) містить значення Y або N, залежно від того, чи дозволений користувачеві доступ до даної таблиці;
- в) указує окремий привілей, який наділений користувач у відносині даної таблиці;
- г) указує, чи існує в таблиці columns_priv елемент, що відповідає даній таблиці.

6 Створіть оператор GRANT, що створює обліковий запис для користувача підім'ям bill з паролем secret, що повинен мати право вибирати, модифікувати, додавати й видаляти дані з таблиці department.

7 Створіть оператор REVOKE, що скасовує надані привілеї для зазначеного користувача.

Розробив: Ланська С.С.

Розглянуто та схвалено
на засіданні предметної (циклової)
комісії програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

План самостійного вивчення теми № 5

з навчальної дисципліни «Бази даних»

Тема	Захист MySQL
Мета	Огляд основних методів захисту даних засобами MySQL

знати:

- систему перевірки привілеїв;
- як захистити облікові записи;
- як захистити файли системи.

вміти:

- створювати облікові записи користувачів;
- встановлювати користувальницькі привілеї.

Час – 4 години

Навчальні питання:

- 1 Робота системи привілеїв на практиці.
- 2 Захист облікових записів
- 3 Потенційно небезпечні привілеї
- 4 Фільтрація даних користувача

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше Система перевірки привілеїв проходить дві стадії. На першій стадії перевіряється, чи дозволено користувачеві з'єднуватися із сервером взагалі. Друга стадія починається при спробі користувача виконати конкретну команду або запит.

По-друге MySQL має дуже об'ємну систему привілеїв. Надаючи своїм користувачам деякі із цих привілеїв, варто бути дуже обережними. Це стосується, насамперед, таких привілеїв, як FILE, PROCESS й WITH GRANT OPTION

По-третє Паролі користувачів в MySQL шифруються

Література

1Малыхина, М.П. Базыданных: основы, проектирование, использование [Текст] / М.П. Малыхина – СПб.: БХВ– Петербург, 2004, – 512с.: ил.

2Карпова, Т.С., Базыданных: модели, разработки, реализации [Текст]: / Т.С. Карпова – СПб.: Питер, 2001. – 303с.

Навчальні матеріали до самостійного вивчення теми

1 Робота системи привілеїв на практиці.

Ми починаємо з обговорення того, яким образом сервер MySQL застосовує привілею, надані користувачам.

Система перевірки привілеїв проходить дві стадії. На першій стадії перевіряється, чи дозволено користувачеві з'єднуватися із сервером взагалі. Для цього використовується таблиця `user` у базі даних `mysql`. MySQL бере ім'я користувача, уведений їм пароль й ім'я хосту, з якого користувач намагається з'єднатися із сервером, і з'ясовує, чи є в таблиці відповідний рядок. Якщо відповідного рядка не виявиться, користувач не одержить дозволи з'єднатися із сервером.

Оскільки таблиця `user` підтримує використання групових символів у стовпці `host`, однієї комбінації `user/hostname` може відповідати кілька рядків. Щоб ухвалити рішення щодо того, який з рядків є підходящим, MySQL розглядає спочатку конкретне ім'я хосту. Наприклад, якщо в таблиці присутні рядки для користувача `test` з хосту `localhost` і користувача `test` з хоста `%` (означаючого будь-який хост), то буде обраний рядок з ім'ям `localhost`. Варто взяти до уваги й те, що ці два рядки можуть містити різні паролі. На практиці це може виявитися причиною більших непорозумінь.

Друга стадія починається при спробі користувача виконати конкретну команду або запит. Перед виконанням кожного запиту перевіряється його допустимість по таблицях привілеїв.

Якщо запит, що ви намагаєтеся виконати, вимагає привілеїв глобального рівня - як, наприклад, виконання `LOAD DATA INFILE` або використання `SHOW PROCESSLIST`, -буде перевірена таблиця `user`. Для запитів до конкретної бази даних спочатку перевіряється таблиця `user`. Якщо користувач має право доступу до всіх

баз даних, цього буде досить. Якщо ні, то додатково перевіряються таблиці db й host. Якщо користувач не має відповідних привілеїв на цьому рівні, то перевіряються привілеї рівня таблиць і стовпців, якщо такі були встановлені.

2 Захист облікових записів

Існує кілька загальних принципів безпеки, що відносяться до керування' обліковими записами користувачів в MySQL. Ці принципи ми зараз і розглянемо.

Установка пароля для кореневого облікового запису

При установці MySQL кореневий пароль за замовчуванням не встановлюється. Перед тим як використати MySQL, абсолютно необхідно встановити цей пароль (цю вимога можна ігнорувати тільки в експериментальному оточенні). Без встановленого кореневого пароля хто завгодно може увійти в систему й робити з вашими даними все, що заманеться. Практично завжди це небажано. Якщо ви ще не встановили кореневий пароль, зробіть це негайно.

Видалення анонімних облікових записів

При установці MySQL в Windows установник автоматично створює деякі облікові записи. В Linux це відбувається при виконанні сценарію mysql_install_db. Ці два облікових записи будуть анонімними - вони являють собою облікові записи, що створюються тоді, коли не вказується ім'я користувача. Один з облікових записів зі значення хосту localhost, а інший - % (тобто будь-який хост, що фактично дозволяє будь-яке вилучене з'єднання). Для цих облікових записів за замовчуванням не встановлено ніякого пароля.

Ви, імовірно, вже розумієте, до чого це може привести, і ми настійно рекомендуємо вам видалити ці облікові записи. Це можна зробити, наприклад, так:

```
delete from user where User=";
```

```
delete from db where User=";
```

Після цього варто застосувати оператор FLUSH PRIVILEGES, щоб оновити таблиці привілеїв.

Другою причиною необхідності видалення анонімних облікових записів є те, що вони можуть створювати проблеми при спробах звичайних користувачів увійти в систему. Якщо ви створите, наприклад, обліковий запис користувача з ім'ям laura для доступу з будь-якого хосту (%), то коли laura спробує увійти в систему з хосту

localhost, у результаті пошуку в таблиці user сервер MySQL знайде laura@% й (anonymous) @localhost. За правилами MySQL спочатку буде перевірений рядок з конкретним ім'ям хосту, тобто (anonymous) @localhost. Зверніть увагу на те, що хоча користувачем laura ім'я користувача було зазначено, у цьому випадку це не має ніякого значення! Анонімні облікові записи не вимагають вказівки імені користувача. Цілком імовірно, що пароль для цього анонімного облікового запису не буде збігатися з паролем з laura (за замовчуванням пароль не встановлений, і це значить, що користувач не повинен його вводити). Тому, коли laura спробує увійти в систему зі своїм ім'ям користувача й паролем з хоста localhost, вона одержить у відповідь Access Denied (доступ заборонений) без якої б те не було видимої причини.

Щоб уникнути виникнення такого роду проблем, найпростіше видалити анонімні облікові записи й забути про їх.

3 Потенційно небезпечні привілеї

MySQL має дуже об'ємну систему привілеїв. Надаючи своїм користувачам деякі із цих привілеїв, варто бути дуже обережними. Це стосується, насамперед, таких привілеїв, як FILE, PROCESS й WITH GRANT OPTION.

Привілей FILE дозволяє користувачеві використати команду LOAD DATA INFILE. Її можна використати для завантаження файлів із сервера (наприклад, файлу пароля /etc/passwd) або навіть файлів дані бази даних, фактично в обхід системи привілеїв. Привілей PROCESS дозволяє користувачеві використати SHOW PROCESSLIST. Це відкриває інформацію про всі виконувані запити, у результаті чого конфіденційна інформація одного користувача може стати доступною іншому.

Привілей GRANT OPT ION дозволяє користувачеві поділитися своїми привілеями з іншими користувачами. Знаючи про цьому й усвідомлюючи наслідку, ви повинні надавати таке право обережно.

Паролі й шифрування

Паролі користувачів в MySQL шифруються. До версії 4.1 можна було запам'ятати шифрований пароль у програмі входу в систему, щоб використати його при реєстрації користувача надалі. Зараз і пароль, і механізм входу в систему стали більше захищеними.

Якщо створений вами додаток припускає збереження імен користувачів (не MySQL) і паролів, рекомендується використати для їхнього шифрування не функцію PASSWORD(), а яку-небудь іншу. Ми рекомендуємо використати функцію MD5() або ENCRYPT().

Захист файлів системи

Крім захисту облікових записів MySQL, необхідно здійснювати контроль доступу до виконують файлам, що, і сценаріям MySQL, а також до файлів даних. Нижче ми пропонуємо деякі дотичні цього рекомендації.

Не запускайте mysqld від імені кореневого користувача

Це рекомендація стосується Linux й інших Unix-подібних операційних систем. Не піддавайтеся спокусі запустити сервер MySQL (mysqld) від імені кореневого користувача. Точно так само, як й у випадку з Web-сервером, створіть спеціальний обліковий запис для запуску сервера MySQL. У такий спосіб ви зможете обмежити права доступу, які сервер MySQL може мати стосовно файлової системи.

Доступ і привілеї в операційній системі

Немає рації витрачати час на правильне оформлення облікових записів користувачів в MySQL, якщо немає можливості управляти доступом до файлів у самій операційній системі. Необхідно управляти доступом користувачів до виконують файлам, Що, MySQL, сценаріям й, зокрема, каталогам даних, Типовим джерелом проблем захисти є не бази даних інших користувачів, а самі користувачі, що мають дозвіл на доступ до машини, на якій розміщений сервер MySQL. Якщо ці користувачі можуть одержати доступ до каталогу даних, вони можуть скопіювати файли даних і завантажити їх, використовуючи інший сервер MySQL.

Загалом кажучи, бажано забезпечити наступні міри безпеки.

- 1 Тільки певні користувачі повинні мати право запускати mysqld.
- 2 Цілком розумно надати таке право тільки користувачеві, обліковий запис якого ви створили спеціально для запуску mysqld.
- 3 Тільки певні користувачі повинні мати доступ до пов'язаним з MySQL програмам і сценаріям, таким як mysqladmin, mysqldump або mysqlhotcopy. Можливо, краще визначити це право на програмному рівні.

4 Тільки певні користувачі повинні мати доступ до каталогів даних MySQL. Якщо сервер mysql запущений від імені деякого користувача, даному користувачеві буде потрібно доступ до цього каталогу. Іншим користувачам такий доступ не обов'язковий, і тому в загальному випадку його краще заборонити.

4 Фільтрація даних користувача

Перш ніж пересилати введені користувачем дані MySQL, необхідно перевірити, немає чи в них помилок на рівні додатка. Те, як саме це робиться, залежить від використовуваної вами платформи розробки, але спочатку розглянемо приклад, що пояснює, чому необхідно виконати таку перевірку.

Породжувати проблеми можуть самі необразливі дії: наприклад, введення користувачем свого імені - Патрик О'генри - у ваш додаток. Якщо ви передасте ці дані прямо в MySQL, через апостроф у прізвищі О'генри виникнуть проблеми. У найгіршому випадку користувачі можуть увести команди MySQL в інтерфейс вашого додатка або в Web-форми. Дії, які вам доведеться почати для перевірки даних, будуть залежати від використовуваного вами мови програмування, але деякі загальні вказівки із цього приводу можна знайти в посібнику з MySQL - вони годяться для більшості мов.

Інші рекомендації захисту

Ми розглянули систему привілеїв й облікові записи користувачів, обговорили доступ до файлів даних у рамках файлової системи й коротко згадали про фільтрації даних. Якщо ви піклуєтеся про безпеку мережних з'єднань (без яких, швидше за все, не обійтися при наявності зовнішніх зв'язків), то MySQL надає можливість шифрувати передачу даних засобами SSL. Не слід зневажати також питаннями фізичного захисту.

Використання з'єднань SSL

Якщо ви не хочете, щоб зломщики мали можливість посипати пакети, що аналізують мережу, у сегмент мережі між сервером MySQL і клієнтами, можна настроїти MySQL на використання захищених з'єднань. Це значить, що весь обмін даними між клієнтом і сервером буде шифруватися за допомогою засобів SSL (Secure Sockets Layer - протокол захищених сокетів).

Для використання SSL потрібно встановити бібліотеку OpenSSL (вона доступна за адресою www.openssl.org), запустити сервер з опціями `-with-openssl` й `-with-ssl`, а також виконати деякі установки в командному рядку. Гарний приклад сценарію, що вам належить при цьому виконати, можна знайти в посібнику з MySQL - тут ми його не відтворюємо.

Після цього можна додати в оператори GRANT відповідне обмеження, зажадавши, щоб для зв'язку використалися з'єднання SSL або був відповідний сертифікат. Наприклад, можна використати наступний оператор GRANT:

```
grant all on employee.*  
to testuser identified by 'password'  
require ssl;
```

У результаті буде створено (або модифіковано) обліковий запис для testuser, що призначає користувачеві пароль password. Відповідний користувач зможе з'єднатися із сервером тільки через SSL. Можна зажадати, щоб або всіх ваших користувачах з'єднувалися із сервером саме так, або тільки ті, які запитують доступ з хостів, відмінних від localhost.

Фізичний захист системи

Раз вже ви збираєтеся подбати про правильну структуру облікових записів користувачів в MySQL й операційній системі, а можливо й про те, щоб користувачі з'єднувалися через SSL, ви повинні бути безпосередньо зацікавлені в тому, щоб забезпечити фізичний захист вашої системи. Якщо хтось сторонній може відключити ваш сервер, висмикнувши кабель харчування з розетки, або викрасти дані, просто привласнивши сервер разом з усіма даними, ви, мабуть, не захищені від проблем. Про фізичний захист часто забувають, особливо в невеликих компаніях. Це може й не дивно для Windows, але навіть такі захищені системи, як Unix/Linux, виявляються уразливими, коли не забезпечена фізичний захист. Наприклад, дуже просто змінити кореневий пароль машини в Linux, якщо користувач має фізичний доступ до машини. Ясно, що з можливістю кореневого доступу всі дані у ваших базах даних MySQL можуть бути скомпрометовані

Питання та завдання для самоконтролю знань студентів

1 Яка з таблиць бази даних mysql використовується для перевірки допустимості з'єднання користувача?

- а) tables_priv;
- б) db;
- в) columns_priv;
- г) user.

2 Яка з таблиць перевіряється першою при визначенні права користувача виконати конкретний запит?

- а) user;
- б) host;
- в) db;
- г) tables_priv.

3 Якщо MySQL виявить у таблиці user кілька рядків з одним ім'ям користувача, то яка із цих рядків використовується для аутентифікації?

- а) рядок з найбільш конкретним ім'ям хоста;
- б) рядок з найбільш загальним ім'ям хоста;
- в) будь-який рядок з відповідним паролем;
- г) жодна з вищевказаних відповідей.

4 Які з наступних привілеїв небажано надавати користувачам?

- а) FILE;
- б) PROCESS;
- в) WITH GRANT OPTION;
- г) всі вищевказані.

5 Яким повинен бути користувач, що запускає mysqld?

- а) користувачем з низьким рівнем привілеїв;
- б) кореневим користувачем;
- в) або пункт а), або б);
- г) ні пункт а), ні б).

6 На основі інформації із прикладу установки, наданого посібником з MySQL, установите OpenSSL для використання із сервером MySQL.

Розробив: Ланська С.С.

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

План самостійного вивчення теми № 6

з навчальної дисципліни «Бази даних»

Тема Резервування й відновлення даних

Мета Вміти використовувати основні команди MySQL по резервуванню та відновленню даних

знати:

- резервування й відновлення за допомогою mysqldump, mysqlhotcopy;
- резервування й відновлення вручну;
- резервування й відновлення за допомогою BACKUP TABLE й RESTORE TABLE;
- відновлення за допомогою журналу двійкової реєстрації;
- перевірка й відновлення таблиць за допомогою CHECK й REPAIR, myisamchk, mysqlcheck.

Час – 8 годин

Навчальні питання

- 1 Резервування й відновлення за допомогою mysqldump
- 2 Резервування й відновлення за допомогою mysqlhotcopy
- 3 Резервування й відновлення вручну
- 4 Резервування й відновлення за допомогою BACKUP TABLE й RESTORE TABLE
- 5 Відновлення за допомогою журналу двійкової реєстрації
- 6 Перевірка резервної копії
- 7 Перевірка й відновлення таблиць
- 8 Перевірка й відновлення за допомогою CHECK й REPAIR
- 9 Перевірка й відновлення за допомогою myisamchk
- 10 Перевірка й відновлення за допомогою mysqlcheck

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше

Резервування:

- Сценарій `mysqldump` створює файл дампу операторів SQL.
- Сценарій `mysqlhotcopy` копіює файли даних у зазначене місце.
- Команда `BACKUP TABLE` копіює файл дані таблиці в зазначене місце.
- Можна виконати резервне копіювання даних вручну, спочатку зберігши даного буфера таблиць на диску й заблокувавши таблиці, а потім безпосередньо скопіювавши відповідні файли.

По-друге

Відновлення:

- Завантаження файлів дампу, отриманих за допомогою сценарію `mysqldump`.
- Завантаження файлів копій даних, отриманих за допомогою сценарію `mysqlhotcopy` або резервного копіювання вручну.
- Відновлення за допомогою команди `RESTORE TABLE` даних, отриманих за допомогою команди `backup table`.
- Повторне виконання операцій з журналу двійкової реєстрації, починаючи з моменту останнього резервування.

По-третьє

Перевірка й відновлення таблиць:

- Перевірка таблиць за допомогою `CHECK TABLE`, `myisamchk`, `isamchk` або `mysqlcheck`.
- Відновлення таблиць за допомогою `REPAIR TABLE`, `myisamchk`, `isamchk` або `mysqlcheck`.
- Не слід застосовувати програму `myisamchk` при працюючому сервері.

Література

1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.: ил.

2 Файли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Файли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464 с.: ил.

Навчальні матеріали до самостійного вивчення теми

Очевидно, як й у випадку будь-яких інших файлів, ви повинні мати резервні копії своїх баз даних. Можна також зробити копію бази даних з метою реплікації або для того, щоб перемістити її на іншу машину.

В MySQL є кілька варіантів резервування баз даних.

1 Застосування сценарію `mysqldump` для створення дампу, тобто файлу, що містять оператори SQL, необхідні для відтворення бази даних.

2 Застосування сценарію `mysqlhotcopy` для створення файлу даних. Цей сценарій безпосередньо копіює файли даних конкретної бази даних.

3 Безпосереднє створення резервної копії файлів даних вручну. Це фактично еквівалентно застосуванню `mysqlhotcopy`, але в цьому випадку все робиться вручну. Якщо ви зволієте використати цей варіант, необхідно перед копіюванням припинити роботу бази даних або оновити й блокувати всі таблиці, щоб забезпечити їхню внутрішню погодженість. І `mysqldump`, і `mysqlhotcopy` оновлять і блокують таблиці за вас, тому застосування зазначених сценаріїв є більше простим і безпечним.

4 Використання команд `BACKUP TABLE` й `RESTORE TABLE` для резервування або відновлення конкретної таблиці або безлічі таблиць.

Ми розглянемо кожний із цих варіантів по черзі.

Не забувайте також про те, що хоча резервування і є життєво важливим для нормальної роботи бази даних, на час виконання резервування доступ користувачів до бази даних обмежується. Навіщо? Щоб одержати погоджений "знімок" бази даних, таблиці повинні бути оновлені й під час копіювання повинні залишатися в незмінному стані. Цього можна домогтися, заблокувавши таблиці (як це й буває в більшості випадків) або виключивши сервер (чого робити не рекомендується), але в

кожному разі на час виконання резервування доводиться зменшити здатність бази даних до реагування на запити.

Одним з рішень цієї проблеми є використання реплікації. Ви одержуєте можливість відключити один компонент системи й виконати резервне копіювання, у той час як користувачі продовжують щасливо робити свою справу.

1 Резервування й відновлення за допомогою mysqldump

Простіше й зручніше за все виконати резервування бази даних за допомогою сценарію mysqldump, запустивши його з командного рядка. Цей сценарій забезпечує з'єднання із сервером MySQL і створення файлу дампу SQL. Файл дампу містить оператори SQL, необхідні для відтворення відповідної бази даних.

Нижче наведений типовий приклад використання цього сценарію.

```
mysqldump --opt -u ім'я_користувача -p пароль  
employee>backup.sql
```

Тут використовується опція -opt, що інкапсулює кілька інших опцій. Ми також указали ім'я бази даних і перенаправляли виведення у резервний файл, що хотіли б використати.

У результаті із зазначеного сценарію до простої бази даних employee створюється файл виведення, подібний показаному в лістингу 1.1.

Лістинг 1.1. Приклад виведення сценарію mysqldump

```
-- MySQLdump 10.2  
-- Host: localhostDatabase: employee  
-- Server version 4.1.0-alpha-max-debug  
--  
-- Tablestructurefortable 'assignment'  
--  
DROP TABLE IF EXISTS assignment;  
CREATE TABLE assignment (  
  clientIDint(11) NOT NULL default '0',  
  employeeIDint(11) NOT NULL default '0',  
  workdateNOT NULL default '0000-00-00',  
  hoursfloatdefault NULL,  
  PRIMARY KEY (clientID,employeeID,workdate)  
) TYPE=InnoDB CHARSET=latin1;  
--  
---- Dumpingdatafortable 'assignment'  
--  
/* 140000 ALTER TABLE assignment DISABLE KEYS */;  
LOCK TABLES assignment WRITE;
```

```

INSERT INTO assignment VALUES
(1,7513, '0000-00-00',5), (1,7513, '2003-01-20',8.5);
UNLOCK TABLES;
/* 140000 ALTER TABLE assignment ENABLE KEYS */;
--
---- Tablestructurefortable 'client'
--
DROP TABLE IF EXISTS client;
CREATE TABLE client (
clientI 'int(11) NOT NULL auto_increment,
namevarchar(40) default NULL,
addressvarchar(100) default NULL,
contactPersonvarchar(80) default NULL,
contactNumbervarchar(12) default NULL,
PRIMARY KEY (clientID)
) TYPE=InnoDB CHARSET=latin1;
--
---- Dumpingdatafortable 'client'
--
/* 140000 ALTER TABLE client DISABLE KEYS */;
LOCK TABLES client WRITE;
INSERT INTO client
VALUES
(1,'Telco Inc','1 CollinsSt
Melbournel','Fred Smith1','95551234'),
(2,'The Bank1','100 BourkeSt
Melbournel','Jan Tristan','95559876');
UNLOCK TABLES;
/*!40000 ALTER TABLE client ENABLE KEYS */;
--
---- Tablestructurefortable 'department'
--
DROP TABLE IF EXISTS department;
CREATE TABLE department (
depar"trnentIDint(11) NOT NULL auto_increment,
namevarchar(30) default NULL,
PRIMARY KEY (departmentID)
) TYPE=InnoDB CHARSET=latin1;
--
---- Dumpingdatafortable 'department1'
--
/*!40000 ALTER TABLE department DISABLE KEYS */;
LOCK TABLES department WRITE;
INSERT INTO department VALUES
(42,'Finance'),
(128,'Research andDevelopment'),
(129,'Human Resources'),
(130,'Marketing'),
(131,'Property Services');
UNLOCK TABLES;

```

```

/*!400DO ALTER TABLE department ENABLE KEYS */;
--
---- Tablestructurefortable 'employee'
--
DROP TABLE IF EXISTS employee;
CREATE TABLE employee (
employeeIDint(11) NOT NULL auto_increment,
namevarchar(80) default NULL,
jobvarchar<30) default NULL,
departmentIDint(11) NOT NULL default '0',
PRIMARY KEY femployeeID)
) TYPE=InnoDB CHARSET=latin1;
--
---- Dumpingdatafortable 'employee'
--
/*!40000 ALTER TABLE employee DISABLE KEYS */;
LOCK TABLES employee WRITE;
INSERT INTO employee
VALUES
(6651,'Ajay Patel','Programmer'/128),
(7513,'Nora Edwards','Programme^",128),
(9006,'Candy Burnett','Systems Administrator',128)
(9842, 'Ben Smith1','DBA1, 42),
(9843,'Fred Smith','DBA1,131);
UNLOCK TABLES;
/*!40000 ALTER TABLE employee ENABLE KEYS */;
--
---- Tablestructurefortable 'employeeSkills'
--
DROP TABLE IF EXISTS employeeSkills;
CREATE TABLEemployeeSkills (
employeeIDint(11) NOT NULL default '0',
skillvarchar(15) NOT NULL default "",
PRIMARY KEY (employeeID,skill)
) TYPE=InnoDB CHARSET=latin1;
--
---- Dumpingdatafortable 'employeeSkills'
--
/* !40000 ALTER TABLE employeeSkills DISABLE KEYS */;
LOCK TABLES employeeSkills WRITE;
INSERT INTO employeeSkills
VALUES
(6651,'Java'),
(6651,1VB'),
(7513,'C'),
(•7513, 'Java'),
(7513, 'Perl'),
(9006,'Linux'),
(9006,'NT'),
(9842,'DB2');

```

```
UNLOCK TABLES;
```

```
/*140000 ALTER TABLE employeeSkills ENABLE KEYS V;
```

Тепер можна перезавантажити або відтворити базу даних employee, виконавши наступні дії.

1 Створити базу даних з підходящим ім'ям на потрібній машині.

2 Завантажити файл копії за допомогою команди

```
mysql -u ім'я_користувача -p <backup.sql
```

Сценарій mysqldump має безліч опцій. Тут ми використали опцію -iort, що можуть супроводжувати наступні параметри.

--iquick. Змушує MySQL направити дані дампу відразу у файл, міняючи буфер пам'яті (що передбачається за замовчуванням). У результаті процес прискориться.

--add-drop-table. Дає MySQL вказівка додавати оператор DROP TABLE перед кожним оператором CREATE TABLE у дампі (див. лістинг 1.1),

--add-locks. Додають оператори LOCK TABLES й UNLOCK TABLES (їх ви теж можете побачити у файлі дампу).

--extend-insert. Змушує MySQL використати багаторядковий синтаксис вставки для додавання безлічі рядків за допомогою одного оператора INSERT. Так, у нашому лістингу відповідний оператор виглядає так:

```
INSERT INTO employeeSkills  
VALUES  
(6651, 'Java'),  
(6651, 'VB'),  
(7513, 'C'),
```

Якщо буде потрібно відтворити базу даних з резервної копії, такий оператор буде виконуватися швидше, ніж відповідна послідовність окремих операторів INSERT.

--lock-tables. Змушує MySQL блокувати всі таблиці, перш ніж почати створення дампу.

Зверніть увагу на те, що опція -opt (означаюча optimized) буде оптимізувати час, необхідний для наступного завантаження дампу, а не час, витрачений на створення дампу. Процес створення файлу дампу може виявитися довгим.

От ще кілька корисних опцій.

--databases. Дозволяє вказати не одну, а кілька баз даних для дампа.

--all-databases. Дає MySQL вказівка створити дамп всіх наявних баз даних.

`--allow-keywords`. Якщо є імена полів, які є ключовими словами MySQL (або можуть стати такими в майбутньому), ця опція жадає від MySQL кваліфікувати (тобто супроводжувати) ім'я кожного стовпця ім'ям таблиці.

`-d` або `--no-data`. Створювати дампи тільки структури бази даних, але не її вмісту. Це може виявитися корисним при розміщенні й тестуванні бази даних на різних машинах.

Перевагами використання `mysqldump` є простота й автоматичне блокування таблиць.

Але є й два недоліки. По-перше, цей сценарій блокує таблиці: запуск цього сценарію на сервері блокує доступ користувачів на кілька секунд (або хвилин, залежно від розміру таблиць). Якщо вам буде потрібно створити дампи на окремому не має реплікації сервері, бажано робити це не під час пікових навантажень, щоб не викликати роздратування користувачів. Якщо ви зберігаєте безліч даних і маєте багато користувачів у будь-який час, необхідно вибрати інший спосіб резервування.

По-друге, оскільки сценарій `mysqldump` працює через сервер MySQL, те буде виконуватися повільніше, ніж `mysqlhotcopy`. Сценарій `mysqlhotcopy` використовує сервер MySQL не занадто активно, а виконує значну частину своєї роботи, звертаючись безпосередньо до файлової системи.

2 Резервування й відновлення за допомогою `mysqlhotcopy`

Сценарій `mysqlhotcopy` відрізняється від `mysqldump` тим, що копіює безпосередньо файли дані бази даних, а не дані, необхідні для відновлення, за допомогою з'єднання із сервером. При цьому зв'язок із сервером однаково потрібно, щоб оновити й блокувати таблиці бази даних, але оскільки даний сценарій в основному використовує операції з файловою системою, а не запити до бази даних, він повинен у цілому виконуватися трохи швидше, ніж сценарій `mysqldump`.

Даний сценарій можна використати так, як показано нижче.

```
mysqlhotcopy -u ім'я_користувача ~p ім'яБДрозміщення_копії
```

Цей сценарій є сценарієм Perl. Якщо ви використаєте Unix або подібну їй операційну систему, у вас майже напевно десь уже є виконуваний файл `perl`. В Windows, щоб використати Perl, вам потрібно встановити його.

Файли, створювані `mysqlhotcopy`, - це точні копії файлів даних бази даних. Щоб використовувати ці копії, необхідно зупинити роботу сервера MySQL і замінити файли даних у каталозі даних MySQL відповідними файлами-копіями.

3 Резервування й відновлення вручну

Замість використання `mysqlhotcopy` можна зробити те ж саме вручну. Це припускає відновлення даних (запис на диск умісту файлових буферів) і блокування таблиць, а потім копіювання файлів даних у резервне місце (при цьому таблиці повинні залишатися заблокованими).

Це значить, що спочатку прийде відкрити сеанс MySQL. Потім можна ввести команду `LOCK TABLES`, щоб блокувати всі таблиці, які планується резервувати:

```
locktables  
employeeeeread,  
departmentread,  
clientread,  
assignmentread,  
employeeSkillsread;
```

Оператор `LOCK TABLES` як параметри одержує список імен таблиць і тип блокування, якому необхідно застосувати: `READ` або `WRITE`. Для резервування блокування для читання звичайно буває досить. Це означає, що інші потоки (об'єкти, що зв'язуються) можуть продовжувати читати дані таблиць, але не зможуть записувати в них дані, поки резервування не буде завершено.

Блокування в таких ситуаціях дуже важливі, оскільки копіювання може зайняти досить багато часу. У нашому випадку буде катастрофою, якщо, наприклад, після копіювання таблиці `employeee`, але ще перед копіюванням таблиці `department`, хтось видалить інформацію про всіх службовців якогось відділу й інформацію про самий відділ. Отримана в результаті копія буде неузгодженою, оскільки буде містити інформацію про службовців неіснуючого відділу.

Потім варто застосувати команду `FLUSH TABLES`:

```
flushtables;
```

Якщо необхідно резервувати всі бази даних, можна сполучити ці два кроки, використовуючи команду

```
flushableswithreadlock;
```

Тепер можна копіювати файли даних. Дуже важливо, щоб сеанс зв'язку (у ході якого ви оновили й блокували таблиці) залишався відкритим, поки виконується

копіювання. Це забезпечить збереження блокувань. У результаті завершення сеансу таблиці будуть розблоковані

По завершенні копіювання файлів треба, звичайно, розблокувати таблиці:
`unlocktables;`

Ця процедура аналогічна сценарію `rnysqlhotcopy`, і точно так само ви можете відновити базу даних.

4 Резервування й відновлення за допомогою `BACKUP TABLE` й `RESTORE TABLE`

Альтернативу тільки що розглянутим варіантам становлять два оператори SQL, які можна використати для одержання тих же результатів. Це - оператори `BACKUP TABLE` й `RESTORE TABLE`. Відповідні команди працюють тільки з таблицями MyISAM.

Ви можете створити резервну копію таблиці MyISAM за допомогою команди
`backuptabletltto 'шлях/до/копії';`

Зверніть увагу на те, що в Windows необхідно також указати букву дисководу, наприклад,

```
backuptabletltto 'c:/шлях/до/копії1;
```

У результаті файли, що представляють зазначену таблицю MyISAM, будуть скопійовані в зазначене місце. Таблиця буде заблокована для читання, поки буде виконуватися копіювання.

Можна також указати список таблиць, розділений комами, однак кожна з таких таблиць буде блокуватися й копіюватися окремо, по черзі. Якщо необхідно з погоджений набір таблиць, варто спочатку застосувати оператор `LOCK TABLES` (про те, як це зробити, говорилося в попередньому розділі, "Резервування й відновлення вручну").

Щоб відновити дані за допомогою копії, уведіть

```
restoretabletlfrotn 'c:/tmp';
```

Це спрацює тільки в тому випадку, якщо відновлюваних таблиць у поточній базі даних не існує. Якщо вже є таблиця з відповідним ім'ям, перед використанням оператора `RESTORE` варто застосувати команду `DROP TABLE`.

І підкреслимо ще раз, `RESTORE` працює тільки з таблицями MyISAM.

5 Відновлення за допомогою журналу двійкової реєстрації

Як правило, при відновленні даних за допомогою резервної копії із часу створення копії вже бувають виконані якісь нові модифікації даних таблиць. Базу даних можна спочатку відновити за допомогою однієї із процедур відновлення, описаних у попередніх розділах, а потім виконати повторно всі операції відновлення даних, зроблені із часу резервування.

Всі зміни запам'ятовуються в журналі двійкової реєстрації або журналі відновлень. От чому журнал двійкової реєстрації так важливий. Ви можете витягти список виконаних операцій з журналу двійкової реєстрації за допомогою команди

```
mysqlbinloglogfile>updates.sql
```

Досить бажано глянути на цей файл перед тим, як повторно запускати відповідні запити, - цілком можливо, що якісь із них ви повторювати не побажаєте. Можливо, що якийсь погано продуманий запит SQL і привів до того, що вам довелося звернутися до резервної копії.

Наприклад, один раз серед безлічі рядків ми виявили, що хтось із програмістів увів

```
updateusersetpassword='password';
```

Природно, що при відновленні таблиці ми не побажали знову вводити цей запит і встановлювати для всіх користувачів нашої системи пароль

```
password!
```

6 Перевірка резервної копії

Який би метод резервування ви не вибрали, дуже важливо з отриману копію або, точніше, можливість відновлення даних. Нерідко зустрічаються адміністратори, які сумлінно й регулярно створюють резервні копії, але ніколи не перевіряють, чи зможуть. вони якщо буде потреба відновити за допомогою цих копій дані.

Процедура резервування повинна займати досить серйозне місце при аналізі ризиків і прийнятті відповідних рішень. Куди помістити копії, щоб вони перебували на іншому фізичному диску? Як забезпечити безпечне зберігання копії? Прийнявши правильне рішення й спланувавши графік резервування, ви не будете турбуватися про працездатність прийнятої схеми. Якщо ви перевірили можливість відновлення даних за допомогою резервної копії, ви зможете виявити проблеми до того, як вони стануть критичними.

Дуже важливою складовою системи MySQL у цілому й процесу резервування зокрема є журнал двійкової реєстрації. За замовчуванням запису нього не виконується, але він виразно необхідний для того, щоб урахувати у відновлюваній базі даних самі останні зміни.

7 Перевірка й відновлення таблиць

Перевірка таблиць на їхню цілісність є частиною системи профілактики таблиць, а також частиною процедури відновлення даних у випадку непередбачених обставин, наприклад, у випадку відмови системи електроживлення.

MySQL пропонує три способи перевірки таблиць: за допомогою CHECK TABLE, myisamchk (або isamchk) або mysqlcheck. Потім можна усунути виявлені в таблицях проблеми за допомогою REPAIR TABLE або за допомогою myisamchk (або isamchk), або mysqlcheck.

Існує ряд факторів, які варто враховувати при виборі кожного із зазначених варіантів. Команди CHECK й REPAIR можна використати в рамках MySQL, тоді як інші повинні застосовуватися з командного рядка. Команди CHECK й REPAIR працюють як з таблицями MyISAM, так і з таблицями InnoDB. Сценарій isamchk застосуємо до таблиць ISAM, тоді як myisamchk й mysqlcheck можуть використатися з таблицями MyISAM.

Не можна застосовувати myisamchk або isamchk до таблиць, які в той момент використовуються. Найкраще перед використанням цих сценаріїв завершити роботу сервера, однак можна обмежитися й блокуванням таблиць. Якщо використати зазначені сценарії з таблицями, використовуваними іншими потоками MySQL, дані можуть бути ушкоджені. Команди CHECK, REPAIR й mysqlcheck можна використати й тоді, коли сервер перебуває в роботі, а таблиці - у використанні.

Ми обговоримо можливості використання кожного із зазначених інструментів.

8 Перевірка й відновлення за допомогою CHECK й REPAIR

Можна перевірити таблицю за допомогою CHECK TABLE, наприклад, так:

```
checktabledepartment;
```

Команда CHECK TABLE працює з таблицями MyISAM й InnoDB. У результаті виконання зазначеної команди (якщо все піде так, як повинне бути) ви повинні одержати приблизно наступне:

Table	Op	Msg_type	Msg_text
employee. department	check	status	OK

1 row in set (0.00 sec)

Ви можете також одержати у відповідь фразу Table is already up to date (таблиця вже задовольняє всім вимогам), що означає, що все в порядку.

Одержання будь-якого іншого повідомлення означає наявність проблем, і в цьому випадку ви повинні спробувати виправити таблицю. Це можна зробити за допомогою команди REPAIR TABLE (якщо це таблиця MyISAM):

```
REPAIR TABLE t1;
```

Якщо відновлення виконується (або ніякого відновлення насправді не потрібно), ви повинні одержати приблизно такий результат:

Table	Op	Msg_type	Msg_text
test.t1	repair	status	OK

1 row in set (0.03 sec)

Якщо ж буде отримане якесь інше повідомлення, а не OK, це значить, що команда REPAIR не спрацювала, і необхідно використати більше діючу програму myisamchk.

9 Перевірка й відновлення за допомогою myisamchk

Розглянемо тільки myisamchk, але не isamchk. Якщо у вас є таблиці ISAM, ми рекомендуємо конвертувати їх в MyISAM.

Програма myisamchk неймовірно корисна й може вивести вас із деяких досить неприємних ситуацій, у яких ви можете виявитися. Але не забувайте про те, що не слід використати програму myisamchk при працюючому сервері, з погляду безпеки сервер краще зупинити.

Найпростіше викликати myisamchk за допомогою введення команди myisamchk таблиця в командному рядку.

Тут таблиця повинна вказувати шлях до файлу .MYI, що представляє таблицю MyISAM.

Виконання зазначеної команди повідомить вас практично про всі помилки. Якщо й це, як вам здасться, не допоможе вирішити проблему, можна спробувати запустити команду з перемикачем -m. За замовчуванням команда шукає помилки в індексах, а із зазначеним перемикачем скануються також і рядка.

За допомогою myisamchk можна також виправити помилки. У такий спосіб можна виправити більшість помилок таблиць MyISAM, з якими ви зіткнетесь. Для швидкого відновлення можна викликати програму myisamchk з опціями -q -r;

```
myisamchk -q -r таблиця
```

Якщо це не спрацює, можна зробити резервну копію даних, а потім спробувати виконати повне відновлення:

```
myisamchk -r таблиця
```

Якщо й це не допоможе, можна спробувати застосувати опцію --safe-recover яка може виправити деякі помилки, не виправлені опцією -r:

```
myisamchk--safe-recover таблиця
```

Програма myisamchk має велику кількість опцій, список яких можна одержати, набравши myisamchk у командному рядку без параметрів.

10 Перевірка й відновлення за допомогою mysqlcheck

Програма mysqlcheck може використатися для перевірки таблиць MyISAM й InnoDB, а також для виправлення таблиць MyISAM при працюючому сервері.

Щоб перевірити таблиці бази даних за допомогою mysqlcheck, можна використати наступну команду:

```
mysqlcheck -i ім'я_користувача -p employee
```

Можна вказати й список таблиць, які необхідно перевірити, але за замовчуванням програма перевірить всі таблиці зазначеної бази даних (дуже зручна можливість). Якщо все в порядку, ви побачите щось подібне наступний:

```
employee.assignment      OK
employee.client           OK
employee.department       OK
employee.employee         OK
employee.employeeSkills   OK
```

Можна також використати перемикач --databases, що дозволяє вказати список баз даних для перевірки, або опцію --all-databases, щоб перевірити всі бази даних даного сервера.

Можна використати програму `mysqlcheck` з опцією `-r`, щоб виправити зіпсовані таблиці MyISAM, якщо такі будуть виявлені.

Питання та завдання для самоконтролю знань студентів

1 Щоб виконати резервування бази даних, необхідно зробити наступне:

- а) завершити роботу сервера;
- б) оновити дані таблиць і заблокувати їх;
- в) зробити зазначене в пунктах а й б;
- г) жодне з вищевказаних дій не є обов'язковим.

2 Необхідно блокувати таблиці вручну перед виконанням

- а) резервування вручну;
- б) `mysqldump`;
- в) `mysqlhotcopy`;
- г) нічого з вищевказаного.

3 Якого типу таблиці можна поовеонть за допомогою команди `CHECK TABLE`?

- а) InnoDB й MyISAM;
- б) MyISAM;
- в) MyISAM і BOB;
- г) InnoDB і BOB.

4 Якого типу таблиці можна виправити за допомогою команди `REPAIR TABLE`?

- а) InnoDB й MyISAM;
- б) MyISAM;
- в) MyISAM і BOB;
- г) InnoDB і BOB.

5 Якщо команда `CHECK TABLE` повідомляє `Table is already up to date`, то

- а) варто запустити `REPAIR TABLE`;
- б) механізм зберігання не підтримує `CHECK TABLE`;
- в) з таблицею все в порядку;
- г) нічого з вищевказаного не вірно.

6 Створіть резервні копії своєї бази даних за допомогою кожного із чотирьох розглянутих у главі методів. Відновите базу даних, використовуючи кожну з отриманих резервних копій.

Розробив: Ланська С.С.

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

План самостійного вивчення теми № 7

з навчальної дисципліни «Бази даних»

Тема	Розподілені бази даних
Мета	Дати огляд основних понять принципів та властивостей розподілених баз даних

знати:

- основні означення та принципи виникнення розподілених баз даних
- логічну архітектуру розподілених баз даних
- архітектуру програмно-технічних засобів розподілених СКБД;
- як здійснюється розподіл даних по мережі.

вміти:

- виконувати фрагментацію даних;
- використовувати реплікацію

Час – 4 години

Навчальні питання:

- 1 Основні означення. Причини виникнення та завдання
- 2 Логічна архітектура розподілених баз даних
- 3 Архітектура програмно-технічних засобів розподілених СКБД
- 4 Розподіл даних по мережі
- 5 Розподілене зберігання даних

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше Розподілена система керування базами даних (РСКБД) — це програмне забезпечення, яке керує РБД і надає такі механізми доступу до них, що їх застосування дає користувачу можливість працювати з РБД як з однією цілісною базою даних.

По-друге Однією з важливих проблем РСКБД є досягнення логічної незалежності даних від місця зберігання, тобто прозорість доступу до даних. Це означає, що користувач повинен мати можливість сприймати всі необхідні йому дані як єдине ціле, не зважаючи на те, в який спосіб вони розподілені у мережі

Література

1Малыхина, М.П. Базыданных: основы, проектирование, использование [Текст] / М.П. Малыхина – СПб.: БХВ– Петербург, 2004, – 512с.: ил.

2Карпова, Т.С., Базыданных: модели, разработки, реализации [Текст]: / Т.С. Карпова – СПб.: Питер, 2001. – 303с.

Навчальні матеріали до самостійного вивчення теми

1 Основні означення. Причини виникнення та завдання

Розподілені БД складають ще один напрям в просторі досліджень і розробок систем керування базами даних. У цих системах доводиться вирішувати всі завдання, властиві централізованим СКБД, але, як правило, в більш складних постановках. У розподілених системах виникають і специфічні проблеми, від вирішення яких багато в чому залежить ефективність, надійність і доступність систем БД. В даний час більшість розподілених СКБД базується на реляційній моделі даних і розрахована на використання в локальних обчислювальних мережах. Багато проблем поширюються і на розподілені СКБД в територіально рознесених мережах, і майже всі проблеми зберігаються для розподілених СКБД, заснованих на інших моделях даних.

Розподілена база даних (РБД) - це множина логічно взаємозалежних баз даних, розподілених у комп'ютерній мережі.

Розподілена система керування базами даних (РСКБД) — це програмне забезпечення, яке керує РБД і надає такі механізми доступу до них, що їх застосування дає користувачу можливість працювати з РБД як з однією цілісною базою даних.

Розподілена система баз даних (РСБД) — це РБД разом із РСКБД.

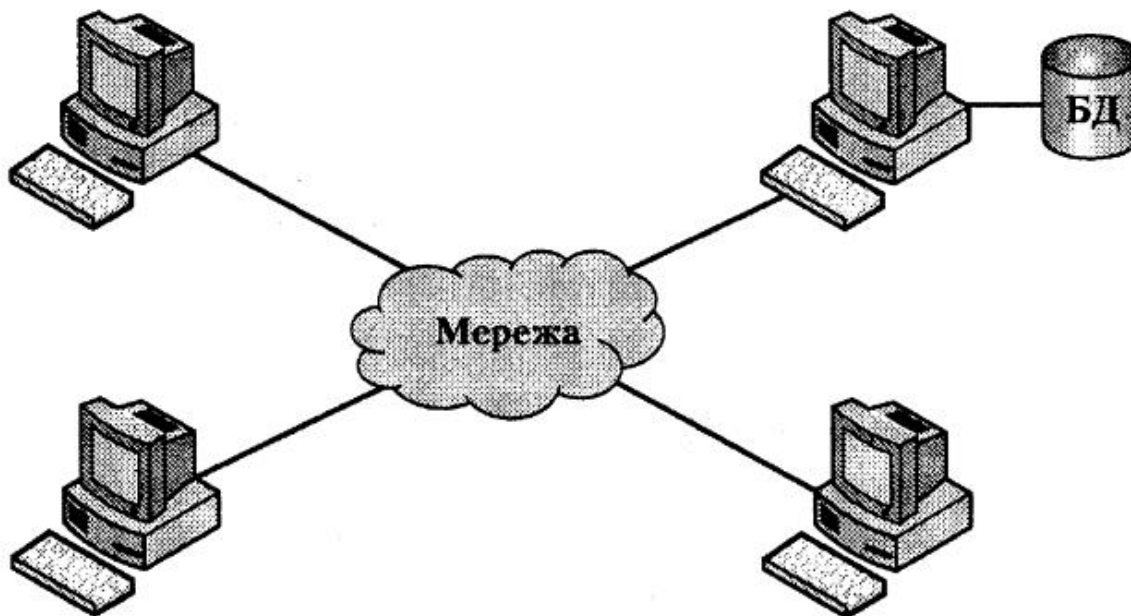


Рисунок 7.1 – Централізована база даних у комп'ютерній мережі

Не слід плутати РСБД з централізованою базою даних, що використовується в мережі (див. рис. 7.1). У цьому випадку база даних розташована на одному з комп'ютерів, а всі інші мають доступ до неї через комунікаційну мережу. Не є розподіленою також база даних, що працює в середовищі багатопроцесорних комп'ютерів. У цьому випадку ми маємо справу з паралельною БД.

Архітектура розподіленої СКБД наведена на рис. 7.2. Кожний з вузлів мережі містить свою базу даних, однак вони розглядаються як логічно єдина база, а не як сукупність розкиданих у мережі файлів. Усі дані є логічно взаємозалежними.

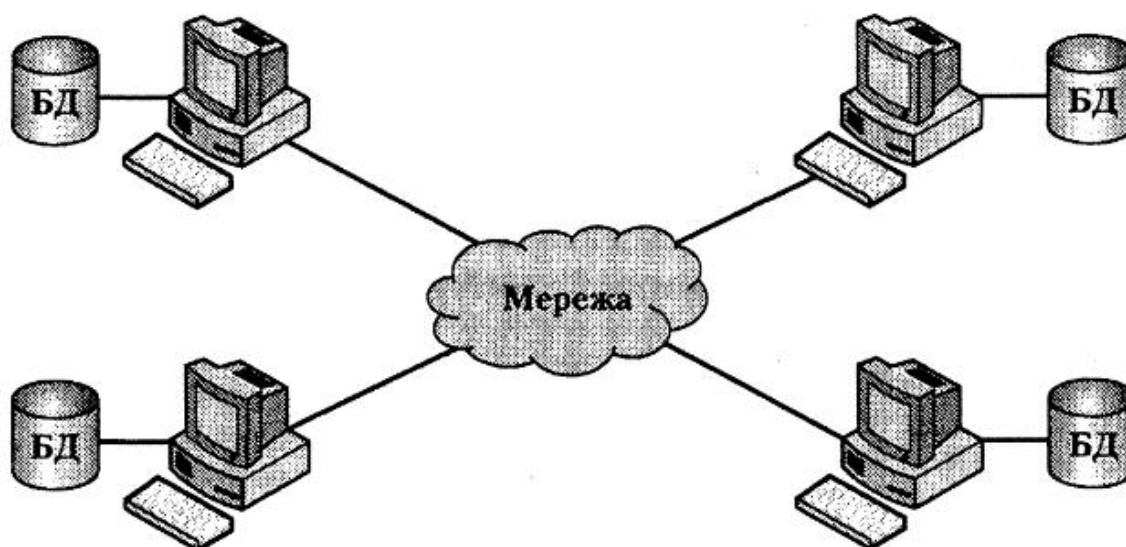


Рисунок 7.2 – Розподілена база даних

Залежно від типу програмного забезпечення розрізняють два типи РСКБД:

- однорідні;
- неоднорідні.

В однорідних РСКБД передбачається, як мінімум, що на всіх вузлах використовуються однотипні СКБД (наприклад, реляційні), які мають схожі функціональні можливості. Як максимум, припускається, що на всіх вузлах використовуються однакові технічні засоби, тобто однакові типи комп'ютерів і програмного забезпечення. Це стосується операційних систем, програмного забезпечення СКБД та моделей даних, що підтримуються.

У неоднорідних РСКБД вузли базуються на різних програмно-технічних платформах, які можуть містити різні типи СКБД. Окрім того, такі СКБД можуть підтримувати різні моделі даних. У цьому випадку ускладнюється вирішення проблеми їхньої взаємодії.

Прозорість доступу до даних

Однією з важливих проблем РСКБД є досягнення логічної незалежності даних від місця зберігання, тобто прозорість доступу до даних. Це означає, що користувач повинен мати можливість сприймати всі необхідні йому дані як єдине ціле, не зважаючи на те, в який спосіб вони розподілені у мережі.

Розглянемо такий приклад. Нехай база даних містить відношення СЛУЖБОВЕЦЬ, ПРОЕКТ, РОБОТА з інформацією стосовно службовців, проектів, що розроблюються, та участі службовців у проектах компанії, яка має філії в Римі, Лондоні, Парижі й То-кіо. Базы даних цих філій містять дані згідно зі схемою, зображеною на рис. 7.3. Будь-який користувач у кожному з вузлів цієї мережі має сприймати логічну структуру бази даних так, ніби всі три відношення містяться в одній локальній базі даних. Розглянемо приклад:

```
SELECT проект.назва, службовець.назва  
FROM службовець, проект, робота  
WHERE службовець.#e = робота.#e  
AND робота.#p = проект.#p  
AND (проект.місце = 'Париж' OR проект.місце = 'Токіо')  
AND службовець.зарплата > 40000  
AND службовець.зарплата < 60000
```

Цей запит, згідно з яким вибираються назви проектів, що виконуються в Парижі чи Токіо, й імена службовців, які беруть у них участь, із зарплатою в діапазоні від 40 000 до 60 000, буде виконано навіть тоді, коли його формулює користувач, що перебуває в Лондоні.

З погляду користувача існує єдина розподілена СКБД і єдина база даних, у якій є всі дані, що визначені у віртуальній таблиці.

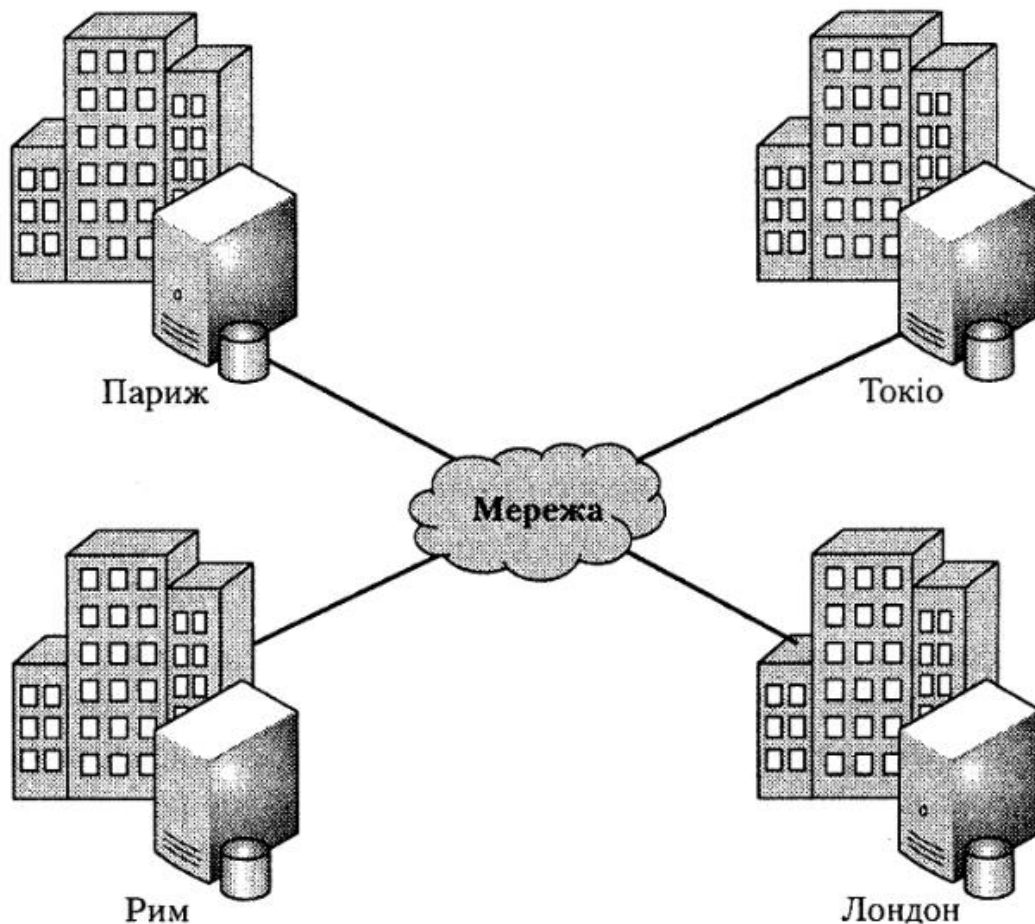


Рисунок 7.3 – Приклад розміщення даних у РБД

Централізовані ІС за допомогою СКБД вирішили основні проблеми файлових систем - усунення надмірності, незв'язність, неузгодженість даних.

Централізація процесів обробки даних дозволяє усунути такі недоліки як незв'язність, суперечливість і надмірність даних в ІС, забезпечує можливість стандартизації представлення даних, санкціонованого доступу тощо. Доводи на користь централізації даних:

- дані, які виникають у різних підрозділах, розглядаються системою як одне ціле (логічно);
- великий обсяг даних загального призначення;

- використовуються централізованими додатками;
- відносно легко реалізується захист даних.

Однак у міру зростання БД, використання їх в територіально рознесених організаціях призводить до того, що централізована СКБД погано справляється із зростанням числа оброблюваних транзакцій. Це призводить до зниження загальної надійності та продуктивності системи при обробці запитів користувачів. Багато додатків пов'язані з децентралізованим введенням і споживанням даних. Як мінімум це приводить до деякої комунікаційної мережі з централізованою СКБД, де виникають наступні проблеми:

- великий обсяг обмінів даними (напружений мережевий трафік);
- низька надійність системи через відмову каналів зв'язку та центральної СКБД;
- низька загальна продуктивність (вузькі місця - мережа і центральний сервер);
- великі витрати на розробку і експлуатацію.

Через низку вищеперелічених проблем, навіть в технології «клієнт-сервер» побудувати по-справжньому велику ІС не представляється можливим. Крім того, багато транснаціональних економічних систем фізично розподілені по всьому світу, що робить практично неможливою централізоване зберігання даних. Очевидно, що в централізованій БД легше забезпечити безпеку і цілісність, але вирішення вищеперерахованих проблем можливе лише на шляху створення децентралізованої обробки та зберігання даних, тобто на шляху переходу до розподілених БД. При децентралізації досягається:

- паралельність обробки внаслідок децентралізації;
- висока живучість системи;
- менші початкові витрати - мережа розвивається поступово;
- більш висока продуктивність;
- вигідно зберігати дані та обробляти на місцях виникнення.

Основне завдання СКБД в розподіленій системі полягає в забезпеченні засобів інтеграції локальних баз даних, що розташовані в деяких вузлах обчислювальної мережі, з тим, щоб користувач, який працює в будь-якому вузлі мережі, мав доступ до всіх цих баз даних як до єдиної бази даних. При цьому повинні забезпечуватися:

- простота використання системи;
- розвинені засоби забезпечення цілісності;
- можливості автономного функціонування при порушеннях зв'язності мережі або при адміністративних потребах;
- висока ефективність.

В загальному випадку ІС можуть бути комбінованими та містити як централізовані дані так і децентралізовані.

Алгоритми виконання реляційних операцій

Якщо говорити тільки про реляційні розподілені СКБД, які найбільш поширені в теоретичному і практичному відношенні, досі проводиться маса досліджень в області оптимізації алгоритмів виконання реляційних операцій (головним чином, з'єднання віддалених таблиць). Таким чином, навіть розглянувши невелику частину проблем розподілених систем, можна переконатися, що вони потребують велику кількість досліджень і експериментів. Розпочатий в Україні перехід до використання локальних (а тепер і високошвидкісних) мереж дає практичну можливість проведення таких робіт.

2 Логічна архітектура розподілених баз даних

Логічна архітектура розподілених баз даних — це архітектура логічно взаємозалежних даних. Вона дещо нагадує архітектуру ANSI/SPARC. Графічне зображення логічної архітектури розподілених баз даних наведено на рисунку 7.5. Особливість архітектури РБД, порівняно з архітектурою ANSI/SPARC, полягає в тому, що виникає ще один рівень, глобальний концептуальний, завданням якого (як і в моделі ANSI/SPARC) є зображення концептуальної моделі предметної області в цілому. На цьому рівні описується характер розподілу даних.

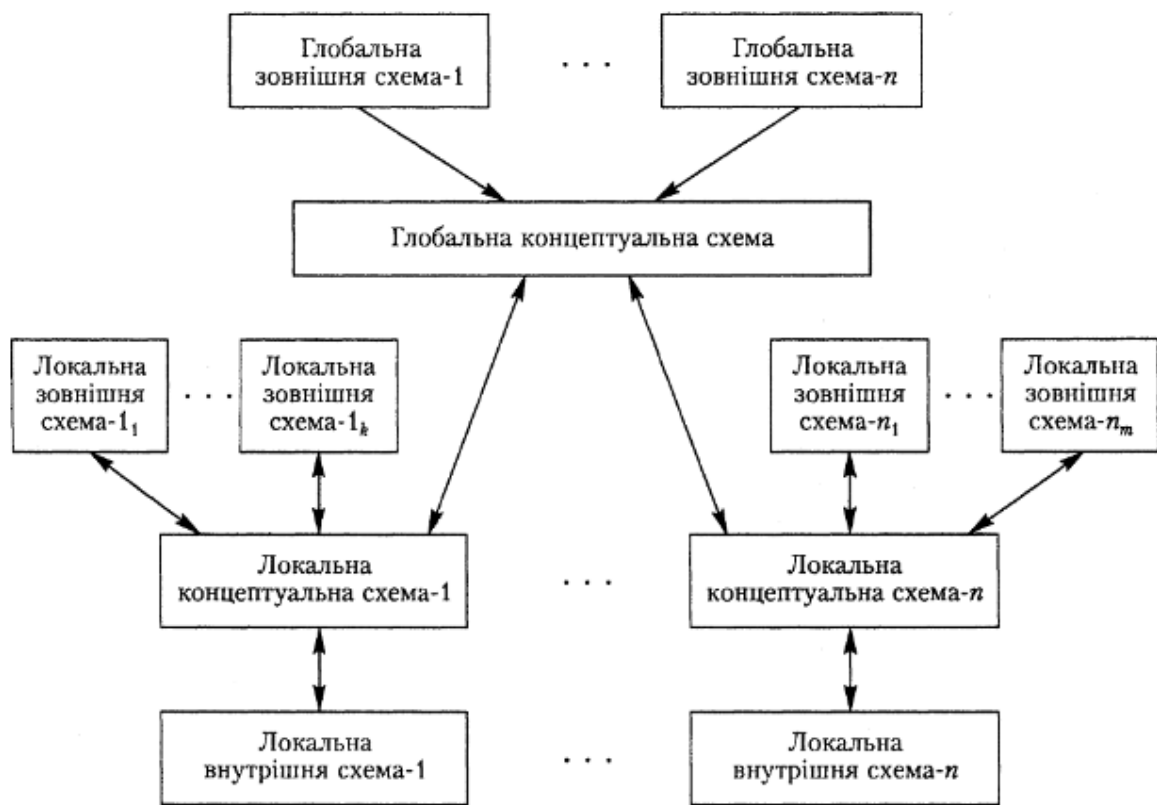


Рисунок 7.5 – Логічна архітектура розподіленої СКБД

На локальному концептуальному рівні здійснюється локальний опис ПО. Тоб-то схема цього рівня містить опис тільки тієї частини предметної області, що є специфічною для конкретного вузла розподіленої ПО; дані інших вузлів розподіленої ПО в схемі не вказуються. Відображення «глобальний-концептуальний/ локальний-концептуальний» дають змогу зобразити глобальну концептуальну схему у вигляді сукупності локальних концептуальних схем, і навпаки.

Глобальна зовнішня схема зображує дані, необхідні користувачам і застосуванням, у бажаному для них вигляді, незалежно від способу розподілу даних за вузлами. Це завдання вирішується завдяки тому, що глобальні зовнішні схеми базуються на глобальній концептуальній схемі.

Локальні зовнішні схеми базуються на локальних концептуальних схемах, від-так вони можуть посилалися лише на ті дані, що розташовані на відповідному вузлі розподіленої ПО.

Локальні внутрішні схеми — це схеми зберігання даних на конкретних вузлах розподіленої ПО. Вони пов'язані з відповідними локальними концептуальними схемами.

3 Архітектура програмно-технічних засобів розподілених СКБД

Архітектура програмно-технічного комплексу розподілених СКБД має три головні характеристики:

- розподіленість;
- неоднорідність;
- автономність.

Розглянемо ці характеристики детальніше.

Розподіленість

Спосіб розподілу компонентів системи баз даних за комп'ютерами мережі визначається тим, чи є в мережі єдиний комп'ютер з повноваженнями розподіленої СКБД, або ж їх кілька. Якщо таких комп'ютерів кілька, то чи є між ними взаємодія тощо.

Неоднорідність

Важливою характеристикою є міра однорідності програмно-технічних засобів СКБД. Ідеться про технічне забезпечення (типи комп'ютерів), комунікаційні засоби зв'язку, операційні системи і типи баз даних (моделі даних, мови запитів, алгоритми керування транзакціями тощо).

Автономність

Автономність характеризує, наскільки самостійно компоненти СКБД можуть виконувати свої функції. До питань автономності належать:

- автономність проектних рішень (наскільки самостійно компоненти СКБД можуть розв'язувати задачі, які були спроектовані для них);
- автономність вирішенням комунікаційних проблем (наскільки самостійно компоненти СКБД можуть встановлювати зв'язки з іншими компонентами);
- автономність обчислень (наскільки самостійно компоненти СКБД можуть приймати рішення щодо виконання локальних операцій). Діаметрально протилежними підходами до вирішення цих питань є створення єдиної централізованої СКБД та встановлення СКБД у кожному з вузлів мережі.

Зазначимо, що перелічені характеристики мають не лише розподілені СКБД, але й усі розподілені системи обробки даних.

Різновиди архітектури

Основними різновидами архітектури програмно-технічних засобів розподіленої СКБД є:

- клієнт-серверна архітектура;
- архітектура з багатьма незалежними серверами;
- архітектура із взаємодіючими серверами;
- архітектура однорангової мережі.

Розглянемо ці типи архітектури детальніше.

Клієнт-серверна архітектура

Така архітектура передбачає наявність єдиного комп'ютера-сервера і багатьох комп'ютерів-клієнтів, що взаємодіють між собою через канали зв'язку. На сервері розташована СКБД та інтегрована (централізована) база даних. Ніякого розподілу баз даних за вузлами мережі немає. На клієнтських комп'ютерах виконуються застосування, які працюють із серверною базою даних, а також розміщене програмне забезпечення для зв'язку з віддаленою СКБД. Як клієнти, так і сервер оснащені комунікаційним програмним забезпеченням.

Архітектура з багатьма незалежними серверами

Ця архітектура передбачає існування багатьох серверів, що мають доступ до своїх локальних баз даних. У такому випадку на комп'ютерах клієнтів має зберігатись інформація стосовно того, які дані на яких серверах розташовані, а також має бути розміщене програмне забезпечення, що дає змогу декомпонувати запити для їхнього виконання на різних серверах і потім об'єднати результати.

Архітектура із взаємодіючими серверами

У цьому випадку передбачається, що кожний із серверів містить повну інформацію стосовно того, які дані на яких серверах зберігаються, а також здатний обробляти розподілені запити. У разі потреби один сервер може звернутися до іншого для одержання необхідних даних. Кожен клієнт має свій сервер, до якого звертається для виконання операцій над розподіленою базою даних.

Архітектура однорангової мережі

Усі комп'ютери мережі є серверами. На кожному комп'ютері розміщено розподілену СКБД і базу даних, з кожного комп'ютера можна надіслати до іншого запит на отримання необхідних даних.

Основне завдання при проектуванні розподіленої БД - розподіл даних по мережі.

Способи вирішення цього завдання:

- в кожному вузлі мережі зберігається і використовується власна БД, проте збережені в ній дані доступні для інших вузлів мережі;
- всі дані розподіленої БД повністю дублюються в кожному вузлі мережі;
- збережені в центральному вузлі дані частково дублюються в тих периферійних вузлах, в яких вони використовуються.

Розподілена обробка даних вимагає вирішення питань:

- розподілені БД можуть бути однорідними або неоднорідними в сенсі використовуваних СКБД. Для користувачів в будь-якому випадку повинна бути забезпечена прозорість перетворень структур даних і ПП (навіть при використанні різних СКБД);
- повинна бути єдина концептуальна схема для забезпечення прозорості даних по всій мережі (користувач не повинен вказувати вузол, де знаходяться дані);
- декомпозиція запиту користувача на складові частини, які можуть пересилатися для виконання в різні вузли мережі в залежності від місця зберігання даних та ефективності обробки (повинна бути забезпечена координація процесів обробки);
- синхронізація процесів оновлення і обробки копій даних;
- захист даних і їх відновлення;
- керування словниками даних тощо.

Розглянемо архітектуру однорідних розподілених БД.

Для опису інформаційної структури всієї мережі вводиться інтерфейс концептуальної моделі даних - глобальна мережева «Компонентна схема даних» (метамодель даних):

- інтерфейс зовнішньої моделі - зовнішня схема мережі для зручності роботи користувача (написання запиту в рамках мережі);
- у кожному вузлі мережі є локальна загальна схема (одна для кожного вузла) містить як опис локальних даних, збережених у цьому вузлі, так і опис даних, збережених в інших вузлах, але використовуваних ПП і користувачами в даному вузлі;

– для реалізації запиту його зовнішня схема транслюється в так звану загальну схему мережі (в якій вже присутня інформація про розміщення необхідних даних в мережі) і починається його виконання;

– СКБД будь-якого вузла мережі зберігає локальні дані та виконує в цьому вузлі необхідні операції над ними. SQL-запит, який поступив, декомпонується на складові операції (підзапити), будується план переміщення і обробки підзапитів в мережі, і починається пересилка підзапитів у відповідні локальні СКБД для виконання;

– СКБД вузла, виконавши відповідний підзапит, результат виконання видає в мережу;

– після надходження відповідей на всі підзапити формується остаточна відповідь.

Підтримка копій даних у декількох вузлах мережі

Основним накладними витратами при виконанні розподіленого запиту є пересилання даних між вузлами мережі. Одним із способів скорочення цих накладних витрат є підтримка копій найбільш часто використовуваних даних у декількох вузлах мережі з урахуванням породжуваних цим додаткових накладних витрат по підтриманню узгодженості копій при модифікації даних.

Іншою підставою для підтримання копій є збільшення рівня доступності даних при виході з ладу вузлів мережі або комунікаційних ліній, внаслідок чого втрачається зв'язність мережі. Теоретично доступ до деякого об'єкту БД може тривати в будь-якому розділі мережі, що містить його копію. Однак доводиться вирішувати ту ж проблему узгодження копій при змінах об'єкта даних. Існує багато підходів до вирішення цієї проблеми від повного вирішення будь-якого доступу до будь-якої копії об'єкта в усіх розділах мережі з проведенням складної процедури узгодження копій після відновлення зв'язності мережі до дозволу модифікації об'єкту тільки в одному розділі. Для виявлення цього розділу зазвичай застосовуються різні варіанти алгоритму голосування.

Фрагментація об'єктів БД

Альтернативний підхід, що забезпечує максимальне розпаралелювання виконання запиту до БД, полягає в тому, що таблиця розбивається на ряд вертикальних або горизонтальних фрагментів, і ці фрагменти зберігаються в різних

вузлах мережі. Теоретично це може забезпечити прискорення виконання запиту, що зачіпає фрагментовані відношення, але практично досягти цього дуже важко. Основна проблема полягає в різкому розширенні простору пошуку варіантів виконання запитів, з яким повинен працювати оптимізатор запитів. Існують пропозиції і дослідження, пов'язані з комбінацією підтримки копій і фрагментацією об'єктів БД. У цьому випадку підтримуються копії фрагментів, що, вирішує обидва завдання, але ще більше ускладнює реалізацію.

5 Розподілене зберігання даних

У розподіленій СКБД дані розподіляються за вузлами мережі. Існують два основні механізми розподіленого зберігання даних:

- фрагментація;
- реплікація.

Розглянемо їхню реалізацію в реляційній моделі даних.

Фрагментація

Суть фрагментації полягає в тому, щоб поділити логічну базу даних на фрагменти з метою зберігання кожного фрагмента на певному вузлі мережі. Одиницями фрагментації можуть бути відношення та складені відношення. У випадку, коли одиницею фрагментації є відношення, вирішується проблема, яке відношення в якій базі даних має зберігатися. За іншого підходу допускається, що будь-яке від-ношення може бути зображене у вигляді сукупності фрагментів, що розподіляються за різними базами даних.

Фрагментацію, що здійснюється розподілом відношень за базами даних, теоретично здійснити нескладно, тому розглянемо проблему фрагментації власне відношень.

Фрагментація відношень

Завдання фрагментації відношень формулюється в такий спосіб. Нехай задане відношення R . Його потрібно зобразити у вигляді сукупності відношень $R_1, \dots, R_n$ так, щоб ця сукупність відповідала критеріям ефективності (за часом доступу, пам'яттю, завантаженістю комп'ютерів тощо).

Фрагментація є коректною, якщо вона повна, не містить перетинів і може бути реконструйована. Пояснимо ці терміни.

Декомпозиція відношення R на фрагменти $R_1, R_2, \dots, R_n$ є повною тоді й лише тоді, коли кожен елемент даних з R належить якомусь із відношень R_i .

Декомпозиція відношення R на фрагменти $R_1, R_2, \dots, R_n$ може бути реконструйована, якщо існує такий реляційний вираз $\phi(R_1, R_2, \dots, R_n)$, що $R = \phi(R_1, R_2, \dots, R_n)$.

Декомпозиція відношення R на фрагменти $R_1, R_2, \dots, R_n$ не містить перетинів, якщо будь-який елемент даних з R міститься не більш ніж в одному фрагменті.

Є три типи фрагментації відношень:

- горизонтальна;
- вертикальна;
- змішана.

Розглянемо кожний з цих різновидів фрагментації.

Горизонтальна фрагментація

Горизонтальна фрагментація полягає в розподілі кортежів відношення за фрагментами. Формально горизонтальну фрагментацію можна визначити в такий спосіб. Нехай задане відношення R і на ньому визначений предикат F_i . Тоді горизонтальний фрагмент R_i , відношення визначається так:

$$R_i = \sigma_{F_i}(R).$$

Тобто горизонтальний фрагмент R_i - це множина кортежів R , що задовольняють умову F_i .

Коректність горизонтальної фрагментації може бути встановлена в такий спосіб. Нехай на відношенні R задано множину предикатів $F = \{F_i\}$, $i = 1, k$. Ця множина породжує відповідну множину фрагментів $R_1, \dots, R_k$. Множина F є повною стосовно R (відтак і горизонтальна фрагментація теж є повною), якщо

$$R = \bigcup \sigma_{F_i}(R), i = 1, k.$$

Фрагментація є повною тоді й лише тоді, коли $F_1 \vee F_2 \vee \dots \vee F_k$ є завжди істинною формулою. Таку фрагментацію можна реконструювати за щойно наведеною формулою. Якщо для будь-якої пари F_i та F_j з множини F , $i \neq j$ формула $F_i \& F_j$ є завжди хибною, то фрагментація, породжувана предикатами з множини F , не міститиме перетинів.

Розглянемо кілька прикладів. Нехай задане відношення

ПРОЕКТ(Рном, Назва, Тип, Вартість).

Очевидно, що множина предикатів

$\{\text{Вартість} < 300000, \text{Вартість} = 300000, \text{Вартість} > 300000\}$

породжує повну фрагментацію відношення ПРОЕКТ, яка не містить перетинів.

Предикати

$\{\text{Вартість} < 300000, \text{Вартість} > 200000\}$

породжують повну фрагментацію, що містить перетин. А предикати

$\{\text{Вартість} < 200, \text{Вартість} > 300\}$

породжують неповну фрагментацію, яка не містить перетинів. Нарешті предикати

$\{\text{Тип} = \text{'держзамовлення'}, \text{Тип} = \text{'приватний'}, \text{Тип} = \text{'ініціативний'}\}$

визначають фрагментацію без перетинів, а їхня повнота залежить від домену атрибута Тип. Усі чотири фрагментації можуть бути реконструйовані.

Породжена горизонтальна фрагментація

Якщо два відношення сполучені зв'язком типу «один-до-багатьох», то під час фрагментації відношення в закінченні «один» (відношення-власник) може породжувати фрагментацію відношення, що розташоване в закінченні «багато» (відношення-член). А саме, розташовуючи той чи інший кортеж відношення-власника на одному з вузлів, можна на ньому ж розташувати всі ті кортежі відношення-члена, що зв'язані з вихідним кортежем. Це буде доцільним, оскільки в багатьох запитах кортежі відношення-власника обробляються разом із відповідними кортежами відношення-члена.

Розглянемо приклад.

Нехай задано відношення

ПРОЕКТИ(#Рном, Назва, Тип, Вартість),

СЛУЖБОВЦІ(#Сном, #Рном, Ім'я)

Зв'язок між ними, встановлений за допомогою зовнішнього ключа #Рном у відношенні СЛУЖБОВЦІ, вказує, в яких проектах беруть участь службовці.

Нехай відношення ПРОЕКТИ поділяється на два фрагменти згідно з такими предикатами:

$\{\text{Вартість} < 300\ 000, \text{Вартість} \geq 300\ 000\}$

Відповідно до цих предикатів породжуються два фрагменти відношення СЛУЖБОВЦІ. Перший із них міститиме всіх службовців, які беруть участь у

проектах вартістю менше 300 000, а другий — службовців, які беруть участь у проектах вартістю не менше 300 000. Породжена фрагментація формально може бути визначена в такий спосіб. Нехай задано відношення R та S і у відношенні S продубльований атрибут A відношення R . Нехай відношення R поділене на горизонтальні фрагменти $\{R_1, R_2, \dots, R_k\}$ згідно з предикатами $\{F_1, F_2, \dots, F_k\}$, визначеними на атрибуті A . Поділ відношення S на фрагменти $\{S_1, S_2, \dots, S_k\}$ називається породженою горизонтальною фрагментацією, якщо будь-який фрагмент S_i - визначається за такою формулою:

$$S_i = \sigma_{S[A] = R_i[A]}(S).$$

Розглянемо властивості повноти, можливості реконструкції і відсутності перетинів стосовно породженої фрагментації.

Якщо у відношенні S атрибут A є обов'язковим зовнішнім ключем з відношення R , то визначена вище породжена горизонтальна фрагментація є повною. Що стосується можливості реконструкції, то відношення S відновлюється з породженої фрагментації так само, як і під час звичайного відновлення відношення з фрагментів. Якщо зв'язок між відношеннями R та S має множинність «один-до-багатьох», то фрагментація відношення S , породжена фрагментацією без перетинів відношення R , також не містить перетинів.

Переваги горизонтальної фрагментації:

- дає змогу паралельно обробляти фрагменти відношення;
- дає можливість поділяти відношення так, щоб кортежі розташовувалися там, де вони будуть використовуватися найчастіше.

Вертикальна фрагментація

Суть вертикальної фрагментації полягає в тому, що відношення поділяється на дві чи більше проекцій, тобто схема відношення поділяється на певну множину підсхем.

Для відновлення вихідного відношення з фрагментів необхідно, щоб усі підсхеми містили первинний ключ. Існує інший підхід, коли під час поділу схеми до кожного фрагмента автоматично додається поле з ідентифікатором кортежу. Значення цього ідентифікатора зазвичай встановлюються системою автоматично.

Розглянемо приклад.

Нехай задано відношення

ПОСТАЧАННЯ(#Рном, Постачальник, Обладнання, Проект) (табл. 7.1.).

Таблиця 7.1 – Вихідне відношення ПОСТАЧАННЯ

#Рном	Постачальник	Обладнання	Проект
P1	Іванов	Двигун	АН-24
P2	Іванов	Двигун	ЯК-40
P3	Іванов	Шасі	АН-24
P4	Петров	Двигун	АН-24
P5	Петров	Електрообладнання	ЯК-40
P6	Петров	Електрообладнання	АН-70

Вертикальна фрагментація з використанням первинного ключа може виглядати так (див. табл. 7.2., 7.3.):

Таблиця 7.2 – Фрагментація ПОСТАЧАННЯ-1

#Рном	Постачальник
P1	Іванов
P2	Іванов
P3	Іванов
P4	Петров
P5	Петров
P6	Петров

Таблиця 7.3 - Фрагментація ПОСТАЧАННЯ-2

#Рном	Обладнання	Проект
P1	Двигун	АН-24
P2	Двигун	ЯК-40
P3	Шасі	АН-24
P4	Двигун	АН-24
P5	Електрообладнання	ЯК-40
P6	Електрообладнання	АН-70

Розглянемо умови коректності вертикальної фрагментації.

Нехай задано відношення R , що визначене на множині атрибутів A з ключем K . Це відношення поділене на вертикальні фрагменти $FR = \{R_1, R_2, \dots, R_n\}$, кожний з яких має атрибути A_{R_i} . Повнота вертикальної фрагментації означає, що

$$A = \cup A_{R_i}, i = 1, n.$$

Можливість реконструкції вертикальної фрагментації означає, що

$$R = \{R_1 * R_2 * \dots * R_n\} [A].$$

Природне з'єднання виконується за первинним ключем чи ідентифікатором кортежу. Відсутність перетинів означає, що будь-який атрибут з множини A , за винятком первинного ключа та ідентифікатора кортежу, не входить до двох чи більше фрагментів водночас.

Переваги вертикальної фрагментації:

- дає змогу поділяти кортежі відношення так, щоб їхні частини розташовувалися там, де вони використовуватимуться найчастіше;
- дає змогу проводити паралельну обробку відношень;
- наявність ідентифікатора кортежу дає можливість здійснювати ефективне з'єднання вертикальних фрагментів.

Змішана фрагментація

Змішана фрагментація передбачає послідовне застосування вертикальної і горизонтальної фрагментацій. Приклад змішаної фрагментації зображений на рисунку 7.6.

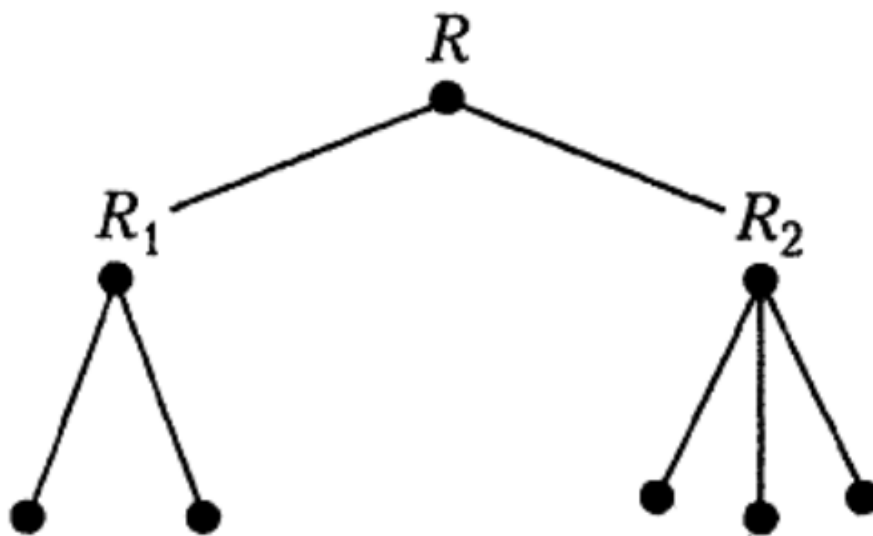


Рисунок 7.6 – Приклад змішаної фрагментації

Після отримання усіх необхідних фрагментів відношень постає проблема розподілу цих фрагментів за вузлами мережі. Єдиних рекомендацій стосовно того, як це робити, немає. Нехай задано:

$F = \{F_1, F_2, \dots, F_n\}$ - множину фрагментів логічної бази даних;

$S = \{S_1, S_2, \dots, S_m\}$ - множину вузлів розподіленої СКБД;

$Q = \{Q_1, Q_2, \dots, Q_k\}$ - множину застосувань, що мають функціонувати в мережі.

Потрібно знайти оптимальний розподіл фрагментів F за вузлами мережі S за умови, що відомий розподіл застосувань Q за вузлами мережі.

Визначаючи оптимальність, слід враховувати різні параметри, зокрема:

- вартість передавання, зберігання й обробки даних;
- часові характеристики;
- продуктивність;
- обмеження (наприклад, ємнісні характеристики вузлів мережі).

Для того щоб розподілити фрагменти за вузлами мережі, необхідна додаткова інформація, яка стосується:

- бази даних та розмірів фрагментів;
- застосувань (місць їхнього розташування, частоти використання тих чи інших фрагментів для вибирання даних, частоти використання фрагментів для відновлення даних);
- вузлів мережі (вартості зберігання й обробки даних у вузлах);
- мережі (вартості та часових характеристик передавання даних між двома вузлами).

Питання та завдання для самоконтролю знань студентів

- 1 Що таке однорідні й неоднорідні розподілені бази даних?
- 2 Чи відповідає логічна архітектура розподілених баз даних архітектурі ANSI/SPARC?
- 3 Що таке розподіленість, неоднорідність й автономність баз даних?
- 4 Які механізми розподіленого зберігання даних ви знаєте?
- 5 Які різновиди фрагментації баз даних ви знаєте?
- 6 Що таке реплікація? Які існують механізми й моделі реплікації?

- 7 Опишіть алгоритми розподіленого виконання операції з'єднання відношень.
- 8 Які є способи обчислення запитів на фрагментованих відношеннях?

Розробив: Ланська С.С.

Розглянуто та схвалено

на засіданні предметної (циклової) комісії
програмної інженерії

Протокол № 4 від 20.11.2017 р.

Голова комісії _____ Ланська С.С.

Критерії оцінювання роботи студента
над темою для самостійного вивчення.

В результаті роботи над темою для самостійного вивчення студент повинен надати письмову роботу, яка містить:

- конспект теми;
- відповіді на питання для самоконтролю знань студентів, надані в плані самостійного вивчення теми.

Згідно з розробленою Модульно – рейтинговою системою оцінювання успішності студентівз дисципліни «Бази даних», максимальна кількість балів, яку може отримати студент за роботу над темою самостійного вивчення, дорівнює 2.

Критерії оцінювання та система нарахування балів за виконання завдань наведено у таблиці № 1:

Таблиця № 1

<i>Види завдань</i>	<i>Кількість балів</i>	<i>Критерії оцінювання виконання завдання</i>	<i>Максимальна кількість балів</i>
Конспект теми	1	Конспект теми повний, логічно-послідовний. Студент зумів виділити та опрацювати основні положення теми, виділені в плані самостійного вивчення теми.	1
	0,75	Конспект повний, але при цьому допущені невеликі пропуски, що не спотворили логічного і інформаційного змісту теми.	
	0,5	Конспект неповно або непослідовно розкриває зміст матеріалу теми, але в конспекті виділені основні положення теми.	
	0,25	Конспект не розкриває основний зміст навчального матеріалу, або студент не зумів виділити та опрацювати основні положення теми.	

Таблиця № 1 (Продовження)

<i>Види завдань</i>	<i>Кількість балів</i>	<i>Критерії оцінювання виконання завдання</i>	<i>Максимальна кількість балів</i>
Відповіді на контрольні питання	1	Студент правильно та повному об'ємі відповідає на всі контрольні питання, надані в плані самостійного вивчення теми.	1
	0,75	Студент правильно та повному об'ємі відповідає на 70% контрольних питань, наданих в плані самостійного вивчення теми.	
	0,5	Студент правильно та повному об'ємі відповідає на 50% контрольних питань, наданих в плані самостійного вивчення теми.	
	0,25	Студент правильно та в повному об'ємі відповідає не більше ніж на 30% контрольних питань, наданих в плані самостійного вивчення теми.	
Разом			2

Відповідність кількості набраних студентом балів оцінці за 4 – бальною шкалою оцінювання наведена в таблиці № 2.

Таблиця № 2

Бальна	Національна
1,8-2	відмінно
1,5-1,7	добре
1,2-1,45	задовільно
0-1,2	незадовільно