Модуль 3.
Проектирование баз данных

Содержание модуля

- Задача проектирования БД
- Проектирование на базе семантических моделей: этапы
- Концептуальное проектирование
- Логическое и физическое проектирование
- Инструмент проектирования MySQL Workbench
- Другие подходы к проектированию БД
- SQL-DDL

Задачи проектирования

- Отображение объектов предметной области средствами модели данных
- Отображение связей объектов
- Достаточность представления предметной области для пользователей
- Формирование ограничений целостности
- Минимизация аномалий

Проектирование БД в реляционной модели

- Плохое отображение семантики
- Трудно моделировать сложные предметные области на основе отношений
- Отсутствие формальных правил для преобразования требований предметной области в схему БД
- Для сложных систем не всегда можно однозначно отделить сущности от связей

Вывод: необходимость использования моделей данных с лучшим отображением семантики данных.

Проектирование с использованием семантических моделей

- Формирование концептуальной схемы данных
- Преобразование к логической схеме данных на основе некоторой модели данных
- Преобразование к физической схеме данных

Модель Сущность-Связь (ER)

- Графическое представление - диаграмма сущность-связь (ER-диаграмма)
- Понятия ER-модели:
  - Сущность, тип сущности
  - Связь, тип связи
  - Атрибут, тип данных, домен
  - Ключи
  - Ограничения целостности
- Ограничения:
  - сущности должны быть отличимы
  - связи бинарные, типизированные

Концептуальное проектирование

- Этапы:
  - Выделение сущностей
  - Определение связей
  - Атрибуты, типы данных, домены
  - Ключи
  - Ограничения целостности
    - допустимые значения
    - разрешенные значения
    - существующие значения

Выделение типов сущностей

- Работа бизнес аналитика, проектировщика БД
- Выяснение потребностей и представления предметной области для каждой роли пользователя
- Выделяются независимые типы сущностей
- Отбрасываются “ненужные”
- Именование типов сущностей

Определение связей

- Наиболее сложный/творческий процесс
- Типы связей:
  - Один ко многим
  - Многие к одному
  - Многие ко многим
  - Один к одному
- Кратность связей
  - 1
  - 0..1
  - 0..n
  - 1..n
- Примеры в БД онлайн-магазина

Определение атрибутов

- Выделяются атрибуты типов сущностей
- Выделяются атрибуты связей
- Простые и составные атрибуты
- Определение типов данных, доменов
- Именование атрибутов

Выбор ключей

- Определяются ключевые атрибуты
- Выбирается первичный ключ
  - Простота
  - Минимальность
  - Редкие изменения
- Для слабых сущностей вводятся синтетические ключевые атрибуты

Проверка

- Наличие связей один к одному
- Наличие избыточных связей
- Проверка выполнимости пользовательских операций – мысленный эксперимент
- Обсуждение с пользователями

Особые случаи. Супертипы и подтипы

- Близко к понятиям ООП
- Типы сущностей содержат общие атрибуты и/или связи
- Ограничения
  - Сущность любого подтипа является сущностью супертипа
  - Сущность супертипа обязательно является сущностью некоторого подтипа
  - Не должно быть сущностей одновременно нескольких подтипов

Пример:

- Пользователь Stepic (имя, логин, пароль, ...)
  - Слушатель курса (выполнение заданий)
  - Преподаватель (авторство курсов)

Особые случаи. Взаимоисключающие связи

- Сущности некоторого типа могут быть альтернативно связаны с сущностями других типов
- Разделение типов сущностей

Логическое проектирование БД

- Реализация схемы данных на основе выбранной модели
- Для РМД может быть в значительной степени автоматизирован переход от концептуальной модели к логической
- РМД. Логическая модель:
  - набор схем отношений
  - выделенные первичные ключи
  - выделенные внешние ключи

Логическое проектирование БД

- Правила перехода:
  - Каждый простой тип сущности отображается в отношение
  - Атрибуты ER-модели отображаются в атрибуты отношений
  - Компоненты уникального идентификатора отображаются в первичный ключ
  - Связи один к одному и один ко многим отображаются в виде внешних ключей
  - Связи многие ко многим отображаются с созданием дополнительного отношения
  - Сложные связи (более, чем между двумя типами сущностей) декомпозируются с выделением нового типа сущности
  - Многозначные атрибуты декомпозируются с выделением нового типа сущности
- Нормализация

Логическое проектирование БД

- Особые случаи:
  - Супертипы и подтипы
    - Одна таблица
    - Несколько таблиц
  - Взаимоисключающие связи
    - Общее хранение внешних ключей
    - Раздельное хранение внешних ключей

Физическое проектирование БД

- Перенос логической модели в конкретную СУБД
  - Определение типов данных для атрибутов
  - Определение допустимости неопределенных значений
  - Выбор стратегии обработки исключительных ситуаций при попытках нарушения ссылочной целостности
  - Уточнение названий таблиц, атрибутов
  - Хранение и/или вычисление производных атрибутов
  - Реализация ограничений целостности, процедурная обработка
  - Формирование описания модели данных в рамках выбранной СУБД
  - Формирование индексов
  - Секционирование и партицирование

Физическое проектирование БД

- Критерии
  - Пропускная способность
  - Время ответа
  - Утилизируемые ресурсы

Пример проектирования БД

- Предметная область: Stepic

SQL/DDL

- Создание схемы
- Создание доменов
- Создание таблиц
- Изменение таблиц

SQL/DDL. Создание схемы

- CREATE SCHEMA

```
<schema definition> ::=
                CREATE SCHEMA <schema name clause>
                  [ <schema character set specification> ]
                  [ <schema element>... ]
```

- Пример

```
CREATE SCHEMA IF NOT EXISTS `store` DEFAULT CHARACTER SET utf8;
```

SQL/DDL. Создание домена

- CREATE DOMAIN

```
domain_definition ::= CREATE DOMAIN domain_name [AS] data_type
    [ default_definition ]
    [ domain_constraint_definition_list ]

default_definition ::= DEFAULT { literal | niladic_function | NULL }

domain_constraint_definition_list ::= [CONSTRAINT constraint_name]
     CHECK (conditional_expression)
```

- Пример (PostgreSQL)

```
CREATE DOMAIN SALE_STATUS AS VARCHAR(45)
    DEFAULT 'new'
    CHECK (VALUE IN ('new', 'process', 'assembly', 'ready', 'delivering',
'issued', 'rejected'))
    CONSTRAINT SALE_STATUS_NOT_NULL CHECK (VALUE IS NOT NULL);
```

SQL/DDL. Создание таблицы

- CREATE TABLE

```
base_table_definition ::= CREATE TABLE base_table_name
     (base_table_element_commalist)

base_table_element ::= column_definition |
     base_table_constraint_definition


column_definition ::= column_name
    { data_type | domain_name }
    [ default_definition ]
    [ column_constraint_definition_list ]
```

SQL/DDL. Создание таблицы

- CREATE TABLE

```
column_constraint_definition ::=
    [ CONSTRAINT constraint_name ]
    NOT NULL
    | { PRIMARY KEY | UNIQUE }
    | references_definition
    | CHECK ( conditional_expression )

references_definition ::=
    REFERENCES base_table_name [ (column_commalist) ]
        [ MATCH { SIMPLE | FULL | PARTIAL } ]
        [ ON DELETE referential_action ]
        [ ON UPDATE referential_action ]
```

SQL/DDL. Создание таблицы

- CREATE TABLE

```
base_table_constraint_definition ::=
    [ CONSTRAINT constraint_name ]
    { PRIMARY KEY | UNIQUE } ( column_commalist )
    | FOREIGN KEY ( column_commalist )
       references_definition
    | CHECK ( conditional_expression )

referential_action ::=
   { NO ACTION | RESTRICT | CASCADE
    | SET DEFAULT | SET NULL }
```

SQL/DDL. Создание таблицы

- Примеры

```
CREATE TABLE IF NOT EXISTS `store`.`category` (
  `id` INT NOT NULL,
  `name` VARCHAR(255) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```

SQL/DDL. Создание таблицы

```
CREATE TABLE IF NOT EXISTS `store`.`category_has_good` (
  `category_id` INT NOT NULL, `good_id` INT NOT NULL,
  PRIMARY KEY (`category_id`, `good_id`),
  INDEX `fk_category_has_good_good1_idx` (`good_id` ASC),
  INDEX `fk_category_has_good_category_idx` (`category_id` ASC),
  CONSTRAINT `fk_category_has_good_category`
    FOREIGN KEY (`category_id`) REFERENCES `store`.`category` (`id`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_category_has_good_good1`
    FOREIGN KEY (`good_id`) REFERENCES `store`.`good` (`id`)
    ON DELETE NO ACTION ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```

SQL/DDL. Изменение таблицы

- ALTER TABLE

```
base_table_alteration ::= ALTER TABLE base_table_name
    column_alteration_action
    |  base_table_constraint_alternation_action

column_alteration_action ::=
        ADD [ COLUMN ] column_definition
    | ALTER [ COLUMN ] column_name
        { SET default_definition | DROP DEFAULT }
    | DROP [ COLUMN ] column_name
            { RESTRICT | CASCADE }
```

SQL/DDL. Изменение таблицы

- Пример

```
ALTER TABLE `store`.`sale`
     ADD COLUMN
          `is_exclusive_case` BOOLEAN NOT NULL DEFAULT 0;


ALTER TABLE `store`.`sale`
     DROP COLUMN dt_created,
     DROP COLUMN dt_modified,
     DROP FOREIGN KEY fk_order_status1,
     DROP COLUMN status_id,
     ADD COLUMN ts_modified TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
     ADD COLUMN sale_status VARCHAR(45) NOT NULL DEFAULT 'new'
      CHECK (VALUE IN ('new', 'process', 'assembly', 'ready',
'delivering', 'issued', 'rejected'));
```

SQL/DDL. Изменение таблицы

- ALTER TABLE

```
base_table_constraint_alternation_action ::=
        ADD [ CONSTRAINT ] base_table_constraint_definition
    | DROP CONSTRAINT constraint_name
        { RESTRICT | CASCADE }
```

- Пример

```
ALTER TABLE `store`.`sale`
     DROP FOREIGN KEY fk_order_client1;
ALTER TABLE `store`.`sale`
     DROP INDEX fk_order_client1_idx;
ALTER TABLE `store`.`sale`
     DROP COLUMN client_id;
```

Заключение

Спасибо за внимание!