*Лекція № **9***
ТЕЗИ

Реляційні бази даних як основа сховищ даних. Поняття про роботу адміністратора СУБД. Основи адміністрування СУБД MySQL

Реляційні бази даних - концепція

Вище зазначалося, що як правило на сьогодні власне дані сховища зберігаються в реляційних базах даних.
Оскільки вони відіграють одну з ключових ролей в трирівневій архітектурі клієнт-сервер, то насправді роль реляційних баз даних на сьогодні взагалі важко переоцінити.

Треба однак розуміти, що на сьогодні також існують і використовуються у відповідних галузях і інші бази даних, наприклад, дуже перспективним видається напрям об'єктно-орієнтованих СУБД. Цілком певну нішу, наприклад, займають деревовидні (ієрархічні) БД.

Концептуально теорія систем управління реляційними базами даних була розроблена в 1970-тих роках ***Едгаром Коддом*** (Edgar Codd), британським вченим, який працював в той час в дослідницьких лабораторіях IBM.

Значення реляційної моделі є сьогодні величезним. Теорія реляційних баз даних Кодда відноситься до найважливіших іновацій останніх 85-ти років (за версією журналу Forbes від 2002 року). Відомими є “12 првил Кодда”, які характеризують реляційну систему. (Кодд сформулював їх, щоб протистояти нечистій рекламі виробників, які стверджували, що їх продукт є реляційною СУБД)

Основна ідея ***реляційної БД*** є дуже простою (все геніальне є простим, але не все просте є геніальним :)):
- дані зберігаються в ***таблицях*** (двомірних структурах), які складаються із ***стовбців*** (полів) та ***рядків*** (записів).
- Кожне поле може містити тільки один тип даних.
- Записи містять значення, що відповідають кожному із стовбців — це т.зв. кортеж, сукупність взаємозв'язаних даних.
- Один із стовбців (або комбінація з кількох стовбців) — ***первинний ключ*** (primary key) - однозначно ідентифікує кожен рядок з таблиці, забезпечує можливість відрізнити один рядок від іншого
- певним чином можна зв'язувати запис однієї таблиці із записом (записами) з інших — з допомогою ***зовнішніх ключів*** (foreign key).

Для взаємодії з реляційними БД було створено різні мовні засоби. В IBM було розроблено:
- ***SQL*** (Structured Query Language) — мова запитів
- ***DDL*** (Data Definition Language) — мова для створення об'єктів БД
- ***DML*** (Data Manipulation Language) — мова для маніпулювання даними

Всіх їх разом як правило називають SQL.

© Maryana Rozhankivska 2008. Ніяка частина даного документу не може бути відтворена чи розповсюджена без згоди автора.

SQL IBM запатентувала в 1985 році.
Стандарт ANSI — 1986
Стандарт ISO — 1987 — SQL-97.
з того часу було випущено декілька версій цього стандарту, зокрема навіть в цьому році SQL:2008.

Існує також багато розширень SQL (кожен виробник додає щось своє), найвідоміші є:
- PL/SQL (Procedural Language/SQL) — Oracle
- PL SQL (Procedural Language) - IBM SQL
- T-SQL (Transact-SQL) — Microsoft/Sybase
- SQL/PSM (SQL/Persistent Stored Module) — MySQL
- PSQL (Procedural SQL) - Interbase/Firebird

Отже дані зберігаються в таблицях.
Структура таблиць має бути дуже добре продумана, очевидно.

Первинний ключ однозначно ідентифікує запис (рядок, кортеж) в таблиці.
Часто він формується штучно — з допомогою генераторів, автоінкрементого поля і т.п.
ПОЯСНИТИ
(але не для типу зв'язку багато до багатьох)
Первинний ключ завжди має значення (заповнений — NOT NULL) і завжди унікальний.

Зовнішній ключ — це стовбець (чи комбінація стовбців) в таблиці, що служать для посилання на рядок іншої таблиці.
Часто зовнішні ключі в таблиці (яка тоді називається ***дочірньою***) — це значення первинного ключа з іншої таблиці (яка називається ***батьківською***).
Зовнішній ключ може також бути порожнім (NULL) і, очевидно, неунікальним.
ПОЯСНИТИ

Зовнішні ключі пов'язані з первинними правилом ***цілісності посилань*** (ссылочной целостности) (reference integrity).
Це означає, що при створенні зовнішнього ключа можна задати певні правила поведінки СУБД:
- ***обмеження на видалення*** (чи навіть зміну) — не можливо видалити запис з батьківської таблиці поки на нього є посилання
- ***каскадні зміна*** (або видалення) — дочірні записи змінюють значення при зміні запису батьківського
- ***обнулення при видаленні*** (або зміні) — дочірні записи приймають значення NULL при видаленні батьківського.

Кілька слів про те, як проектуються реляційні бази даних

Для пректування реляційних БД також протягом років вже розроблено кілька підходів і засобів.
Найважливішим з них є ***техніка концептуального моделювання даних згідно моделі "сутність-зв'язок"*** . ***ERM — entity-relationship model.***
Модель даних "сутність-зв'язок" має також важливий ефект: з її допомогою не тільки аналізується предметна область а й уточняється задача.

© Maryana Rozhankivska 2008. Ніяка частина даного документу не може бути відтворена чи розповсюджена без згоди автора.

Така модель пізніше цілком природньо трансформується в таблиці БД. Як правло така модель не має ***надлишковості*** (дублювання) даних.
ПОЯСНИТИ

Важливо однак розуміти, що не зважаючи на концептуальні поняття “правильності” проектування БД, найчастіше в гру це вступає проблема оптимізації роботи БД (по швидкодії як правило). Тому часто буває так, що таблиці проектуються також з огляду на цей фактор на шкоду академічному підходу.
ПОЯСНИТИ

Також для перевірки якості структури БД існують т.зв. ***правила нормування***.
Ці правила дають змогу оцінити якість моделі БД.

Перше правило нормалізації (перша нормальна форма - 1NF) визначає саме поняття таблиці:
перша номальна форма має місце тоді, коли відношення включають фіксоване число стовбців з <u>елементарними</u> значеннями.
Тобто значення мають бути елементарним — НЕ складеними. Наприклад (в залежності від предметної області і х-теру використання, правда) таблиця повинна містити 2 стовбці “ім'я” і “прізвище” замість одного “ім'яІпрізвище”.

Друга нормальна форма (2NF) визначає допустимі залежності між стовбцями і первинним ключем таблиці.
Друга номальна форма має місце тоді, коли таблиця має першу нормальну форму і всі записи (стрічки) таблиці <u>однозначно</u> визначаються первинним ключем таблиці.
Іншими словами таблиця повинна мати первинний ключ і не повинно бути таких стовбців, значення яких залежить тільки від частини первинного ключа.
Напр в таблиці з полями (orderNo, CustomerNo, CustomerName, OrderDate) з первинним ключем (orderNo, CustomerNo) CustomerName залежить тільки від частини ключа (тільки від CustomerNo) — очевидна надлишковість і таблицю треба розбити на дві.

Друга номальна форма регулює т.зв. ***функціональні залежності.***

Третя нормальна форма (3NF) виключає ***транзитивні залежності***
Третя нормальна форма має місце тоді, коли таблиця приведена до другої нормальної форми і <u>неключові стовбці таблиці не мають транзитивних</u> <u>залежностей від первинного ключа</u>.
Іншими словами, неключові стовбці Неповинні залежати від первинного ключа <u>через інші стовбці</u>. Залежність має бути прямою.
Наприклад в таблиці з полями (CustomerID, Customername, CustomerCategoryID, CustomerCategoryName) поле CustomerCategoryName залежить від CustomerCategoryID. Очевидно, що треба розбити цю таблицю на дві.

Нормальних форм є більше, ніж три (шість плюс нормальна форма Бойса-Кода), але ці три є обов'язковими для всіх таблиць грамотно спроектованої бази даних.

Важливо однак зауважити, що є випадки, коли, наприклад, з міркувань швидкодії проектант порушує правила третьої нормальної форми.

© Maryana Rozhankivska 2008. Ніяка частина даного документу не може бути відтворена чи розповсюджена без згоди автора.

Завдання адміністратора СУБД

Загально кажучи, завдання адміністратора БД полягає в установці, настройці та супроводі СУБД в процесі її функціонування.

Трохи конкретніше перелік задач виглядає так:
- установка програмного забезпечення СУБД
- запуск і зупинка сервера БД
- підтримка рахунків доступу користувачів до БД
- підтримка/перегляд лог-файлів чи іншої інформації про роботу сервера БД
- створення резервних копій БД
- налаштування сервера — швидкодія, продуктивність, локалізація і т.п.
- налаштування реплікації (при потребі)
- оновлення програмного забезпечення БД
- захист СУБД на рівні файлової системи і мережевих засобів
- захист сервера від внутрішнього несанкціонованого доступу
- відновлення БД після збоїв
- превентивна підтримка сервера БД

ПОЯСНИТИ кожен пункт.

Очевидно, що від сервера БД до сервера БД конкретний зміст цих задач дещо міняється, але всі вони залишаються актуальними.

Далі ми коротко розглянемо дві СУБД з точки зору роботи системного адміністратора, тобто в контексті посталених вище завдань.
MySQL — як один з найпростіших, дуже широко використовуваних сьогодні і достатньо функціональних серверів БД і
Firebird/Interbase як інший приклад повнофункціонального сервера БД промислового масштабу.
Пару слів також скажемо про ще дві СУБД: Oracle і PostgreSQL.

© Maryana Rozhankivska 2008. Ніяка частина даного документу не може бути відтворена чи розповсюджена без згоди автора.