МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г. Локтионова
«28» 02 2023 г.

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Методические указания по проведению
лабораторных занятий по дисциплине Базы данных
для студентов направления подготовки 09.03.01
Информатика и вычислительная техника

Курск 2023 г.

УДК 004.652
Составители Е.Ю. Емельянова, И.Е. Чернецкая, Е.Н. Иванова

Рецензент
доцент кафедры программной инженерии,
кандидат технических наук
Т.Н. Конаныхина

Проектирование баз данных: методические указания по проведению лабораторных занятия по дисциплине Базы данных / Юго-Зап. гос. ун-т; сост.: Е.Ю. Емельянова, И.Е. Чернецкая, Е.Н. Иванова. – Курск, 2023. – 61 с.

Руководство к проектированию реляционной базы данных, содержит этапы проектирования: получение модели сущность-связь, переход к реляционной модели данных, проведение нормализации, реализации базы данных.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по направлению Информатика и вычислительная техника.

Предназначены для студентов направления 09.03.01 Информатика и вычислительная техника очной и заочной формы обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16.
Усл.печ.л. Уч.-изд.л. . Тираж 20 экз. Заказ 40 . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

1 Цель работы

Научиться проектировать реляционные базы данных; изучить операторы языка SQL для создания баз данных и таблиц; создать таблицы учебной базы данных под управлением одной из стандартных реляционных СУБД,

Теоретическая часть

2 Общие положения

2.1 Понятия и определения

База данных (Database) – электронное хранилище информации. Самые распространенные сейчас – реляционные базы данных – хранят данные в таблицах. Физически база данных представляет собой один или несколько файлов специального формата.

СУБД, Система Управления Базами Данных (DBMS – Database managment system) – комплекс программ, предназначенный для создания и сопровождения баз данных. СУБД способна одновременно управлять несколькими базами данных, с которыми в одно и то же время, параллельно, работают многие пользователи, Обычно СУБД включает в себя следующие программы:

– ядро СУБД – программа или служба, которая постоянно «висит» в памяти и занимается обслуживанием поступивших от пользователей запросов на обработку данных (найти данные по критерию поиска, вставить, удалить или изменить строки таблицы и т.д.

– программы для создания и сопровождения баз данных – интерактивные программы, где можно просмотреть содержимое баз данных, находящихся под управлением данной СУБД, отредактировать его, зарегистрировать новых пользователей базы данных и разрешить им работать с определенными таблицами, создавать и удалять базы данных и др.

– мониторы производительности – программы, с помощью

которых можно просмотреть в реальном времени загрузку СУБД, статистику наиболее часто выполняющихся запросов, выявить запросы, требующие большого времени на выполнение и их оптимизировать.

Приложение базы данных (Database Application) – это программа, с помощью которой пользователи работают с базой данных. Приложение обычно пишется на языках высокого уровня (Java, Delphi, C++) или с применением Web-технологий (HTML+язык web-сценариев: PHP, Python, Java Script, Perl), Приложение направляет SQL-запросы к СУБД, в ответ получает данные и отображает их в удобном для пользователя виде.

2.2 Этапы проектирования баз данных

Методика нисходящего проектирования баз данных предполагает создание базы данных за пять этапов:

1. Словесное описание предметной области – описывается область реального мира, данные из которой будут храниться в БД, назначение и функции базы данных, алгоритмы обработки данных, особенности пользовательского интерфейса.
2. Объектное моделирование (инфологическое моделирование) – частичная формализация предметной области. Из словесного описания выделяются объекты, информация о которых будет храниться в базе данных, и логические связи между ними в терминах «реального мира».
3. Выбор СУБД – каждая СУБД хранит данные в форме одной из даталогических моделей. Самой распространенной моделью сейчас является реляционная модель данных. Кроме того, каждая СУБД имеет специфические особенности реализации модели данных, которые необходимо учитывать на последующих этапах проектирования.
4. Даталогическое проектирование. Если речь идет о реляционной базе данных, на этом этапе определяется структура таблиц, ограничений целостности и способы их поддержания средствами выбранной СУБД.
5. Реализация. Создание базы данных на ЭВМ.

3 Модель «сущность-связь»

3.1 Основные понятия модели «сущность-связь»

Для второго этапа проектирования баз данных – объектного моделирования – наибольшее распространение получила модель «сущность-связь» (Entity-Relationship), или ER-модель.

Создание ER-модели очень похоже на создание системы классов в объектно-ориентированном программировании (ООП). Базовыми понятиями ER-модели являются сущность и связь, которые по аналогии с ООП можно представить как особые типы классов.

Сущность – объект, информация о котором сохраняется в базе данных. Этот объект имеет свойства – атрибуты сущности. Сущностью может быть сотрудник, заказ, товар и т.п. По аналогии с ООП, следует различать класс сущности и экземпляр сущности. Класс сущности моделирует набор однотипных объектов. Экземпляр сущности представляет конкретный физический объект. Каждому классу сущности дается уникальное в пределах ER-модели имя. Имя класса принято записывать заглавными буквами. Например, в базе данных библиотеки класс сущности ЧИТАТЕЛЬ может быть представлен множеством экземпляров, где каждый экземпляр – это пользователь библиотеки (Иванов, Петров, Сидоров и пр.).

Связи моделируют смысловые ассоциации между сущностями Различают класс связи и экземпляр связи. Каждому классу связи дается уникальное имя. Например, связь АБОНЕМЕНТ между сущностями ЧИТАТЕЛЬ и КНИГА (см. рис. 1) означает, что книга в настоящее время находится на руках у читателя, Сущности, включенные в данную связь, называются участниками связи, а количество участников связи называется её степенью. В данном примере связь. АБОНЕМЕНТ имеет степень «2».

Сущности обозначаются прямоугольниками, связи – линиями. Внутри ромба пишется максимальная кардинальность связи.

Рисунок 1 – Сущности и связи изображаются в виде диаграммы «сущность-связь» (ER-диаграммы)

Связи второй степени (их еще называют бинарными связями) являются самыми распространенными. Известно три типа бинарных связей: 1:1 («один-к-одному»); 1:M («один-ко-многим») или обратная ей M:1 («многие-к-одному»); и M:N («многие-ко-многим»). Тип связи устанавливается по количеству связанных между собой экземпляров сущностей.

На рисунке 2 приведен пример детализации связи АБОНЕМЕНТ. Эта связь имеет тип 1:M, поскольку каждый читатель может взять несколько книг, но экземпляр книги может быть записан только за одним человеком.

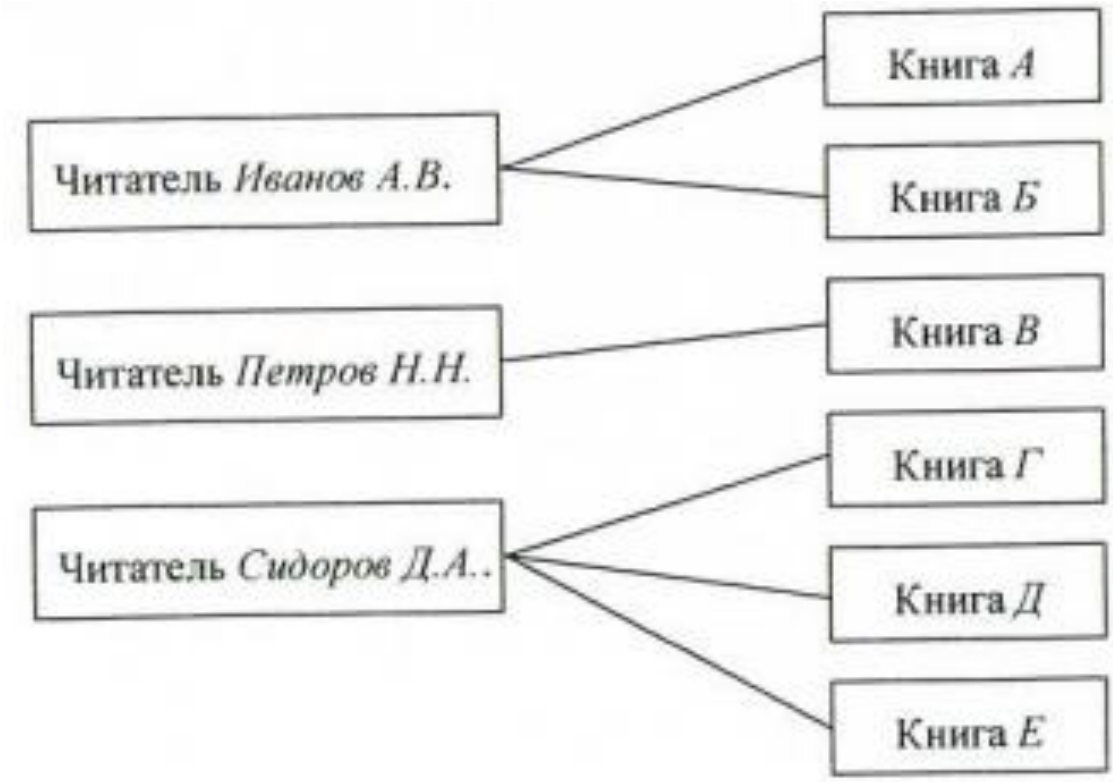


Рисунок 2 – Детализация ЕВ-диаграммы

Максимальное количество экземпляров сущностей, которые могут участвовать в связи, называется максимальным кардинальным числом связи. Со стороны сущности ЧИТАТЕЛЬ максимальное кардинальное число равно 1, а со стороны сущности КНИГА определяется максимальным количеством книг, которые читатель может одновременно держать на руках. Так, в Курской областной библиотеке имени Асеева – это 5 книг, в библиотеке ЮЗГУ – М (не ограничено).

Аналогично определяется минимальное кардинальное число связи – это минимальное количество экземпляров сущностей, объединенных одним экземпляром связи. Для сущностей ЧИТАТЕЛЬ и КНИГА минимальные кардинальные числа равны 0, потому что читатель может не брать ни одной книги, а книга может находиться в библиотеке, не будучи взятой ни одним читателем. Связь с минимальным кардинальным числом 0 называется необязательной (по отношению к данной сущности), связь с кардинальным числом ≥1 является обязательной (тоже по отношению к данной сущности). На ER-диаграмме минимальные кардинальные числа пишутся над линиями связи (см. рис.3,а) или обозначаются: вертикальной чертой – обязательная связь, овалом – необязательная (см. рис.3,6).

а)

б)

Рисунок 3 – Различные способы изображения кардинальных чисел

Связь БИБЛИОТЕЧНАЯ КАРТОЧКА показывает, в каких разделах систематического каталога есть ссылка на книгу. Связь имеет тип M:N, поскольку одну книгу может быть несколько ссылок из разных разделов каталога. В то же время один раздел каталога содержит ссылки на несколько разных книг. Раздел каталога может быть пустым (с его стороны связь необязательная), а книга должна где-то упоминаться (со стороны книги связь обязательная).

В результате построения ER-диаграммы получается связанный граф, вершинами которого являются сущности, а дугами – связи. В полученном графе необходимо избегать циклических связей – это признак того, что модель может быть составлена некорректно.

Объекты реального мира, описываемые сущностями и связями, могут иметь некоторые свойства, или характеристики, которые требуется хранить в БД. В ER-модели подобные свойства моделируются атрибутами. Атрибуты могут быть как у сущностей, так и у связей. Например, класс сущности ЧИТАТЕЛЬ может иметь атрибуты НомерЧитательскогоБилета, ФИО, ДатаРождения, ДомашнийАдрес, Телефон; а класс связи. АБОНЕМЕНТ (см. рис. 1) – атрибут СрокВозврата.

Различают следующие виды атрибутов:

Простой или составной (композитный) атрибут. Составной атрибут состоит из набора простых атрибутов. Например, НомерЧитательскогоБилета – это простой атрибут; ДомашнийАдрес – составной, он включает простые атрибуты Город, Улица, Дом, Квартира.

Однозначный или многозначный атрибут. Многозначный атрибут – это список (массив) однотипных значений, Многозначным атрибутом может быть атрибут СписокКлючевыхСлов сущности КНИГА.

Базовый или производный (вычислимый) атрибут. Базовый атрибут хранит (в памяти) присвоенное ему значение, производный – вычисляется по формуле из других атрибутов (в памяти не хранится). Например, ДатаРождения читателя – это базовый атрибут, а ВозрастЧитателя – вычислимый, он может изменяться со временем, а базовый остается неизменным.

Обязательный или необязательный (отсутствующий) атрибут. Обязательный атрибут должен быть заполнен у каждого экземпляра сущности. Необязательный атрибут может иметь "пустое" значение (МИГ), например, Телефон читателя.

Ключевой атрибут (идентификатор) — атрибут, который может использоваться для различения (идентификации) экземпляров сущностей. У каждого экземпляра сущности своё неповторимое значение идентификатора. У сущности ЧИТАТЕЛЬ идентификатором является НомерЧитательскогоБилета. Идентификатор может быть составным (например, {СерияПаспорта, НомерПаспорта}). Ключевые атрибуты – всегда обязательные (NOT NULL).

Изображение атрибутов на ER-диаграмме показано на рисунке 4.

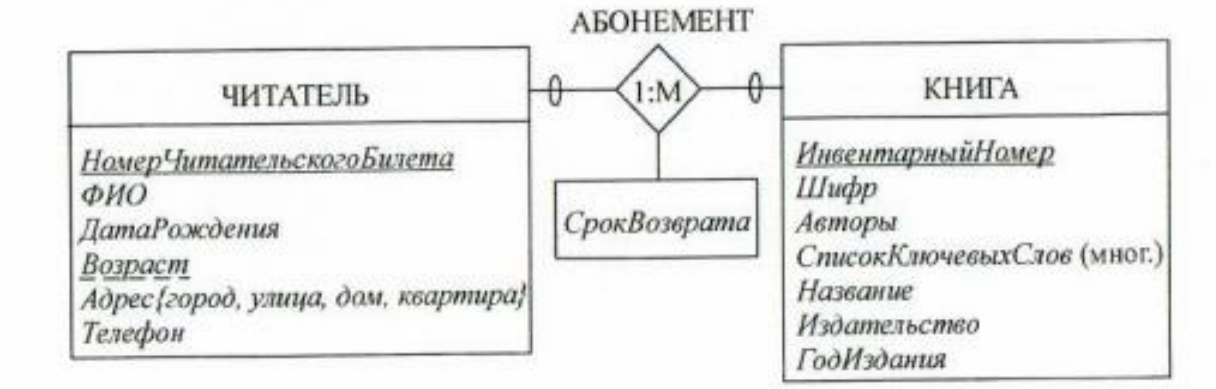


Рисунок 4 - Атрибуты сущности перечисляются под заголовком, отделённые чертой. Ключевые атрибуты подчеркиваются сплошной линией, вычислимые – пунктиром. Атрибуты связи изображаются в прямоугольнике под связью

В модели «сущность-связь» существует понятие сильных и слабых сущностей, Сильная (независимая) сущность – такая, которая может существовать и идентифицироваться независимо от того, связана она с другими сущностями или нет. Слабая (зависимая) сущность не может существовать, не будучи связанной с другой сущностью (эта сущность является сильной по отношению к этой, слабой, сущности). Примером слабой сущности может быть ТРЕБОВАНИЕ_НА_КНИГУ (см. рис. 5).

Сущность ТРЕБОВАНИЕ_НА_КНИГУ становится

бессмысленной, если из модели исключить сущность КНИГА. Обратное неверно. Сущность КНИГА продолжает существовать и не теряет своей полезности, если исчезнет сущность ТРЕБОВАНИЕ НА_КНИГУ. Таким образом, сущность КНИГА является сильной по отношению к слабой сущности ТРЕБОВАНИЕ _НА_КНИГУ.

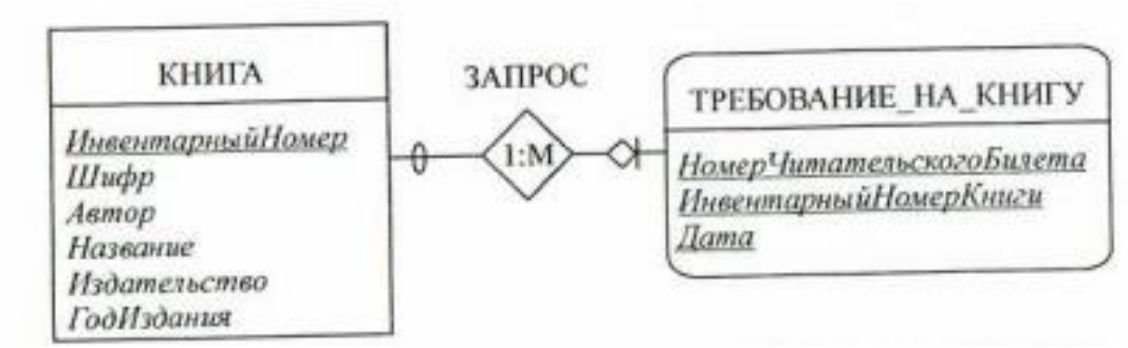


Рисунок 5 – Слабая сущность ТРЕБОВАНИЕ_НА_КНИГУ обозначается прямоугольником со скругленными углами. На линии связи между сильной и слабой сущностью, рядом со слабой сущностью ставится ромб

Ключевой атрибут сущности ТРЕБОВАНИЕ НА_КНИГУ составной, он включает идентификатор сущности КНИГА – поле ИивентарныйНомерКниги. Слабая сущность, зависящая от сильной по ключевому полю, как в данном примере, – называется идентификационно-зависимой, а связь – идентификационной (identifying relationship).

На вопрос о том, являются ли две сущности одинаково «сильными» или образуют пару «сильная-слабая», может быть ответ, теряет ли первая сущность свой логический смысл, если из модели исключить сущность другого класса. Если да, то сущности зависимы

По аналогии с наследованием в объектно-ориентированном программировании, в ER-модели допускается иерархия классов сущностей. Базовый класс сущностей называется супертипом (supertype) или подтипом, У супертипа может быть любое количество дочерних классов – подтипов (subtype). Подтип наследует все атрибуты и связи базового класса и при необходимости добавляет к ним свои атрибуты и связи. Например,

от класса сущности КНИГА можно создать подтип МНОГОТОМНОЕ_ ИЗДАНИЕ (рис.6). Многотомные издания наследуют все атрибуты базового класса КНИГА, но в тоже время появляются новые свойства, не применимые к обычным, однотомным, книгам: количество томов, номер тома, название тома, количество страниц в томе и др. Аналогично можно создать еще один подтип сущности КНИГА – класс УЧЕБНИК с новыми атрибутами специальность и название предмета (см. рис.6).

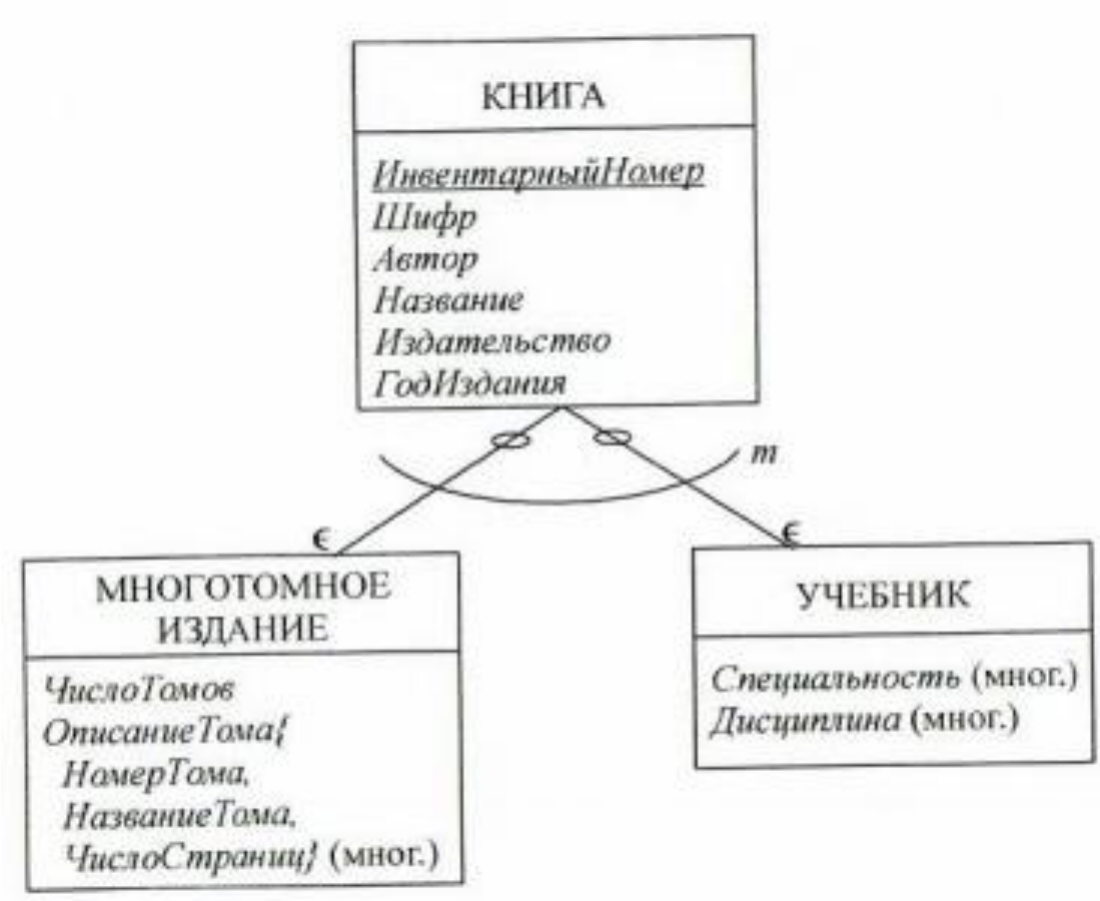


Рисунок 6 – Иерархия типов сущностей. Значок ∈ показывает, что подтипы наследуют все атрибуты и связи супертипа. Пустые овалы на связях со стороны класса КНИГА означают, что подтипы не являются обязательными: то есть существуют книги, которые нельзя отнести ни к многотомным изданиям, ни к учебникам, а только к надтипу КНИГА

Подтипы МНОГОТОМНОЕ ИЗДАНИЕ и УЧЕБНИК являются не взаимоисключающими (not exclusive). То есть могут

существовать такие книги, которые одновременно являются и учебниками, и многотомными изданиями. В отличие от принципов ООП, где экземпляр объекта принадлежит ровно одному классу, в модели «сущность-связь» экземпляр сущности может одновременно принадлежать нескольким подтипам одного супертипа. Максимальное число этих подтипов записывается рядом с дугой (m на рис.6). Для нашего примера это число ограничено только числом подтипов.

Существуют также взаимоисключающие (exclusive) подтипы, когда экземпляр сущности может принадлежать только одному из подтипов. Поясним на примере. Читателями вузовской библиотеки могут быть студенты, аспиранты и преподаватели. Каждой категории читателей соответствует подтип, различающийся набором атрибутов (см. рис.7) Подтипы СТУДЕНТ, АСПИРАНТ, ПРЕПОДАВАТЕЛЬ являются взаимоисключающими, поскольку читатель библиотеки выступает только в одной из ролей.

Рисунок 7 – Иерархия взаимоисключающих подтипов (дуга помечается цифрой 1)

Поперечные черточки на связях со стороны сущности ЧИТАТЕЛЬ означают, что подтипы обязательные, то есть каждый читатель должен быть либо студентом, либо аспирантом, либо преподавателем, а экземпляров базового класса ЧИТАТЕЛЬ не

существует.

3.2 Пример проектирования модели «сущность-связь»

Пример 1

Описание предметной области: транспортное предприятие выполняет грузовые перевозки по заявкам организаций и населения. Каждый автомобиль обслуживают два водителя, работающие посменно. Расписание работы каждого водителя хранится в базе данных.

Грузоперевозки выполняются по предварительным заявкам. Заявка включает: марку автомобиля, пункт назначения, время начала и окончания работ, задание, информацию о заказчике. Если заказчиком является организация, хранится ее название, ИНН, юридический адрес, телефон, Если заказчик – частное лицо, то хранятся паспортные данные и контактный телефон. В базе учитывается, какой именно автомобиль фактически обслужил заявку.

Одно из возможных решений задачи показано на рисунке 8. Имеются сущности АВТОМОБИЛЬ и ВОДИТЕЛЬ, связанные связью с кардинальностью 1:2, поскольку один автомобиль обслуживают два водителя. Связь со стороны автомобиля и со стороны водителя обязательная, поскольку к каждому исправному автомобилю прикреплен хотя бы один водитель, и водители, сами по себе, без автомобилей, не предусмотрены.

Расписание работы каждого водителя отражает сущность РАСПИСАНИЕ, она является слабой по отношению к сущности ВОДИТЕЛЬ. Атрибуты ВремяНачалаСмены и ВремяОкончанияСмены определяют время работы соответствующего водителя.

Сущность ЗАКАЗЧИК имеет два обязательных взаимоисключающих подтипа: ОРГАНИЗАЦИЯ и ЧАСТНОЕ ЛИЦО, которые отличаются наборами атрибутов. Идентификация сущностей этих подтипов происходит разными способами: у организаций ключевым атрибутом является ИНИ, а у граждан – паспортные данные.

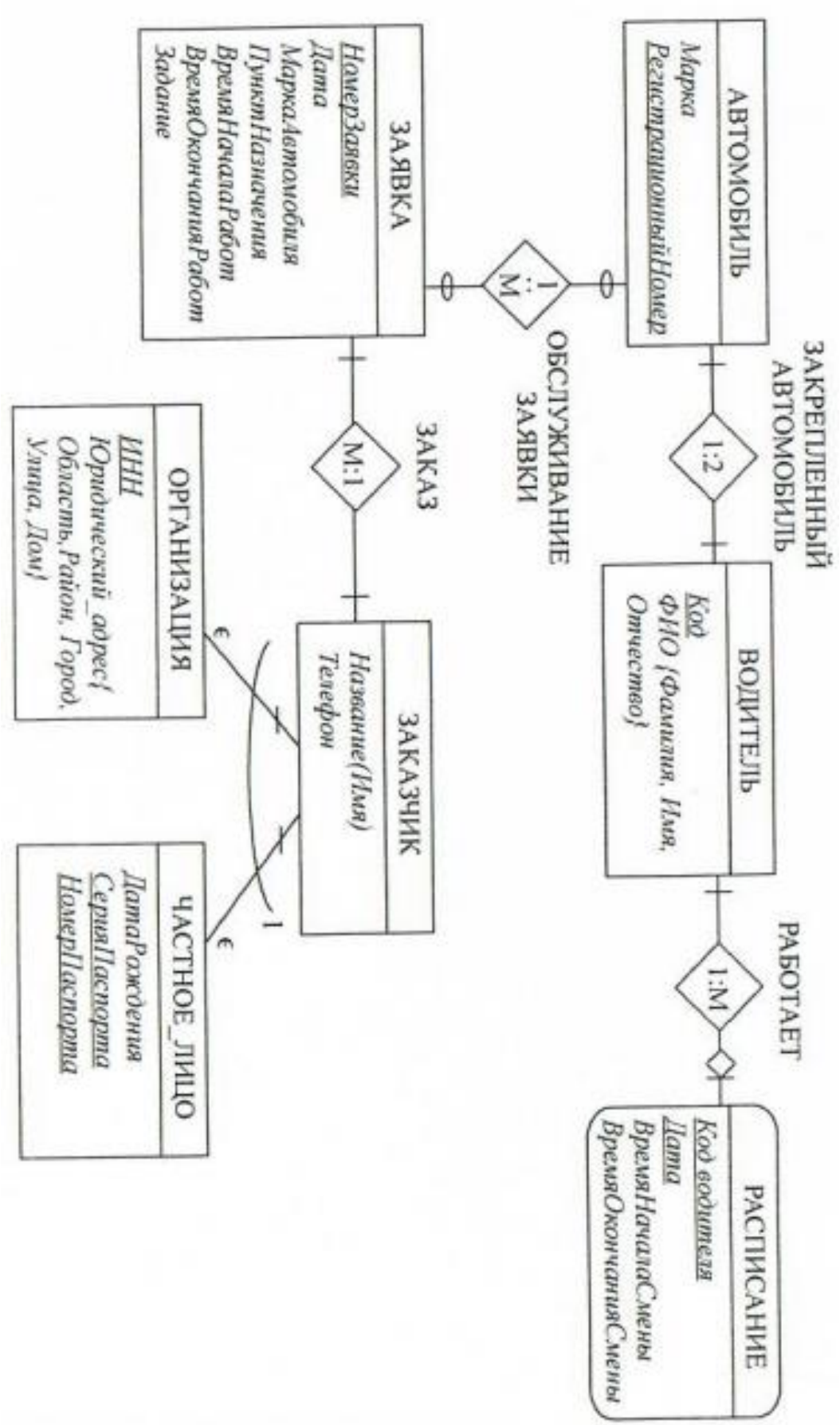


Рисунок 8 – Диаграмма «сущность-связь для примера 1

Каждый заказчик может делать несколько заказов, но заявка принадлежит ровно одному заказчику, поэтому связь ЗАКАЗЧИКА с сущностью ЗАЯВКА имеет тип 1:М. Связь обязательная с обеих сторон, потому что заявка обязательно чья-то, а заказчики регистрируются в базе только после того, как они подали заявку. Учет, была ли обслужена заявка или нет, н какой автомобиль обслужил данную заявку, отражает связь ОБСЛУЖИВАНИЕ ЗАЯВКИ. Она является необязательной с обеих сторон, Необязательность со стороны заявки показывает, что заявка может быть не обслужена вовсе. Необязательность связи со стороны автомобиля отражает случай, когда автомобиль (например, новый) еще не сделал ни одного выезда. Водитель, который обслужил данную заявку, определяется по времени своей работы в сущности

РАСПИСАНИЕ. Приведенная модель является одним из возможных способов решения задачи, но отнюдь не единственным.

4 Реляционная модель данных

4.1 Основные понятия реляционной модели

Модель «сущность-связь» не пригодна для машинного хранения данных. Большинство современных СУБД использует для внутреннего представления данных реляционную модель. В реляционной модели данные хранятся в форме таблиц. Столбцы таблицы описывают свойства (атрибуты) объекта, строка содержит описание экземпляра объекта.

Реляционная таблица называется отношением (relation). От «любой другой» таблицы отношение отличается тем, что:

1) ячейки содержат простые, неделимые (атомарные) значения,

2) порядок следования строк не важен,

3) не должно быть двух одинаковых строк.

Первое положение показывает, что в отношении не допускаются объединения ячеек, и не допускается разбиения одной ячейки на несколько. Второе и третье положение говорят, что строки таблицы рассматриваются как элементы множества (set). Из математического понятия множества следует, что элементы (в

данном случае строки) не могут повторяться и не имеют порядковых номеров. Столбы таблицы- отношения называются атрибутами, строки – кортежами.

Для однозначной идентификации строк используется ключ. Ключ – это один или более столбцов отношения, значения ячеек которых уникальны у каждой строки. Если ключ состоит из одного столбца, он называется простым, если группой столбцов – составным. Ключевые столбцы выбираются при проектировании реляционной таблицы с таким расчетом, чтобы ключе сохраняли свойство уникальности независимо от количества строк в таблице.

По определению отношение не может содержать двух одинаковых строк, поэтому в тривиальном случае ключ образуют все столбцы этой таблицы

Пример 2

Найти все нетривиальные ключи отношения. Отношение хранит информацию о работниках организации.

Работники

Табель-ный номер	Фамилия	Имя	Отчество	Дата рожде-ния	Сер. пас-порта	Номер пас-порта	Отдел	Должность
406	Поляков	Иван	Иванович	01.02.1998	38 00	015799	5	наладчик
409	Котова	Нина	Петровна	24.12.2002	38 21	200773	2	бухгалтер
519	Жилин	Егор	Ильич	18.06.1997	38 21	267005	5	сборщик

Все столбцы таблицы, кроме табельного номера, могут содержать повторяющиеся значения. Табельный номер у каждого работника свой – это простой ключ. Серия и номер паспорта в совокупности также обеспечивают уникальность – это составной ключ, хотя по отдельности значения серий и номера могут повторяться.

Итак, в таблице Работники есть два нетривиальных ключа Табельный номер и составной ключ (Серия паспорта, Номер паспорта).

Как показал пример, в отношении может быть несколько ключей. Все эти ключи называются потенциальными, или возможными, ключами. Один из них выбирают в качестве первичного ключа (Primary Key, PK). Он будет использоваться для идентификации строк и для организации связей между таблицами. Остальные потенциальные ключи с этих пор именуют

альтернативными ключами.

Реляционные таблицы могут связываться между собой посредством внешних ключей. Поясним организацию связей на примере.

Пример 3.

Информация об отпусках работников организации сохраняется в таблице Отпуск:

Отпуск

Табельный номер	Начало	Окончание
406	01.06.2022	03.07.2022
409	11.05.2022	20.05.2022
409	09.08.2022	31.08.2022
406	02.09.2022	02.10.2022

Таблица Отпуск связана с таблицей Работники из примера 2 по столбцу ТабельныйНомер. Каждая строка таблицы Отпуск указывает на строку таблицы Работники, содержащую соответствующее значение табельного номера. Строки таблицы Отпуск, где ТабельныйНомер=406, относятся к работнику Полякову И.И.; строки ТабельныйНомер=409 – к работнику Котовой Н.П.

Подобное разбиение данных на несколько таблиц делается для того, чтобы не было дублирования информации. При дублировании одинаковых данных в нескольких таблицах, во-первых, расходуется лишняя память, а, во-вторых, могут появиться противоречие в данных, если изменения забыли внести в дубликат.

Ссылки между таблицами организуются путем добавления в ссылающуюся таблицу столбцов, в точности повторяющих формат первичного ключа базовой таблицы. Эти столбцы называются внешним ключом (Foreign Key, FK). Внешний ключ может содержать только реально существующие значения первичного ключа базовой таблицы. То есть не допускается ситуация, когда внешний ключ указывает на не существующую строку базовой таблицы (ото называется ссылочной целостностью). Ссылочную целостность СУБД проверяют автоматически. При попытке ввода не действительного внешнего ключа СУБД генерируют ошибку.

Отношение может иметь несколько потенциальных ключей. Выбирая из них первичный ключ, стараются выбрать его самым

«легковесным», то есть занимающим минимальную память. Это делается потому что внешние ключи ссылающихся таблиц должны иметь тот же формат, что и первичный ключ целевой таблицы. А чем длиннее внешний ключ, тем больше памяти он займет при хранении. Во-вторых, поиск по короткому ключу быстрее, чем по длинному.

Однако выбирать первичный ключ, исходя только лишь из его легковесности, не стоит. Порой столбцы, входящие во внешние ключи, несут прямую смысловую нагрузку в подчиненных таблицах. Поэтому при выборе первичного ключа следует учитывать не только размер, но и роль этого ключа в ссылающихся таблицах.

Бывает, что все потенциальные ключи таблицы занимают довольно большой объем памяти, Чтобы не создавать такие же «тяжелые» внешние ключи, прибегают к использованию суррогатного ключа. Суррогатный ключ (Ersatz Key) – это новый, обычно числовой, столбец, добавляемый в целевую таблицу в качестве первичного ключа,

Он используется только для нумерации строк, и никакого иного смысла: не несёт. Суррогатный ключ обычно делают автоинкрементным (auto increment), или выбирающим свои значения случайным образом из некоего диапазона. Для таблиц, которые впоследствии могут быть объединены в одну (например, в распределенных базах данных), суррогатный ключ можно заполнять 8-байтовыми значениями GUID (Global Universal ID), чтобы избежать совпадения ключей.

4.2 Ограничения целостности

Каждый столбец СУБД хранит данные в одном из стандартных форматов (подробнее см. п. 6.1 «Стандартные типы данных»). На практике не все значения из диапазона, который способен хранить выбранный формат, являются логически корректными с точки зрения предметной области, Например, пусть для представления количества товара на складе выбран тип SMALLINT (2-байтовое целое). Его диапазон от 32768 до +32767, из него логически допустимыми являются числа >0, так как количество не отрицательно.

Ограничения целостности (integrity constraints) – это правила, ограничивающие логически допустимые значения столбцов с целью поддержания правильности и полноты хранящейся в базе данных информации.

Ограничения целостности делятся на декларативные и отложенные. Декларативные объявляются при создании таблиц БД, и за их выполнением следит, как правило, СУБД. Отложенные ограничения целостности – это более сложные правила, которые невозможно реализовать через декларативные ограничения, и приходится для этого применять языковые средства СУБД (триггеры, хранимые процедуры и функции). В этом параграфе будут рассмотрены виды декларативных ограничений. Об их программировании на языке SQL см. п.6.3.

1. Обязательное наличие данных (NULL-значения)

Ограничение целостности накладывается на столбец, и определяет, может ли этот столбец иметь пустые, не заполненные ячейки (так называемые NULL-значения).

2. Значение по умолчанию (DEFAULT)

Ограничение целостности задает значение по умолчанию для столбца таблицы. Это значение заносится в ячейку данного столбца, когда в команде вставки или обновления строки новое значение для этой ячейки не указано.

3. Первичный ключ (PRIMERY KEY)

Первичный ключ у таблицы всегда один. Это один или несколько столбцов, значения которых уникальны для каждой строки таблицы. Первичный ключ используется для организации связей между таблицами. Столбцы, входящие в первичный ключ, всегда обязательные (NOT NULL).

4. Уникальные столбцы (UNIQUE)

Это ограничение целостности используется для объявления альтернативных ключей. Подобно primary key указывает, что столбец или группа столбцов не могут сдержать повторяющихся значений. Все столбцы, входящие в ограничение UNIQUE, должны быть NIT NULL. Для организации связи 1:1 между таблицами ограничение UNIQUE накладывается на внешний ключ.

5. Ограничения на значения столбца (CHECK)

Это ограничение позволяет указать диапазон, список или

«маску» логически допустимых значений столбца.

6. Ссылочная целостность

СУБД, поддерживающие целостность ссылок по внешним ключам, автоматически проверяют, чтобы внешний ключ указывал только на существующую строку целевой таблицы. Внешние ключи могут ссылаться на столбцы первичного ключа и столбцы UNIQUE.

Зададимся вопросом: что станет со ссылающимися строками при удалении целевой строки таблицы, ведь внешний ключ должен указывать только на существующую строку целевой таблицы? На подобную ситуацию СУБД может реагировать одним из стандартных способов:

1) каскадное удаление (DELETE CASCADE) – при удалении целевой строки автоматически удаляются все ссылающиеся на нее строки подчиненных таблиц.

2) запрещение удаления (DELETE NO ACTION, DELETE RESTRICT) – СУБД запрещает удаление строки, пока на нее есть ссылки. Нужно сначала удалить все ссылающиеся строки, а затем – целевую строку.

3) установка внешнего ключа в NULL, или в значение по умолчанию (DELETE SET NULL / DELETE SET DEFAULT) – при удалении целевой строки поле внешнего ключа устанавливается в NULL, или значение по умолчанию.

Аналогичные типы реакции предусматриваются при изменении значения первичного ключа целевой таблицы: UPDATE CASCADE, UPDATE NO ACTION, UPDATE SET DEFAULT, UPDATE SET NULL.

Описанные выше декларативные ограничения целостности СУБД автоматически проверяет перед каждой операцией изменения таблицы. Если ограничения нарушаются, СУБД отменяет операцию и выдает сообщение об ошибке.

4.3 Правила преобразования ЕВ-модели в реляционную

1. Каждой сущности ставится в соответствие реляционная таблица. Однозначные атрибуты сущности становятся столбцами таблицы, идентификатор сущности – первичным ключом. Обязательные атрибуты сущности получают свойство NOT NULL,

необязательные – NULL. Производные атрибуты сущности преобразуются в вычислимые столбцы.

2. Каждому многозначному атрибуту сущности ставится в соответствие отдельная таблица. В нее добавляется внешний ключ, ссылающийся на соответствующую строку базовой таблицы. На него ставится ограничение DELETE CASCADE и UPDATE CASCADE.

3. Связь типа 1:1 реализуется так: в таблицу, соответствующую сущности с кардинальным числом M добавляется внешний ключ, ссылающийся на таблицу с кардинальным числом 1. Если связь не обязательная, внешний ключ получает свойство NULL, при обязательной связи – NOT NULL.

4. Связь типа 1:1 организуется несколькими способами:

– если связь с обеих сторон обязательная, две сущности сливаются в одну (одна общая таблица).

– если связь хотя бы с одной стороны не обязательная, то таблица, соответствующая не обязательной связи, становится базовой, а во вторую таблицу добавляется внешний ключ, отмеченный свойством UNIQUE.

5. Моделирование связи M:M происходит путем введения дополнительной таблицы, которая связана с каждой из исходных таблиц связью M:1. Она состоит из двух внешних ключей, ссылающихся соответственно на первую и вторую таблицы. Первичный ключ этой таблицы составной, состоит из совокупности всех внешних ключей.

6. Атрибуты связей преобразуются в дополнительные столбцы, добавляемые в таблицу, содержащую внешний ключ. Так, атрибут связи АБОНЕМЕНТ СрокВозврата (см. рис. 4) войдет в состав таблицы Книга. Атрибуты связей M:N добавляются как дополнительные столбцы в таблицу, представляющую связь (п.5).

7. Слабые сущности моделируются отдельной таблицей, связанной по внешнему ключу с таблицей, представляющей сильную сущность. Если слабая сущность идентификационно-зависимая, ее первичный ключ будет составным, он будет включать столбцы внешнего ключа (id сильной сущности). На внешний ключ слабой сущности накладывается правило DELETE

CASCADE и UPDATE CASCADE.

8. Для представления иерархии сущностей есть несколько способов:

– каждому типу сущности (супертипу и каждому подтипу) соответствует отдельная таблица. В таблицу супертипа переходят все атрибуты базовой сущности. В таблицы подтипов – атрибуты, специфические для каждого подтипа. Первичный ключ таблицы супертипа, как правило, суррогатный (идентификатор связи). В таблицы, представляющие подтипы, добавляют внешний ключ, указывающий на таблицу супертипа (внешний ключ одновременно является первичным ключом таблицы подтипа) со свойствами DELETE CASCADE, UPDATE CASCADE. Этот способ пригоден для любых иерархий сущностей.

– таблицы создаются только для подтипов, супертип таблицы не имеет. Набор столбцов каждой таблицы состоит из наследуемых атрибутов супертипа и атрибутов соответствующего подтипа. Способ подходит для моделирования взаимоисключающих подтипов. В реализации возникает ряд сложностей:

1) если пространство первичных ключей таблиц-подтипов общее;

2) если на супертип есть ссылки из других сущностей.

– третий способ пригоден только для взаимоисключающих подтипов. Все сущности, входящие в иерархию, представляются одной общей таблицей. Набор её столбцов является объединением атрибутов всех супер- и подтипов. Атрибуты, различающиеся у подтипов сущностей, помечаются как необязательные (NULL). Для различения одних подтипов от других обычно добавляется дополнительный столбец, где кодируется тип сущности.

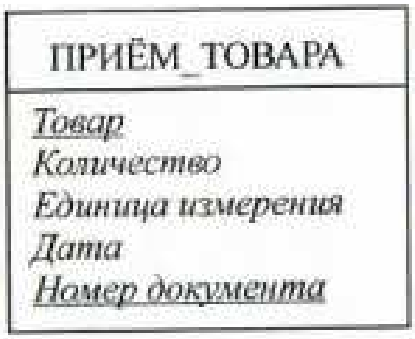
9. Связи между сущностями со степенью 3 и выше стараются разложить на бинарные связи, и затем промоделировать стандартным образом. Другой вариант – для моделирования связи степени и создается отдельная таблица, содержащая n внешних ключей (по числу связываемых классов сущностей), указывающих на соответствующие таблицы.

10. В полученной реляционной модели проводится нормализация (см. п.5).

5 Нормализация

5.1 Аномалии модификации

Случается, что в реляционных таблицах, имеющих определенную структуру. называемую аномальной, операции модификации (вставка, удаление, редактирование) могут приводить к нежелательным последствиям. Рассмотрим пример. Простенькая база данных склада хозтоваров хранит, когда какой товар получен, и в каком количестве. По одному документу (накладной) на склад могут поступить несколько различных товаров. ER-модель состоит из одной сущности ПРИЁМ_ТОВАРА (см. рис. 9, а), которая преобразуется в реляционную таблицу (см. рис. 9, б).



а)

Прием_товара

Товар	Кол-во	Единицы измерения	Дата	Номер док-та
Гвозди 100	10	кг	2021-10-21	58
Гвозди 120	10	кг	2021-10-21	58
Шланг резиновый	150	метр	2021-10-28	59
Брус 6000х40х40	12	куб.м	2021-11-02	60

б)

Рисунок 9 – ER-диаграмма (а); соответствующее ей отношение (б).

Если мы удалим строку, относящуюся к документу 59 (см. рис. 9, б), то удалим не только информацию о том, что по документу получен шланг, но и то, шланг обычно измеряется в метрах. Удаляя факты, относящиеся к одной области (по накладной № 59 получен шланг), мы непроизвольно удаляем факты,

относящиеся к другой области (единицей измерения шлангов является погонный метр). Это называется аномалией удаления (deletion anomaly).

Таблица Прием_товара имеет аномалию вставки (insertion anomaly). Аномалия вставки проявляется в том, что мы не можем записать в таблицу некоторый факт, не указав дополнительно другой факт. Предположим, мы хотим записать в базу, что фанера измеряется в листах, однако мы не можем внести эти данные в таблицу Прием_Товара, пока на склад не поступит хотя бы один лист фанеры.

Третий тип аномалий – аномалия обновления (update anomaly). Допустим, оператор ошибся пои вводе, и документ 58 нужно было отнести не к 21, а к 22 октября 2021 г. (см. рис. 9, 6). Чтобы исправить эту ошибку, нужно найти все строки, относящиеся к документу 58, и изменить в них ячейку Дата на 2021-10-22. Здесь аномалия модификации проявляется в том, что при изменении одного факта (накладная №58 создана 21.10.2021), мы вынуждены модифицировать несколько строк таблицы.

Аномальная структура таблиц приводит к избыточности данных. Факт, что документ №58 создан 21.10.2021, продублирован в нескольких строках. Избыточность не только тратит лишнюю память, но допускает существование противоречивых данных (один и тот же документ может относится к разным датам, если ошибиться при вводе).

Аномалии, присутствующие в таблице Прием_Товара, можно описать следующим образом: проблемы возникают из-за того, что таблица содержит факты, относящиеся к различным темам:

1) в каких единицах измеряет товар;

2) когда создан документ;

3) какие товары получены по данному документу.

Когда мы добавляем строку, нам приходится добавлять информацию, затрагивающую различные темы; точно так же, когда мы удаляем строку, мы вынуждены удалять данные, относящиеся сразу к трем темам.

Суть нормализации состоит в том, чтобы разбивать таблицы, содержащие несколько тем, на две или более таблицы, каждая из которых будет содержать ровно одну тему. Так, аномальную

таблицу Приём_Товара можно было бы разбить па три таблицы:

Товары (Товар(РК), Единица измерения)

Документы(Номер документа(РК), Дата)

Поступления_Товар(FK, Номер документа (FK), Количество)

Нетрудно проверить, что полученные таблицы лишены аномалий.

5.2 Нормальные формы

В 70-х годах XX века теоретики реляционных баз данных обнаруживали различные типы аномалий модификации, вызванные структурой отношений. Классы отношений, лишенные аномалий определенного типа, называются нормальными формами (normal forms). Известно семь нормальных форм: первая, вторая, третья, четвертая, пятая нормальные формы (1НФ, 2НФ, ЗНФ, 4НФ, 5НФ), нормальная форма Бойса-Кодда (НФБК) и доменно-ключевая нормальная форма (ДКНФ). Нормальные формы являются вложенными друг в друга (см. рис. 10). То есть отношение во второй нормальной форме является отношением в первой нормальной форме, а отношение в 5НФ одновременно находится в 4НФ, НФБК, ЗНФ, 2НФ, 1НФ.

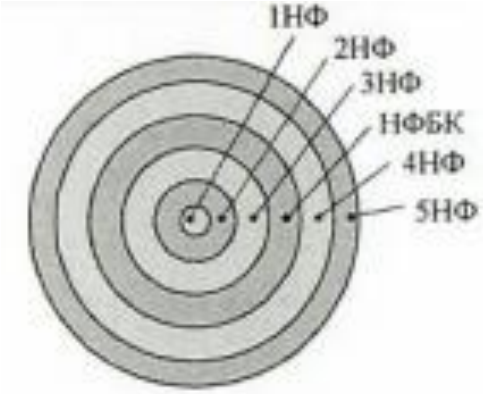


Рисунок 10 – Нормальные формы

Первая нормальная форма (1st normal form) - 1НФ

Таблица находится в 1НФ, если она удовлетворяет определению отношения.

Таблица, находящаяся в 1НФ может быть подвержена аномалиям. Более старшие нормальные формы позволяют избежать определенных типов аномалий.

Прежде чем перейти к изучению второй нормальной формы, необходимо ввести понятие Функциональной зависимости между атрибутами отношения.

Функциональная зависимость (functional dependency) – такая логическая связь между атрибутами отношения, при которой по известному значению одного атрибута можно найти (или вычислить) значение другого атрибута. Например, зная название товара, по таблице Прием_товара (см. рис. 9, 6), можно определить единицу измерения этого товара, и это значение будет единственным: атрибут «Единица измерения» функционально зависит от атрибута «Товар». Обратное утверждение, что «Товар» функционально зависит от «Единиц измерения», не верно. Зная единицу измерения (например, куб.м), можно определить множество товаров, но не один единственный.

Функциональная – зависимость обозначается стрелкой Товар – Единица измерения. Атрибуты, стоящие слева от стрелки, называются детерминантом, справа от стрелки – зависимостью.

Более формально можно определить функциональную зависимость так: если A и B – атрибуты в таблице T, то функциональная зависимость: A→B обозначает, что если две строки в таблице T имеют одно и то же значение атрибута A, то они имеют одно и то же значение атрибута B. Это определение также применимо, если A и B – множества столбцов, а не просто отдельные столбцы.

Вторая нормальная форма (2^{nd} normal form) –2НФ

Известно, что по значению первичного ключа можно однозначно определить значения остальных ячеек этой строки. Следовательно, все неключевые атрибуты функционально зависят от первичного ключа.

Первичный ключ таблицы Приём_Товара составной – (Товар, Номер документа), от него зависят неключевые атрибуты:

(Товар, Номер документа) → Количество

(Товар, Номер документа) → Единица измерения

(Товар, Номер документа) → Дата

Доказано, что отношение будет иметь аномалии модификации, если существуют зависимости между его атрибутами, при которых неключевые атрибуты функционально

зависят от части составного первичного ключа.

В таблице Приём_Товара такие зависимости есть:

Товар → Единица измерения

Номер документа → Дата

Для ликвидации аномалий исходную таблицу следует разбить на более мелкие так, чтобы детерминанты функциональных зависимостей образовали первичные ключи этих таблиц В нашем примере три детерминанта: Товар, Номер документа и (Товар, Номер документа), – образуют первичные ключи трех таблиц:

Т1(Товар, Единица измерения)

Т2(Номер документа, Дата)

Т3(Товар, Номер документа, Количество)

Получившийся результат совпадает с полученным из интуитивных соображений (п. 5.1). Таблицы Т1, Т2, Т3 лишены аномалий, они соответствуют второй нормальной форме:

Определение: таблица находится во второй нормальной форме (2НФ), если она удовлетворяет определению 1НФ и все ее атрибуты, не входящие в первичный ключ, функционально зависят от первичного ключа н не зависят от части первичного ключа. Отсюда следует вывод: все таблицы с простым первичным ключом находятся во 2НФ.

Рекомендации: если в таблицу введен суррогатный ключ (для замены длинного информационного ключа на более короткий), то при проведении нормализации следует мысленно подменять суррогатный ключ на информационный ключ.

Третья нормальная форма (3^d normal form) – 3НФ

Рассмотрим отношение

Поставщики(Имя, Город, Страна, Код межгорода, Телефон)

Атрибут Имя – первичный ключ. Поскольку первичный ключ простой, отношение находится в 2НФ. В отношении существуют следующие функциональные зависимости:

Имя → Город /*вытекает из определения первичного ключа*/

Имя → Страна /*вытекает из определения первичного ключа */

Имя → Код межгорода /*из определения первичного ключа*/

Имя → Телефон /* из определения первичного ключа*/

Город → Страна

Город → Код межгорода

Таблица имеет аномалии. Например, переход населенного пункта в ведение другого государства вызывает модификацию всех строк, в которых он упоминается. Кроме того, таблица Поставщики избыточна, и допускает ввод противоречивых данных (один и тот же город в разных государствах). Аномалии в таблице обусловлены функциональной зависимостью между неключевыми атрибутами

Город → Страна

Город → Код межгорода

Определение: таблица находится в третьей нормальной форме {3НФ), если она удовлетворяет определению 2НФ и не существует функциональных зависимостей между неключевыми атрибутами.

Для ликвидации аномалий таблицу Поставщики следует разбить на две таблицы:

Города(Город, Страна, Код межгорода)

с функциональными зависимостями:

Город → Страна, Город → Код межгорода, и таблицу

Поставщики(Имя, Город (FK), Телефон)

с функциональными зависимостями:

Имя → Город, Имя → Телефон

Каждая из полученных таблиц находится в ЗНФ и лишена аномалий модификации.

Нормальная форма Бойса-Кодда (Boyce-Codd normal form, BC/NF) – НФБК

Первичный ключ таблицы Города – это строковый столбец. С целью экономии памяти введем числовой первичный ключ КодГор:

Города(КодГор, Город (Unique), Страна, Код межгорода)

Выпишем функциональные зависимости:

/*из определения первичного ключа следует:*/

КодГор → Город

КодГор → Страна

КодГор → Код межгорода

/*прежние функциональные зависимости остались в силе:*/

Город → Страна,

Город → Код межгорода

/*поскольку Город и КодГор взаимозаменяемы:*/

Город → КодГор

Таблица Города лишена аномалий модификации, однако после добавления суррогатного первичного ключа она противоречит определению ЗНФ. Для приведения к ЗНФ се следовало бы разбить на две таблицы:

КодыГородов(КодГор, Город (Unique))

СведенияОГородах(КодГор, Страна, Код межгорода),

хотя интуитивно ясно, что это абсурд. Столкнувшись с подобными несуразностями, которые могут возникать не только из-за введения суррогатных первичных ключей, Бойс и Кодд обосновали и предложили более строгое определение ЗНФ, которое учитывает, что в таблице может быть несколько возможных ключей.

Определение: таблица находится в нормальной форме Бойса-Кодла (НФБК), если неключевые атрибуты функционально зависят только от возможных ключей, и не зависят от частей этих потенциальных ключей.

Таблица Города находится в НФБК, поскольку КодГор и Город являются возможными ключами отношения.

НФБК заключает в себе ограничения ЗНФ (в смысле отсутствия функциональных зависимостей между неключевыми атрибутами) и 2НФ (в смысле запрета на функциональные зависимости от части составного первичного ключа).

Четвертая нормальная форма (4^{th} normal form) – 4НФ

В отношениях возможны другие виды аномалий, связанные с наличием многозначных зависимостей (multivalue dependency) между атрибутами. По определению, атрибут А многозначно определяет атрибут В той же таблицы, если для каждого значения атрибута А существует хорошо определенное множество соответствующих значений В (обозначается А→>>В, читается: «атрибут А многозначно определяет В»).

Например, в таблице Города по названию страны можно было определить множество названий (или кодов) городов, следовательно, Страна →> Город (Страна →> КодГор). При рассмотрении 2НФ, ЗНФ, НФБК мы не обращали внимания на многозначные зависимости в таблицах Прием Товара, Поставщики, хотя они и были: Единица_измерения →> Товар, (Номер документа, Дата) →>Товар, Страна →> Город. Здесь аномалии

были вызваны наличием нежелательных функциональных зависимостей.

В модели «сущность-связь» вводилось понятие многозначных атрибутов. Многозначный атрибут – это массив однотипных значений. Например, список ключевых слов сущности КНИГА (см. рис. 4). При переходе от ER-модели к реляционной многозначные атрибуты выносятся в отдельную таблицу, связанную связью M:1 с таблицей, представляющей сущность (п.4.3,(2)). Если этого не сделать, а поместить многозначные атрибуты вместе с однозначными в одну таблицу, то возникает избыточность и, возможно, аномалии. Например, если все атрибуты сущности КНИГА (см. рис.4) поместить в одну таблицу, то строки, относящиеся к одной книге, различаются только в последнем столбце.

Инв №	Шифр	Автор	Название	Изд-во	Год	Ключ. слово
11706	681.3С65	Сорокина С.Н.	Программирование драйверов	СПБ:БХВ	2002	драйвер
11706	681.3С65	Сорокина С.Н.	Программирование драйверов	СПБ:БХВ	2002	пакет запроса
11706	681.3С65	Сорокина С.Н.	Программирование драйверов	СПБ:БХВ	2002	сервис
40067	82.3Ш23	Шапарова Н.А.	Энциклопедия славянской мифологии	М: Астрель	2003	баба-Яга

Наличие в отношении нескольких многозначных атрибутов (зависимостей) помимо избыточности приводит к аномалиям вставки, удаления и обновления. Рассмотрим пример. Таблица хранит информацию о совместных проектах:

Проекты(Код, Название, Исполнители (FK), Партнёры (FK), Начало, Окончание)

Столбец Исполнители содержит коды сотрудников, участвующих в проекте (многозначный атрибут). Столбец Партнёры содержит ссылки на сторонние организации, вовлеченные в проект (также многозначный атрибут). Столбцы Начало и Окончание указывают период работы над проектом. В отношении имеются следующие функциональные и многозначные зависимости:

Код $\rightarrow$ Название

Код → Начало

Код → Окончание

Код →> Исполнители

Код →> Партнёры

Заполнение строк таблицы денными, если исполнителей три, а организаций-партнеров две, выглядит так:

Код	Название	Исполнители	Партнеры	Начало	Окончание
401	Расшифровка генома	10	6	01.04.2022	31.03.2023
401	Расшифровка генома	11	6	01.04.2022	31.03.2023
401	Расшифровка генома	12	6	01.04.2022	31.03.2023
401	Расшифровка генома	10	7	01.04.2022	31.03.2023
401	Расшифровка генома	11	7	01.04.2022	31.03.2023
401	Расшифровка генома	12	7	01.04.2022	31.03.2023

Проекту 401 посвящено 6 строк: сочетание каждого исполнителя с каждой организацией. Если бы те же данные хранились в меньшем количестве строк, то удаление одной строки могло бы повлечь потерю нескольких фактов.

Несовершенство структуры таблицы проявляется в избыточности (столбцы Название, Начало и Окончание повторяются), аномалиях удаления (при увольнении исполнителя приходится удалять несколько строк), аномалиях вставки (при добавлении новой организации приходится вставлять новые строки по числу исполнителей) и аномалии модификации (изменение любого факта вызывает модификацию нескольких строк).

Аномалии в таблице Проекты обусловлены наличием двух многозначных зависимостей в одной таблице Код →> Исполнители и Код →> Партнёры. Многозначные атрибуты Исполнители и Партнеры зависят от общего атрибута Код, и не зависят друг т друга.

Определение: отношение находится в четвертой нормальной форме (4НФ), если оно находится в НФБК, и содержит единственную многозначную зависимость А→>В, где А – возможный ключ отношения.

Чтобы привести таблицу Проекты к 4НФ, ее нужно разбить на две таблицы (по числу многозначных зависимостей): в одну вынести Код-Исполнители, во вторую – Код-Партнеры.

T1(Код, Название, Исполнители (FK), Начало, Окончание)

T2(Код, Партнёры (FK))

А еще лучше добавить третью таблицу, и в нее переместить однозначные атрибуты проекта, чтобы избавиться от избыточности данных.

Проекты(Код, Название, Начало, Окончание)

Участники(Код(FK), Исполнители (FK))

Организации(Код(FK), Партнёры (FK))

Пятая нормальная форма (5^{th} normal form) – 5НФ

Пятая нормальная форма затрагивает отношения, которые имеют несколько многозначных атрибутов, и эти атрибуты зависимы между собой. В таблице

Обучение(Специальность, Дисциплина, Семестр, Преподаватель)

три многозначных атрибута:

1) дисциплина – список дисциплин, изучаемых специальностью;

2) семестр – список семестров, в которых дисциплина изучается на данной специальности (например, Информатика на «ВМ» изучается в 1,2 семестрах, на «ПО» – в 1 семестре, на «ЭК» – во 2 и 3 семестрах);

3) преподаватель – список преподавателей, которые ведут занятия по данной дисциплине у соответствующей специальности.

Многозначные зависимости

Специальность $\rightarrow>$ Дисциплина

(Специальность, Дисциплина) $\rightarrow>$ Семестр

(Преподаватель, Специальность) $\rightarrow>$ Дисциплина

Если таблицу Обучение не нормализовывать, придется хранить всевозможные сочетания многозначных атрибутов (Специальность, Дисциплина, Семестр, Преподаватель) – данные избыточные и запутанные, Для упрощения исходную таблицу следует разбить на более мелкие таблицы по количеству многозначных зависимостей:

T1(Код1, Специальность, Дисциплина)

/*Код1 – заменитель пары значений (Специальность, Дисциплина)*/

T2(Код1(FK), Семестр)

Т3(Код1(FK), Преподаватель)

Доменно-ключевая нормальная форма (domain/key normal form) – ДКНФ

Определение; отношение находится в доменно-ключевой нормальной форме, если каждое ограничение целостности, накладываемое на это отношение, является логическим следствием определения доменов и ключей.

Доказано, что таблицы, находящиеся в ДКНФ, лишены каких бы то ни было аномалий модификации. К сожалению, общего подхода, позволяющего привести таблицу к ДКНФ) пока не существует.

5.3 Процедура нормализации и денормализации

При составлении модели «сущность-связь», а затем реляционной модели данных, следует планировать данные так, чтобы каждая таблица содержала ровно одну тему. Это поможет избежать аномалий в таблицах.

Далее каждую таблицу необходимо проверить на соответствие нормальным формам в следующем порядке (и при необходимости разбить на более мелкие таблицы):

1НФ $\Rightarrow$ 2НФ $\Rightarrow$ НФБК $\Rightarrow$ 4НФ $\Rightarrow$ 5НФ $\Rightarrow$ ДКНФ

Нормализация таблиц имеет свои плюсы и минусы. Существенным плюсом является то, что пропадают аномалии модификации, избыточность, хранение противоречивой информации. Минус нормализации проявляется в замедленной выборке данных. После нормализации количество таблиц возрастает; информация, которая раньше лежала в одной таблице, теперь разбросана по нескольким. Чтобы составить комплексный отчет, приходится просматривать несколько таблиц, что занимает больше времени, чем поиск в одной ненормализованной таблице. В некоторых базах данных это замедление поиска оказывается столь существенным, что выгоднее пренебречь нормализацией, зато выиграть в производительности. Тогда выполняют обратный процесс – денормализацию – намеренное соединение нормализованных таблиц в не нормализованные. Логику работы прикладных программ, обрабатывающих базу данных, дополняют процедурами дополнительной поддержки – целостности и

непротиворечивости ненормализованных данных.

5.4 Пример проектирования реляционной базы данных

Пример 4

Задание: преобразовать модель «сущность-связь» из примера 1 в реляционную модель данных. Провести нормализацию таблиц.

Проектирование реляционной базы данных заключается в определении структуры таблиц, связей между ними, доменов и ограничений целостности. На рисунке 11 показана схема связи реляционных таблиц. Она получена из модели «сущность-связь» (см. рис. 8) по правилам, изложенным выше. Каждой сущности соответствует отдельная таблица:

сущности АВТОМОБИЛЬ – таблица AUTO,

сущности ВОДИТЕЛЬ – таблица DRIVER,

сущности РАСПИСАНИЕ – таблица SHEDULE,

сущности ЗАЯВКА – таблица REQUEST.

Моделирование взаимоисключающих подтипов ОРГАНИЗАЦИЯ и ЧАСТНОВ_ЛИЦО выполнено по третьему варианту правила (см. п.4.3, правило 8); таблица CUSTOMER (ЗАКАЗЧИК) содержит атрибуты, общие для обоих подтипов (название Name и телефон Phone). Первичный ключ таблицы CUSTOMER суррогатный – столбец ID_Cust. Значения этого столбца являются внешним и первичным ключом одновременно в таблицах PERSON и ORGANIZATION, моделирующих подтипы сущностей ЧАСТНОЕ _ЛИЦО и ОРГАНИЗАЦИЯ. Например, для регистрации заказчика Сидорова П.И. в таблице CUSTOMER добавляется строка (315; Сидоров П.И; NULL). 315 – его регистрационный номер, NULL – телефона нет. В таблице PERSON добавляется строка (315; 19.02.1994; 38 21; 169805). Число 315 является ссылкой на строку таблицы CUSTOMER, и первичным ключом таблицы PERSON.

Моделирование связей M:1 (в том числе связи 2:1 между водителем и автомобилем) происходит путем добавления внешних ключей в таблицу, со стороны которой кардинальное число связи «M». Если связь обязательная, внешний ключ NOT NULL, иначе – NULL.

Связь	Таблица.Столбец	Обязательность
ЗАКРЕПЛЕННЫЙ_АВТОМОБИЛЬ	DRIVER.RegNumAuto	NOT NULL
РАБОТАЕТ	SHEDULE.Driver_ID	NOT NULL
ОБСЛУЖИВАНИЕ_ЗАЯВКИ	REQUEST.RegNumAuto	NULL
ЗАКАЗ	REQUEST.Cust_ID	NOT NULL

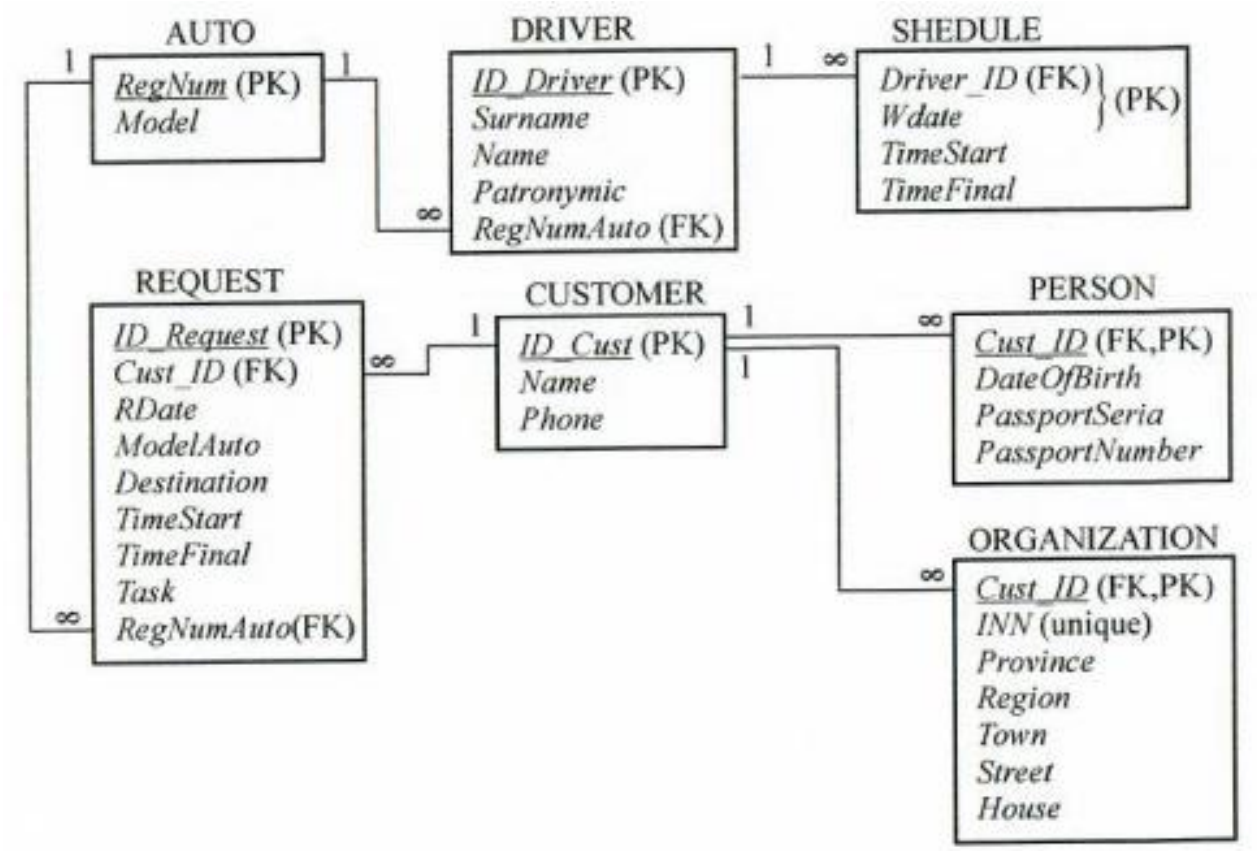


Рисунок 11 – Схема связи таблиц реляционной модели. Первичные ключи таблиц обозначены PK, внешние FK. Связи по внешним ключам имеют тип M:1 (∞:1).

Обязательность связи со стороны сущности, где кардинальное число равно «1», никак не моделируется в реляционной модели (как, например, связь ЗАКАЗ со стороны сущности ЗАКАЗЧИК).

Проектирование доменов (п. 6.2). В каждый домен для нашего примера входит один первичный ключ и все внешние ключи, ссылающиеся на него.

Имя домена	Назначение домена	Столбцы, принадлежащие домену	Базовый тип	Ограничения целостности домена
TRegNum	Регистрационные номера автомобилей	AUTO.RegNum DRIVER.RegNumAuto REQUEST.RegNumAuto	CHAR(14)	NUT NULL, маска «б ччч бб» бб – буквы,

				чч – числа
TModelAuto	Модели автомобилей	AUTO.Model REQUEST.ModelAuto		
TDrive_ID	Код водителя	DRIVER.ID_Driver SHEDULE.Driver_ID	SMALLINT	NOT NULL, >0
TDate	Дата	SHEDILE.WDate REQUEST.Rdata	DATA	Нет
TTime	Время	SHEDULE.TimeStart SHEDULE.TimeFinal REQUEST.TimeStart REQUEST.TimeFinal	TIME	Нет
TCust_ID	Номера заказчиков	CUSTOMER.ID_Cust PERSON.Cust_ID ORGANIZATION.Cust_ID	SMALLINT	NOT NULL, >0

Примечание: модели автомобилей можно записывать в виде строки символов («Газ», «Камаз»). Недостатки этого способа проявляются в следующем: 1) если при оформлении заявки записать название модели заглавными буквами, то поиск свободных автомобилей данной марки не даст результатов, так как строки «Газ» и «ГАЗ» не равны; 2) строка занимает больше байтов памяти, чем число. Рациональнее будет создать дополнительную таблицу МОДЕЛИ_АВТОМОБИЛЕЙ (Models) со столбцами (КодМодели, НазваниеМодели), и везде вместо названия модели записывать ее код. Таким образом, домен TModelAuto будет числовым.

Таблица [Models]

Имя столбца	Тип	NULL/ NOT NULL	Default	PRIMARY KEY/ Unique	Check	FOREIGN KEY и другие огран. целостности	Примечание
ID_Model	TModel Auto	См. домен	Авто-инкрем.	Primary key	См. домен		Код модели
ModelType	VARCHAR(20)	NOT NULL					Название модели

Таблица [Auto]

Имя столбца	Тип	NULL/ NOT NULL	Default	PRIMARY KEY/ Unique	Check	FOREIGN KEY и другие огран. целостности	Примечание
RegNum	TRegNum	См. домен	Авто-инкрем.	Primary key	См. домен		Рег.номер автомобиля
Model	TModelAuto	См. домен			См. домен	FK Models.ID_Model DELETE NO ACTION)	Ссылка на модель

Таблица [Driver]

Имя столбца	Тип	NULL/ NOT NULL	Default	PRIMARY KEY/ Unique	Check	FOREIGN KEY и другие огран. целостности	Примечание
ID_Drver	TDriver_ID	См. домен	Авто-инкрем.	Primary key	См. домен		Код водителя
Surname	NVARCHAR(30)	NOT NULL					Фамилия
Name	NVARCHAR(20)	NOT NULL					Имя
Patronymic	NVARCHAR(30)	NOT NULL					Отчество
RegNumAuto	TModelAuto	См. домен			См. домен	*FK Auto.RegNum DELETE NO ACTION	Ссылка на авто

* - не более двух строк таблицы могут иметь равные значения столбцы RegNumAuto.

Таблица [Shedule]

Имя столбца	Тип	NULL/ NOT NULL	Default	PRIMARY KEY/ Unique	Check	FOREIGN KEY и другие огран. целостности	Примечание
Driver_ID	TDriver_ ID	См. домен		Primary key	См. домен	FK Driver. .ID_Driver	Код водителя DELETE CASCADE UPDATE CASCADE
WDate	TDate	NOT NULL			См. домен		Дата
TimeStart	TTime	NOT NULL			>TimeFinal		Время начала смены
TimeFinal	TTime	NOT NULL					Время конца смены

Таблица [Customer]

Имя столбца	Тип	NULL/ NOT NULL	Default	PRIMARY KEY/ Unique	Check	FOREIGN KEY и другие огран. целостности	Примечание
ID_Cust	TCust_ID	См. домен	Авто-инкрем	Primary key	См. домен		Код заказчика
Name	NVARCHAR(70)	NOT NULL					Имя заказчика
Phone	VARCHAR(12)	NULL					Телефон

В ER-модели подтипы сущностей ОГАНИЗАЦИЯ и ЧАСТНОЕ_ЛИЦО идентифицировались по-разному. Идентификатором организации был атрибут ИНН, идентификатором человека (СерияПаспота, НомерПаспорта) (рис. 8). При переходе к реляционной модели в таблицы PERSON и ORGANIZATON был добавлен новый первичный ключ Cust_ID, идентифицирующие атрибуты сущностей превратились в обычные столбцы с ограничением Unique.

Таблица [Person]

Имя столбца	Тип	NULL/ NOT NULL	Default	PRIMARY KEY/ Unique	Check	FOREIGN KEY и другие огран. целостности	Примечание
Cust_ID	TCust_ID	См. домен		Primary key	См. домен	FK Customer. .ID_Cust	Код заказчика
DateOfBirth	DATE	NOT NULL					Дата рождения
PassportSeria	CHAR(5)	NOT NULL		Unique			Серия паспорта
PassportNumber	NUMERIC (6,0)	NOT NULL			>0		Номер паспорта

INN	NUMERIC(15,0)	NOT NULL		Unique	>0		ИНН
Province	NVARCHAR(31)	NULL					Область
Region	NVARCHAR(31)	NULL					Район
Town	NVARCHAR(31)	NOT NULL					Город, село
Street	NVARCHAR(47)	NULL					Улица
House	VARCHAR(12)	NULL					Дом, корпус

6 Реализация баз данных

6.1 Стандартные типы данных

Тип данных определяет формат, в котором хранится значение столбца. Стандарт SQL предусматривает несколько базовых типов данных. Каждая СУБД обычно помимо базовых типов предоставляет программисту свои собственные типы данных для повышения функциональности системы. В таблице 1 приведены типы данных, принятые в СУБД Microsoft SQL Server, MySQL, InterBase(=Firebird).

6.2 Домены и пользовательские типы данных

В реляционной модели доменом называется множество допустимых значений столбца таблицы: у каждого столбца – свой домен. Некоторые СУБД поддерживают специальный объект базы данных «домен» (domain), который представляют собой именованный пользователем стандартный тип данных (из таблицы 1), с которым могут быть связаны декларативные ограничения целостности. Домены применяются, когда несколько столбцов, обычно из разных таблиц, хранят логически одинаковые данные.

Например, в таблице Сотрудники есть столбец АдресСотрудника, и в таблице Клиенты столбец АдресКлиента. Адреса предполагается хранить в виде текстовой строки VARCHAR длиной 60 байт. Для адресов можно объявить домен с именем (например, TAddress). И связать его с типом VARCHAR(60). Далее при создании таблиц вместо типа столбцов АдресСотрудника и АдресКлиента указывается имя домена TAddress. В дальнейшем, если потребуется изменить формат хранения адресов (например, увеличить длину строки, или поменять тип с VARCHAR на NVARCHAR), достаточно изменить настройки домена, и СУБД автоматически поменяет формат всех

столбцов этого домена. Если создавать базу данных вообще без доменов, то администратору базы данных придется ручками менять типы столбцов в нескольких таблицах – долго и больше вероятность ошибиться.

Таблица 1

Название типа	Описание	Поддержка СУБД		
		SQL Server	MySQL	Fire-bird
1	2	3	4	5
Целые числа				
TINYINT	1-байтовое число (со знаком или без него: –128..+127 или 0..255)	+	+	–
SMALLINT*	2-байтовое число (со знаком или без него)	+	+	+
MEDIUMINT	3-байтовое число (со знаком или без него)	–	+	–
INT*, INTEGER*	4-байтовое число (со знаком или без него)	+	+	+
BIGINT	8-байтовое число (со знаком или без него)	+	+	–
Числа с плавающей запятой				
FLOAT*	Вещественное число одинарной точности в диапазоне: 1,17E-38..3.4E+38 (MySQL), 2,22E-308..1,79E+308 (SQL Server)	+	+	+
DOUBLE* DOUBLE PRECISION*	Вещественное число двойной точности в диапазоне 2,22E–308..1,79E+308 (MySQL)	–	+	+
REAL*	Вещественное число в диапазоне 2,22E-308..1,79E+308 (MySQL) 1,17E-38..3.4E+38 (SQL Server)	+	+	–
Числа с фиксированной запятой				
DECIMAL(d,p)* DEC(d,p)* NUMERIC(d,p)*	Используется для точного представления вещественных десятичных чисел со знаком: d (*dimension*) – *размерность*, общее количество знаков; p (*precision*) – *точность*, количество знаков после запятой. Например, числа: три знака перед запятой, два знака после (±000,00..999,99) определяются как DECIMAL(5,2)	+	+	+
Денежный тип				
MONEY	Совместим с типом DECIMAL; в MySQL это эквивалент DECIMAL(12,2), в SQL Server аналог DECIMAL(15,4)	+	+	–
SMALLMONEY	В SQL Server аналог DECIMAL(10,4)	+	–	–
Строки символов				
CHAR(n)* CHARACTER(n)*	Строка фиксированной длины, занимает в памяти *n* байт независимо от реальной длины содержащейся в ней строки.	+	+	+
NCHAR(n)* NATIONAL CHAR(n)*	Строка фиксированной длины в кодировке Unicode, занимает 2*n* байт.	+	+	+
VARCHAR(n)*	Строка переменной длины. Занимает в памяти количество байт, равное реальной длине строки +1. При объявлении *n* – максимальная длина строки.	+	+	+
NVARCHAR(n)* NATIONAL VARCHAR(n) NCHAR VARYING(n)*	Строка переменной длины в кодировке Unicode. Используется для хранения строк с национальными символами алфавита.	+	+	+
Биты и битовые массивы				
BIT (MySQL) BOOL (SQL Server)	Булевский тип (0 или 1), занимает в памяти один бит.	+	+	–
BIT(n)* BINARY(n)	Массив из *n* бит, каждый элемент принимает значение 0 или 1	+	+	–
BIT VARYING(n)* VARBINARY(n)	Массив бит переменной длины (максимальная длина *n*)	+	+	–
Дата и время				
DATE*	Дата от '1000-01-01' до '9999-12-31'	–	+	+
DATETIME	Дата и время от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'	+	+	–
SMALLDATETIME	Дата и время от '1900-01-01 00:00' до '2079-06-06 23:59'	+	–	–
TIMESTAMP*	Дата и время, диапазоны зависят от СУБД	+	+	+
TIME*	Время с точностью до секунд от 00:00:00 до '838:59:59'	–	+	+
YEAR	Год	–	+	–
INTERVAL	Временной интервал	–	–	–

Таблица 1 (продолжение)

1	2	3	4	5
Перечисления и множества				
ENUM	Множество до 65535 элементов	–	+	–
SET	Множество до 255 элементов	–	+	–
Неструктурированные объекты				
TINYBLOB, TINYTEXT	Блок памяти размером до 255 байт	–	+	–
BLOB, TEXT	Блок памяти размером до 64Кб	–	+	+
TEXT(n)	Блок памяти размером до 2Гб	+	–	–
NTEXT(n)	Блок памяти размером до 2Гб (вмещает до 1 Гб символов кодировки Unicode)	+	–	–
MEDIUMBLOB, MEDIUMTEXT	Блок памяти размером до 16 Мб	–	+	–
LONGBLOB, LONGTEXT	Блок памяти размером до 4 Гб	–	+	–
IMAGE	Массив бит длиной до $2^{31}-1$ байт	+	–	–

В InterBase/Firebird домен создается командой

CREATE DOMAIN Имя_домена AS ТИП
[DEFAULT {значения_по_умолчанию | NULL | USER}]
[NOT NULL]
[CHECK (условие_проверки)]

Здесь ТИП – имя стандартного типа данных из таблицы 1.

Имя домена указывается вместо стандартного типа данных при объявлении столбца.

Пример

- создаем домен «табельный номер сотрудника»

```
CREATE DOMAIN DTabNum
AS INTEGER
    DEFAULT 1000 NOT NULL
    CHECK (Value BETWEEN 1000 AND 9999)
    -- использование домена при создании таблиц

CREATE TABLE Workers{                    -- таблица «Работники»
    TabNum DTabNum PRIMERY KEY           -- табельный номер
    Name VARCHAR(60) NOT NULL,           -- имя сотрудника
    ...)

CREATE TABLE Leave{                      -- таблица «Отпуск»
    TabNum DTabNum FOREGN KEY REFERENCES
Workers(TabNum),                         -- табельный номер
    StartDate DATE NOT NULL,             -- начало отпуска
    FinDate DATE);
```

Бывает, что Firebird некорректно выполняет запросы из-за того, что столбцы, принадлежащие одному базовому типу, не объявлены принадлежащими одному домену

Домен можно изменить командой ALTER DOMAIN и удалить командой DROP DOMAIN.

В MS SQL вместо термина «домен» используется понятие «пользовательский тип данных» (user-defined data type). Его создание проходит в два этапа: 1) объявление типа данных; 2) связывание с этим типом ограничений целостности.

Объявляется пользовательский тип данных системной хранимой процедурой sp_addtype. Вот се формат (имена параметров начинаются с символа @)

sp_addtype
[@typename=] Имя_пользовательского_типа_данных,
[@phystype=] стандартный тип данных
[, [@nulltype=] «NULL» или «NOT NULL»
[, [@owner=] «имя_пользователя_создавшего_тип»]

Вызов этой процедуры для объявления пользовательского типа «табельный номер сотрудника» в SQL будет выглядеть так:

EXEC sp_addtype @typename=DTabNum, @phystype=INTEGER, @nulltype='NOT NULL'

SQL-операор EXEC вызывает хранимую процедуру. Имена параметров, начинающиеся с @, можно не писать, тогда фактические значения параметров следует перечислять в том же порядке, что и в заголовке процедуры. Предыдущий вызов можно переписать так:

EXEC sp_addtype DTabNum, INTEGER, 'NOT NULL'

Второй этап – связывание с пользовательским типом данных ограничений целостности. Значение по умолчанию задается объектом базы данных «умолчание» (Default). «Умолчание» создастся SQL-оператором

CREATE DEFAULT Имя_Умолчания AS значение по умолчанию

«Умолчание» связывается с пользовательским типом хранимой процедурой sp_bindefault. Et формат:

sp_bindefault

[@defname=] 'Имя_Умолчания',

[@objname=] 'Имя_пользовательского_типа_данных'

[, [@futureonly=] ‘futureonly’ или ‘NULL’ по умолчанию]

CHECK-подобные ограничения целостности задаются объектом ‘базы данных «правило» (rule). Правило создается оператором

CREATE RULE Имя правила AS условие_проверки

/*в условии проверки значение проверяемой переменной обозначается любым именем, которое начинается с символа @*/

Правило связывается с пользовательским типом данных хранимой процедурой sp_bindrule

sp_bindrule

[@defname=] ‘Имя Правила’,

[@objname=] ‘Имя пользовательского_типа_данных’

[, [@futureonly=] 'futureonly' или 'NULL’ по умолчанию]

Полный SQL-скрипт для создания домена «табельный номер сотрудника» в MS SQL будет выглядеть так:

EXEC sp_addtype DTabNum, INTEGER, ‘NOT NULL’;

CREATE DEFAULT TabNumDef AS 1000;

EXEC sp_bindefault 'TabNumDef', 'DTabNum';

CREATE RULE ‘TabNumRange AS @Value BETWEEN 1000 AND 9999;

EXEC sp_bindrule 'TabNumRang', ‘DTabNum’;

Пользовательский тип данных удаляется хранимой процедурой sp_droptype:

sp_droptype Имя_пользовательского_типа_данных

Для переименования пользовательского типа данных служит процедура sp_rename. Кстати, эта процедура может переименовывать и другие объекты СУБД SQL базы данных, таблицы, столбцы, представления, хранимые процедуры и пр.

6.3 Создание баз данных и таблиц

Разновидность языка, применяемая в конкретной СУБД, называется диалектом SQL. Например, диалект СУБД Oracle называется PL/SQL; в MS SQL и DB2 применяется диалект Transact-SQL; в Interbase и Firebird – isql. Каждый диалект SQL совместим до определенной степени со стандартом SQL, но может

иметь отличия и специфические расширения языка, поэтому для выяснения синтаксиса того или иного SQL-оператора следует в первую очередь смотреть Help конкретной СУБД.

Для операций над базами данных и таблицами в стандарте SQL предусмотрены операторы:

CREATE DATABASE – создать новую базу данных

DROP DATABASE – удалить базу данных

SET DARABASE, – сделать базу данных текущей

USE

CREATE TABLE – создать таблицу

ALTER TABLE – изменить структуру существующей таблицы (добавить/удалить столбцы или ограничения целостности)

DROP TABLE – удалить таблицу.

Ниже приводится синтаксис этих операторов по стандарту SQL. Поскольку их синтаксис в СУБД может отличаться от стандарта, при выполнении лабораторной работы рекомендуется обращаться к справочной системе СУБД.

Имена объектов базы данных (таблиц, столбцов и др.) могут состоять из буквенно-цифровых символов и символа подчеркивания. Специальные символы (@$# и т.п.) обычно указывают на особый тип таблицы (системная, временная и др.). Не рекомендуется использовать в именах национальные (русские) символы, пробелы и зарезервированные слова, но если они всё же используются, то такие имена следует писать в кавычках «...» или в квадратных скобках [..].

Далее при описании конструкций операторов SQL будут использоваться следующие обозначения: в квадратных скобках [] записываются необязательные части конструкции; альтернативные конструкции разделяются вертикальной чертой |; фигурные скобки {} выделяют логические блоки конструкции; многоточие ... указывает на то, что предшествующая часть конструкции может многократно повторяться. «Раскрываемые» конструкции записываются в угловых скобках < >.

Создание базы данных

CREATE DATABASE Имя_базы_ данных

Удаление одной и более баз данных

DROP DATABASE Имя_базы_данных [, Имя_базы данных ...]

Объявление текущей базы данных

USE Имя_базы_данных – в MySQL

SET DATABASE Имя_базы_данных – в Firebird

Создание таблицы.

CREATE TABLE Имя таблицы (

<описание столбца> [, <описание_столбца> |

<ограничение_целостности_таблицы>...]

)

<описание столбца> →

Имя_столбца ТИП [DEFAULT значение_по_умолчанию]

[NOT NULL]

{[UNIQUE |PRIMARY KEY] }

|

{[REFRENCES Имя_таблицы(Имя_столбца)]

[ON DELETE

{NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

[ON UPDATE

{NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

}

[CHECK (условие_проверки)]

ТИП столбца может быть либо стандартным типом данных (см. таблицу 1), либо именем домена (см. п.6.2).

Некоторые СУБД позволяют создавать вычислимые столбцы (computed columns). Это виртуальные столбцы, значение которых не хранится в физической памяти, а вычисляется сервером СУБД при всяком обращении к этому столбцу по формуле, заданной при объявлении этого столбца. В формулу могут входить значения других столбцов этой строки, константы, встроенные функции и глобальные переменные.

Описание вычислимого столбца в SQL имеет вид:

<описание_столбца> → Имя_столбца AS выражение

Описание вычислимого столбца в Firebird имеет вид:

<описание_столбца> → Имя_столбца COMPUTED BY <выражение>

СУБД MySQL вычислимые столбцы не поддерживает.

<ограничение_целостности_таблицы>

CONSTRAINT Имя_ограничения_целостности

{UNIQUE | PRIMARY KEY} (список_столбцов_образующих_ключ)

| FOREIGN KEY (список_столбцов_FK)

REFRENCES Имя_таблицы(список_столбцов_PK)

[ON DELETE

{NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

[ON UPDATE

{ NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

| CHECK (условие_проверки)

Некоторые СУБД допускают объявление временных таблиц (существующих только во время сеанса). В SQL имена временных таблиц должны начинаться с символа # (локальные временные таблицы, видимые только создавшему их пользователю) или ## (глобальные таблицы, видимые всем пользователям); в MySQL для создания временных таблиц используется ключевое слово TEMPORARY, например:

CREATE TEMPORARY TABLE ... (далее синтаксис см. CREATE TABLE).

Изменение структуры таблицы

Используется для изменения типа столбцов существующих таблиц, добавления и удаления столбцов и ограничений целостности.

ALTER TABLE Имя_таблицы

– изменение типа столбца (в SQL и Firebird)

ALTER COLUMN Имя_столбца новый_ТИП

– изменение типа, имени и ограничений столбца (в MySQL)

CHANGE COLUMN Имя_столбца <описание_столбца>

– добавление обычного или вычислимого столбца

| ADD <описание_столбца>

– добавление ограничения целостности

| [WITH CHECK | WITH NO CHECK] ADD <ограничение_целостности_таблицы>

– удаление столбца

| DROP COLUMN Имя_столбца

– удаление ограничения целостности

| DROP CONSTRAINT Имя_ограничения_целостности

– включение или отключение проверки ограничений целостности

– в My SQL

| {CHECK | NO CHECK} CONSTRAINT

{Список_имен_ограничений_целостности | ALL}

Удаление таблицы

DROP TABLE Имя_таблицы

Далее рассмотрим, как при создании новых таблиц командой CREATE TABLE или изменении структуры существующих таблиц командой ALTER TABLE объявить декларативные ограничения целостности (подробнее они описаны в п.4.2)

1. Обязательное наличие данных (NULL–значения)

Объявляется словом NULL (столбец может иметь пустые ячейки) или NOT NULL (столбец обязательный). По умолчанию принимается NULL.

Пример создания таблицы:

CREATE TABLE Clients(

ClientName NVARCHAR(60) NOT NULL,

DataOfBirth DATE NULL,

Phone CHAR(12)); – по умолчанию тоже NULL.

2. Значение по умолчанию (DEFAULT)

Значение по умолчанию можно задать для каждого столбца таблицы, Если при модификации ячейки ее новое значение не указано, сервер вставляет значение по умолчанию. Значение по умолчанию может быть NULL, константой, вычислимым выражением или системной функцией.

Рассмотрим пример создания таблицы Orders (Заказы). Столбец OrderDate принимает по умолчанию значение текущей даты, а столбец Quantity (количество) по умолчанию равен 0.

CREATE TABLE Orders(

OrderNum INT NOT NULL, – номер заказа

OrderDate DATETIME NOT NULL – дата заказа

DEFAULT GetDate(),

– функция GetDate() возвращает текущую дату

Quantity SMALLINT NOT NULL – кол-во товара,

DEFAULT 0);

3. Объявление первичных ключей (PRIMARY KEY)

Простой первичный ключ объявляется словами PRIMARY

KEY при создании таблицы. Например,

```
CREATE TABLE Staff( – таблица «Работники»
TabNum INT PRIMARY KEY, – первичный ключ
WName NVARCHAR(40) NOT NULL, – ФИО
... – описание прочих столбцов );
```

Составной первичный ключ объявляется иначе:

способ 1 (объявление PK при создании таблицы)

```
CREATE TABLE Clients(
PasSeria NUMERIC(4,0) NOT NULL, – серия паспорта
PasNumber NUMERIC(6,0) NOT NULL, – номер паспорта
Name NVARCHAR(40) NOT NULL,
Phone CHAR(12),
– объявление составного первичного ключа
CONSTRAINT Clients_PK
PRIMARY KEY(PasSeria, PasNumber) );
– способ 2(PK объявляется после создания таблицы)
– сначала создаем таблицу без PK
CREATE TABLE Clients(
PasSeria NUMERIC(4,0) NOT NULL, – серия паспорта
PasNumber NUMERIC(6,0) NOT NULL, – номер паспорта
ClientName NVARCHAR(40) NOT NULL,
Phone CHAR(12) );
– модификация таблицы - добавляем PK
ALTER TABLE Clients
ADD CONSTRAINT Clients_PK
PRIMARY KEY (PasSeria, PasNumber);
```

4. Уникальность столбцов (UNIQUE)

Подобно Primary Key указывает, что столбец или группа столбцов не могут содержать повторяющихся значений, но не являются PK. Все столбцы, объявленные UNIQUE, должны быть NOT NULL. Пример объявления простого уникального столбца:

```
CREATE TABLE Students(
SCode INT PRIMARY KEY,  суррогатный PK
FIO NVARCHAR(40) NOT NULL, – ФИО
RecordBook CHAR(5) NOT NULL UNIQUE); – № зачетки
```

Пример объявления составного уникального поля:

```
CREATE TABLE Staff( – таблица «Работники»
```

TabNum INT PRIMARY KEY, – табельный номер
WName NVARCHAR(40) NOT NULL, – ФИО
PasSeria NUMERIC(4,0) NOT NULL, – серия паспорта
PasNumber(6,0) NOT NULL, – номер паспорта
– объявление составного уникального поля
CONSTRAINT Staff_UNQ UNIQUE(PasSeria, PasNumber));

5. Ограничения на значения столбца (CHECK)

Это ограничение позволяет указать диапазон, список или «маску» логически допустимых значений столбца.

Пример создания таблицы Workers (Работники):

CREATE TABLE Workers(
– табельные номера 4-значные
TabNum INT PRIMARY KEY
CHECK(TabNum BETWEEN 1000 AND 9999),
Name VARCHAR(60) NOT NULL, – ФИО сотрудника
– пол - буква «м» или «ж»
Gentry CHAR(1) NOT NULL
CHECK(Gentry IN ('м', 'ж')),
– возраст не менее 14 лет
Age SMALLINT NOT NULL CHECK(Age>=14),
– № свидет-ва пенсионного страхования (по маске)
PensionCert CHAR(14)
CHECK (PensionCert LIKE ‘_ _ _-_ _ _-_ _ _-_ _’));

В этом примере показаны разные типы проверок. Диапазон допустимых значений указывается конструкцией BETWEEN ... AND; обычные условия (как для столбца Age) используют знаки сравнений =, <>, >, >=, <, <=, связанные при необходимости логическими операциями AND, OR, NOT (например, Age>=14 AND Age<=70); для указания списка допустимых значений используется предикат IN и его отрицание NOT IN; конструкция

LIKE маска_допустимых_значений EXCEPT список_исключений

используется для задания маски допустимых значений строковых столбцов. В маске применяются два спецсимвола: «%» – произвольная подстрока, и «_» – любой единичный символ. Конструкция EXCEPT является необязательной.

В условии отбора CHECK могут сравниваться значения двух

столбцов одной таблицы и столбцы разных таблиц.

6. Ссылочная целостность

Простой внешний ключ объявляется конструкцией FOREIGN KEY REFERENCES ...

```
CREATE TABLE Leave( – таблица «Отпуск»
– табельный номер работника – внешний ключ на таблицу Staff
TabNum INT FOREIGN KEY REFERENCES Staff(TabNum),
StartDate DATE NOT NULL, – начало отпуска
FinDate DATE); – окончание отпуска
```

Составной внешний ключ объявляется через constraint при создании таблицы командой CREATE TABLE или при модификации структуры командой ALTER TABLE:

```
CREATE TABLE Order( – таблица «Заказы»
PrderNum INT PRIMARY KEY – номер заказа
ODate DATE NOT NULL, – дата заказа
– столбцы внешнего ключа
ClPastSeria NUMERIC(4,0) NOT NULL, – серия паспорта
ClPasNumber NUMERIC(6,0) NOT NULL, – номер паспорта
– объявление составного внешнего ключа, ссылающегося на таблицу Clients
CONSTRAINT Order_FK
FOREIGN KEY(ClPasSeria, ClPasNumber)
REFERENCES Clients(PastSeria, PasNumber) );
```

Для организации связи 1:1 между таблицами ограничение UNIQUE накладывается на внешний ключ.

Стандартный тип реакции сервера на удаление или модификацию строки, на которую ссылаются через внешний ключ строки других таблиц: CASCADE, NO ACTION, RESTRICT, SET NULL, SET DEFAULT, объявляется при создании внешнего ключа после слов ON DELETE и ON UPDATE.

```
CREATE TABLE Leave( – таблица «Отпуск»
TabNum INT FOREIGN KEY REFERENCES Staff(TabNum)
ON DELETE CASCADE
ON UPDATE CASCADE,
StartDate DATE NOT NULL,
FinDate DATE NULL);
```

СУБД MySQL внешние ключи позволяет объявлять, но целостность по внешним ключам не проверяет (в целях совместимости с будущими версиями).

6.4 Создание автоинкрементных столбцов

Автоинкрементный столбец (autoincrement field) – это целочисленный столбец, значение которого автоматически увеличивается или уменьшается с постоянным шагом при добавлении новой строки. Например, у первой строки этот столбец =1, у второй =2, у третьей =3, и т.д. Когда строка удаляется (например; с номером 3), значение 3 все равно больше использоваться не будет, поскольку автоинкрементный столбец всегда только увеличивается, не возвращаясь назад. Автоинкрементные столбцы используются для создания суррогатных первичных ключей.

Создание автоинкрементных столбцов в MS SQL

Автоинкрементный столбец объявляется при описании столбца командами CREATE TABLE или ALTER TABLE с помощью конструкции

IDENTITY(начальное_значение, шаг_приращения)

Например,

```
CREATE TABLE Records(
RecNum INT IDENTITY(0,1) PRIMARY KEY,
/*первичный ключ – это автоинкрементное поле с начальным значением 0 и шагом приращения +1*/
Field_1 CHAR(10), ... – описание прочих столбцов
);
```

Тип автоинкрементного столбца должен быть целочисленным (tinyint, smallint, int, bigint) или с фиксированной запятой (decimal(p,0); numeric(p,0)). Свойство IDENTITY не применяется совместно со свойствами DEFAULT и NULL. Разрешается иметь только один автоинкрементный столбец в таблице.

Для генерации значений автоинкрементного столбца сервер использует скрытую переменную-счетчик. Сервер следит за тем, чтобы при удалении какой-либо записи ее порядковый номер не был использован повторно.

Создание автоинкрементных столбцов в MySQL.

MySQL разрешает иметь в таблице только один автоинкрементный столбец (счётчик). Он объявляется флагом AUTO_INCREMENT при создании таблицы. Тип столбца-счетчика – беззнаковый целый.

Таблицы типа MyISAM и InnoDB следят за тем, какие значения генерируются счетчиком, поэтому при удалении какой-либо записи ее порядковый номер не будет использован повторно. В таблицах других типов порядковый номер вычисляется путем прибавления единицы к максимальному значению столбца. Если из таблицы удалить все записи, нумерация снова начнется с единицы.

Тип таблицы н начальное значение счетчика можно задать табличными опциями

TYPE = тип таблицы

AUTO_INCREMENT = начальное_значение /*действует только на таблицы типа MyISAM*/

Опция тип_таблицы определяет формат хранения таблицы и функциональные возможности. MySQL поддерживает семь типов таблиц: ISAM, MyISAM (по умолчанию), Heap, Merge, InnoDB, Gemini, BerkleyDB (BDB). Поддержка транзакций реализована только в последних трех типах таблиц. Табличные опции указываются после операторов CREATE TABLE или ALTER TABLE, например:

```
CREATE TABLE Records(
/*столбцы*/
RecNum INT UNSIGNED NOT NULL AUTO_INCREMENT,
Field_1 CHAR(10) NULL,
... /*описание прочих столбцов*/
/*ограничения целостности*/
PRIMARY KEY (RecNum)
)
/*табличные опции*/
TYPE = MYISAM
AUTO_INCREMENT = 100
```

Создание автоинкрементных столбцов в Firebird

Автоинкрементные столбцы реализуются с помощью генераторов и триггеров.

Генератор (generator) – это специальный объект базы данных.

Он представляет собой целочисленную 4-байтовую переменную.

Генератор создается командой

CREATE GENERATOR Имя_генератора

По умолчанию значение нового генератора равно 0. Его можно изменить командой

SET GENERATOR Имя_генератора TO новое_значение

Новое значение может быть числом от -2^{31} до $+(2^{31}-1)$.

Для получения следующего значения генератора предназначена функция

GEN_ID(Имя_генератора, шаг_приращения)

Функция прибавляет шаг_приращения к текущему значению генератора и возвращает полученное число. Шаг приращения может быть положительным или отрицательным.

Для создания автоинкрементного столбца нужно предварительно создать генератор. Значение генератора будет заноситься в ячейку автоинкрементного столбца при добавлении новой строки. Эта операция автоматически выполняется триггером.

Триггер (trigger) – блок команд SQL, которые автоматически выполняются сервером в ответ на операцию с таблицей. Каждый триггер привязан к конкретной таблице. По типам операций, вызывающих запуск триггера, триггеры делятся на три типа:

INSERT – добавление новых строк,

UPDATE – модификация существующих строк,

DELETE – удаление строк.

По времени запуска триггеры Firebird делятся на два типа:

BEFORE – триггер запускается до внесения изменений в таблицу;

AFTER – триггер запускается после фиксации изменений в таблице. Если с одним и тем же событием (например, модификацией строк таблицы) связано несколько триггеров, то они вызываются по очереди в соответствии со своей позицией (0, 1, 2, ...).

Триггер создается оператором CREATE TRIGGER

CREATE TRIGGER Имя_триггера FOR Имя_таблицы
{BEFORE|AFTER} {INSERT|UPDATE|DELETE}
[POSITION значение_позиции]

AS BEGIN

<тело триггера. Операторы разделяются ; >

END <символ-ограничитель>

Символ-ограничитель по умолчанию – точка с запятой. Но поскольку «;» используется в теле триггера, то требуется переопределить ограничитель. Для этого используется оператор

SET TERM новый_ограничитель

Новый ограничитель – это произвольная (в пределах разумного) подстрока.

Ниже приведен пример создания триггера на вставку строк. Триггер заносит в ячейку автоинкрементного столбца новое значение генератора.

```
/*создаем таблицу – столбец RecNum будет автоинкрементным*/
CREATE TABLE Records(
RecNum INT NOT NULL PRIMARY KEY,
...); /*прочие столбцы*/
/*создаем генератор (по умолчанию равен 0)*/
CREATE GENERATOR RecNum_Gen;
/*создаем триггер: предварительно меняем символ-ограничитель с «;» на «!!»*/
SET TERM !!
/*объявление триггера*/
CREATE TRIGGER CreateRecNum FOR Records
BEFORE INSERT
POSITION 0
AS BEGIN
NEW.RecNum – GEN_ID(RecNum_Gen, 1);
END !!
/*меняем символ-ограничитель обратно*/
SET TERM ; !!
```

NEW – это имя виртуальной таблицы, содержащей добавляемую строку. Таблица NEW состоит из одной строки, и существует только внутри тела триггера. Для изменения ее ячейки RecNum используется оператор

NEW.RecNum = GEN_ID(RecNum_Gen, 1);

Функция GEN_ID автоматически увеличивает значение

генератора на 1, и возвращает его новое значение.

7 Содержательная часть

Лабораторное занятие №1

Составление моделей «сущность-связь». Решение задач на проектирование баз данных с использованием модели «сущность-связь»

Цель занятия: научиться составлять модель «сущность-связь»

Ход занятия

1. У преподавателя получить индивидуальный вариант задания.
2. Изучить теоретическую часть методических указаний.
3. Для своего варианта разработать модель «сущность-связь». Показать модель преподавателю для исправления возможных ошибок и неточностей
4. Оформить отчет.

Лабораторное занятие №2

Семантическое моделирование базы данных по индивидуальным вариантам. Для заданной предметной области проектирование модели «сущность-связь»

Цель занятия: научиться выделять сущности, определять связи между ними, составлять модель «сущность-связь»

Ход занятия

1. У преподавателя получить индивидуальный вариант задания. Задание выдается на бригаду 1-3 человека и остается постоянным на все последующие занятия.
2. Изучить теоретическую часть методических указаний.
3. Для своего варианта выделить все сущности, установить типы связей между ними.
4. Разработать модель «сущность-связь». Показать модель преподавателю для исправления возможных ошибок и неточностей
5. Оформить отчет.

Лабораторное занятие №3

Проектирование баз данных в рамках реляционной модели. Знакомство с реляционной моделью. Решение задач на преобразование модели «сущность-связь» в реляционную модель. Нормализация

Цель занятия: научиться проектировать реляционную модель базы данных, проводить нормализацию

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Для своего варианта модели «сущность-связь» выполнить преобразование в реляционную модель.
3. Выполнить нормализацию таблиц при необходимости. Показать модель преподавателю для исправления возможных ошибок и неточностей
5. Оформить отчет.

Лабораторное занятие №4

Создание реляционной базы данных с СУБД Firebird. Знакомство с СУБД Firebird. Создание в ней реляционной базы данных по заданной структуре

Цель занятия: получить навыки работы с СУБД Firebird

Ход занятия

1. Для заданной структуры модели «сущность-связь» создать реляционную базу данных.
2. Показать базу преподавателю для исправления возможных ошибок и неточностей
3. Оформить отчет.

Лабораторное занятие №5

Создание реляционной базы данных по индивидуальному варианту. Проектирование модели «сущность-связь» и ее реализация в СУБД

Цель занятия: получить навыки реализации модели «сущность-связь» в СУБД Firebird

Ход занятия

1. Для своего варианта структуры модели «сущность-связь» создать реляционную базу данных.
2. Задать ограничения целостности, накладываемые на столбцы таблиц.
3. Заполнить базу информацией (не менее 10 записей). Показать базу преподавателю для исправления возможных ошибок и неточностей
4. Оформить отчет.

Лабораторное занятие №6

Изучение языка SQL: однотабличные запросы. Синтаксис SQL-оператора SELECT, поиск данных в таблице, вычислимые столбцы, сортировка (решение задач)

Цель занятия: изучить синтаксис SQL-оператора SELECT

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы на поиск данных в таблице, на вычисление столбцов, на сортировку (решение задач)
3. Оформить отчет.

Лабораторное занятие №7

Изучение языка SQL: агрегатные функции. Синтаксис SQL-оператора SELECT, группировка и агрегатные функции

Цель занятия: изучить синтаксис SQL-оператора SELECT

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы на группировку данных, запросы, выполняющие агрегатные функции (решение задач)
3. Оформить отчет.

Лабораторное занятие №8

Изучение языка SQL: многотабличные запросы. Операция соединения таблиц (INNER, LEFT, RIGHT, FULL JOIN), сортировка в многотабличных запросах

Цель занятия: изучить синтаксис языка SQL

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы на соединение таблиц в различных вариантах, сортировку данных в связанных таблицах (решение задач)
3. Оформить отчет.

Лабораторное занятие №9

Изучение языка SQL. Групповые функции в многотабличных запросах

Цель занятия: изучить синтаксис языка SQL

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы на использование функций, работающих с несколькими связанными таблицами (решение задач)

3. Оформить отчет.

Лабораторное занятие №10

Изучение языка SQL: вложенные запросы. Использование в SQL-запросах предикатов IN, EXIST, ANY, ALL. Измерение времени выполнения и оптимизация SQL-запросов.

Цель занятия: изучить синтаксис языка SQL, получить навыки использования предикатов в запросах.

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы на использование предикатов IN, EXIST, ANY, ALL (решение задач)
3. Оформить отчет.

Лабораторное занятие №11

Изучение языка SQL: объединения. Использование в SQL-запросах оператора UNION. Использование групповых функций и сортировки с объединенных запросах

Цель занятия: изучить синтаксис языка SQL, получить навыки использования оператора UNION.

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы на объединение таблиц (решение задач).
3. Составить SQL-запросы, содержащие групповые функции и функции сортировки (решение задач)
4. Оформить отчет.

Лабораторное занятие №12

Изучение языка SQL: вставка, удаление, модификация записей. Изучение операторов INSERT, UPDATE, DELETE

Цель занятия: изучить синтаксис языка SQL, изучение операторы INSERT, UPDATE, DELETE.

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Составить SQL-запросы с операторами INSERT, UPDATE, DELETE (решение задач).
3. Оформить отчет.

Лабораторное занятие №13

Хранимые процедуры и триггеры. Изучение синтаксиса хранимых процедур. Создание хранимых процедур

Цель занятия: изучить правила создания хранимых процедур и триггеров, получить навыки создания хранимых процедур.

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Создать хранимую процедуру (для своего варианта базы данных).
3. Создать триггер (для своего варианта базы данных)
4. Оформить отчет.

Лабораторное занятие №14

Текущий контроль знаний. Контрольная работа по языку SQL

Цель занятия: проверить усвоение знаний по синтаксису языка SQL.

Ход занятия

1. Получить задание от преподавателя.
2. Выполнить задание.
3. Оформить отчет.

Лабораторное занятие №15

Администрирование базы данных. Создание пользователей и выдача прав доступа к объектам базы данных

Цель занятия: изучить процедуру администрирования базы данных, получить навыки определения прав доступа к базе данных.

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Создать администратора базы данных (своего варианта), наделить правами администратора.
3. Создать пользователя базы данных, наделить правами пользователя.
4. Оформить отчет.

Лабораторное занятие №16

Построчная обработка данных. Курсоры. Иерархические запросы. Средства современных СУБД для построчной обработки таблиц (хранимые процедуры, курсоры). Операторы SQL для работы с курсорами. Обзор средств SQL для реализации иерархических запросов. Решение задач

Цель занятия: изучить операторы языка SQL для работы с курсорами

Ход занятия

1. Изучить теоретическую часть методических указаний.
2. Создать SQL-запрос для работы с курсорами.
3. Оформить отчет.

Лабораторное занятие №17

Проектирование приложений баз данных на языке высокого уровня. Обзор технологий доступа к базам данных (ADO, ODBC). Компоненты языков программирования для работы с базами данных (C++ Builder, Delphi). Пример проектирования пользовательского интерфейса

Цель занятия: научиться проектировать пользовательский интерфейс.

Ход занятия

1. Разработать интерфейс пользователя.
2. Используя компоненты языков высокого уровня для работы с базами данных реализовать спроектированный интерфейс.
3. Оформить отчет.

Лабораторное занятие №18

Контроль результатов работы. Тестирование баз данных

Цель занятия: тестирование программных продуктов – баз данных.

Ход занятия

1. Проверить работоспособность программных приложений, используя подготовленные комплекты тестов.
2. Провести устное собеседование по различным темам.

8 Контрольные вопросы к лабораторным занятиям

1. Опишите, что такое: база данных, СУБД, приложение базы данных.

2. Охарактеризуйте понятия: сущность, связь. В чем различия между классом и экземпляром сущности (или связи)?

3. Перечислите типы бинарных связей. Приведите примеры связи каждого типа.

4. Какие виды атрибутов допускаются в модели «сущность-связь»?

5. Что показывает кардинальное число связи: минимальное и максимальное?

6. Как отличить, являются ли две связанные друг с другом

сущности сильной и слабой по отношению друг к другу, или независимыми по силе?

7. Что представляет собой *отношение* в реляционной модели? Чем отношение отличается от любых других таблиц?

8. Какие декларативные ограничения целостности можно наложить на реляционную таблицу?

9. Зачем нужен первичный ключ в реляционной таблице? Может ли существовать таблица без ключа (если да, то когда; если нет, то почему). Как выбрать первичный ключ?

10. Что такое внешний ключ и зачем он нужен.

11. Как организовать связь 1:1 между реляционными таблицами? Какие ограничения целостности при этом используются?

12. Как при переходе от ER-модели к реляционной моделируется связь «многие-ко-многим»?

13. Подумайте, как при переходе к реляционной модели реализуется троичная связь: *отец–мать→дети* с атрибутом связи: *дата рождения ребенка* (здесь «отец», «мать», «дети» – классы сущностей).

14. Что понимается под аномальной и нормальный структурой таблиц? Приведите примеры аномалий вставки, удаления и модификации.

15. В чем состоит суть нормализации.

16. Дайте определения нормальным формам (1НФ … 5НФ).

17. Что такое функциональная зависимость между атрибутами отношения. Найдите функциональный зависимости в таблице, содержащей информацию о подписке граждан на журналы и газеты.

Подписка(ФИО_подписчика, Адрес, НазваниеИздания,
ПодписнойИндекс, Год, Месяц)

Один человек может подписаться на несколько изданий, и в каждом издании – на несколько месяцев.

Если таблица имеет аномалии, нормализуйте ее.

18. В чем, по вашему мнению, преимущества и недостатки доменов?

19. Как в выбранной вами СУБД сделать так, чтобы значение первичного ключа генерировалось автоматически при создании

новой записи?

Список литературы

1. Базы данных [Текст] : учебное пособие / Г.В. Верхова [и др.]. – СПб. : Пеолитехника, 2008. – 171 с.

2. Щелоков, С.А. Базы данных [Электронный ресурс] : учебное пособие / С.А. Щелоков. – Оренбург : Оренбургский гос. ун-т, 2014. – 298 с. // Режим доступа – http://biblioclub.ru.