МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ПРОЕКТУВАННЯ ТА ВИКОРИСТАННЯ БАЗ ДАНИХ -1

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою-професійною програмою «Комп'ютерний моніторинг та геометричне моделювання процесів та систем»-спеціальності 122 Комп'ютерні науки*

Київ

КПІ ім. Ігоря Сікорського

2021

Проектування та використання баз даних -1. Комп’ютерний практикум: навчальний посібник для здобувачів ступеня бакалавра за освітньо-професійною програмою "Комп'ютерний моніторинг та геометричне моделювання процесів і систем" / КПІ ім. Ігоря Сікорського; уклад.: І.В.Сегеда.– Електронні текстові дані (1 файл: 61,9 Кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 49с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 2 від 09.12.2021) за поданням Вченої ради теплоенергетичного факультету (протокол №3 від 26.10.2021)

Електронне мережне навчальне видання

ПРОЕКТУВАННЯ ТА ВИКОРИСТАННЯ БАЗ ДАНИХ - 1
КОМ’ЮТЕРНИЙ ПРАКТИКУМ

Укладач: Сегеда Ірина Василівна, канд. екон. наук, доцент

Відповідальний редактор: Сидоренко Ю.В., канд. техн. наук, доцент

Рецензент: Яковлева А. П., Канд. ф-м. наук., ст.н.с., доцент

За редакцією укладача

Посібник «Проектування та використання баз даних-1» розроблений на підставі Ф-каталогу вибіркових освітніх компонентів циклу професійної підготовки освітньо-професійної програми Комп'ютерний моніторинг та геометричне моделювання процесів і систем спеціальності 122 «Комп’ютерні науки» та призначений для якісної організації виконання комп’ютерного практикуму студентами. Спрямований на формування у студентів умінь та досвіду роботи з реляційними базами даних засобами мови SQL. Забезпечує студентів теоретичними відомостями і необхідними прикладами виконання завдань, запланованих впродовж семестру.

ЗМІСТ

ВСТУП

Проектування та використання баз даних та інформаційних систем стає невід'ємною складовою діяльності сучасної людини та функціонування успішних організацій. У зв'язку з цим великої актуальності набуває опанування принципів побудови та ефективного застосування відповідних технологій та програмних продуктів: систем управління базами даних, систем автоматизації проектування.

Ефективність функціонування розроблених систем залежить від вірного вибору інструментальних засобів створення інформаційних систем, визначення відповідної моделі даних, обґрунтування раціональної схеми побудови баз даних, організації запитів до бази даних. Все це потребує усвідомленого застосування теоретичних положень та практичних навиків при розробці баз даних.

Мета циклу комп'ютерного практикуму полягає в тому, щоб студенти отримали навички у проектуванні та використанні баз даних, виконувати проектування інформаційного забезпечення (логічну та фізичну структури баз даних) інформаційних систем. Та отримали досвід використання одержаних знань та вміння самостійно виконувати моделювання і проектування структур та елементів баз даних із застосуванням різноманітних сучасних методик та технологій.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №1.

Тема: **Створення збережених процедур.**

Мета: Навчитись використовувати збережені процедури.

Завдання:

Для заданої предметної області

1. Створити 1 збережену процедуру без параметрів;
2. Створити 2 збережені процедури із вхідним параметром;
3. Створити 1 збережену процедуру із вихідним параметром;
4. Створити 2 користувацькі функції.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Створення процедур

Збережена процедура (Stored procedure) – програма, що містить SQL-конструкції, яка зберігається у базі даних і виконується на стороні сервера.

Створення процедури

CREATE PROCEDURE *ім'я_процедури*
[(*вхідний_параметр тип* [, *вхідний_параметр тип* **...])]**
[RETURNS (*вихідний_параметр тип* **[,***вихідний_параметр тип***...])]**
AS
[DECLARE VARIABLE *ім'я_змінної тип_змінної*;]
…
[DECLARE VARIABLE *ім'я_змінної тип_змінної;]*
BEGIN
команди_InterBase
END

Синтаксис збереженої процедури складається з двох частин: заголовка і тіла. Заголовок включає команду **CREATE PROCEDURE**, ім'я процедури, список вхідних параметрів і список параметрів, які повертаються з процедури. Дані списки можуть бути відсутні. В процедурах використовуються стандартні типи даних *InterBase.*

Тіло процедури починається після слова **AS**, містит перелік описів локальних змінних і блок операторів.

Кожна змінна описується окремо за допомогою команди **DECLARE VARIABLE**. В кінці кожної команди ставиться крапка з комою (;). Опис локальних змінних може бути відсутній.

Блок операторів береться в операторні дужки **BEGIN … END**. Він може містити *DML-SQL*-конструкції, а також, програмні конструкції **FOR SELECT…DO, IF-THEN, WHILE...DO** та інші.

При використанні процедур, як і тригерів, у скриптах або в ISQL-режимі використовуються терми.

Тобто, процедура повинна мати такий вигляд

SET TERM ^ ;
CREATE PROCEDURE Pr1
…
END
^
SET TERM ; ^

Після **END** крапка з комою не ставиться. Завершенням команди є символ ^.

Модифікація процедури

ALTER PROCEDURE *ім'я_процедури*
[(*вхідний_параметр тип [, вхідний_параметр тип* ...])]
[**RETURNS** (*вихідний_параметр тип [,вихідний_параметр тип*...])]
AS
[**DECLARE VARIABLE** *ім'я_змінної тип_змінної*;
[**DECLARE VARIABLE** *ім'я_змінної тип_змінної*; …]]
BEGIN
команди_InterBase
END

За допомогою цієї команди можна змінити список вхідних і вихідних параметрів і тіло процедури. Заголовок і тіло процедури повинні бути включені в дану команду повністю. Синтаксис такий самий, як і в команді CREATE PROCEDURE.

Виконання процедур

Збережені процедури можна викликати з клієнтських додатків, тригерів або інших процедур. Існує два види збережених процедур: **SELECT і EXECUTE.**

SELECT-процедури (selectable procedures) або процедури-вибірки обов'язково повертають один або декілька наборів значень (якщо вони існують), які згруповані по рядках, і можуть використовуватись при створенні запитів разом з таблицями і представленнями.

EXECUTE-процедури (executable procedures) або виконані процедури можуть повертати тільки один набір значень, який перерахований у вихідних параметрах, або не повертати жодного значення у викликаючу програму.

SELECT-процедури і EXECUTE-процедури описуються однаково, але викликаються по різному.

SELECT-процедури викликаються за допомогою конструкції:

SELECT * FROM *ім'я_процедури [(факт_параметр [, факт_параметр…]*)];

EXECUTE-процедури викликаються за допомогою конструкції:

EXECUTE PROCEDURE *ім'я_процедури [(*
факт_параметр [, факти_параметр, …])]

Знищення процедури

DROP PROCEDURE *ім'я_процедури*;

Процедуру не можна знищити, якщо вона виконується або використовується в інших процедурах, представленнях, тригерах.

Приклади створення процедур

Приклад збереженої процедури, який ілюструє використання операторів циклу і перевірки умови.

```
CREATE PROCEDURE P1 RETURNS (R INTEGER)
AS
BEGIN
 R = 0;
 WHILE (R < 5) DO
 BEGIN
 R = R + 1;
 SUSPEND;
 IF (R = 3) THEN
 EXIT;
 END
END
```

Після виклику **SELECT * FROM** P; процедура поверне значення 1, 2 и 3 у викликаючий додаток.

Приклад: Визначити максимальну стипендію.

```
CREATE PROCEDURE PR2 RETURNS (Max_stip
DECIMAL (8,2))
AS
```

```
BEGIN
 SELECT MAX(STIP) FROM STUDENTS INTO
:Max_stip;
 SUSPEND;
END
```

Виклик

```
SELECT * FROM PR2;
```

або

```
EXECUTE PROCEDURE PR2;
```

Приклад: Вивести прізвища студентів, які отримують вказану стипендію.

```
CREATE PROCEDURE PR3 (ST DECIMAL (8,2))
RETURNS (Fam VARCHAR(15), Stip DECIMAL (8,2))
AS
BEGIN
 FOR SELECT SFAM, STIP FROM STUDENTS
WHERE STIP=:ST INTO :Fam, :Stip
 DO
 SUSPEND;
END
```

Виклик

```
SELECT * FROM PR3(40);
```

КОНТРОЛЬНІ ЗАПИТАННЯ:

1) Що таке збережені процедури?
2) Якою командою створюються збережені процедури?
3) Які команди і оператори може включати тіло процедури?
4) Які є види збережених процедур?
5) Яким чином викликаються збережені процедури?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2.

Тема: **Побудова представлень та курсорів.**

Мета: Навчитись працювати з представленням та курсором.

Завдання:
1) створити представлення по одній таблиці;
2) створити представлення по декільком таблицям;
3) спробувати внести зміни у предсставлення та створити представлення, яке передає зміну даних у таблицю;
4) створити курсор.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Представлення – це віртуальна таблиця, створена на основі запиту до реальних таблиць. Представлення реалізується як SQL-запит, який можна зберегти. Представлення виконується кожен раз, коли відбувається до нього звертання.

Створення представлення

CREATE VIEW *ім'я представлення* **[(***ім'я_поля* **[,** *ім'я_поля* **...])]**
AS
<select>
[WITH CHECK OPTION];

ім'я_поля – задає ім'я поля в представленні. Кількість імен стовпців, визначені в представленні, має відповідати кількості і порядку стовпців, перерахованих в **<select>.**

<select> – команда, яка вибирає дані з таблиць або інших представлень відповідно заданим умовам.

В *IBExpert* представлення **Views** створюється як будьякий об'єкт бази даних. В результаті формується SQL-команда, яка коригується користувачем. Крім того, у вікні SQL builder будь-який SQL-запит можна зберегти як представлення, якщо натиснути кнопку **Create View from SELECT**

Представлення можуть бути модифіковані, тобто до них можна застосувати команди **INSERT, DELETE** або **UPDATE**.

Але це можливо в тому випадку, якщо усі поля представлення допускають наявність NULL-значень, а <select>-команда, що використовується в представленні, не містить підзапитів, агрегатних функцій, виразів **DISTINCT і HAVING**.

WITH CHECK OPTION – забороняє модифікацію представлень (застосування команд INSERT, DELETE або **UDATE** відповідно до представлень), якщо порушуються умови, задані в конструкції **WHERE**.

Представлення не можуть включати конструкцію **ORDER BY.**

Приклад:

Створити представлення, яке містить інформацію про студентів, які отримують стипендію.

CREATE VIEW STIPEN (NAME, STIPENDIA) AS
SELECT SFAM, STIP
FROM STUDENTS
WHERE STIP<>0;

Знищення представлення

DROP VIEW *ім'я_представлення*;

Представлення можна видалити в тому випадку, якщо воно не використовується в інших представленнях.

Обмеження на створення й використання представлень даних

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

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

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

Переваги та недоліки представлень даних

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

1. Незалежність від даних

Незалежність від даних для користувача полягає в стабілізації структури БД, яка буде залишатися незмінною навіть у разі перегляду структури вихідних відношень.

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

2. Актуальність

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

3. Підвищення захищеності даних

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

4. Додаткові зручності й можливості настроювання

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

5. Зниження складності

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

6. Забезпечення цілісності даних

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

Недоліки

Використання представлень дозволяє досягти істотних переваг, але необхідно звернути увагу на таке:

1. Обмежені можливості оновлення

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

2. Структурні обмеження

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

3. Зниження продуктивності

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

Курсори

Курсори підтримуються в збережених процедурах, функціях і тригерах. Синтаксис такий же, як і у упровадженому SQL. Курсори поки тільки для читання, односпрямовані (тобто по набору можна ходити тільки вперед без можливості повернутися) та несприйнятливі. Несприйнятливість означає, що сервер може створювати копію результуючої таблиці, а може і не створювати, формуючи її на льоту. Курсори повинні бути оголошені до їх використання. Змінні з умовами оголошуються перш курсорів. Обробники оголошуються строго після оголошення курсорів.

Приклад

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE a CHAR(16);
```

```
  DECLARE b,c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

  OPEN cur1;
  OPEN cur2;

REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
      IF b < c THEN
        INSERT INTO test.t3 VALUES (a,b);
      ELSE
        INSERT INTO test.t3 VALUES (a,c);
      END IF;
    END IF;
UNTIL done END REPEAT;

  CLOSE cur1;
  CLOSE cur2;
END
```

Оголошення курсорів

DECLARE cursor_name CURSOR FOR select_statement
Цей вираз оголошує курсор з ім'ям cursor_name. select_statement вказує на конструкцію типу SELECT ... FROM ... Можна оголосити багато курсорів в підпрограмі, але кожен курсор в даному блоці повинен мати унікальне ім'я. Вираз SELECT не повинен містити вказівку INTO.

Відкривання курсорів

OPEN cursor_name
Вираз відкриває раніше оголошений курсор

Вибірка з курсора в змінну

FETCH cursor_name INTO var_name [, var_name] ...
Цей вираз вибирає наступний рядок (якщо рядок існує), використовуючи зазначений відкритий курсор, і просуває покажчик курсору. Якщо більш рядків не є доступними, відбувається зміна значення змінної SQLSTATE в 02000. Для вилову цієї події ви повинні встановити обробник: HANDLER FOR SQLSTATE '02000'

Закриття курсору

CLOSE cursor_name
Закриває курсор cursor_name. Якщо явно не вказано, то курсор закривається автоматично при закритті відповідного блоку підпрограми.

Як використовувати курсори

Застосування курсора в процедурах здійснюється шляхом послідовного виконання наступних кроків:
За допомогою оператора DECLARE оголошується курсор для окремого оператора SELECT або для окремої процедури.
Оператором OPEN проводиться відкриття курсору.
Використовуючи оператор FETCH, здійснюється встановлення покажчика на необхідний запис курсору. При цьому значення полів поточного запису присвоюються змінним, вказуються в операторі FETCH. Зазвичай це конструкція поміщається в ітеративний елемент (простіше кажучи цикл), який переривається по деякому умові.
В процесі переміщення покажчика поточного запису курсора при виході покажчика за межі курсора встановлюється значення SQLSTATE = 02000.
Після того як курсор стає непотрібним, він закривається оператором CLOSE.

Приклади курсорів

Курсор призначений для вибірки даних (ідентифікаторів записів) в рядок з роздільником у вигляді коми по переданим параметрам. Курсор знаходиться всередині збереженої функції get_pedplan (). У неї передається три параметра: lip-номер лабораторії, ti-номер пари і dt - дата проведення заняття.

```
01: CREATE DEFINER = 'for_spammers'@'zoonman.ru' FUNCTION `get_pedplan`(lip INTEGER(11), ti INTEGER(11), dt DATE)
02:     RETURNS char(64) CHARSET latin1
03:     DETERMINISTIC
```

```
04:     CONTAINS SQL
05:     SQL SECURITY INVOKER
06:     COMMENT 'Функція повертає список id з таблиці raspisanie'
07: BEGIN
08:
09:  DECLARE done INT DEFAULT 0;
10:  DECLARE a INT;
11:  DECLARE retv CHAR(64);
12:  DECLARE flg INT;
13:
14:  DECLARE cur1 CURSOR FOR SELECT id FROM raspisanie  WHERE timeintv=ti AND rdate=dt AND labip=lip ;
15:  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;
16:  OPEN cur1;
17:   SET retv  ='' '' ;
18:   SET flg' = 0;
19:   REPEAT
21:    FETCH cur1 INTO a;
22:      IF NOT done THEN
23:        IF flg! =0 THEN
24:          SET retv:= CONCAT(retv,',' , a);
25:        ELSE
26:          SET retv:=a;
27:          SET flg:=1;
28:        END IF;
29:      END IF;
30:   UNTIL done  END REPEAT;
31:   CLOSE cur1;
32:   RETURN retv;
33: END;
```

Курсор визначено на рядку 14. Відкритий на 16 сходинці. З 19-ї розпочато прохід по вибірці отриманої курсором. На кожному кроці циклу відбувається зчитування запису (21-а). Потім, якщо не досягнуть кінець вибірки (22-а), виконується перевірка прапора на перший запис (23-я). Якщо запис перший, то присвоюємо retv поточне значення вибірки (26-а) і встановлюємо прапор (27-а), інакше об'єднуємо значення retv з поточним значенням вибірки (рядок 24). Після проходу по курсору закриваємо його (31-а) і повертаємо значення (32-й рядок).

КОНТРОЛЬНІ ЗАПИТАННЯ:

1) Яка існує різниця між обновлюваним представленням даних та тим, що не оновлюється?
2) Пояснити відмінність між курсорами та представленнями.
3) Пояснити які представлення передають зміни до БД.
4) Пояснити, що таке параметризований запит.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3.

Тема: **Створення тригера.**

Мета: Вивчити поняття тригера.

Завдання:

1. Створити тригер за допомогою оператора CREATE trigger;
2. Для заданої предметної області написати два тригера для різних таблиць бази даних;
3. Видалити тригер за допомогою оператора DROP trigger.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Тригер (Trigger) – це частина програмного коду, що асоціюється з таблицею або виглядом, яка автоматично виконує дії при додаванні, зміні або видаленні запису в таблиці або представленні.

Створення тригера

CREATE TRIGGER *ім'я_тригера* FOR *ім'я_таблиці*
[ACTIVE | INACTIVE]
{BEFORE | AFTER}
{DELETE | INSERT | UPDATE}
[POSITION *номер*]
AS
[DECLARE VARIABLE *ім'я_змінної тип_змінної1*;]
…
[DECLARE VARIABLE *ім'я_змінної тип_змінноїN*;]
BEGIN
команди_InterBase
END

[ACTIVE | INACTIVE] – необов'язковий параметр, який визначає дію тригера після завершення транзакції.

ACTIVE – тригер використовується (за замовчуванням).

INACTIVE – тригер не використовується.

{BEFORE | AFTER} – обов'язковий параметр, який визначає коли відбудеться активізація тригера до події **(BEFORE)** чи після **(AFTER).**

{DELETE | INSERT | UPDATE} – визначає операцію над таблицею, яка викликає тригер для виконання.

[POSITION *номер*] – визначає порядок виклику тригера для обробки однієї події, наприклад при додаванні запису в таблицю. Номер має бути цілим від 0 до 32 767, за замовчуванням дорівнює нулю. Тригер, який має менший номер, виконується раніше.

Синтаксис тригера складається з двох частин: заголовка і тіла. Заголовок включає команду **CREATE TRIGGER**, ім'я тригера, і вище наведені параметри.

Ім'я тригера пропонується формувати по такому правилу: ім'я таблиці_початкові літери параметрів тригера.

Наприклад, **STUDENTS_BI0** – ім'я тригера для таблиці **STUDENTS**, який виконується перед **(BEFORE)** вставкою **(INSERT)** даних з початковим номером 0.

Тіло тригера починається після слова AS, містить перелік описів локальних змінних і блок операторів.

Кожна змінна описується окремо за допомогою команди

DECLARE VARIABLE *ім'я_змінної тип_змінної*;

де тип_змінної – стандартні типи даних InterBase.

В кінці кожної команди ставиться крапка з комою (;).

Опис локальних змінних може бути відсутній.

Блок операторів береться в операторні дужки **BEGIN …END**. Він може містити DML-SQL-конструкції, а також, програмні конструкції **FOR SELECT…DO, IF-THEN,**

WHILE...DO та інші. Кожна команда в блоці завершується крапкою з комою (;). Але крапка з комою є стандартним розділювачом команд в SQL при написанні скриптів. Отже, щоб уникнути помилок при виконанні, необхідно перед командою

створення тригерів, змінювати роздільник команд скрипту на інший символ, а після завершення процедури відновлювати крапку з комою назад.

Це може бути будь-який символ, який не використовується в даній команді, наприклад ^ або !! .

Терм визначається перед створенням тригера командою

SET TERM *терм* ;

А після завершення тригера – командою

SET TERM ; *терм*

Потрібно зазначити, що зміна терма використовується

при роботі в isql-режимі або при написанні скриптів.

Тобто, в скрипті тригер повинний мати такий вигляд

```
SET TERM ^ ;
CREATE TRIGGER Pr1
…
END
^
SET TERM ; ^
```

Після **END** крапка з комою не ставиться. Завершенням команди є символ ^.

У блоці програмного коду тригера можуть використовуватись дві контекстні змінні: **NEW і OLD**. Змінна **OLD**.*ім'я_стовпця* відповідає за старі значення стовпця, змінна **NEW.***ім'я_стовпця* – за нові значення.

Дана таблиця ілюструє використання контекстних змінних **NEW і OLD** в залежності від типу тригера і можливої дії зі змінною.

Таблиця 3.1.

Використання контекстних змінних NEW і OLD в тригерах

Тип тригера	Контекстні змінні			
	NEW		OLD	
	Читання	Зміна	Читання	Зміна
BEFORE INSERT	Y	Y	N/A	N/A
ALTER INSERT	Y	N/A	N/A	N/A
BEFORE UPDATE	Y	Y	Y	N/A
ALTER UPDATE	Y	N/A	Y	N/A
BEFORE DELETE	N/A	N/A	Y	N/A
AFTER DELETE	N/A	N/A	Y	N/A

Y – означає, що використання змінної в даному типі тригера можливе, N/A – означає, що ні.

Приклад:
Використати генератор **GEN_STUDENTS** як лічильник
для поля **SNOM** при додаванні записів у таблицю **STUDENTS**.

```
CREATE TRIGGER STUDENTS_BI0 FOR STUDENTS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
 IF (NEW.SNOM IS NULL) THEN
 NEW.SNOM = GEN_ID(GEN_STUDENTS, 1);
END
```

Модифікація тригера

ALTER TRIGGER *ім'я_тригера*
[ACTIVE | INACTIVE]
[{BEFORE | AFTER} {DELETE | INSERT | UPDATE}]
[POSITION *номер*]
[AS *тіло_тригера*]
[*терм*]

При модифікації можна змінити статус тригера, порядок і спосіб його виконання, внести зміни у тіло тригера. Терм вказується, якщо модифікується тіло тригера.

Приклад:
Змінити статус активного тригера TR1 на пасивний.
ALTER TRIGGER TR1 INACTIVE;

Видалення тригера

DROP TRIGGER *ім'я_тригера*;

КОНТРОЛЬНІ ЗАПИТАННЯ:

1) Що таке тригер?
2) Які є види тригерів?
3) Чи може тригер використовуватись як окрема програма?
4) До яких дій по відношенню до таблиць приводить виконання тригерів?
5) Які оператори використовуються в тригерах?
6) Який зміст мають змінні NEW та OLD в тілі тригера?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4

Тема: **Вивчити механізм формування транзакцій, рівні ізоляції транзакцій і взаємні блокування.**

Мета: Вивчити поняття транзакція, блокування.
Завдання:

1) продемонструвати роботу транзакції (підтвердження та відкат транзакції, точки збереження SAVEPOINT транзакцій);
2) продемонструвати таку властивість транзакції як блокування доступу до даних;
3) продемонструвати робту інших команд блокування даних (lock / unlock та ін.);
4) вивчити які команди проводять автоматичне блокування даних під час роботи;

ТЕОРЕТИЧНІ ВІДОМОСТІ:

В процесі функціонування багатокористувацької БД необхідно забезпечити такі умови, щоб при паралельній обробці декількох завдань дії одних користувачів не чинили непередбаченого впливу на роботу інших. Іноді мета полягає в тому, щоб в умовах паралельної обробки користувач отримав той же результат, як у випадку, якби він був єдиним користувачем БД. В інших випадках мається на увазі, що дії різних користувачів будуть впливати один на одного, але очікуваним чином.

На жаль, не існує стратегії організації паралельної обробки даних, що ідеально підходить для всіх випадків, - все способи припускають певний компроміс між зручністю роботи і забезпеченням несуперечності БД.

Важлива складова частина вирішення зазначеного завдання - використання механізму транзакцій.

Транзакція - це послідовність дій з БД, в якій або всі дії виконуються успішно, або не виконується жодна з них. Транзакція є атомарної, вона виконується як єдине ціле.

Транзакцію можна розглядати як перетворення одного логічно узгодженого стану БД в інше, причому в проміжних точках (тобто під час виконання транзакції) БД може перебувати в неузгоджену стані.

Наприклад, при переказі грошей на оплату покупки з рахунку продавця на рахунок покупця (будемо вважати, що вся інформація зберігається в одній БД) треба з одного рахунку списати зазначену суму, а на іншій - зарахувати. Якщо сума списана, але не зарахована (наприклад, через що стався в момент між цими операціями збою живлення сервера), то виникає внутрішнє протиріччя - гроші втрачаються. Тому така послідовність операцій повинна бути оформлена у вигляді транзакції.

Для вказівки рамок транзакції прикладна програма повинна дати команди на початок транзакції, збереження транзакції, відкат транзакції. Транзакція починається з оператора BEGIN TRANSACTION (в MS SQL Server допустимо скорочення BEGIN TRAN, а деякі СУБД використовують інструкцію START TRANSACTION або SET TRANSACTION). Закінчується транзакція виконанням оператора COMMIT (в запису без скорочень для MS SQL Server - COMMIT TRANSACTION або COMMIT TRAN) або оператора ROLLBACK (ROLLBACK TRANSACTION, ROLLBACK TRAN). Оператор COMMIT встановлює точку фіксації, яка позначає закінчення логічної одиниці роботи. Виконання оператора ROLLBACK повертає БД до попередньої точки фіксації.

Розглянемо приклад з перекладом суми з рахунку покупця на рахунок продавця. Нехай для простоти сума буде фіксована - 100 грн., Дані про рахунки знаходяться в таблиці Accounts, у продавця і покупця унікальні в рамках таблиці імена ("Покупець!" І "Продавец1") і тільки за одним рахунком. Тоді транзакція може виглядати наступним чином:

```
BEGIN TRAN
UPDATE Accounts SET AccountSum = AccountSum-100 WHERE Name = 1 Покупець1 ';
UPDATE Accounts SET AccountSum = AccountSum + 100 WHERE Name = 'Продавець1';
COMMIT TRAN
```

Якщо в процесі виконання подібної транзакції до виконання оператора COMMIT відбудеться збій електроживлення, після процедури відновлення БД буде повернута в стан, що передує початку транзакції (докладніше про це далі в цьому розділі).

Можливо, знадобиться зробити відкат транзакції і в інших випадках. Для цього, зокрема, може використовуватися умовний оператор IF, який визначає умова успішного завершення транзакції:

```
BEGIN TRANSACTION
IF (<умова завершення: ")
COMMIT TRANSACTION
ELSE
ROLLBACK TRANSACTION
```

Крім явних транзакцій, початок і кінець які згідно чітко зазначених у коді, існують ще неявні транзакції. Наприклад, СУБД MS SQL Server за замовчуванням обробляє як неявні транзакції такі оператори SQL, як INSERT, DELETE, UPDATE. Зокрема, якщо оператор UPDATE буде змінювати кілька рядків таблиці і хоча б в один рядок внести зміну не вийде, то вся операція буде скасована, і таблиця повернеться в стан на початок операції.

Управління транзакціями

Для управління транзакціями використовуються наступні команди:

- COMMIT - Зберігає зміни
- ROLLBACK - відкочується (скасовує) зміни
- SAVEPOINT - Створює точку до якої група транзакцій може відкотитися
- SET TRANSACTION - Розміщує ім'я транзакції.

Команди управління транзакціями використовуються тільки для DML команд: INSERT, UPDATE, DELETE. Вони не можуть бути використані під час створення, зміни або видалення таблиці.

Приклади:

Перед початком виконайте наступну команду, для того, щоб відключити автоматичне виконання транзакції:

mysql> SET autocommit=0;

Припустимо, що у нас є таблиця developers, яка містить такі записи:

```
+----+------------------+-----------+------------+--------+
| ID | NAME             | SPECIALTY | EXPERIENCE | SALARY |
+----+------------------+-----------+------------+--------+
|  1 | Eugene Suleimanov | Java      |          2 |   2500 |
|  2 | Peter Romanenko   | Java      |          3 |   3500 |
|  3 | Andrei Komarov    | C++       |          3 |   2500 |
|  4 | Konstantin Geiko  | C#        |          2 |   2000 |
|  5 | Asya Suleimanova  | UI/UX     |          2 |   1800 |
|  7 | Ivan Ivanov       | C#        |          1 |    900 |
|  8 | Ludmila Geiko     | UI/UX     |          2 |   1800 |
+----+------------------+-----------+------------+--------+
```

Видалимо всіх C ++ розробників за допомогою наступної команди:

```
mysql> DELETE FROM developers
    WHERE SPECIALTY = 'C++';
```

mysql> COMMIT;

В результаті виконання даного запиту наша таблиця буде містити такі записи:

```
+----+-------------------+-----------+------------+--------+
| ID | NAME              | SPECIALTY | EXPERIENCE | SALARY |
+----+-------------------+-----------+------------+--------+
|  1 | Eugene Suleimanov | Java      |          2 |   2500 |
|  2 | Peter Romanenko   | Java      |          3 |   3500 |
|  4 | Konstantin Geiko  | C#        |          2 |   2000 |
|  5 | Asya Suleimanova  | UI/UX     |          2 |   1800 |
|  7 | Ivan Ivanov       | C#        |          1 |    900 |
|  8 | Ludmila Geiko     | UI/UX     |          2 |   1800 |
+----+-------------------+-----------+------------+--------+
```

Тепер спробуємо виконати команду **ROLLBACK**:

mysql> ROLLBACK;

Після виконання даної команди наша таблиця містить наступні дані:

```
+----+-------------------+-----------+------------+--------+
| ID | NAME              | SPECIALTY | EXPERIENCE | SALARY |
+----+-------------------+-----------+------------+--------+
|  1 | Eugene Suleimanov | Java      |          2 |   2500 |
|  2 | Peter Romanenko   | Java      |          3 |   3500 |
|  3 | Andrei Komarov    | C++       |          3 |   2500 |
|  4 | Konstantin Geiko  | C#        |          2 |   2000 |
|  5 | Asya Suleimanova  | UI/UX     |          2 |   1800 |
|  6 | Ludmila Geiko     | UI/UX     |          2 |   1800 |
|  7 | Ivan Ivanov       | C#        |          1 |    900 |
+----+-------------------+-----------+------------+--------+
```

Як ми бачимо, запис C ++ розробника знову в таблиці.

Тепер подивимось як працює SAVEPOINT.

Для початку створимо точку збереження, використовуючи наступний запит:

mysql> SAVEPOINT SP1;

Тепер виконаємо наступні запити:

mysql> DELETE FROM developers WHERE ID = 7;
Query OK, 1 row affected (0.00 sec)

mysql> DELETE FROM developers WHERE ID = 6;
Query OK, 1 row affected (0.02 sec)

mysql> DELETE FROM developers WHERE ID = 5;
Query OK, 1 row affected (0.00 sec)

На даний момент наша таблиця містить такі записи:

ID	NAME	SPECIALTY	EXPERIENCE	SALARY
1	Eugene Suleimanov	Java	2	2500
2	Peter Romanenko	Java	3	3500
3	Andrei Komarov	C++	3	2500
4	Konstantin Geiko	C#	2	2000

Тепер ми повернемося до точки збереження SP1 за допомогою команди:

mysql> ROLLBACK TO SP1;

Після виконання даного запиту, наша таблиця буде зберігати такі записи:

ID	NAME	SPECIALTY	EXPERIENCE	SALARY
1	Eugene Suleimanov	Java	2	2500
2	Peter Romanenko	Java	3	3500
3	Andrei Komarov	C++	3	2500
4	Konstantin Geiko	C#	2	2000
5	Asya Suleimanova	UI/UX	2	1800
6	Ludmila Geiko	UI/UX	2	1800
7	Ivan Ivanov	C#	1	900

Як ми бачимо, ми відкотилися до стану таблиці на момент створення точки збереження SP1.

Тепер, коли нам більше не потрібна ця точка збереження, ми можемо її звільнити:

mysql> RELEASE SAVEPOINT SP1;

Розглянемо команду SET TRANSACTION, яка використовується для того, щоб ініціювати транзакцію БД. Дана команда, дозволяє нам визначити характеристики транзакції.

Наприклад, якщо ми хочемо, вказати, що транзакція призначена тільки для читання, то ми повинні виконати такий запит:

SET TRANSACTION READ ONLY;

Якщо ж ми хочемо, щоб транзакція дозволяла виконувати запис даних, то запит буде мати вигляд, вказаний нижче:

SET TRANSACTION READ WRITE;

КОНТРОЛЬНІ ЗАПИТАННЯ:

1) Що таке транзакція?
2) Що означає абревіатура ACID?
3) Якими властивостями володіє транзакція?
4) Який механізм виконання транзакцій?
5) Які проблеми допомагають вирішувати транзакції?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №5.

Тема: **Робота з блокуванням та транзакцією.**

Мета: Проаналізувати рівні ізоляції даних при роботі з транзакціями.

Завдання:
1) створити двох корстувачів та продемонструвати їх паралельну роботу.
2) Написати транзакції та здійснити блокування для власної бази даних;
1 транзакція на оновлення даних в таблиці та її повний відкат;
1 транзакція із точкою відкату та субтранзакціями після неї та відкат до цієї точки;
1 блокування таблиці на читання, спроба перегляду даних та розблокування таблиці.
3) Зміна рівня ізоляції транзакцій на READ UNCOMMITED.
1 транзакція на оновлення даних в таблиці без явного режиму транзакцій.
Перегляд внесених змін. Відкат цієї транзакції.

ТЕОРЕТИЧНІ ВІДОМОСТІ:

Транзакції володіють чотирма наступними властивостями (їх ще називають властивостями ACID - Atomicity, Consistency, Isolation, Durability):

- *атомарність* - транзакція є неподільною, виконуються або всі дії, або нічого:
- *узгодженість* - База даних знаходиться у стабільному, несуперечливому стані. Транзакція переводить базу даних у інший стан, який є стабільним і несуперечливим, тобто не повинна порушуватись цілісність бази даних.
- *ізоляція* - На роботу транзакції не повинні впливати інші транзакції. Результати виконання транзакції будуть доступні іншим транзакціям тільки після її підтвердження.;
- *довговічність* - коли транзакція виконана, її поновлення зберігаються, навіть якщо в наступний момент станеться збій системи.

Коли одночасно відбувається виконання декількох транзакцій, що використовують одні ті ж об'єкти БД, ці транзакції називаються *паралельними* або *конкуруючими*. Якщо не зробити додаткових дій, спрямованих на забезпечення перерахованих вище властивостей ACID, поява конкуруючих транзакцій може призвести до таких проблем, як:

• *проблема втраченого поновлення (англ.* Lost update) - кілька користувачів змінюють одну і ту ж рядок, ґрунтуючись на її початковому значенні, в результаті

частина даних буде втрачено, так як кожна наступна транзакція перезапише зміни, зроблені попередньої;

• *проблема "брудного читання" (англ.* Dirty read) може виникнути при читанні транзакцією записи, яка змінена, але ще не збережена в БД (дані змінені ще не завершилася транзакцією, яка після буде скасована);

• *проблема неповторюваного читання (англ.* Non-repeatable reads) - при повторному читанні даних, вже лічених раніше, транзакція виявляє модифікації або видалення, викликані інший завершеною транзакцією; подібна зміна може порушити логіку роботи транзакції;

• *проблема читання фантомів (англ.* Phantom reads) з'являється, коли при повторному читанні даних транзакція виявляє нові рядки, вставлені або змінені інший транзакцією, завершеною після попереднього читання цього набору даних.

Розглянемо приклад "втраченого поновлення". Нехай БД містить інформацію про товари в магазині іграшок. Перед початком розглянутих паралельних транзакцій в торговому залі знаходилося 10 червоних гумових м'ячів. Користувач А (продавець) вносить інформацію про продаж п'яти м'ячів, користувач В (працівник складу) - про доставку в торговий зал ще двох м'ячів. Проблема полягає в тому, що завдання надійшли одночасно, і СУБД змушена забезпечувати їх "псевдопаралельне" виконання.

Таблиця 5.1

Приклад втраченого поновлення

транзакція А	транзакція В
А1. Вважати запис про кількість червоних гумових м'ячів; (Результат: 10).	В 1. Вважати запис про кількість червоних гумових м'ячів; (Результат: 10).
А2. Зменшити кількість м'ячів на 5; (Результат: 5).	
А3. Записати нове значення в БД; (Результат: в БД 5)	В 2. Збільшити число м'ячів на 2; (Результат: 12).
	У 3. Записати нове значення в БД; (Результат: в БД 12)

Таблиця 5.1 ілюструє виникнення втраченого поновлення через те, що транзакції в ході виконання розщеплюються на окремі чергуються дії. У наведеному прикладі порядок виконання дій був A1 - B1 - A2 - A3 - B2 - B3, в результаті в БД залишився запис про наявність 12 м'ячів, тоді як насправді їх сім.

Щоб уникнути подібних проблем, необхідно забезпечити ізоляцію транзакцій. Основний засіб тут - *блокування ресурсів*-. СУБД нс дає конкуруючим транзакціях отримувати копії однієї і тієї ж записи, коли передбачається швидку зміну цього запису.

У випадку з нашим прикладом транзакція користувача В повинна буде чекати, поки користувач А чи не закінчить роботу з товаром "червоні гумові м'ячі", а тільки потім зможе отримати дані про їх кількість. Якщо послідовність обробки транзакцій буде дотримана, то підсумковий результат буде 7 (10-5 + 2), як і повинно бути.

Блокування можуть накладатися або автоматично, з ініціативи СУБД, або командою, яка віддається СУБД прикладної програмою або запитом користувача. Накладаються СУБД блокування називаються *неявними блокуваннями (англ.* Implicit locks), а блокування, що накладаються за командою користувача, - *явними блокуваннями (англ.* Explicit locks).

Блокування, в залежності від СУБД, можуть накладатися на окремі рядки, сторінки (області фіксованого розміру, наприклад 8 Кбайт, що включають кілька рядків), таблиці, всю БД цілком. Розмір блокується ресурсу називається *глибиною деталізації блокування (англ.* Lock granularity). При більшій глибині деталізації, тобто при блокуванні більших об'єктів, СУБД краще справляється з адмініструванням блокування, але такі блокування частіше викликають конфлікти. При зменшенні розміру блокується фрагмента ймовірність виникнення конфлікту знижується, але такими блокуваннями складніше управляти: СУБД необхідно відстежувати набагато більше деталей.

Два основних типи блокування:

1) монопольна *(англ.* Exclusive lock, X-lock) - блокуються всі види доступу до елементу;

2) колективна *(англ.* Shared lock, S-lock) - елемент стає недоступним для зміни, але його можна прочитати.

Якщо одна транзакція встановила блокування, то інша транзакція при спробі накласти несумісну блокування або виконати заборонене блокуванням дію буде переведена в режим очікування до тих пір, поки блокування не буде знято.

У деяких випадках може виникати ситуація взаємного блокування *(англ.* Deadlock), коли дві транзакції блокують роботу один одного і жодна з них не може продовжити виконання, щоб в результаті зняти блокування. Такі

ситуації повинні відслідковуватися СУБД: зазвичай одна з транзакцій примусово "відкочується", а інша продовжує роботу.

Розібравшись з проблемою втраченого поновлення і питаннями блокування, розглянемо приклади виникнення інших проблем: "брудного" читання, неповторюваного читання, читання фантомів. Нехай вихідна таблиця містить дані про бронювання квитків на авіарейси (табл. 5.2).

Таблиця 5.2

Таблиця Reservations

Flight	Seat	Name
10	7в	NULL
10	7C	NULL

Приклад "брудного" читання може бути наступним. Транзакція А оновлює дані в рядку, після чого відбувається відкат транзакції. Через те, що між цими подіями транзакція В прочитала дані з цього рядка, було отримано некоректне значення. У табл. 5.3 приведена послідовність операцій, яка привела до "брудному" читання.

Таблиця 5.3

Приклад "брудного читання"

транзакція А	транзакція В
UPDATE Reservations SET Name = 'John' WHERE Seat = '7C' AND Flight = 10;	SELECT Name FROM Reservations WHERE Seat = *НС '* AND Flight = 10; (Результат: John)
ROLLBACK	

У табл. 5.4 для тих же вихідних даних показаний приклад неповторюваного читання. Через те, що між зчитування даних транзакцією А транзакція В змінила одну з рядків, один і той же вираз дало різні результати - вдруге отримано менше число вільних місць.

Таблиця 5.4

Приклад неповторюваного читання

транзакція A	транзакція B
SELECT Seat FROM Reservations WHERE Name is NULL AND Flight = 10; (Результат: 7B, 7C)	
	UPDATE Reservations SET Name = 'John' WHERE Seat = T7C 'AND Flight = 10;
SELECT Seat FROM Reservations WHERE Name is NULL AND Flight = 10; (Результат: 7B)	

Проблема читання фантомів може привести до збільшення числа записів, отриманих при повторному виконанні одного і того ж запиту: в табл. 4.5 показано, як повторне читання транзакцією A числа вільних місць повертає більшу кількість місць через внесені транзакцією B змін.

Таблиця 5.5

Приклад читання фантомів

транзакція A	транзакція B
SELECT Seat FROM Reservations WHERE Name is NULL AND	
Flight = 10; (результат: 7B, *1C)*	INSERT INTO Reservations VALUES (',' 7D ', NULL);
SELECT Seat FROM Reservations WHERE Name is NULL AND Flight = 10; (Результат: 7B, 7C, 7D)	

Для вирішення перерахованих вище проблем в стандарті SQL-92 були визначені чотири рівні ізоляції транзакцій. Розробник може вказати бажаний рівень ізоляції (або використовувати рівень, прийнятий за замовчуванням), а СУБД здійснює управління блокуванням відповідно до заданим рівнем. Кожен наступний рівень підтримує вимоги попереднього і накладає додаткові обмеження.

Рівень "незавершене читання" *(англ.* READ UNCOMMITTED) дозволяє уникнути тільки втраченого поновлення. Цей рівень вимагає, щоб змінювати дані могла тільки одна транзакція; якщо інший транзакції необхідно змінити ті ж дані, вона повинна очікувати завершення першої транзакції.

Рівень "завершене читання" *(англ.* READ COMMITTED) додатково дозволяє уникнути "брудного читання". Якщо транзакція початку зміна даних, то ніяка інша транзакція не зможе прочитати їх до завершення першої.

Рівень "відтворюється читання" *(англ.* REPEATABLE READ) на додаток забезпечує повторюваність читання: якщо транзакція зчитує дані, то ніяка інша транзакція не зможе їх змінити, і при повторному читанні вони будуть знаходитися в тому ж стані.

Рівень "серіалізуемость" *(англ.* SERIALIZABLE), найвищий рівень, що дозволяє впоратися з проблемою читання фантомів. Якщо транзакція звертається до даних, то ніяка інша транзакція не зможе додати нові або змінити існуючі рядки, які можуть бути лічені при виконанні транзакції. Подібна блокування накладається нема на конкретні рядки таблиці, а на рядки, що задовольняють певному логічному умові.

Серіалізуемость означає, що дві транзакції обробляються паралельно таким чином, що їх результати узгоджуються з результатами, які можуть бути отримані при послідовній обробці цих транзакцій.

У табл. 5.6 перераховані рівні ізоляції транзакцій.

Треба розуміти, що більш високий рівень ізоляції транзакцій призводить до більшої кількості блокувань і знижує загальну продуктивність системи баз даних. У Microsoft SQL Server 2008 рівень ізоляції за замовчуванням - READ COMMITED.
Можна явно встановити рівень ізоляції (в прикладі - рівень SERIALIZABLE):

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

Різні СУБД можуть реалізовувати відрізняються набори рівнів ізоляції. Так, в Microsoft SQL Server 2008 це READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SNAPSHOT (формує "моментальний знімок" даних на момент їх зчитування в транзакцію, але не підтримує блокування даних), SERIALIZABLE.

Таблиця 5.6

Рівні ізоляції

проблеми	рівні ізоляції			
	Read uncommitted	Read committed	Repeatable read	Serializable
втрачене оновлення	неможливо	неможливо	неможливо	неможливо
"Брудне читання"	можливо	неможливо	неможливо	неможливо
неповторюване читання	можливо	можливо	неможливо	неможливо
фантомне читання	можливо	можливо	можливо	неможливо

Розглянемо тепер механізм забезпечення довговічності транзакції. Потрібно розуміти, що дані з таблиць перед зміною зчитуються з диска в пам'ять, і якийсь час всі зміни БД існують тільки в оперативній пам'яті. При збої системи (наприклад в результаті відключення електроживлення) відбувається втрата вмісту оперативної пам'яті.

Паралельно зі змінами даних з БД СУБД здійснює журнал транзакцій - внесення записів про транзакції в спеціальний файл реєстрації. Для того щоб після перезавантаження система могла визначити, які транзакції необхідно відкинути, а які - ні, проводяться наступні дії.

Перед кожним завершенням операції COMMIT файл реєстрації фізично записується на диск. Це правило ведення файлу реєстрації називається "протокол попереднього запису в журнал".

З певною періодичністю система автоматично встановлює контрольну точку *(англ.* Checkpoint). Це означає, що вміст всіх змінених сторінок БД з оперативної пам'яті записується на диск, і створюється спеціальна фізична запис контрольної точки, яка надає список всіх здійснюваних в даний момент транзакцій. Контрольна точка може бути встановлена і явно по команді користувача або прикладної програми.

Припустимо, що файли на диску пошкоджені не були, і в результаті збою втрачено тільки вміст оперативної пам'яті. Подібний збій іноді називається м'яким. Аналіз журналу транзакцій дозволить привести файли БД в актуальний стан.

На рис. 5.1 показані можливі варіанти виконання транзакцій перед збоєм системи.

Порівнюючи після перезавантаження записи в журналі транзакцій і записи контрольної точки, СУБД приймає рішення про відкат транзакцій Т3 і Т5: вони не завершилися до моменту збою.

Транзакції Т2 і Т4 завершилися, але результати їх роботи залишалися в оперативній пам'яті і пропали через збій. Для цих транзакцій потрібне повторне виконання, так званий накат транзакцій. Транзакція Т1 завершилася до контрольної точки, а значить, результати її виконання записані в БД, і в процесі відновлення вона не зачіпається.

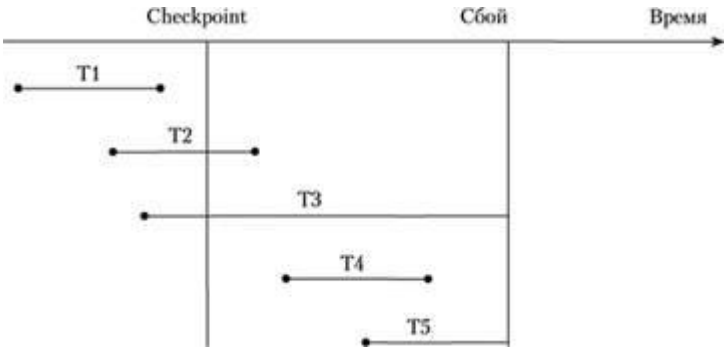

Рисунок 5.1. **Варіанти стану транзакцій перед збоєм**

Збій системи може призвести не тільки до втрати вмісту пам'яті, але і до пошкодження файлів БД, - це так званий жорсткий збій. В цьому випадку необхідно відновлення БД з резервної копії. Якщо з моменту створення резервної копії збереглися всі журнали транзакцій, з'являється можливість привести базу в той стан, який у неї було на момент передував збою. Конкретні дії в даному випадку визначаються тими стратегіями резервного копіювання та відновлення, які пропонує СУБД. Великі системи БД часто мають розподілену структуру. Якщо в розподіленої БД транзакція зачіпає кілька вузлів і на одному з них вироблені дії не пройшли, то відкат змін повинен проводитися також на всіх вузлах. Завдання *управління розподіленими транзакціями* виконує системний компонент, званий *координатором.* Щоб гарантувати дотримання атомарності розподіленої транзакції (то, що вона або виконається цілком, або буде анульована на всіх вузлах), координатор може використовувати *протокол двофазної фіксації*. Суть його полягає в наступному:

- кожному з яких торкається транзакцією вузлів надсилається повідомлення про те, що він повинен зробити;

- кожному вузлу надсилається запит, чи готовий він завершити дану розподілену транзакцію;

- кожен вузол відповідає "так" або "ні";

- коли координатор отримав від всіх вузлів повідомлення "так", він посилає друге повідомлення про те, що транзакцію можна завершити; якщо хоча б один вузол не підтвердив свою готовність завершити транзакцію, то всім надсилається повідомлення про відкат.

Протокол двофазної фіксації не гарантує обов'язкового виконання транзакції. Крім того, конкретні реалізації даного протоколу, як правило, включають вказівку тимчасового інтервалу, протягом якого транзакція повинна бути виконана.

КОНТРОЛЬНІ ЗАПИТАННЯ.

1) Які проблеми виникають при роботі з транзакціями?
2) Які є рівні ізоляції транзакцій?
3) В яких випадках слід застосовувати READ COMMITED транзакції?
4) Назвіть статуси роботи транзакцій.
5) Як може завершуватись транзакція?
6) Які параметри можна задавати транзакціям?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 6

Тема: **Управління правами користувачів.**

Мета: створення, редагування і видалення облікових записів користувачів; призначення і скасування привілеїв.

Завдання:
Створити обліковий запис нового користувача і наділити його певними привілеями;

ТЕОРЕТИЧНІ ВІДОМОСТІ

СУБД MySQL є багатокористувацьким середовищем, тому для доступу до таблиць БД можуть бути створені різні облікові записи з різним рівнем привілеїв. Обліковому запису редактора можна надати привілеї на перегляд таблиці, додавання нових записів і оновлення вже існуючих. Адміністратору БД можна надати більш широкі повноваження (можливість створення таблиць, редагування та видалення вже існуючих). Для користувача БД достатньо лише перегляду таблиць.

Розглянемо наступні питання:

- створення, редагування і видалення облікових записів користувачів;
- призначення і скасування привілеїв.

Обліковий запис є складовою і приймає форму 'username' @ 'host', де username – ім'я користувача, а host – найменування хосту, з якого користувач може звертатися до сервера. Наприклад, записи 'root' @ '127.0.0.1' і 'wet' @ '62.78.56.34' означають, що користувач з ім'ям root може звертатися з хосту, на якому розташований сервер, а wet – тільки з хоста з IP-адресою 62.78.56.34.

IP-адреса 127.0.0.1 завжди відноситься до локального хоста. Якщо сервер і клієнт встановлені на одному хості, то сервер прослуховує з'єднання за цією адресою, а клієнт відправляє на нього SQL-запити.

IP-адреса 127.0.0.1 має псевдонім localhost, тому облікові записи виду 'root' @ '127.0.0.1' можна записувати у вигляді 'root' @ 'localhost'.

Число адрес, з яких необхідно забезпечити доступ користувачеві, може бути значним. Для завдання діапазону в імені хоста використовується спеціальний символ

"%". Так, обліковий запис 'wet' @ '%' дозволяє користувачеві wet звертатися до сервера MySQL з будь-яких комп'ютерів мережі.

Усі облікові записи зберігаються в таблиці user системної бази даних з ім'ям mysql. Після першої інсталяції вміст таблиці user виглядатиме так, як показано в лістингу.

```
mysql> SELECT Host,User,Password FROM mysql.user;
+----------------------+------+----------+
| Host                 | User | Password |
+----------------------+------+----------+
| localhost            | root |          |
| production.mysql.com | root |          |
| 127.0.0.1            | root |          |
| localhost            |      |          |
| production.mysql.com |      |          |
+----------------------+------+----------+
5 rows in set (0.27 sec)
```

Створення нового облікового запису.

Створити обліковий запис дозволяє оператор

CREATE USER 'username' @ 'host' [IDENTIFIED BY [PASSWORD] 'пароль'];

Оператор створює новий обліковий запис з необов'язковим паролем. Якщо пароль не вказано, в його якості виступає порожній рядок. Розумно зберігати пароль у вигляді хеш-коду, отриманого в результаті незворотного шифрування. Щоб скористатися цим механізмом авторизації, необхідно помістити між ключовим словом identified by і паролем ключове слово password.

Видалення облікового запису.

Видалити обліковий запис дозволяє оператор

DROP USER 'username' @ 'host';

Зміна імені користувача облікового запису.

Здійснюється за допомогою оператора

RENAME USER старе_ім'я ТО нове_ ім'я;

Призначення привілеїв.

Розглянуті вище оператори дозволяють створювати, видаляти і редагувати облікові записи, але вони не дозволяють змінювати привілеї користувача – повідомляти MySQL, який користувач має право тільки на читання інформації, який на

читання і редагування, а кому надані права змінювати структуру БД і створювати облікові записи.

Для вирішення цих завдань призначені оператори grant (призначає привілеї) і revoke (видаляє привілеї). Якщо облікового запису, яка показана оператора grant, не існує, то вона автоматично створюється. Видалення всіх привілеїв з допомогою оператора revoke не призводить до автоматичного знищення облікового запису. Для видалення користувача необхідно скористатися оператором drop user.

У найпростішому випадку оператор grant виглядає наступним чином:

```
mysql> GRANT ALL ON *.* TO 'wet'@'localhost' IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.17 sec)
```

Даний запит створює користувача з ім'ям wet і паролем pass, який може звертатися до сервера з локального хоста (localhost) і має всі права (all) для всіх баз даних (*. *). Якщо такий користувач існує, то його привілеї будуть змінені на all.

Замість ключового слова all можна використовувати будь-яке з ключових слів, представлених в табл. 6. 1. Ключове слово on в операторі grant задає рівень привілеїв, які можуть бути задані на одному з чотирьох рівнів, представлених в табл. 6. 2. Для таблиць можна встановити тільки такі типи привілеїв: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX і ALTER. Це слід враховувати при використанні конструкції grant all, яка призначає привілеї на поточному рівні. Так, запит рівня бази даних grant all on db. * Не надає ніяких глобальних привілеїв.

Скасування привілеїв.

Для скасування привілеїв використовується оператор revoke:

```
mysql> REVOKE DELETE, UPDATE ON *.* FROM 'wet'@'localhost';
Query OK, 0 rows affected (0.02 sec)
```

Оператор revoke скасовує привілеї, але не видаляє облікові записи, для їх видалення необхідно скористатися оператором drop USER.

Таблиця 6. 1

Операції дозволені привілеями

Привілея	Операція, дозволена привілеєм
ALL [PRIVILEGES]	Комбінація всіх привілеїв, за винятком привілеї *GRANT option*, яка задається окремо
ALTER	Дозволяє редагувати таблицю за допомогою оператора *ALTERTABLE*
ALTER ROUTINE	Дозволяє редагувати або видаляти збережену процедуру
CREATE	Дозволяє створювати таблицю за допомогою оператора *CREATETABLE*
CREATE ROUTINE	Дозволяє створювати збережену процедуру
CREATE TEMPORARY TABLES	Дозволяє створювати тимчасові таблиці
CREATE USER	Дозволяє працювати з обліковими записами с допомогою *CREATE USER, DROP USER, RENAME USER і REVOKE ALL PRIVILEGES*
CREATE VIEW	Дозволяє створювати уявлення за допомогою оператора *CREATE VIEW*
DELETE	Дозволяє видаляти записи за допомогою оператора *delete*
DROP	Дозволяє видаляти таблиці за допомогою оператора *DROPTABLE*
EXECUTE	Дозволяє виконувати збережені процедури
INDEX	Дозволяє працювати з індексами, зокрема, використовувати оператори *CREATE INDEX і DROP INDEX*
INSERT	Дозволяє додавати в таблицю нові записи оператором insert
LOCK TABLES	Дозволяє здійснювати блокування таблиць за допомогою операторів *LOCK TABLES і UNLOCK TABLES*. Для вступу в дію цього привілею повинна бути встановлена привілей *SELECT*
select	Дозволяє здійснювати вибірки таблиць оператором *SELECT*
Show databases	Дозволяє переглядати список всіх таблиць на сервері за допомогою оператора *show databases*
Show view	Дозволяє використовувати оператор *show create view*

UPDATE	Дозволяє оновлювати вміст таблиць оператором UPDATE
USAGE	Синонім для статусу «відсутні привілеї»
GRANT OPTION	Дозволяє управляти привілеями інших користувачів, без цього привілею неможливо виконати оператори grant і *REVOKE*

Таблиця 6. 2

Рівні привілеїв

ключове слово *ON*	**Рівень**
*ON *.**	Глобальний рівень - користувач із правами на глобальному рівні може звертатися до всіх БД і таблиць, що входять до їх складу
*ON db.**	Рівень бази даних - привілеї поширюються на таблиці бази даних *db*
ON db.tbl	Рівень таблиці - привілеї поширюються на таблицю *tbl* бази даних db
ON db.tbl	Рівень стовпчика - привілеї стосуються окремих стовпців в таблиці tbl бази даних db. Список стовпців вказується в дужках через кому після ключових слів *select, insert, update*

Приклад.

Для роботи виберемо два комп'ютери, які підключені до локальної мережі. На одному необхідно розгорнути сервер MySQL, на інший - скопіювати клієнт командного рядка mysql.exe. Визначимо IP-адреса сервера:

```
C:\Documents and Settings\admin>ipconfig

Настройка протокола IP для Windows


Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 192.168.67.6
        Маска подсети . . . . . . . . . . : 255.255.255.0
        Основной шлюз . . . . . . . . . . : 192.168.67.254
```

Створимо новий обліковий запис, дозволивши користувачеві user1 звертатися до сервера MySQL з будь-яких комп'ютерів мережі:

```
mysql> CREATE USER 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.01 sec)
```

Призначимо цьому користувачеві привілеї глобального рівня:

```
mysql> GRANT ALL ON *.* TO 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.00 sec)
```

На клієнтському комп'ютері в командному рядку (наприклад, за допомогою FAR), запустимо клієнт командного рядка в наступному форматі:

```
The FAR manager, version 1.70 beta 5 (build 1634)
Copyright (C) 1996-2000 Eugene Roshal, Copyright (C) 2000-2003 FAR Group
Зарегистрирован: xUSSR регистрация
C:\>mysql -u user1 -p123 -h 192.168.67.6
```

Спостерігаємо відгук віддаленого сервера і працюємо з ним як зазвичай:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.51b-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| book               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

КОНТРОЛЬНІ ЗАПИТАННЯ:

1) Яку інформацію містить зареєстрований запис про користувача?
2) Який користувач має повний доступ до бази даних?
3) Які права мають користувачі?
4) Яка команда використовується для надання прав користувачам?

Варіанти завдань

1. *Бібліотека.* Бібліотека вирішила заробляти гроші, видаючи напрокат книги, які є в невеликій кількості екземплярів. Кожна книга, що видається в прокат, має назву, автор, жанр. Залежно від цінності книги для кожної з них визначена заставна вартість (сума, яку клієнт вносить при взятті книги напрокат) і вартість прокату (сума, яку клієнт платить при поверненні книги, одержуючи назад заставу). Читачі реєструються в картотеці, яка містить стандартні анкетні дані (прізвище, ім'я, по батькові, адресу, телефон). Кожен читач може звертатися до бібліотеки кілька разів. Всі звернення читачів фіксуються, при цьому за кожним фактом видачі книги запам'ятовуються дата видачі та очікувана дата повернення. Вартість прокату книги має залежати не лише від самої книги, а й від терміну її прокату. Крім того, необхідно додати систему штрафів за шкоду, завдану книзі, та систему знижок для деяких категорій читачів.

2. *Прокат автомобілів.* Фірма, що займається прокатом автомобілів, має автопарк, що містить кілька автомобілів різних марок, цін і типів. Кожен автомобіль має власну вартість прокату. До пункту прокату звертаються клієнти. Клієнти проходять обов'язкову реєстрацію, під час якої про них збирається стандартна інформація (прізвище, ім'я, по батькові, адресу, телефон). Кожен клієнт може звертатися до пункту прокату кілька разів. Звернення клієнтів фіксуються, при цьому за кожною угодою запам'ятовуються дата видачі та очікувана дата повернення. Вартість прокату автомобіля має залежати не тільки від самого автомобіля, а й від терміну його прокату, а також від року випуску. Також потрібно ввести систему штрафів за повернення автомобіля у неналежному вигляді та систему знижок для постійних клієнтів.

3. *Страхова компанія.* Страхова компанія має філії, які характеризуються найменуванням, адресою та телефоном. До філій звертаються клієнти з метою укладання договору страхування. Залежно від об'єктів, що приймаються на страхування, та страхованих ризиків договір укладається за певним видом страхування (страхування автотранспорту від угону, страхування домашнього майна, добровільне медичне страхування). При укладанні договору фіксуються: дата укладання, страхова сума, вид страхування, тарифна ставка та філія, в якій укладався договір. Договори укладають страхові агенти. Крім інформації про агентів (прізвище, ім'я, по батькові, адреса, телефон) потрібно зберігати філію, в якій вони працюють. Потрібно мати можливість розраховувати заробітну плату агентам. Заробітна плата становить певний відсоток від страхового платежу (платіж – страхова сума, помножена на тарифну ставку). Відсоток залежить від виду страхування, за яким укладено договір.

4. *Готель.* Готель надає номери клієнтам. Кожен номер характеризується місткістю, комфортністю (люкс, напівлюкс, звичайний) та ціною. Про клієнтів збирається певна інформація (прізвище, ім'я, по батькові, паспортні дані, адреса проживання та деякий коментар). Надання номера клієнту здійснюється за наявності вільних місць у номерах, що

підходять клієнту за вказаними вище параметрами. Під час заселення фіксується дата заселення. При виїзді з готелю кожного місця запам'ятовується дата звільнення. Необхідно також бронювати номери. Для постійних клієнтів, а також для певних категорій клієнтів передбачено систему знижок. Знижки можуть сумуватися.

5. *Оптово-роздрібний продаж товарів.* Компанія торгує товарами із певного спектру. Кожен товар характеризується найменуванням, оптовою ціною, роздрібною ціною та довідковою інформацією. До компанії звертаються покупці, кожному з яких у базі даних фіксуються стандартні дані (найменування, адреса, телефон, контактна особа). По кожній угоді складається документ, у якому поруч із покупцем фіксуються кількість купленого ним товару та дата купівлі. Зазвичай покупці у межах однієї угоди купують не один товар, а відразу кілька. Також компанія вирішила надавати знижки в залежності від кількості закуплених товарів та їхньої загальної вартості.

6. *Ведення замовлень.* Компанія займається оптовим продажем різних товарів. Кожен із товарів характеризується ціною, довідковою інформацією та ознакою наявності чи відсутності доставки. До компанії звертаються замовники. Для кожного з них у базі даних запам'ятовуються стандартні дані (найменування, адреса, телефон, контактна особа). За кожним замовленням складається документ, у якому поряд із замовником фіксуються кількість купленого ним товару та дата купівлі. Доставка товарів може здійснюватися способами, різними за ціною та швидкістю. Потрібно зберігати інформацію про те, якими способами може здійснюватися доставка кожного товару, та інформацію про те, який вид доставки (і яку вартість доставки) вибрав клієнт під час укладання угоди.

7. *Бюро з працевлаштування.* Бюро готове шукати працівників для різних роботодавців та вакансії для фахівців різного профілю, що шукають роботу. При зверненні до бюро роботодавця його стандартні дані (назва, вид діяльності, адреса, телефон) фіксуються у базі даних. При зверненні до бюро претендента його стандартні дані (прізвище, ім'я, по батькові, кваліфікація, професія, інші дані) також фіксуються у базі даних. За кожним фактом задоволення інтересів обох сторін складається документ. У документі зазначаються претендент, роботодавець, посада та комісійні (дохід бюро). У базі має фіксуватися не лише угода, а й інформація щодо відкритих вакансій. Крім того, для автоматичного пошуку варіантів необхідно вести довідник «Види діяльності».

8. *Фірма з продажу запчастин.* Фірма продає запасні частини для автомобілів. Фірма має певний набір постачальників, за якими відомі назва, адреса та телефон. У постачальників купуються деталі. Кожна деталь характеризується назвою, артикулом та ціною. Деякі постачальники можуть постачати однакові деталі (один артикул). Кожен факт купівлі запчастин у постачальника фіксується у базі даних, причому обов'язковими для запам'ятовування є дата купівлі та кількість придбаних деталей. Ціна деталі може змінюватися від постачання до постачання. Постачальники заздалегідь повідомляють фірму

про дату зміни ціни та її нове значення. Потрібно зберігати як поточне значення ціни, а й усю історію зміни цін.

9. *Курси підвищення кваліфікації.* У навчальному закладі організовано курси підвищення кваліфікації. Групи слухачів формуються залежно від спеціальності та відділення. У кожну з них включено певну кількість слухачів. Проведення занять забезпечує штат викладачів, кожному з яких у базі даних зареєстровані стандартні анкетні дані (прізвище, ім'я, по батькові, телефон) та стаж роботи. Внаслідок розподілу навантаження отримано інформацію про те, скільки годин занять проводить кожен викладач із відповідними групами. Зберігаються також відомості про вид занять (лекція, практика), дисципліну та оплату за 1 годину. Розмір погодинної оплати залежить від предмета та типу заняття. Крім того, кожен викладач може вести не всі предмети, а тільки деякі.

10. *Нотаріальна контора.* Нотаріальна контора готова надати клієнту певний комплекс послуг. Послуги формалізовані, тобто складено їх список з описом кожної послуги. При зверненні клієнта його стандартні дані (назва, вид діяльності, адреса, телефон) фіксуються у базі даних. За кожним фактом надання послуги клієнту складається документ, у якому зазначаються дата, послуга, сума угоди, комісійні (дохід контори), опис угоди. У межах угоди клієнту може бути надано кілька послуг. Вартість кожної послуги фіксована. Крім того, компанія надає в рамках однієї угоди різні види знижок. Знижки можуть підсумовуватись.

11. *Визначення факультативів для студентів.* Викладачі кафедри у вищому навчальному закладі забезпечують проведення факультативних занять із деяких предметів. Є відомості про студентів, які включають стандартні анкетні дані (прізвище, ім'я, по батькові, група, адреса, телефон). По кожному факультативу існує певна кількість годин і вид занять, що проводяться (лекції, практика, лабораторні роботи). В результаті роботи зі студентами з'являється інформація про те, хто з них записався на якісь факультативи. Існує певний мінімальний обсяг факультативних предметів, які має прослухати кожен студент. Після закінчення семестру до бази даних заноситься інформація про оцінки, отримані студентами на іспитах. Деякі з факультативів можуть тривати більше семестру. У кожному семестрі для предмета встановлюється обсяг лекцій, практик та лабораторних робіт у годинах. Як підсумкову оцінку за предмет береться остання оцінка, отримана студентом.

12. *Розподіл навчального навантаження.* Необхідно розподіляти навантаження між викладачами кафедри. Є відомості про викладачів, що включають поряд з анкетними даними відомості про їх науковий ступінь, посаду та стаж роботи. Викладачі кафедри мають забезпечити проведення занять із деяких дисциплін. По кожній із них існує певна кількість годин. В результаті розподілу навантаження необхідно отримати інформацію такого роду: «Такий-то викладач проводить заняття з такої дисципліни з такою групою». Усі заняття діляться на лекційні і практичні. По кожному виду занять встановлюється кількість годин. Крім того, дані з навантаження потрібно зберігати кілька років.

13. *Облік телекомпанією вартості реклами, що пройшла в ефірі.* Робота комерційної служби телевізійної компанії побудована так: замовники просять помістити свою рекламу у певній передачі у певний день. Кожен рекламний ролик має певну тривалість. Для кожної організації-замовника відомі банківські реквізити, телефон та контактна особа для проведення переговорів. Передачі мають певний рейтинг. Вартість хвилини реклами у кожній передачі визначається, з рейтингу передачі та інших міркувань. Необхідно зберігати інформацію про агентів, які уклали договори реклами. Зарплата рекламних агентів становить певний відсоток загальної вартості реклами, що пройшла в ефірі.

14. *IT-компанія.* Компанія надає IT-послуги організаціям та підприємствам. У компанії працюють співробітники, про яких має зберігатися стандартна інформація та дані про кваліфікацію (володіння мовами та системами програмування, знання СУБД, операційних систем). До компанії звертаються клієнти, про яких збираються стандартні дані (найменування та адреса організації, телефон, адреса електронної пошти, прізвище, ім'я та по батькові контактної особи для зв'язку). Завдання для клієнта виконує певний співробітник, фіксується дата видачі завдання і трудомісткість виконання (у годинах). При повторному зверненні клієнт переходить до категорії постійних та отримує знижку. Зі зростанням компанії виникла необхідність поділу її на відділи. Збільшилися масштаби проектів і тепер завдання клієнта доручається відділу. У межах договору може виконуватися кілька завдань різними відділами компанії.

15. *Вантажні перевезення.* Компанія здійснює перевезення вантажів різними маршрутами. Необхідно відстежувати вартість перевезень із урахуванням заробітної плати водіїв. Для кожного маршруту визначено назву, обчислено приблизна відстань та встановлено деяку оплату для водія. Інформація про водіїв включає прізвище, ім'я, по батькові та стаж. Для проведення розрахунків зберігається повна інформація про перевезення (маршрут, водій, дати відправлення та прибуття). За фактом деяких перевезень водіям сплачується премія. Фірма вирішила запровадити гнучку систему оплати. Оплата водіям має залежати не лише від маршруту, а й від стажу водія. Крім того, потрібно врахувати, що перевезення можуть здійснювати два водії.

16. *Облік телефонних переговорів.* Телефонна компанія надає абонентам телефонні лінії для міжміських переговорів. Абонентами компанії є юридичні особи, які мають телефонну точку, ідентифікаційний номер, розрахунковий рахунок у банку. Вартість переговорів залежить від міста, в яке здійснюється дзвінок, та часу доби (день, ніч). Кожен дзвінок абонента автоматично фіксується у базі даних. При цьому запам'ятовуються місто, дата, тривалість розмови та доби. Компанія вирішила запровадити гнучку систему знижок. Так, вартість хвилини тепер зменшується залежно від тривалості розмови. Розмір знижки для кожного міста різний.

17. *Врахування внутрішньоофісних витрат.* Співробітники приватної фірми можуть здійснювати дрібні покупки потреб фірми, надаючи в бухгалтерію товарний чек. Бухгалтерія

відстежує внутрішньоофісні витрати. Фірма складається з відділів, кожен із яких має назву. У кожному відділі працює певна кількість працівників. Співробітники можуть купувати відповідно до видів витрат. Кожен вид витрат має назву, деякий опис та граничну суму коштів, які можуть бути витрачені за цим видом витрат на місяць. При кожній купівлі працівник оформляє документ, де вказує вид витрати, дату, суму та відділ. Потрібно зберігати дані про витрати не тільки загалом по відділу, а й по окремих співробітниках. Нормативи щодо витрачання коштів встановлюються не загалом, а щодо кожного відділу за кожен місяць. Невикористані у поточному місяці гроші можуть бути використані пізніше.

18. *Платна клініка.* У поліклініці працюють лікарі різних спеціальностей, які мають різну кваліфікацію. Щодня до поліклініки звертаються пацієнти. Усі пацієнти проходять обов'язкову реєстрацію, коли до бази даних заносяться стандартні анкетні дані (прізвище, ім'я, по батькові, рік народження, адресу). При зверненні до поліклініки пацієнт обстежується та проходить лікування у різних спеціалістів. Кожен пацієнт може звертатися до поліклініки кілька разів, потребуючи різної медичної допомоги. Усі звернення пацієнтів фіксуються, встановлюється діагноз, визначається вартість лікування, запам'ятовується дата звернення. Загальна вартість лікування залежить від вартості консультацій та процедур, призначених пацієнтові. Для певних категорій громадян передбачені знижки.

19. *Аналіз динаміки показників фінансової звітності підприємств.* Інформаційно-аналітичний центр холдингу відстежує динаміку показників підприємств холдингу. До структури холдингу входять кілька підприємств. Кожне підприємство має стандартні характеристики (назва, реквізити, телефон, контактна особа). Роботу підприємства можна оцінити так: на початку кожного звітного періоду з урахуванням фінансової звітності обчислюється певний набір показників. Важливість показників характеризується деякими числовими константами. Значення кожного показника вимірюється у певній системі одиниць. Деякі показники вважаються у гривнях, деякі у доларах, деякі у євро. Для зручності роботи з показниками необхідно зберігати зміни курсів валют відносно один до одного.

20. *Видача банком кредитів.* Комерційний банк видає кредити юридичним особам. В залежності від умов отримання кредиту, відсоткової ставки і терміну повернення всі кредитні операції діляться на кілька основних видів. Кожен із цих видів має свою назву. Кредит може отримати юридична особа (клієнт), що при реєстрації надав такі відомості: назву, вид власності, адресу, телефон, контактну особу. Кожен факт видачі кредиту реєструється банком, причому фіксуються сума кредиту, клієнт і дата видачі. Щоб відстежувати динаміку повернення кредитів, прийнято рішення враховувати в системі ще й дату фактичного повернення грошей. Потрібно ще врахувати, що кредит може гаситися частинами, і за затримку повернення кредиту нараховуються штрафи.

21. *Інвестиційна компанія.* Компанія займається вкладенням коштів у цінні папери, що характеризуються рейтингом, прибутковістю за минулий рік, мінімальною сумою угоди та деякою додатковою інформацією. Клієнтами компанії є підприємства, які довіряють їй

керувати їх вільними коштами на певний період. Необхідно вибрати вид цінних паперів, які дозволять отримати прибуток компанії та клієнту. При роботі з клієнтом суттєвою є інформація про підприємство – назву, вид власності, адресу та телефон. Кожна інвестиція характеризується інформацією про клієнта, інформацією про цінні папери, котируванням паперу, датою його купівлі та датою його продажу. Необхідно зберігати історію котирувань кожного цінного паперу. Крім того, крім вкладень у цінні папери, існує можливість вкладати гроші в банківські депозити.

22. *Зайнятість акторів театру.* Комерційний директор театру організує залучення акторів та укладання контрактів. Щороку театр здійснює постановку різних вистав. Кожна вистава має певний бюджет. Для участі у конкретних постановках у певних ролях залучаються актори. З кожним із акторів укладається персональний контракт на певну суму. Кожен актор має стаж роботи, деякі з них удостоєні різних звань. В рамках одного спектаклю на ту саму роль залучається кілька акторів. Договір визначає базову зарплату актора, а за підсумками реально відіграних вистав актору призначається премія. У базі даних слід зберігати інформацію протягом кількох років.

23. *Ювелірна майстерня.* Ювелірна майстерня здійснює виготовлення ювелірних виробів для приватних осіб на замовлення. Майстерня працює з певними матеріалами (платина, золото, срібло, дорогоцінне каміння). При зверненні потенційного клієнта з'ясовується, який виріб йому необхідний. Всі вироби, що виготовляються, належать до деякого типу (сережки, кільця, брошки, браслети), виконуються з певного матеріалу, мають деяку вагу і ціну (що включає вартість матеріалів і роботи). Ювелірний виріб може складатися з кількох матеріалів. Крім того, постійним клієнтам майстерня надає знижки.

24. *Перукарня.* Перукарня стриже клієнтів відповідно до їх побажань та деякого каталогу різних видів стрижки. Для кожної стрижки визначено назву, категорію (чоловічу, жіночу, дитячу), вартість роботи. Для наведення порядку складається база даних клієнтів, де зберігаються їх анкетні дані (прізвище, ім'я, по батькові). Починаючи з п'ятої стрижки, клієнт переходить у категорію постійних і отримує знижку в 3% при кожній наступній стрижці. Після того, як закінчено чергову роботу, в БД фіксуються стрижка, клієнт і дата виконання робіт. Крім того, у перукарні з'явилася філія і необхідна окрема статистика з філій. Вартість стрижки може змінюватися з часом. Потрібно зберігати не лише останню ціну, а й усі дані щодо зміни ціни стрижки.

25. *Хімчистка.* Хімчистка здійснює прийом у населення речей для виведення плям. Для наведення порядку складається база даних клієнтів, де зберігаються їх анкетні дані (прізвище, ім'я, по батькові, адресу, телефон). Починаючи з 3-го звернення, клієнт переходить у категорію постійних клієнтів і отримує знижку в 5% під час чищення кожної наступної речі. Всі послуги поділяються на види, що мають назву, тип і вартість, що залежить від складності робіт. Робота з клієнтом спочатку полягає у визначенні обсягу робіт, виду послуги та, відповідно, її вартості. Якщо клієнт погоджується, він залишає річ (при цьому

фіксується послуга, клієнт та дата прийому) та забирає її після обробки (при цьому фіксується дата повернення). Хімчистка укладає із клієнтом договір. Клієнт може одночасно здавати в чищення кілька речей. У хімчистки з'явилися філії, і потрібна окрема статистика з філій. Введені надбавки за терміновість та складність.

26. *Оренда торгових площ.* Торговий центр здає у найм комерсантам свої торгові площі. В результаті планування визначено деяку кількість торгових точок у межах будівлі, яка може здаватися в оренду. Для кожної з торгових точок важливими даними є поверх, площа, кондиціонер і вартість оренди в день. З потенційних клієнтів збираються стандартні дані (назва, адреса, телефон, реквізити, контактна особа). З появою потенційного клієнта йому показують вільні площі. При досягненні угоди оформляється договір і базі даних фіксується торгова точка, клієнт, період (термін) оренди. Деякі клієнти в рамках одного договору орендують відразу кілька торгових точок, причому для кожної точки можливий термін оренди. Дата укладання договору може збігатися з датою початку оренди. Необхідно збирати інформацію про щомісячні платежі, що надходять від орендарів.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА:

1. Берко А. Ю. Системи баз даних та знань: навч. посіб. / А. Ю. Берко, О. М. Верес, В. В. Пасічник; - Львів : Магнолія, 2008. - 456 с. - Мін-во освіти і науки України. - (Серія «Комп'ютинг» / за заг. ред. В. В. Пасічника; [кн. 1]).

2. James R. Groff, Paul N. Weinberg SQL: The Complete Reference (Second Edition) Copyright 2002 by The McGraw-Hill Companies, Inc. Click Here for Terms of Use. -2002 -1025 p.

3. Пасічник В. В., Резніченко В. А. Організація баз даних та знань: Підручник К.: Видавнича група BHV, 2006. -384 с.

4. Фиайли К. SQL: Руководство по изучению языка [Електронний ресурс] / Крис Фиайли. – 2013. – Режим доступу до ресурсу: http://smotrim.by/books/4221-sql-rukovodstvo-po-izucheniyu-yazyka-kris-fiayli2013.html).

5. Уроки SQL [Електронний ресурс] - Режим доступу до ресурсу: http://moonexcel.com.ua/уроки-sql.

6. Управляючі Конструкції sql. [Електронний ресурс] - Режим доступу до ресурсу: https://studfiles.net/preview/5210288/page:2/)

7. Організація баз даних та знань. Реляційна алгебра. [Електронний ресурс] – Режим доступу до ресурсу: http://bookwu.net/book_organizaciya-baz-danih-iznan_997/27_2.10-realizaciya-relyacijno-algebri.

8. Жежнич П. І. Консолідовані інформаційні ресурси баз даних та знань. -.Львів, Вид. Львівської політехніки, 2010 г. - 212 с.

9. Ситник Н. В. Проектування баз і сховищ даних: Навч.-метод. посіб. для самост. вивч. Дисц. / Н. В. Ситник, М. Т. Краснюк – К. : КНЕУ, 2005. – 264 с.

10. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ / Дейт, К. Дж. – М.: Издательский дом «Вильямс», 2005. – 1328 с. – (8-е издание)