АНАТОЛИЙ ХОМОНЕНКО
ВЛАДИМИР ГОФМАН

РАБОТА С БАЗАМИ ДАННЫХ В DELPHI

3-Е ИЗДАНИЕ

- ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ
- ТЕХНОЛОГИИ BDE, ADO, dbExpress И INTERBASE EXPRESS
- РАБОТА С ЛОКАЛЬНЫМИ И УДАЛЕННЫМИ БАЗАМИ ДАННЫХ
- ПРОГРАММИРОВАНИЕ НА SQL

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

УДК 681.3.06
ББК 32.973.26-18.2
Х76

Хомоненко А. Д., Гофман В. Э.

Х76 Работа с базами данных в Delphi. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2005. — 640 с.: ил.

ISBN 5-94157-361-8

Рассматривается использование средств Delphi 7 для разработки приложений баз данных. Даются понятия баз данных, характеризуются элементы и описываются этапы проектирования реляционных баз данных, изложена технология разработки информационных систем, освещаются приемы работы с данными, создание таблиц и приложений баз данных, подготовка отчетов. Рассматриваются навигационный и реляционный способы доступа к данным с помощью технологий BDE, ADO, dbExpress и Interbase Express, основы программирования на SQL. Показывается использование локальных и удаленных баз данных, включая создание многоуровневых информационных систем и публикацию баз данных в Интернете. Также рассматриваются возможности Delphi 2005 по работе с базами данных. Благодаря подробному изложению тем и большому числу примеров книга может служить практическим руководством по работе с базами данных.

Для программистов

УДК 681.3.06
ББК 32.973.26-18.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

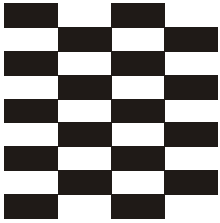
Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.02.05.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 51,6.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04
от 11.11.2004 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-361-8

Оглавление



Глава 2

Реляционные базы данных и средства работы с ними

Большинство современных баз данных для персональных компьютеров являются реляционными. Достоинства реляционной модели организации данных — простота, гибкость структуры, удобство реализации на компьютере, наличие теоретического описания.

Элементы реляционной базы данных

Реляционная база данных (БД) состоит из взаимосвязанных таблиц. Каждая таблица содержит информацию об объектах одного типа, а совокупность всех таблиц образует единую БД.

Таблицы баз данных

Таблицы, образующие БД, находятся в каталоге (папке) на жестком диске. Таблицы хранятся в файлах и похожи на отдельные документы или электронные таблицы (например, табличного процессора Microsoft Excel), их можно перемещать и копировать обычным способом, скажем, с помощью Проводника Windows. Однако в отличие от документов, таблицы БД поддерживают *многопользовательский* режим доступа, это означает, что их могут одновременно использовать несколько приложений.

Для одной таблицы создается несколько файлов, содержащих данные, индексы, ключи и т. п. Главным из них является файл с данными, имя этого файла совпадает с именем таблицы, которое задается при ее создании. В некотором смысле понятия таблицы и ее главного файла являются синонимами, при выборе таблицы выбирается именно ее главный файл: для таблицы dBase это файл с расширением dbf, а для таблицы Paradox — файл с расширением db. Имена остальных файлов таблицы назначаются автоматически — все файлы имеют одинаковые имена, совпадающие с именами таблиц, и разные расширения, указывающие на содержимое соответствующего файла. Расширения файлов приведены ниже в данной главе *в разд. "Таблицы форматов dBase и Paradox"*.

Каждая таблица БД состоит из строк и столбцов и предназначена для хранения данных об однотипных объектах информационной системы. Строка таблицы называется *записью*, столбец таблицы — *полем* (рис. 2.1). Каждое поле должно иметь уникальное в пределах таблицы имя.

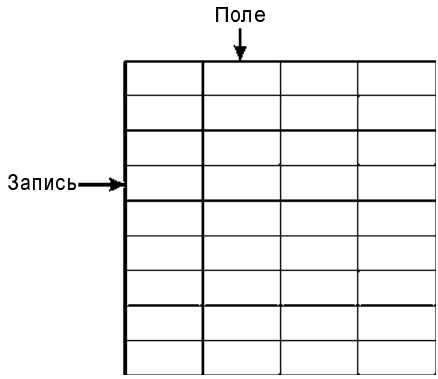


Рис. 2.1. Схема таблицы базы данных

Поле содержит данные одного из допустимых типов, например, строкового, целочисленного или типа "дата". При вводе значения в поле таблицы автоматически производится проверка соответствия типа значения и типа поля. В случае, когда эти типы не совпадают, а преобразование типа значения невозможно, генерируется исключение.

Особенности организации таблиц зависят от конкретной СУБД, используемой для создания и ведения БД. Например, в локальной таблице dBase и в таблице сервера InterBase нет поля автоинкрементного типа (с автоматически наращиваемым значением), а в таблице dBase нельзя определить ключ. Подобные особенности необходимо учитывать при выборе типа (формата) таблицы, т. к. они влияют не только на организацию БД, но и на построение приложения для работы с этой БД. Однако, несмотря на все различия таблиц, существуют общие правила создания и ведения БД, а также разработки приложений, которые и будут далее рассмотрены.

Замечание

В зависимости от типа таблиц и системы разработки приложений может использоваться различная терминология. Например, в InterBase поле таблицы называется столбцом.

Основу таблицы составляет описание ее полей, каждая таблица должна иметь хотя бы одно поле. Понятие структуры таблицы является более широким и включает:

- описание полей;
- ключ;

- индексы;
- ограничения на значения полей;
- ограничения ссылочной целостности между таблицами;
- пароли.

Иногда ограничения на значения полей, ограничения ссылочной целостности между таблицами, а также права доступа называют одним общим термином "ограничения".

Отметим, что отдельные элементы структуры зависят от формата таблиц, например, для таблиц dBase нельзя задать ограничения ссылочной целостности (т. к. у них нет ключей). Все элементы структуры задаются на физическом уровне (уровне таблицы) и действуют для всех программ, выполняющих операции с БД, включая средства разработки и ведения БД (например, программу Database Desktop). Многие из этих элементов (например, ограничения на значения полей или поля просмотра) можно также реализовать в приложении программно, однако в этом случае они действуют только в пределах своего приложения.

С таблицей в целом можно выполнять следующие операции:

- создание (определение структуры);
- изменение структуры (реструктуризация);
- переименование;
- удаление.

При *создании* таблицы задаются структура и имя таблицы. При сохранении на диске создаются все необходимые файлы, относящиеся к таблице. Их имена совпадают с именем таблицы.

При *изменении структуры* таблицы в ней могут измениться имена и характеристики полей, состав и наименования ключа и индексов, ограничения. Однако имена таблицы и ее файлов остаются прежними.

При *переименовании* таблица получает новое имя, в результате чего новое имя также получают все ее файлы. Для этого используются соответствующие программы (утилиты), предназначенные для работы с таблицами БД, например, Database Desktop или Data Pump.

Замечание

Таблицу нельзя переименовать, просто изменив названия всех ее файлов, например, с помощью Проводника Windows.

При *удалении* таблицы с диска удаляются все ее файлы. В отличие от переименования, удаление таблицы можно выполнить посредством любой программы (в том числе и с помощью Проводника Windows).

Ключи и индексы

Ключ представляет собой комбинацию полей, данные в которых однозначно определяют каждую запись в таблице. Простой ключ состоит из одного поля, а составной (сложный) — из нескольких полей. Поля, по которым построен ключ, называют *ключевыми*. В таблице может быть определен только один ключ. Ключ обеспечивает:

- однозначную идентификацию записей таблицы;
- ускорение выполнения запросов к БД;
- установление связи между отдельными таблицами БД;
- использование ограничений ссылочной целостности.

Ключ также называют *первичным ключом* или *первичным (главным) индексом*.

Информация о ключе может храниться в отдельном файле или совместно с данными таблицы. Например, в БД Paradox для этой цели используется отдельный файл (ключевой файл или файл главного индекса) с расширением px. В БД Access вся информация содержится в одном общем файле с расширением mdb. Значения ключа располагаются в определенном порядке. Для каждого значения ключа имеется уникальная ссылка, указывающая на расположение соответствующей записи в таблице (в главном ее файле). Поэтому при поиске записи выполняется не последовательный просмотр всей таблицы, а прямой доступ к записи на основании упорядоченных значений ключа.

Ценой, которую разработчик и пользователь платят за использование такой технологии, является увеличение размера БД вследствие необходимости хранения значений ключа, например, в отдельном файле. Размер этого файла зависит не только от числа записей таблицы (что достаточно очевидно), но и от полей, составляющих ключ. В ключевом файле, кроме ссылок на соответствующие записи таблицы, сохраняются и значения самих ключевых полей. Поэтому при вхождении в состав ключа длинных строковых полей размер ключевого файла может оказаться соизмеримым с размером файла с данными таблицы.

Таблицы различных форматов имеют свои особенности построения ключей. Вместе с тем существуют и общие правила.

- Ключ должен быть уникальным. У составного ключа значения отдельных полей (но не всех одновременно) могут повторяться.
- Ключ должен быть достаточным и не избыточным, т. е. не содержать поля, которые можно удалить без нарушения уникальности ключа.
- В состав ключа не могут входить поля некоторых типов, например, графическое поле или поле комментария.

Выбор ключевых полей не всегда является простой и очевидной задачей, особенно для таблиц с большим количеством полей. Нежелательно выбирать в качестве ключевых поля, содержащие фамилии людей в таблице сотрудников организации или названия товаров в таблице данных склада. В этом случае высока

вероятность существования двух и более однофамильцев, а также товаров с одинаковыми названиями, которые различаются, к примеру, цветом (значение другого поля). Для указанных таблиц можно использовать, например, поле кода сотрудника и поле артикула товара. При этом предполагается, что указанные значения являются уникальными.

Удобным вариантом создания ключа будет использование для него поля соответствующего типа, которое автоматически обеспечивает поддержку уникальности значений. Для таблиц Paradox таким является поле автоинкрементного типа, еще одним достоинством которого является небольшой размер (4 байта). В то же время в таблицах dBase и InterBase поле подобного типа отсутствует, и программист должен обеспечивать уникальность значений ключа самостоятельно, например, используя специальные генераторы.

Отметим, что при создании и ведении БД правильным подходом считается задание в каждой таблице ключа даже в случае, если на первый взгляд он не нужен. В примерах таблиц, которые приводятся при изложении материала, как правило, ключ создается, и для него вводится специальное автоинкрементное поле с именем `Code` или `Number`.

Индекс, как и ключ, строится по полям таблицы, однако он может допускать повторение значений составляющих его полей — в этом и состоит его основное отличие от ключа. Поля, по которым построен индекс, называют *индексными*. Простой индекс состоит из одного поля, а составной (сложный) — из нескольких полей.

Индексы при их создании именуются. Как и в случае с ключом, в зависимости от СУБД индексы могут храниться в отдельных файлах или совместно с данными. Создание индекса называют *индексированием таблицы*.

Использование индекса обеспечивает:

- увеличение скорости доступа к данным (поиска);
- сортировку записей;
- установление связи между отдельными таблицами БД;
- использование ограничений ссылочной целостности.

В двух последних случаях индекс применяется совместно с ключом второй таблицы.

Как и ключ, индекс представляет собой своеобразное оглавление таблицы, просмотр которого выполняется перед обращением к ее записям. Таким образом, использование индекса повышает *скорость доступа* к данным в таблице за счет того, что доступ выполняется не последовательным, а индексно-последовательным методом.

Сортировка представляет собой упорядочивание записей по полю или группе полей в порядке возрастания или убывания их значений. Можно сказать, что индекс служит для сортировки таблиц по индексным полям. В частности, в Delphi записи набора `Table` можно сортировать только по индексным полям.

Набор данных `Query` позволяет выполнить средствами SQL сортировку по любым полям, однако и в этом случае для индексированных полей упорядочивание записей выполняется быстрее.

Для одной таблицы можно создать несколько индексов. В каждый момент времени один из них можно сделать текущим, т. е. активным. Даже при существовании нескольких индексов таблица может не иметь текущего индекса (текущий индекс важен, например, при выполнении поиска и сортировки записей набора данных `Table`).

Ключевые поля обычно автоматически индексируются. В таблицах Paradox ключ также является главным (первичным) индексом, который не именуется. Для таблиц dBase ключ не создается, и его роль выполняет один из индексов.

Замечание

Создание ключа может привести к побочным эффектам. Так, если в таблице Paradox определить ключ, то записи автоматически упорядочиваются по его значениям, что в ряде случаев является нежелательным.

Таким образом, использование ключей и индексов позволяет:

- однозначно идентифицировать записи;
- избегать дублирования значений в ключевых полях;
- выполнять сортировку таблиц;
- ускорять операции поиска в таблицах;
- устанавливать связи между отдельными таблицами БД;
- использовать ограничения ссылочной целостности.

Одной из основных задач БД является обеспечение *быстрого доступа к данным* (поиска данных). Время доступа к данным в значительной степени зависит от используемых для поиска данных методов и способов.

Методы и способы доступа к данным

Выделяют следующие методы доступа к данным таблиц:

- последовательный;
- прямой;
- индексно-последовательный.

При *последовательном* методе выполняется последовательный просмотр всех записей таблицы и поиск нужных из них. Этот метод доступа является крайне неэффективным и приводит к значительным временным затратам на поиск, прямо пропорциональным размеру таблицы (числу ее записей). Поэтому его рекомендуется использовать только для относительно небольших таблиц.

При *прямом* доступе нужная запись выбирается в таблице на основании ключа или индекса. При этом просмотр других записей не выполняется. Напомним, что значения ключей и индексов располагаются в упорядоченном виде и содержат ссылку, указывающую на расположение соответствующей записи в таблице. При поиске записи выполняется не последовательный просмотр всей таблицы, а непосредственный доступ к записи на основании ссылки.

Индексно-последовательный метод доступа включает в себя элементы последовательного и прямого методов доступа и используется при поиске группы записей. Этот метод реализуется только при наличии индекса, построенного по полям, значения которых должны быть найдены. Суть его заключается в том, что находится индекс первой записи, удовлетворяющей заданным условиям, и соответствующая запись выбирается из таблицы на основании ссылки. Это является прямым доступом к данным. После обработки первой найденной записи осуществляется переход к следующему значению индекса, и в таблице выбирается запись, соответствующая значению этого индекса. Так последовательно перебираются индексы всех записей, удовлетворяющих заданным условиям, что является последовательным доступом.

Достоинством прямого и индексно-последовательного методов является максимальная возможная скорость доступа к данным, плата за которую — расход памяти на хранение информации о ключах и индексах.

Указанные методы доступа реализуются СУБД и не требуют специального программирования. Задачей разработчика является определение соответствующей структуры БД, в данном случае — определение ключей и индексов. Так, если для поля создан индекс, то при поиске записей по этому полю автоматически используется индексно-последовательный метод доступа, в противном случае — последовательный метод.

Замечание

При создании составного индекса важен порядок составляющих его полей. Например, если индекс создан по полям "фамилия", "номер отдела" и "дата рождения", а поиск ведется одновременно по полям "фамилия", "дата рождения" и "номер отдела", то такой индекс использован не будет. В результате доступ к таблице осуществляется последовательным методом. В подобной ситуации (для таблицы с большим числом записей) разработчик должен создать также индекс, построенный по полям "фамилия", "дата рождения" и "номер отдела".

При выполнении операций с таблицами используется один из следующих *способов доступа к данным*:

- навигационный;
- реляционный.

Навигационный способ доступа заключается в обработке каждой отдельной записи таблицы. Этот способ обычно используется в локальных БД или в удаленных БД небольшого размера. Если необходимо обработать несколько записей, то все они обрабатываются поочередно.

Реляционный способ доступа основан на обработке сразу *группы записей*, при этом если необходимо обработать одну запись, то обрабатывается группа, состоящая из одной записи. Так как реляционный способ доступа основывается на SQL-запросах, его также называют *SQL-ориентированным*. Этот способ доступа ориентирован на выполнение операций с удаленными БД и является предпочтительным при работе с ними, хотя его можно использовать и для локальных БД.

Способ доступа к данным выбирается программистом и зависит от средств доступа к БД, используемых при разработке приложения. Например, в приложениях, создаваемых в Delphi, реализацию навигационного способа доступа можно осуществить посредством компонентов `Table` или `Query`, а реляционного — с помощью компонента `Query`.

Таким образом, методы доступа к данным определяются структурой БД, а способы доступа — приложением.

Связь между таблицами

В частном случае БД может состоять из одной таблицы, содержащей, например, дни рождения сотрудников организации. Однако обычно реляционная БД состоит из набора взаимосвязанных таблиц. Организация связи (отношений) между таблицами называется *связыванием* или *соединением таблиц*.

Связи между таблицами можно устанавливать как при создании БД, так и при выполнении приложения, используя средства, предоставляемые СУБД. Связывать можно две или несколько таблиц. В реляционной БД, кроме связанных таблиц, могут быть и отдельные таблицы, не соединенные ни с одной другой таблицей. Это не меняет сути реляционной БД, которая содержит единую информацию об информационной системе, связанную не в буквальном смысле (связь между таблицами), а в функциональном смысле — вся информация относится к одной системе.

Для связывания таблиц используются *поля связи* (иногда применяется термин "совпадающие поля"). Поля связи обязательно должны быть индексированными. В подчиненной таблице для связи с главной таблицей задается индекс, который также называется *внешним ключом*. Состав полей этого индекса должен полностью или частично совпадать с составом полей индекса главной таблицы.

Особенности использования индексов зависят от формата связываемых таблиц. Так, для таблиц dBase индексы строятся по одному полю и нет деления на ключ (главный или первичный индекс) и индексы. Для организации связи в главной и подчиненной таблицах выбираются индексы, составленные по полям совпадающего типа, например, целочисленного.

Для таблиц Paradox в качестве полей связи главной таблицы должны использоваться поля ключа, а для подчиненной таблицы — поля индекса. Кроме того, в подчиненной таблице обязательно должен быть определен ключ. На рис. 2.2 показана схема связи между таблицами БД Paradox.

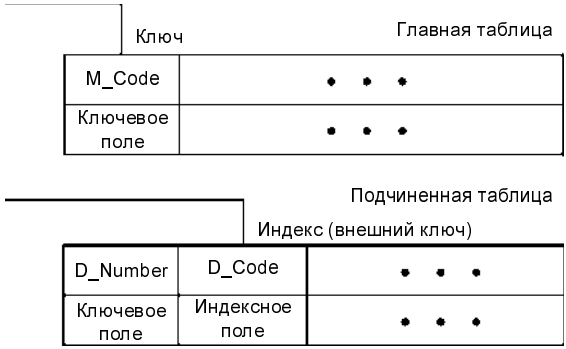


Рис. 2.2. Схема связи между таблицами базы данных Paradox

В главной таблице определен ключ, построенный по полю `M_Code` автоинкрементного типа. В подчиненной таблице определен ключ по полю `D_Number` также автоинкрементного типа и индекс, построенный по полю `D_Code` целочисленного типа. Связь между таблицами устанавливается по полям `D_Code` и `M_Code`. Индекс по полю `D_Code` является внешним ключом. В названия полей включены префиксы, указывающие на принадлежность поля соответствующей таблице. Так, названия полей главной таблицы начинаются с буквы `M` (Master), а названия полей подчиненной таблицы начинаются с буквы `D` (Detail). Подобное именование полей облегчает ориентацию в их названиях, особенно при большом количестве таблиц.

Замечание

Как отмечалось, поля связи должны быть индексированными, хотя, строго говоря, это требование не всегда является обязательным. При доступе к данным средствами языка SQL можно связать (соединить) между собой таблицы и по неиндексированным полям. Однако в этом случае скорость выполнения операций будет низкой.

Связь между таблицами определяет отношение подчиненности, при котором одна таблица является *главной* (родительской, или мастером — Master), а вторая — *подчиненной* (дочерней, или детальной — Detail). Саму связь (отношение) называют связь "главный-подчиненный", "родительский-дочерний" или "мастер-детальный". Существуют следующие виды связи:

- отношение "один-к-одному";
- отношение "один-ко-многим";
- отношение "много-к-одному";
- отношение "много-ко-многим".

Отношение *"один-к-одному"* означает, что одной записи в главной таблице соответствует одна запись в подчиненной таблице. При этом возможны два варианта:

- для каждой записи главной таблицы есть запись в подчиненной таблице;
- в подчиненной таблице может не быть записей, соответствующих записям главной таблицы.