Белорусский государственный университет информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики и технологий программирования

Проволоцкий В.Е., Бугара Д.А.

Методические рекомендации к лабораторному практикуму по курсу МДиСУБД

7 семестр, 2014\2015 уч.год

Количество лабораторных работ - 9

15-09-2014

Минск, 2014

Перечень лабораторных работ по курсу МДиСУБД (7 семестр, 2014\2015 уч.год)

Лабораторная работа №1. Моделирование базы данных

Лабораторная работа №2. Нормализация базы данных

Лабораторная работа №3. SQL: Data Definition Language (DDL)

Лабораторная работа №4. SQL: Data Manipulation Language (DML)

Лабораторная работа №5. SQL: Функции

Лабораторная работа №6. SQL: Группировки

Лабораторная работа №7. SQL: Подзапросы

Лабораторная работа №8. SQL: Составные запросы

Лабораторная работа №9. Основы миграции и синхронизации баз данных

Лабораторная работа №1. Моделирование базы данных

Основные понятия

Всякая профессиональная деятельность связана с информацией, с организацией ее сбора, хранения, выборки. Можно сказать, что неотъемлемой частью повседневной жизни стали базы данных, для поддержки которых требуется некоторый организационный метод, или механизм. Такой механизм называется системой управления базами данных (СУБД).

База данных (БД) – совместно используемый набор логически связанных данных (и их описание), предназначенный для удовлетворения информационных потребностей.

СУБД (система управления базами данных) – программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также получать к ней контролируемый доступ.

Этапы проектирования базы данных

Проектирование базы данных – одна из наиболее сложных и ответственных задач. В результате её решения должны быть определены содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными.

Функциональный подход к проектированию БД

Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Предметный подход к проектированию БД

Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чёткое представление о самой предметной области и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию предметной области и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

Проектирование с использованием метода "сущность-связь"

Метод "сущность–связь" (entity–relation, ER–method) является комбинацией двух предыдущих и обладает достоинствами обоих.

Этап инфологического проектирования начинается с моделирования предметной области. Проектировщик разбивает её на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи).

Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов предметной области. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 5-7 сущностей.

Реляционные базы данных

Управление основными потоками информации осуществляется с помощью, так называемых систем управления реляционными базами данных, которые в ряде случаев поддерживают и клиент-серверные технологии, позволяющие успешно управлять данными.

В реляционной модели объекты реального мира и взаимосвязи между ними представляются с помощью совокупности связанных между собой таблиц (отношений).

Даже в том случае, когда функции СУБД используются для выбора информации из одной или нескольких таблиц (т.е. выполняется запрос), результат также представляется в табличном виде. Более того, можно выполнить запрос с применением результатов другого запроса.

Каждая таблица БД представляется как совокупность строк и столбцов, где строки (записи) соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы (поля) – атрибутам (признакам, характеристикам, параметрам) объекта.

В каждой таблице БД необходимо наличие **первичного ключа** – так именуют поле или набор полей, однозначно идентифицирующий каждый экземпляр объекта или запись. Значение первичного ключа в таблице БД должно быть уникальным, т.е. в таблице не допускается наличие двух и более записей с одинаковыми значениями первичного ключа. Он должен быть минимально достаточным и не зависеть от полей, удаление которых может отразиться на его уникальности.

Реляционные связи между таблицами баз данных

Связи между объектами реального мира могут находить свое отражение в структуре данных, а могут и подразумеваться, т.е. присутствовать на неформальном уровне.

Между двумя или более таблицами базы данных могут существовать отношения подчиненности, которые определяют, что для каждой записи главной таблицы (называемой еще **родительской**) возможно наличие одной или нескольких записей в подчиненной таблице (называемой еще **дочерней**).

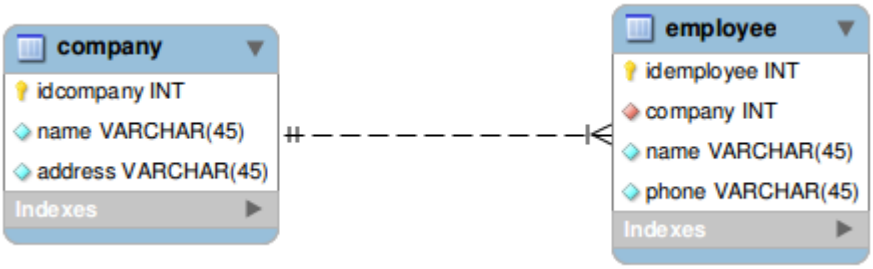
Выделяют три разновидности связи между таблицами базы данных:

- "один–ко–многим";
- "один–к–одному";
- "многие–ко–многим".

Отношение "один–ко–многим"

Отношение "один–ко–многим" имеет место, когда одной записи родительской таблицы может соответствовать несколько записей дочерней. Связь "один–ко–многим" иногда называют связью "многие–к–одному". И в том, и в другом случае сущность связи между таблицами остается неизменной.

В качества примера такого отношения можно рассмотреть случай, когда у нескольких компаний есть свои сотрудники, которые относятся только к одной компании. В тоже время, каждая из компаний имеет в своём штате нескольких сотрудников.



Пример 1.1 Сотрудники и компании

Между таблицами существуют отношения подчинённости. Так таблица сотрудников является дочерней по отношению к родительской таблице компаний.

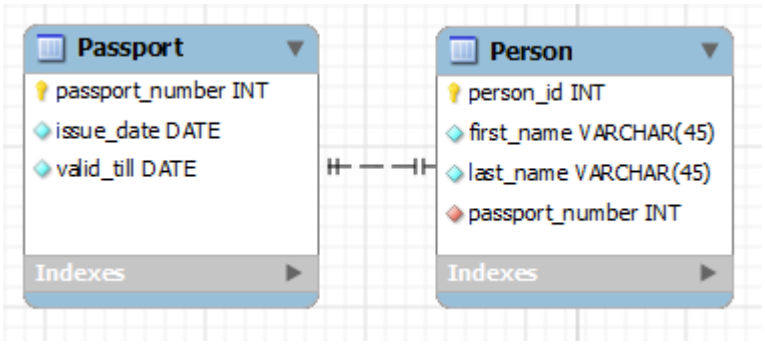
Отношение "один–к–одному"

Отношение "один–к–одному" имеет место, когда одной записи в родительской таблице соответствует одна запись в дочерней. Это отношение встречается намного реже, чем отношение "один–ко–многим".

Связь один-к-одному может быть реализована в одной таблице, тогда записи таблицы содержат данные, которые находятся в связи один-к-одному с первичным ключом или записью.

В редких случаях связь один-к-одному моделируется с использованием двух таблиц. Такой вариант иногда необходим, чтобы преодолеть ограничения СУБД или с целью увеличения производительности, или порой вы можете решить, что вы хотите разделить две сущности в разные таблицы, в то время как они все еще имеют связь один-к-одному. Но обычно наличие двух таблиц в связи один-к-одному считается дурной практикой.

В качестве примера рассмотрим, как может быть смоделирована связь между людьми и их паспортами.



Пример 1.2 Люди и их паспорта

Каждый человек в стране имеет только один действующий паспорт, а каждый паспорт принадлежит только одному человеку.

Отношение "многие–ко–многим"

Отношение "многие–ко–многим" применяется в следующих случаях:

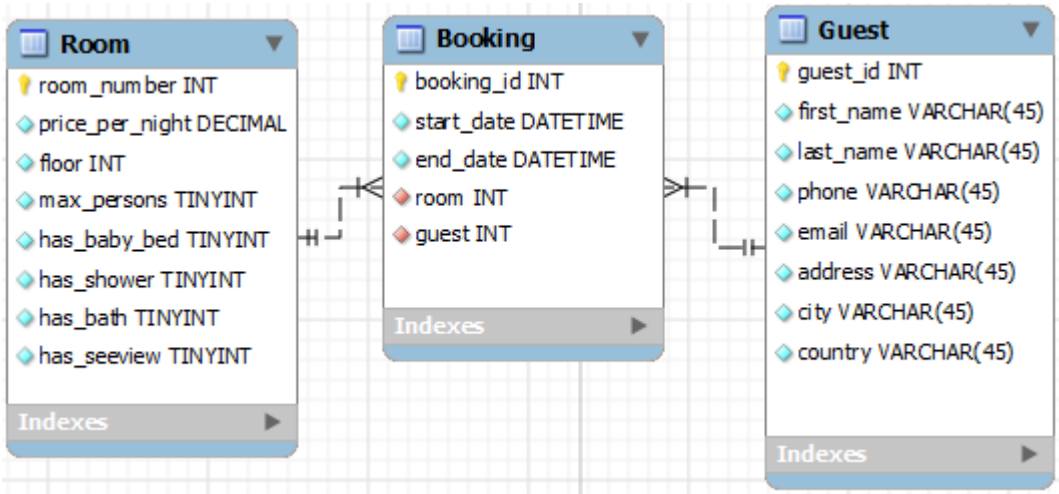
- одной записи в родительской таблице соответствует более одной записи в дочерней;

- одной записи в дочерней таблице соответствует более одной записи в родительской.

Всякую связь "многие–ко–многим" в реляционной базе данных необходимо заменить на связь "один–ко–многим" (одну или более) с помощью введения дополнительных таблиц.

В качестве примера рассмотрим, как может быть смоделирована таблица заказов номеров гостиницы посетителями.

Так как реляционная модель данных требует заменить отношение "многие–ко-многим" на несколько отношений "один–ко-многим", поэтому соединительная таблица связи многие-ко-многим имеет дополнительные поля.



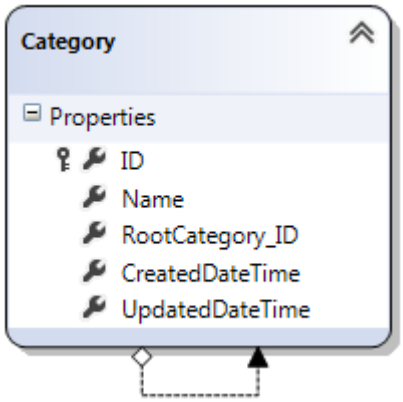
Пример 1.3 Заказ номеров гостиницы посетителями

В этом примере вы видите, что между таблицами гостей и комнат существует связь многие-ко-многим. Одна комната может быть заказана многими гостями с течением времени и с течением времени гость может заказывать многие комнаты в отеле. Соединительная таблица в данном случае является не классической соединительной таблицей, которая состоит только из двух внешних ключей. Она является отдельной сущностью, которая имеет связи с двумя другими сущностями.

Вы часто будете сталкиваться с такими ситуациями, когда совокупность двух сущностей будет являться новой сущностью.

Возвратные отношения

Большинство отношений в базе данных устанавливают между двумя различными таблицами. Однако бывает, когда таблица связывается сама с собой либо через отношение "один-к-одному", либо через отношение "один-ко-многим". Подобные отношения называются **возвратными отношениями.**



Пример 1.4 Категории товаров в интернет-магазине

Возвратные отношения чаще всего применяются для реализации иерархий. Типичный пример – иерархия категорий товаров в интернет-магазине. Категория может быть подчинена другой категории, а та, в свою очередь, может иметь родительскую или дочернюю. Это отношение оформляется путем включения первичного ключа категории в качестве столбца таблицы и установки возвратного отношения "один-ко-многим".

Такие отношения очень часто называются рекурсивными, унарными или собственными.

Лабораторная работа №2. Нормализация базы данных

Лабораторная работа №3. SQL: Data Definition Language (DDL)

Лабораторная работа №4. SQL: Data Manipulation Language (DML)

Лабораторная работа №5. SQL: Функции

Лабораторная работа №6. SQL: Группировки

Лабораторная работа №7. SQL: Подзапросы

Лабораторная работа №8. SQL: Составные запросы

Лабораторная работа №9. Основы миграции и синхронизации баз данных