Н.Р. Балик,
В.І. Мандзюк

БАЗИ ДАНИХ MySQL

Навчальний посібник

ТЕРНОПІЛЬ
НАВЧАЛЬНА КНИГА – БОГДАН

ББК 32.973.26-018.2
Б20

Балик Н.Р., Мандзюк В.І.

Б20 Бази даних MySQL: Навчальний посібник. — Тернопіль: Навчальна книга – Богдан, 2010.— 160 с.

ISBN 978-966-10-0906-5

У пропонованому посібнику розглянуто мову структурованих запитів для взаємодії з базами даних MySQL, починаючи з доступного викладу теорії нормалізації відношень. Основою матеріалу є вивчення прикладів готової бази даних. На практичних прикладах детально описано основні конструкції мови, розглядаються прийоми конструювання запитів: від найпростіших до найскладніших.

Книга орієнтована, перш за все, на початківців (учнів, студентів, учителів), котрі бажають вивчити основи роботи із сучасними базами даних.

ББК 32.973.26-018.2

ISBN 978-966-10-0906-5

1. Основи проектування баз даних

Розглянемо основні принципи проектування баз даних і питання нормалізації. Правильне проектування бази даних мінімізує надлишковість, причому без втрати даних. Метою проектування є використання мінімуму дискового простору і пам'яті для бази даних за умови збереження всіх зв'язків між даними.

Об'єкти та відношення

У процесі моделювання предметна галузь подається у вигляді сукупності об'єктів та зв'язків між ними. Об'єкти — це елементи, процеси реального світу, дані про які зберігають у базі даних. Візьмемо для прикладу сукупність книг, виставлених у бібліотеці. Наприклад, можна зберігати дані про книги і про рубрики (за тематикою), до яких ці книги відносяться. У цьому випадку книги представлятимуть один об'єкт, а групи — інший. Відношення — це зв'язки між об'єктами. Наприклад, книжка належить до певної групи. У цьому випадку «належить до» представляє зв'язок між вказаною книгою і об'єктами, що представляють групи.

Відношення можуть бути різного типу. Вони можуть бути взаємно-однозначними (відношення типу «один до одного»), типу «один до багатьох» (або «багато до одного», залежно від напряму) або типу «багато до багатьох». Взаємно-однозначне відношення зв'язує два об'єкти. Якби до кожної групи належала тільки одна книжка, вказане вище відношення було б взаємно-однозначним. Відношення «належить до» в даному прикладі зазвичай виявляється зв'язком типу «багато до одного», тобто багато книжок належать до тієї самої групи, але кожна з них належить тільки до однієї групи. Обидва типи відношень показані на рис. 1.1.

Зверніть увагу на те, що об'єкти, відношення і типи відношень залежать від умов і правил реального завдання, яке ви намагаєтеся моделювати. Так, у деяких книгарнях одну книгу можуть віднести до кількох груп (наприклад, «Між планетами» до групи «Зарубіжна література» і «Фантастика»). У цьому випадку зв'язок «належить до» має тип «багато до багатьох». Також, як правило, є в наявності більш ніж одна книжка певного автора, в цьому випадку зв'язок «написано ким» вже не буде взаємно-однозначним.

Це слід враховувати при розробці структури бази даних, що моделює відповідну систему, тому що жодні дві системи не будуть точнісінько однаковими.

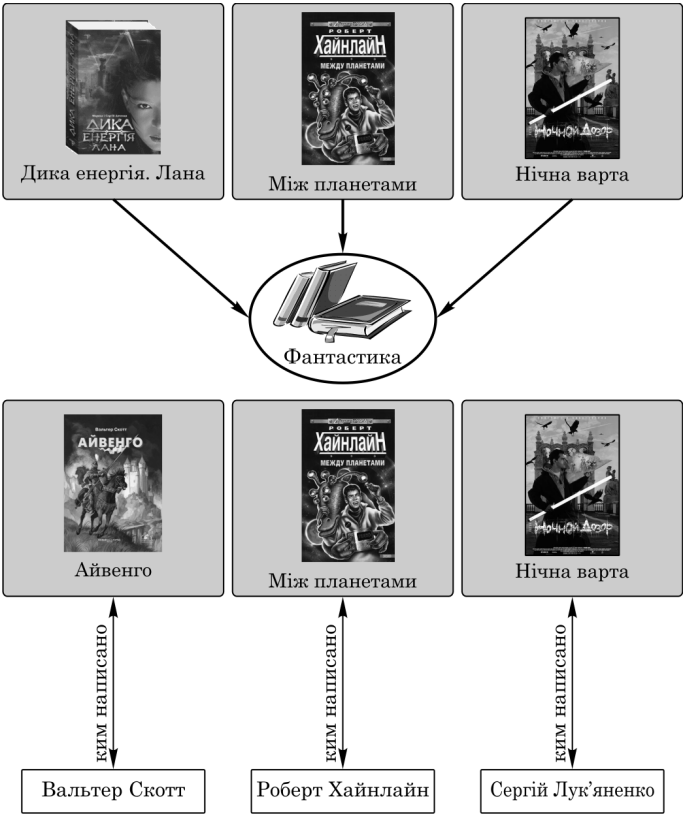

Рис. 1.1. Відношення «ким написано» є взаємно-однозначним, а відношення «належить до» визначає зв'язок типу «багато до одного»

Відношення, або таблиці

MySQL — це система управління реляційними базами даних. Термін «*реляційна*» (від латинського `relatio` — відношення) вказує, перш за все, на те, що така модель зберігання даних побудована на взаємовідношенні складових її частин. Реляційна база даних, як правило, складається з багатьох відношень. Відношення — це таблиця даних. Зверніть увагу, *таблиця* і *відношення* означають те саме. Приклад таблиці показаний на рис. 1.2.

`knygy`

knygaID	nazva	vydavnyctvo	grupID
7513	Дика енергія. Лана	Теза	128
9842	Вій	Навчальна книга – Богдан	42
6651	Між планетами	Наукова думка	128
9006	Айвенго	Глобус	129

Рис. 1.2. У таблиці `knygy` зберігаються кодові номери книг, їхні назви, а також дані про видавництво і групу кожної книги

Ця таблиця містить дані про книги в деякій бібліотеці. (Ми не відобразили тут дані про всі книги, а тільки про деякі з них, для прикладу).

Стовпці, або атрибути

У таблицях бази даних кожен стовпець, або атрибут, описує деяку частину даних, якими характеризується кожен запис у таблиці. Терміни «стовпець» і «атрибут» (властивість) у цьому контексті є взаємозамінними, але стовпець — це насправді частина таблиці, тоді як атрибут характеризує реальний об'єкт, для моделювання якого використовують таблицю. На рис. 1.2 кожна книга має атрибути `knygaID` (кодовий номер), `nazva` (назва), `vydavnyctvo` (видавництво) і `grupID` (номер групи). Ці атрибути формують стовпці таблиці `knygy`, які називають ще *полями*.

Рядки, записи, кортежі

Кожен рядок таблиці `knygy` — це набір даних про одну книгу. Такий набір у базах даних називають рядком, записом або кортежем. Кожен рядок в таблиці складається з відповідних значень стовпців таблиці.

Ключі

Суперключ — це стовпець (або набір стовпців), який можна використовувати для однозначної ідентифікації рядків у таблиці. *Ключ* — це мінімальний суперключ. Наприклад, в таблиці `knygy` для ідентифікації будь-якого рядка можна використовувати пару атрибутів `knygaID` і `nazva`. З цією ж метою можна використовувати і набір, що складається зі всіх стовпців (`knygaID, nazva, vydavnyctvo, grupID`). Обидва ці набори є суперключами.

Проте для ідентифікації рядків зовсім не обов'язково використовувати усі стовпці. Достатньо, наприклад, використовувати `knygaID`. Цей стовпець є мінімальним суперключем — мінімальним набором стовпців, за допомогою якого можна однозначно ідентифікувати будь-який рядок. Таким чином, `knygaID` є ключем.

Ідентифікувати книгу в таблиці `knygy` можна як за іменем (стовпець `nazva`), так і за його кодовим номером (стовпець `knygaID`). Обидва вказані стовпці є ключами і називаються потенційними ключами, оскільки вони є «кандидатами» на роль первинного ключа. Первинний ключ — це стовпець або набір стовпців, який використовують для ідентифікації рядків у таблиці. У нашому випадку як первинний ключ ми використовуватимемо стовпець `knygaID`. Цей ключ є набагато кращим, ніж назва, оскільки цілком можлива наявність двох книг з такою самою назвою.

Зовнішні ключі визначають зв'язки між таблицями. Наприклад, стовпець `grupID` (рис. 1.2) містить номер групи. Це зовнішній ключ: повна інформація про кожну групу зберігатиметься в іншій, окремій таблиці, в якій первинним ключем буде `grupID`.

Функціональні залежності

Термін *функціональна залежність* використовують менше, ніж терміни, що згадувались вище, але нам він знадобиться для того, щоб пояснити суть процесу нормалізації відношення.

Якщо у таблиці є функціональна залежність між стовпцями A і B, яку можна позначити A→B, це означає, що значення стовпця A визначає значення стовпця B. Наприклад, у таблиці `knygy` стовпець `knygaID` функціонально визначає стовпець `nazva` (так само, як і всі інші атрибути у цьому конкретному прикладі).

Схеми

Термін *схема* або *схема бази даних* позначає структуру бази даних, тобто форму бази даних без даних як таких. Іншими словами схема — це шаблон для даних у базі даних.

Схему вказаної вище конкретної таблиці можна подати у такому вигляді:

```
knygy (knygaID, nazva, vydavnyctvo, grupID).
```

Принципи проектування баз даних

При розробці бази даних необхідно враховувати таке:

- ✓ Які дані необхідно пам'ятати, або дані про які об'єкти або процеси потрібно зберігати?
- ✓ Які запитання задаватимуться базі даних? (Такі запитання називають *запитами*.)

Роздумуючи над цим, ми повинні постійно пам'ятати про «правила предметної галузі», які ми намагаємося моделювати, тобто про об'єкти, дані про які слід зберігати, і про те, які зв'язки є між ними.

Крім того, слід створити базу даних так, щоб не виникало структурних проблем типу надлишковості або аномалій даних.

Надлишковість проти втрати даних

При проектуванні схеми ми хотіли б мінімізувати надлишковість даних, але при цьому не допустити втрати даних. Під надлишковістю розуміють повторення даних у різних рядках однієї таблиці або в різних таблицях бази даних.

Уявіть собі, що замість таблиці з даними про книги (`knygy`) і таблиці з даними про групи (`grupy`) ми маємо єдину таблицю, що назива-

ється GrupyKnyg. Щоб зробити це, досить додати до таблиці knygy ще один стовпець nazvaGrupy (назва групи), щоб схема виглядала так:

knygaID	nazva	vydavnyctvo	grupID	nazvaGrupy
7513	Дика енергія. Лана	Теза	128	Фантастика
9842	Вій	Навчальна книга – Богдан	42	Українська класика
6651	Між планетами	Наукова думка	128	Фантастика
9006	Айвенго	Глобус	129	Зарубіжна класика

Рис. 1.3. Така схема породжує надлишковість: дані про назви груп зберігаються знову і знову

```
GrupyKnyg (knygaID, nazva, vydavnyctvo, grupID,
nazvaGrupy)
```

Для кожної книги групи 128 (Фантастика) ми повторюватимемо дані «128, Фантастика», як показано на рис. 1.3. Те саме буде зроблено і для інших груп, на які поділяють книги в цій бібліотеці.

Замість вказаної схеми можна запропонувати таку:

```
knygy (knygaID, nazva, vydavnyctvo, grupID)
grupy (grupID, nazva)
```

У цьому випадку назва кожної з груп зберігається у базі даних тільки в одному місці, а не багато разів, внаслідок чого мінімізується обсяг дискового простору і вдається уникнути деяких проблем.

Зверніть увагу, що ми повинні залишити стовпець grupID в таблиці knygy, інакше будуть втрачені деякі дані — зокрема дані про зв'язок між книгою і групою, до якої віднесли цю книгу. Таким чином, при вдосконаленні схеми завжди потрібно уникати повторення даних, але не допускати при цьому втрати даних.

Аномалії

Аномалії є складнішим поняттям. Аномалії — це проблеми, що виникають через дефекти проектування бази даних. Існують три види аномалій, і ми з'ясуємо, як вони з'являються, на прикладі схеми, показаної на рис. 1.3.

Аномалії вставки

Аномалії вставки виявляються тоді, коли ми намагаємося додати дані в «дефектну» таблицю. Припустимо, що в магазині з'явилася нова книга. При додаванні даних про неї в таблицю `GrupyKnyg` ми повинні додати туди як номер, так і назву відповідної групи. Але що відбудеться, якщо ми введемо невідповідні дані, наприклад, введемо 42 для номера групи, але вкажемо назву «Зарубіжна література»? У результаті буде не зрозумілим, який з рядків бази даних містить правильні дані. Це і є аномалія *вставки*.

Аномалії видалення

Аномалії видалення виникають тоді, коли ми видаляємо дані з дефектної схеми. Припустимо, що всі книги групи 42 продали того самого дня (можливо, в школах почали вивчати їх за програмою і діти розкупили усі). Після видалення записів цих книг у базі даних більше не буде жодного запису, що містить дані про назву групи 42. Це аномалія *видалення*.

Аномалії модифікації

Аномалії модифікації виникають тоді, коли ми змінюємо дані дефектної схеми. Припустимо, що назву групи 128 вирішили змінити і тепер вона називатиметься «Фантастика і фентезі». Необхідно буде змінити відповідні дані про кожну книгу з цієї групи. Ясно, що при цьому цілком можливо яку-небудь пропустити. І якщо ми справді пропустимо хоч би один із відповідних записів, виникне аномалія модифікації.

Порожні значення

Важливим правилом для розробки хорошої структури бази даних є необхідність уникати схем, в яких допускається наявність великого числа порожніх атрибутів. Наприклад, якщо ми хочемо вказати, що одна із приблизно тисячі книг має якусь особливу обкладинку, для зберігання цієї інформації не слід додавати в таблицю ще один стовпець, оскільки для інших 999 книг значенням такого стовпця буде `NULL`. Натомість слід додати нову таблицю, в якій зберігатимуться тільки кодові номери (`knygaID`) і дані про обкладинку тих книг, яких це стосується.