

ДМИТРИЙ ОСИПОВ



Базы данных и Delphi

Теория и практика

+ ПРОБНЫЕ
ВЕРСИИ **PRO**



ВВЕДЕНИЕ В ТЕОРИЮ
РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

ПРОЕКТИРОВАНИЕ
ЛОКАЛЬНЫХ
И КЛИЕНТ-СЕРВЕРНЫХ
БАЗ ДАННЫХ

СТРУКТУРИРОВАННЫЙ ЯЗЫК
ЗАПРОСОВ SQL

РАСШИРЯЕМЫЙ ЯЗЫК
РАЗМЕТКИ XML

ТЕХНОЛОГИИ DBEXPRESS,
INTERBASE EXPRESS И ADO

МНОГОУРОВНЕВЫЕ БАЗЫ
ДАННЫХ НА ОСНОВЕ
DATASNAP

НЕСТАНДАРТНЫЕ ПРИЕМЫ
РАЗРАБОТКИ ПРИЛОЖЕНИЙ
БАЗ ДАННЫХ

PRO
ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

УДК 681.3.06
ББК 32.973.26-018.2
О-74

Осипов Д. Л.

О-74 Базы данных и Delphi. Теория и практика. — СПб.: БХВ-Петербург, 2011. — 752 с.: ил. + DVD — (Профессиональное программирование)
ISBN 978-5-9775-0659-5

Книга основана на материалах лекций и практических занятий, разработанных автором, и объединяет теоретические основы и практические аспекты разработки реляционных баз данных. В первой части рассмотрена концепция реляционных баз данных: реляционная модель данных, жизненный цикл информационной системы, концептуальное и логическое моделирование БД, нормализация отношений, обеспечение многопользовательского доступа к данным, вопросы обеспечения безопасности БД, языки SQL и XML и др. Во второй части описаны возможности современных версий Delphi в области разработки приложений баз данных: подробное описание технологий dbExpress, Interbase Express и ADO, особенности использования компонентов управления данными визуальной библиотеки Delphi, механизм разработки многоуровневых приложений на основе технологии DataSnap, порядок создания отчетов для печати и многое другое. На DVD размещены дополнительные главы, а также материалы и пробные версии ПО компании Embarcadero, включая Delphi XE.

Для студентов и программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Леонид Кочин</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.01.11.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 60,63.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09 от 26.05.2009 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

Оглавление

ВВЕДЕНИЕ	1
ЧАСТЬ I. ВВЕДЕНИЕ В РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ	5
Глава 1. СИСТЕМЫ, ОСНОВАННЫЕ НА ФАЙЛАХ	7
Принцип построения систем, основанных на файлах.....	9
Недостатки систем, основанных на файлах	10
Пути устранения недостатков систем, основанных на файлах	13
Резюме	14
Глава 2. ЭВОЛЮЦИЯ МОДЕЛЕЙ РЕАЛИЗАЦИИ ДАННЫХ	15
Необходимость моделирования	17
Иерархическая модель	18
Сетевая модель.....	20
Попытки разработки стандарта БД.....	21
Реляционная модель	23
Объектно-ориентированная модель.....	25
Резюме	26
Глава 3. ФУНКЦИИ И КОМПОНЕНТЫ СУБД	27
Функциональные обязанности СУБД.....	27
Компоненты СУБД.....	29
Архитектурные решения доступа к БД	32
Файл-сервер.....	32
Клиент-сервер	34
Многоуровневые решения	36
Резюме	37
Глава 4. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ.....	38
Сущность и атрибуты.....	39
Тип данных и домен	41

Связь	43
Реляционная таблица.....	45
Ключ	47
Целостность данных.....	48
Целостность доменов	48
Целостность сущностей	49
Ссылочная целостность.....	50
Корпоративная целостность	50
Реляционная алгебра	51
Резюме	56
ГЛАВА 5. МОДЕЛЬ "СУЩНОСТЬ-СВЯЗЬ"	57
Сущности и атрибуты в ER-модели.....	58
Подтипы сущностей	61
Связи в ER-модели	63
Сильные и слабые связи.....	66
Рекурсивная связь	67
Связи высокого порядка	68
Вариации ER-моделей.....	70
Резюме	72
ГЛАВА 6. НОРМАЛИЗАЦИЯ	73
Первая нормальная форма	76
Функциональная зависимость атрибутов	79
Порядок определения первичного ключа	81
Вторая нормальная форма	82
Третья нормальная форма.....	84
Нормальная форма Бойса—Кодда	86
Четвертая нормальная форма	87
Пятая нормальная форма	88
Резюме	90
ГЛАВА 7. ИНДЕКСИРОВАНИЕ	91
Индексы на основе хеширования.....	93
Хеш-функции	95
Хеширование текстовых данных	96
Борьба с коллизиями	96
Индексы на основе В-деревьев	98
Битовые индексы	103
Правила назначения вторичных индексов	103
Резюме	104

ГЛАВА 8. ТРАНЗАКЦИИ И ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ.....	105
Понятие транзакции	106
Проблемы параллельного доступа к данным.....	108
Потерянные обновления	108
Неактуальные чтения ("грязное чтение")	109
Несогласованная обработка.....	110
Чтение строк-фантомов.....	111
Управление параллельными транзакциями	112
Метод блокировок	113
Метод временных меток	115
Метод двухфазной блокировки	116
Оптимистический подход	116
Уровни изоляции SQL-транзакций	117
Резюме	118
ГЛАВА 9. ПРОЕКТИРОВАНИЕ БД.....	119
Понятие информационной системы	119
Этапы жизненного цикла базы данных	122
Планирование разработки БД.....	123
Определение и анализ требований к системе	123
Проектирование БД.....	127
Выбор СУБД	130
Создание прикладного программного обеспечения.....	131
Тестирование.....	132
Реализация	133
Эксплуатация и сопровождение	133
Резюме	134
ГЛАВА 10. ЗАЩИТА БД	135
Откуда исходят угрозы.....	136
Правила защиты БД.....	139
Аутентификация и авторизация	140
Криптографическая защита	141
Резервное копирование	143
Аудит событий безопасности	144
Модернизация системного и прикладного ПО	145
Доступ к данным только при посредничестве представлений и хранимых процедур	145
Резюме	146

ГЛАВА 11. ЗНАКОМСТВО С SQL	147
Назначение SQL.....	149
Типы данных SQL	150
Предопределенные типы данных	152
Непредопределенные типы данных	157
Массив.....	158
Мультимножество.....	158
Пользовательский тип	159
Другие типы.....	160
Определение констант	160
Преобразование данных.....	161
Операторы	162
Встроенные функции	163
Резюме	164
ГЛАВА 12. ПОСТРОЕНИЕ ЗАПРОСОВ	165
Порядок сортировки — <i>ORDER BY</i>	167
Условие отбора данных — <i>WHERE</i>	167
Сравнение	168
Попадание в диапазон — <i>BETWEEN</i>	169
Соответствие шаблону — <i>LIKE</i>	169
Проверка неопределенного значения — <i>IS NULL</i>	170
Принадлежность множеству — <i>IN, ALL, ANY, SOME</i>	170
Предикат существования — <i>EXISTS</i>	171
Многотабличные запросы.....	171
Слияние <i>UNION</i>	173
Объединение <i>ON</i>	173
Объединение <i>USING</i>	176
Агрегирующие функции	176
Группировка данных — <i>GROUP BY</i>	177
Дополнительная фильтрация группы строк — <i>HAVING</i>	178
Оконные функции.....	178
Рекурсивные запросы.....	182
Резюме	184
ГЛАВА 13. МАНИПУЛИРОВАНИЕ ДАННЫМИ И УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ	185
Язык манипулирования данными DML	185
Вставка, инструкция <i>INSERT</i>	185
Редактирование, инструкция <i>UPDATE</i>	187
Удаление, инструкция <i>DELETE</i>	188
Слияние данных, инструкция <i>MERGE</i>	190

Транзакции	191
Диагностирование ошибок в работе транзакции	194
Настройка уровня изоляции	196
Резюме	197
ГЛАВА 14. ОПРЕДЕЛЕНИЕ ДАННЫХ В SQL	198
Базы данных (схемы)	198
Таблицы	200
Индексы	205
Домены	208
Представления (виртуальные таблицы)	209
Хранимые процедуры	210
Триггеры	212
Курсоры	217
Управление доступом к данным	220
Управление наборами привилегий	221
Предоставление привилегий	222
Лишение привилегий	224
Резюме	225
ГЛАВА 15. ОСНОВЫ XML	226
Правильность и допустимость документа	229
Построение простейшего документа XML	229
Элементы	230
Специальные символы	232
Атрибуты	232
Пространство имен	233
Определение документа	236
DTD	236
Хранение DTD во внешнем файле	240
Резюме	242
ГЛАВА 16. XML SCHEMAS	243
Определение элемента <i><element></i>	246
Тип данных	248
Производные типы <i><simpleType></i>	248
Глобальное и локальное объявление	252
Квалифицирование элемента	252
Ограничения на число элементов	253
Значение по умолчанию и фиксированное значение	253
Создание сложных структур <i><complexType></i>	254

Определение атрибута <code><attribute></code>	256
Подключение XML-схемы к документу	257
Пример схемы <code>computers.xsd</code>	258
Пример документа <code>computers.xml</code>	261
Резюме	262
ЧАСТЬ II. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЙ БД В DELPHI	263
Глава 17. Концепция приложения БД DELPHI	265
Пример простейшего приложения БД	265
Архитектура приложения БД	269
Общая характеристика компонентов соединения с БД	271
Общая характеристика компонентов наборов данных	271
Источник данных	273
Общая характеристика компонентов управления данными	274
Резюме	275
Глава 18. Универсальный набор данных <i>TDataSet</i>	276
Функционал набора данных	278
Открытие и закрытие набора данных	278
Обновление набора данных	279
Перемещение по набору данных	280
Создание закладок и переход к закладке	282
Состояние набора данных	284
Редактирование записей в наборе	286
Организация доступа к отдельному полю	288
Вычисляемые поля	290
Агрегатное поле	290
Фильтрация набора данных	291
Организация поиска данных	292
Обработка событий	294
Кэширование данных	295
Взаимодействие с элементами управления	296
Резюме	297
Глава 19. Общая характеристика полей набора данных	298
Поле таблицы — класс <i>TField</i>	298
Классификация полей по функциональному назначению	301
Классификация полей по типу хранимых данных	301

Обращение к полю.....	303
Доступ к данным поля.....	304
Низкоуровневый доступ к данным.....	306
Преобразование типа данных.....	307
Размер поля.....	308
Значение по умолчанию.....	309
Ограничения на ввод данных.....	310
Маска ввода.....	311
Индексные поля.....	312
Отображение данных.....	312
Обработка событий.....	314
Поля подстановки.....	315
Вычисляемые поля.....	322
Числовые поля, класс <i>TNumericField</i>	324
Поля целых чисел <i>TLargeintField</i> , <i>TIntegerField</i> , <i>TSmallintField</i> , <i>TWordField</i> , <i>TAutoIncField</i>	326
Поля вещественных чисел <i>TFloatField</i> и <i>TCurrencyField</i>	326
Бинарно-кодированные десятичные поля <i>TBCDField</i> и <i>TFMTBCDField</i>	327
Текстовые поля, <i>TStringField</i>	328
Поле глобального идентификатора, <i>TGuidField</i>	329
Логическое поле, <i>TBooleanField</i>	330
Бинарные поля <i>TBinaryField</i> , <i>TBytesField</i> и <i>TVarBytesField</i>	330
Дата и время, поля <i>TDateTimeField</i> , <i>TDateField</i> и <i>TTimeField</i>	330
Дата и время, поле <i>TSQLTimeStampField</i>	331
Поля больших бинарных объектов, <i>TBlobField</i> , <i>TGraphicField</i> , <i>TMemoField</i> и <i>TWildMemoField</i>	331
Композитные поля, <i>TObjectField</i>	338
Резюме.....	340
ГЛАВА 20. ВСПОМОГАТЕЛЬНЫЕ КЛАССЫ НАБОРА ДАННЫХ.....	341
Коллекция <i>TParams</i> и динамический SQL.....	341
Параметр <i>TParam</i>	344
Описание структуры таблицы и ее индексов.....	347
Коллекция полей структуры таблицы <i>TFieldDefs</i>	349
Определение поля, класс <i>TFieldDef</i>	351
Коллекция структуры индексов таблицы, класс <i>TIndexDefs</i>	352
Определение индекса, класс <i>TIndexDef</i>	353
Пример создания таблицы.....	354
Резюме.....	356
ГЛАВА 21. КЛИЕНТСКИЙ НАБОР ДАННЫХ <i>TCLIENTDATASET</i>.....	357
Проекты БД, основанные на файлах.....	358
Загрузка и сохранение данных.....	361

Требования к структуре XML.....	365
Хранение данных в компоненте.....	366
Управление данными.....	367
Работа с индексами.....	370
Определение диапазона.....	373
Ограничения.....	376
Применение агрегирующих функций.....	376
Агрегат, класс <i>TAggregate</i>	377
Агрегатные поля <i>TAggregateField</i>	378
Место <i>TClientDataSet</i> в многоуровневых проектах БД.....	379
Работа в клиентских приложениях DataSnap ранних версий.....	380
Сохранение данных на сервере.....	381
Отмена изменений.....	382
Применение точек сохранения.....	383
Обработка ошибок.....	383
Оптимизация пакета с данными.....	385
Обновление данных.....	386
Выполнение команд SQL.....	387
Резюме.....	387
ГЛАВА 22. ВВЕДЕНИЕ В ТЕХНОЛОГИЮ ADO.....	388
Взаимодействие ADO и компонентов VCL.....	389
Строка соединения ADO.....	392
Соединение с хранилищем данных, компонент <i>TADOConnection</i>	393
Установка соединения.....	393
Пример соединения без регистрации пользователя.....	397
Регистрация пользователя.....	402
Разрыв соединения.....	408
Информирование о соединении.....	408
Отправка команд.....	410
Управление подчиненными компонентами dbGo.....	413
Транзакции.....	413
Кэширование.....	415
Сервисные методы модуля ADODB.....	416
Резюме.....	418
ГЛАВА 23. НАБОРЫ ДАННЫХ ADO.....	419
Командный объект — <i>TADOCommand</i>	419
Коллекция ошибок <i>Errors</i> и объект ошибки <i>Error</i>	424
Набор данных ADO, компонент <i>TADODataSet</i>	425
Интерфейс множества записей <i>_Recordset</i>	426
Доступ к хранилищу данных.....	427

Выбор библиотеки курсора.....	428
Редактирование данных	429
Перемещение по множеству строк	430
Особенности применения поля BCD	432
События <i>TCustomADODataSet</i>	432
Работа с индексами.....	436
Сортировка записей	436
Поиск данных	436
Особенности изоляции транзакций.....	437
Фильтрация множества записей	438
Кэширование записей.....	438
Фильтрация записей в кэше	440
Организация отложенного обновления данных в ADO	441
Портфельный режим обработки данных.....	443
Управление данными ADO в стиле Delphi	444
Таблица <i>TADOTable</i>	445
Организация отношения "главная – подчиненная таблица".....	446
Запрос <i>TADOQuery</i>	448
Хранимая процедура <i>TADOStoredProc</i>	450
Резюме	452
ГЛАВА 24. ДОСТУП К БД INTERBASE	453
Доступ к базе данных, компонент <i>TIBDatabase</i>	454
Выбор диалекта ISQL.....	455
Создание и уничтожение базы данных.....	456
Соединение с базой данных.....	457
Регистрация пользователя.....	459
Разрыв соединения	460
Информирование о составе БД	462
Управление транзакциями	463
Контроль за событиями.....	464
Совместная работа с SQL монитором	464
Транзакция, компонент <i>TIBTransaction</i>	465
Управление транзакцией.....	466
Тайм-аут транзакции	468
Диагностика состояния транзакции	469
Параметры транзакции.....	469
Информация об объектах БД, компонент <i>TIBExtract</i>	471
События InterBase, компонент <i>TIBEvents</i>	473
Информация о БД, компонент <i>TIBDatabaseInfo</i>	474
Монитор SQL, <i>TSQLMonitor</i>	476
Файл инициализации БД, <i>TIBDataBaseINI</i>	476
Резюме	478

ГЛАВА 25. НАБОРЫ ДАННЫХ INTERBASE.....	479
Инструкция SQL, компонент <i>TIBSQL</i>	480
Подготовка к работе	480
Обслуживание полученного набора данных.....	482
Наборы данных InterBase, компонент <i>TIBDataSet</i>	483
Подготовка к работе	483
Обработка событий.....	485
Генератор значений	486
Особенности редактирования данных	488
Работа в режиме кэширования обновлений	488
Перемещение по записям.....	490
Фильтрация данных.....	490
Запрос, компонент <i>TIBQuery</i>	491
Редактирование данных, доступных только для чтения	492
Хранимая процедура, компонент <i>TIBStoredProc</i>	492
Таблица, компонент <i>TIBTable</i>	494
Экспорт-импорт данных	494
Модифицируемый запрос, компонент <i>TIBUpdateSQL</i>	496
Диалог фильтрации, компонент <i>TIBFilterDialog</i>	498
Резюме	500
ГЛАВА 26. АДМИНИСТРИРОВАНИЕ СЕРВЕРА INTERBASE	501
Свойства сервера, <i>TIBServerProperties</i>	505
Сервис лицензирования, <i>TIBLicensingService</i>	510
Конфигурирование сервера, <i>TIBConfigService</i>	511
Ведение журнала транзакций	514
Протокол работы сервера, <i>TIBLogService</i>	517
Статистика, <i>TIBStatisticalService</i>	518
Проверка БД, <i>TBDValidationService</i>	520
Управление учетными записями, <i>TIBSecurityService</i>	522
Резервное копирование и восстановление, <i>TIBBackupService</i> и <i>TIBRestoreService</i> ..	528
Резюме	532
ГЛАВА 27. ТЕХНОЛОГИЯ ДОСТУПА К ДАННЫМ DBEXPRESS	533
Соединение с сервером БД, компонент <i>TSQLConnection</i>	534
Настройка компонента	535
Управление соединением.....	537
Создание БД.....	539
Создание подключения в Data Explorer.....	539
Пример подключения	542
Управление подчиненными наборами данных.....	547

Управление транзакциями	547
Выполнение SQL-инструкций	548
Ограничение числа выполняющихся инструкций	549
Информирование о БД	549
Аутентификация пользователя в DataSnap	551
Мониторинг работы программы, <i>TSQLMonitor</i>	551
Резюме	553
ГЛАВА 28. НАБОРЫ ДАННЫХ DBEXPRESS.....	554
Базовый класс <i>TCustomSQLDataSet</i>	555
Формирование инструкций SQL	556
Получение системной информации	557
Набор данных dbExpress, компонент <i>TSQLDataSet</i>	561
Особенности обслуживания BLOB-полей	563
Таблица <i>TSQLTable</i>	563
Запрос <i>TSQLQuery</i>	565
Хранимая процедура <i>TSQLStoredProc</i>	565
Простой набор данных <i>TSimpleDataSet</i>	567
Резюме	569
ГЛАВА 29. ИНТЕРФЕЙС ПРИЛОЖЕНИЯ И КОМПОНЕНТЫ DATA ACCESS	570
Источник данных — компонент <i>TDataSource</i>	570
Общие черты компонентов отображения данных	572
Сетка базы данных — компонент <i>TDBGrid</i>	573
Одновременный выбор нескольких строк	575
Колонки сетки	576
Коллекция колонок — класс <i>TDBGridColumns</i>	576
Колонка — класс <i>TColumn</i>	578
Обработка событий	582
События прорисовки данных	583
Статический текст — компонент <i>TDBText</i>	585
Строка ввода — компонент <i>TDBEdit</i>	586
Многострочный редактор — <i>TDBMemo</i>	587
Редактор расширенного формата — <i>TDBRichEdit</i>	588
Изображение — компонент <i>TDBImage</i>	588
Список — <i>TDBListBox</i>	589
Комбинированный список — <i>TDBComboBox</i>	590
Группа переключателей — <i>TDBRadioGroup</i>	590
Флажок — <i>TDBCheckBox</i>	591
Компонент — <i>TDBCtrlGrid</i>	591
Поля подстановки	594
Список подстановки — <i>TDBLookupListBox</i>	595
Комбинированный список подстановки — <i>TDBLookupComboBox</i>	596

Навигатор — <i>TDBNavigator</i>	596
Резюме	598
ГЛАВА 30. НЕСТАНДАРТНЫЕ РЕШЕНИЯ ДЛЯ СТАНДАРТНЫХ КОМПОНЕНТОВ.....	599
Компоненты-списки	599
Компонент <i>TListView</i>	604
Сетка, компонент <i>TStringGrid</i>	607
Иерархические данные.....	610
Многотабличное представление иерархических данных	610
Рекурсивная связь	611
Инициализация проекта	614
Новая запись	615
Сбор данных	617
Очистка данных.....	618
Редактирование записи.....	619
Удаление записи.....	620
Сортировка узлов	621
Переподчинение узлов	623
Резюме	628
ГЛАВА 31. МНОГОУРОВНЕВЫЕ БД НА ОСНОВЕ DATASNAP	629
Архитектура трехзвенного проекта БД DataSnap.....	630
Сервер <i>TDSServer</i>	633
Класс сервера <i>TDSServerClass</i>	637
Обмен данными между клиентом и сервером, компоненты <i>TDSTCPSTransport</i> и <i>TDSHTTPService</i>	638
Аутентификация, <i>TDSHTTPServiceAuthenticationManager</i>	641
Метод сервера <i>TSQLServerMethod</i>	641
Пример проекта DataSnap	642
Регистрация службы	648
Подготовка клиентского приложения	649
Подключение сервера приложений к БД	651
Получение данных клиентским приложением	652
Реализация на сервере метода вставки новой записи	654
Доступ к методу вставки записи на стороне клиента.....	655
Архитектура DataSnap, совместимая со старыми клиентскими приложениями.....	655
Интерфейс <i>IAPPServer</i>	656
Провайдер набора данных, компонент <i>TDataSetProvider</i>	658
Подключение к провайдеру набора данных, компонент <i>TDSProviderConnection</i> ..	664
Клиентское приложение БД на основе <i>IAppServer</i>	665
Механизм обратного вызова	666
Резюме	669

ГЛАВА 32. УПРАВЛЕНИЕ СЛУЖБОЙ СЕРВЕРА ПРИЛОЖЕНИЙ DATASNAP	670
Менеджер управления службами	671
Работа со службой	672
Пример управляющего приложения SCP	673
Доработка сервиса DataSnap	680
Создание модуля панели управления	682
Резюме	686
ГЛАВА 33. ОТЧЕТЫ RAVE REPORTS	687
Обзор компонентов Rave Reports	687
Соединение <i>TRvCustomConnection</i>	688
Проект <i>TRvProject</i>	688
Системный компонент <i>TRvSystem</i>	689
Компоненты экспорта отчета в файл	690
Пример работы с редактором Rave Reports	690
Вызов отчета из приложения	696
Резюме	697
ГЛАВА 34. РАЗРАБОТКА ДИНАМИЧЕСКИХ БИБЛИОТЕК ДЛЯ ПРОЕКТОВ БД.....	698
Общая характеристика DLL	698
Создание шаблона динамической библиотеки в Delphi	700
Экспортирование функций DLL	702
Пример простой DLL	703
Взаимодействие динамической библиотеки с проектом	704
Размещение файла DLL	705
Явная загрузка DLL	705
Неявная загрузка DLL	706
Пример DLL универсального генератора отчетов	707
Резюме	712
ЗАКЛЮЧЕНИЕ.....	713
СПИСОК ЛИТЕРАТУРЫ	715
СОДЕРЖИМОЕ DVD	717
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ	719

МАТЕРИАЛ НА ДИСКЕ

ГЕНЕРАТОР ОТЧЕТОВ СВОИМИ РУКАМИ

Технические характеристики принтера
Описание принтера в Delphi, класс TPrinter
Выбор принтера
Управление страницей документа
Формирование и отправка задания на печать
Отмена задания
Печать многострочного текста
Пример универсального отчета
Окно предварительного просмотра
Вывод отчета на печать
Резюме

ДИАГРАММЫ И ГРАФИКИ

Компонент TChart
Хранение графиков в диаграмме
Базовый класс графиков — TChartSeries
Точки графика, класс TChartValueList
Методы вставки новой точки
Удаление точек, очистка графика
Оформление графика
Взаимодействие с мышью
Метки графика — TSeriesMarks
Легенда диаграммы — TChartLegend
Координатные оси диаграммы — TChartAxis
Масштабирование
Многостраничные диаграммы
Экспорт диаграмм
Печать диаграммы
Упорядочивание графиков внутри диаграммы
Обеспечение объемного вида диаграммы
Особенности компонента TDBChart
Графики диаграммы в качестве набора данных
Резюме

ОТЧЕТЫ В MICROSOFT OFFICE

Константы Microsoft Office

Обращение к серверу автоматизации

Понятие коллекции

Текстовый процессор Microsoft Word

Приложение Word — Application

Коллекция документов Documents и документ Document

Параметры страницы, объект PageSetup

Область документа Range

Объект выделения Selection

Коллекция абзацев Paragraphs и абзац документа Paragraph

Коллекция списков Lists и список List

Форматирование списка

Коллекция списков

Коллекция разделов Sections и раздел документа Section

Коллекция таблиц Tables и таблица Table

Внедрение в документ OLE-объектов

Оформление документа

Граница области

Пример универсального генератора отчетов

Электронные таблицы Microsoft Excel

Приложение Excel — Application

Коллекция WorkBooks и книга Workbook

Листы Excel

Универсальная коллекция листов Sheets

Общие свойства листов Worksheet и Chart

Параметры страницы — объект PageSetup

Особенности листа электронной таблицы Worksheet

Коллекция диаграмм Charts и лист диаграммы Chart

Область ячеек Range

Слияние ячеек

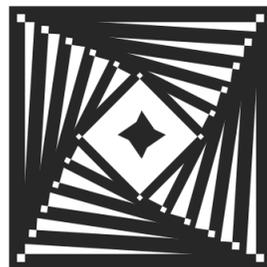
Оформление ячеек

Заливка области — Interior

Усовершенствование универсального генератора отчетов

Резюме

ГЛАВА 4



Реляционная модель данных

Создателем реляционной модели считается математик Эдгар Фрэнк Кодд (Edgar Frank Codd, 1923–2003). Однако работа талантливого ученого начиналась не с чистого листа. Научный базис теории отношений, положенной Коддом в основу реляционной модели, закладывался еще на рубеже XIX и XX веков. Авторами основ будущей реляционной теории являются два исследователя: Чарльз Содерс Пирс (1839–1914) и Эрнст Шредер (1841–1902). Несмотря на столь глубокие корни реляционной теории, первая реляционная база данных появилась на свет спустя семьдесят лет. Датой рождения реляционной БД можно считать июнь 1970 года. Именно тогда Кодд (на тот момент времени сотрудник одной из лабораторий корпорации IBM) опубликовал свою знаменитую статью "Реляционная модель данных для больших совместно используемых банков данных" (Codd E. F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6), в которой впервые прозвучал столь популярный сегодня термин "реляционная модель".

ЗАМЕЧАНИЕ

За свой вклад в науку Эдгар Кодд был удостоен премии имени Тьюринга. Это знаменательное событие произошло в 1981 году.

Первопричиной возникновения нового по тем временам подхода к проектированию баз данных послужили существенные ограничения предыдущих моделей. Ни сетевая, ни иерархическая модели не были способны просто и доступно описывать подлежащие учету данные. Кодд сумел объединить, на первый взгляд, несовместимые вещи — с одной стороны, реляционная модель опиралась на математические выкладки, а с другой стороны, была понятна рядовому пользователю, состоящему в конфронтации даже с таблицей умножения.

ЗАМЕЧАНИЕ

Модель данных представляет собой логически взаимосвязанную совокупность описаний объектов данных и операторов, управляющих этими объектами. Применяя термин "модель", мы акцентируем внимание на то, что это абстрактная конструкция. Чтобы модель превратилась в действующую базу данных, она должна быть физически реализована средствами какой-либо СУБД. Нетрудно догадаться, что реляционная база данных может получиться только из реляционной модели.

Модель данных не представляет существенного интереса для обычного пользователя, но без нее не обойтись разработчику БД. Ведь это не что иное, как чертеж будущей БД. Ни один строитель не начнет строить дом без чертежа, так и ни один программист не начнет создание физической БД без ее подробной схемы. Фундамент реляционной модели построен на понятии сущности и связей между сущностями.

Реляционной модели данных посвящено много фундаментальных трудов, в которых подробно изложены все ключевые аспекты. Обязательно рекомендую читателю хотя бы в обзорном порядке ознакомиться с работами ведущих специалистов в этой области [7, 11, 17]. Моя задача несколько прозаичнее, далее я попытаюсь сформулировать тот минимум информации, без которого понять реляционную модель будет просто невозможно.

Любая область знаний, будь то математика, радиоэлектроника, физика, химия или информатика по мере своего развития формирует свой набор терминов и определений. Без этого никак не обойтись, ведь в своих ученых беседах специалисты оперируют понятиями, не имеющими аналогов в окружающем мире. Зачастую размерность специфичных понятий разрастается до такой степени, что существенная часть времени отводимого на изучение интересующего нас предмета затрачивается только на формирование словарного запаса. Это утверждение подтвердят студенты, выбравшие в качестве своей профессии медицинскую стезю. Вы никогда не присутствовали при общении двух эскулапов? Даже если они говорят о банальном насморке, у стороннего наблюдателя неизбежно создается впечатление, что перед ним два иностранца с неподдающейся идентификации языковой принадлежностью.

С точки зрения сложности семантики, системы управления базами данных (в общем) и реляционная модель (в частности) не являются исключением из правил. Эта сравнительно новая область знаний практически мгновенно успела обзавестись специфичным набором слов. Многие термины перекочевали в БД из словарного запаса программистов, часть определений позаимствована из теории множеств, в тезаурусе разработчиков моделей данных даже нашлось место для понятий, обычно встречающихся в справочниках по философии.

Сущность и атрибуты

Любой более-менее подготовленный пользователь твердо знает, что базы данных предназначены для хранения и обработки информации. Если в руки такому человеку попадет база городских телефонов, то, даже не запуская программу, он сможет предположить, что в ней он обнаружит телефонные номера граждан и организаций. База данных городской библиотеки обязана содержать информацию об авторах и их бессмертных произведениях. Столкнувшись с базой данных авиакомпании с высокой степенью вероятности, можно утверждать, что в ней мы найдем информацию о расписании полетов, марках самолетов и городах, в которые они рассчитывают доставить пассажиров.

То, что кажется элементарным для стороннего наблюдателя, для профессионального разработчика БД может стать очень непростым делом. На самом первом этапе проектирования будущей БД он ломает голову над двумя ключевыми вопросами:

1. Что необходимо хранить в БД?
2. Какие связи между данными следует поддерживать в БД?

Для получения ответа на первый вопрос разработчик должен составить первоначальный перечень типов сущностей. *Тип сущности* (entity) — это класс однотипных объектов, о которых следует хранить информацию в БД. Немного забегаю вперед, проговорюсь, что на заключительной стадии проектирования типы сущности превратятся в реляционные таблицы. В 90% случаев сущность представляет собой какой-то объект, отвечающий на вопрос "Кто?" или "Что?". В базе данных библиотеки на роль сущностей претендуют такие одушевленные и неодушевленные объекты, как автор, книга, издательство, читатель. Совсем не обязательно, чтобы сущность соответствовала объектам реального мира, напротив, она может моделировать и отвлеченные понятия, допустим, выраженные в особых отношениях между сущностями. Именно в таком случае сущность ловко маскируется под глагол. Как правило, это происходит, когда учету подлежат некоторые события. Например, когда читатель получает (это и есть глагол) книгу из библиотечного фонда, логика базы данных требует сохранить информацию о том, кому, когда и какая из книг была выдана. При более подробном рассмотрении выясняется, что при построении первоначального списка мы просто не выявили такой тип сущности, как читательская библиотечная карточка, в которую и делаются все перечисленные пометки. Библиотечная карточка поддерживает двустороннюю связь между сущностями "читатель" и "книга".

Если тип сущности — это класс однотипных объектов, то как же величать отдельный объект, входящий в этот класс? В реляционной модели такой объект называют *экземпляром сущности*. Экземпляр сущности (или просто сущность) — это конкретная реализация объекта. Для сущности "автор" это Уильям Шекспир и Лев Толстой, для сущности "книга" это "Ромео и Джульетта" и "Война и мир".

ЗАМЕЧАНИЕ

Экземпляр сущности — это то, о чем (или о ком) следует хранить информацию в базе данных.

У любой сущности есть свои характеристики. У сущности "самолет" из базы данных авиакомпании таких характеристик миллион. Это марка, максимальная скорость полета, число посадочных мест, грузоподъемность, дата выпуска, цвет фюзеляжа и многое-многое другое. У сущности "читатель" отличительных признаков не меньше, а может быть и больше, чем у летательного аппарата. В счет может пойти все, начиная с цвета глаз и заканчивая размером обуви. Однако в библиотечной базе данных вряд ли стоит учитывать рост, вес и отпечатки пальцев клиента. Гораздо полезнее знать имя, домашний адрес и номер телефона. Отличительные свойства сущности называют *атрибутами*.

Атрибуты могут быть *простыми* (например, название города, в который летит самолет), а может быть и *составными* (т. е. включать в себя несколько простых атрибутов). В качестве составного атрибута выступает адрес читателя. Здесь найдется место почтовому индексу, городу, улице, дому и номеру квартиры. Атрибут может быть *производным*, т. е. полученным в результате проведения какой-то операции над другими атрибутами. Подобные атрибуты часто встречаются при построении различных бухгалтерских систем, например, значение взимаемого с сотрудников предприятия налога составляет определенный процент от заработной платы этих служащих. Можно столкнуться с *многозначными* атрибутами, они описывают те свойства сущности, в которых одновременно хранится несколько значений. Например, у одного человека может быть несколько адресов проживания, несколько контактных телефонных номеров и т. п. Есть и особая разновидность атрибутов, без которых невозможно построение реляционной модели, — атрибуты, однозначно идентифицирующие экземпляр сущности. На роль такого атрибута в состоянии претендовать индивидуальный номер налогоплательщика или уникальный заводской номер самолета. Позднее подобные атрибуты превратятся в ключевые поля реляционных таблиц.

Построив исчерпывающий список типов сущностей и подготовив перечень их атрибутов, разработчик модели базы данных должен задуматься над особенностями реализации типов сущностей. Для этого в первую очередь следует разобраться с особенностями физического представления каждого из атрибутов типа сущности, для этого предназначены тип данных и домен.

Тип данных и домен

Отправной точкой процесса построения любой системы хранения и обработки данных по праву считается выбор *типа данных*. Независимо от того, какой язык программирования вы изучали, при построении новой программы самым первым шагом становится рассмотрение типов данных, имеющих в распоряжении системы. Программистов Delphi, Си или поклонников любого другого языка программирования не удивит типами данных `INTEGER`, `REAL` или `CHAR`. Указанные типы данных специализируются соответственно на обслуживании целочисленных, вещественных и символьных значений.

Типы данных определяют порядок хранения данных в памяти компьютера. Физическая концепция хранения данных зависит от особенностей конкретной платформы, на базе которой функционирует программное обеспечение. Поэтому в разных операционных системах, в разных системах управления базами данных число байтов, отводимых для описания целого или вещественного числа, может отличаться. Например, в 32-разрядных ОС размерность целого числа `INTEGER` равняется 4 байтам. В 64-разрядных системах это утверждение может оказаться ошибочным, ведь здесь проще адресовать 8-байтовую переменную.

Типизация хранимых значений не просто указывает на размерность в байтах, которую должна выделить система для размещения в памяти того или иного значения. Преследуемая цель еще более значима. Типизация определяет, какие операции могут быть осуществлены с теми или иными данными. Система программирования не позволит новичку передать результаты деления в целочисленную переменную, ведь в этом случае есть риск утратить дробную часть результата. Система станет отчаянно сопротивляться, если мы попробуем просуммировать символьный и вещественный тип данных. Пусть даже в символьной переменной хранится числовое значение. В последнем случае перед проведением математической операции необходимо провести преобразование данных. Точно так же СУБД не допустит ввода текстовых данных в поля таблиц, предназначенных для хранения числовой информации.

Подытожим все сказанное. Понятие "тип данных" интегрирует в себе три компонента:

1. Ограничение множества значений, принадлежащих типу.
2. Дефиниция набора операций, применяемых к типу.
3. Определение способа отображения (внешнего представления) значений типа.

Как показала практика, при обработке данных в современных информационных системах возможностей обычной типизации становится недостаточно. Примеров тому великое множество. Формально, при определении атрибута, хранящего дату рождения сотрудников какого-нибудь предприятия, следует ограничиться типом данных `DATE`. Но такой подход не способен в полной мере отразить задачи отдела кадров. Стандартизированный в SQL тип данных `DATE` допускает ввод даты в диапазоне от 1/01/0000 г. до 12/31/9999 г. Вряд ли начальник нынешнего отдела кадров примет на работу сотрудника в несовершеннолетнем возрасте или, наоборот, ровесника Ивана Грозного. Поэтому атрибут "дата_рождения" должен уметь сопротивляться данным, некорректным, с точки зрения пользователя, но абсолютно верным с точки зрения типа данных `DATE`. Есть и другой пример. Допустим, что в базе данных хранится почтовый индекс. Каждый поклонник эпистолярного жанра знает, что это комбинация из шести цифр, отражающая региональную принадлежность почтового отделения. Порядок описания индекса не сложен для человека — нужно заполнить все шесть значений, не четыре, не пять и не семь, а только шесть. Теперь попробуйте подыскать стандартный тип данных, способный однозначно решить эту простейшую задачу. Тип `INTEGER` не подходит, потому что обладает существенно бóльшим диапазоном значений. Строка из шести элементов `CHAR` также не вполне адекватна, ведь она позволяет пользователю вводить любые (не только цифровые) символы. Рассмотрим еще один пример. Формально номер телефона можно типизировать как целое число — `INTEGER`. Целые числа поддерживают все основные арифметические операции. Однако никому в здравом уме и в доброй памяти не придет в голову идея суммировать значения телефонных номеров или вычитать из одного номера другой, хотя формально это обычные целочисленные значения. Просто очень сомнительна практическая ценность полученного результата. В подобных и во многих других схожих ситуациях одна лишь типизация бессильна — она в недостаточной степени поддерживает че-

ловеческую логику. Если я вас убедил, что возможностей типа данных недостаточно для описания характеристик атрибутов, закончим с примерами и поговорим о том, как это ограничение можно преодолеть.

Фундаментальная для современных языков программирования идея типизации в реляционной модели получила дальнейшее развитие. На сцену вышло новое понятие — *домен*. Домен ни в коем случае не подменяет понятие типизации, а выступает в качестве дополнительного ограничителя информации, обрабатываемой в базе данных. Ограничение накладывается за счет введения дополнительных логических правил. Так, принимая нового сотрудника на работу, при вводе даты рождения можно контролировать его возраст, путем элементарного сравнения текущей системной даты компьютера и даты рождения. Обслуживая атрибут почтового индекса, описываемого шестью символами типа CHAR, наложим дополнительное ограничение на диапазон приемлемых значений — наш домен должен пропускать только цифровые знаки. Надо сразу оговориться, что логические правила являются не столько описанием отдельного домена, сколько средством поддержания целостности данных в масштабах всей базы данных. Допустим, вы вводите почтовый индекс города Москвы, а в атрибут "город" передаете Монреаль. В этом случае разумная база данных должна намекнуть нам, что мы сегодня перетрудились и следует прерваться на чашечку кофе... Но для того, чтобы система смогла выявить подобную логическую ошибку, разработчик БД должен построить не только строгий набор доменных ограничений, но и реализовать развитую систему поддержания корпоративной целостности.

ЗАМЕЧАНИЕ

Тип данных определяет особенности физического хранения данных. Домен, опираясь на концепцию типа данных, несет дополнительную нагрузку — отвечает за поддержание логических правил описания данных. Благодаря доменным ограничениям в БД реализуется одно очень важное качество — доменная целостность данных.

В профессиональных СУБД доменные ограничения могут накладываться на несколько атрибутов одновременно, таким образом, одновременно контролируя непротиворечивость хранящихся в них данных.

СВЯЗЬ

Реляционная модель призвана не только описать перечень типов сущностей, но и отразить особенности взаимодействия между ними. Взаимодействие возникает там, где между типами сущностей проявляются отношения, выраженные в виде глагола, например, продавец оформил заказ. Здесь "продавец" и "заказ" — это две сущности, а связь между ними просматривается в глаголе "оформил".

Различают три типа связи между сущностями: "один к одному", "один ко многим" и "многие ко многим". Появление в реляционной БД связи "один к одному" — крайне редкий случай. Чаще всего он свидетельствует о том, что разработчик не

вполне корректно определился с атрибутами и вместо одного из атрибутов создал лишний тип сущности. На практике наиболее распространена связь "один ко многим", например, автор написал много книг, завод выпустил много автомобилей, в одном отделе работает много сотрудников (рис. 4.1).



Рис. 4.1. Отношение "один ко многим"

Связь "многие ко многим" встречается реже: офисы разных компаний размещены в разных городах, множество издательств опубликовало произведения различных авторов и т. п. Реляционная модель напрямую не поддерживает отношение "многие ко многим", поэтому его приходится разбивать на два отношения "один ко многим", но об этом поговорим позднее.

Есть еще один классификационный признак связи — *степень* (размерность). При выявлении связей между сущностями разработчик обычно встречается с *унарными* (unary relationship) и *двойными связями* (binary relationship), но в реальной жизни имеют место связи и большей кратности. Например, выражение "авиакомпания доставляет пассажира в город" подразумевает *тернарную связь* (ternary relationship). Здесь мы сталкиваемся с тремя существительными-сущностями "авиакомпания", "пассажир" и "город" и всего одним глаголом "доставляет". Дальнейшее развитие модели такой базы данных во многом зависит от опыта и интуиции разработчика, ведь разные пассажиры вправе летать в различные города, пользуясь услугами разных авиакомпаний. Ограничившись лишь двойными связями, можно утратить суть сохраняемой информации — какая из авиакомпаний, в какой именно из городов доставила конкретного пассажира. К вопросу построения связей класса "многие ко многим" мы вернемся в *главе 5* при описании процесса ER-моделирования, а пока обсудим еще одну особенность связи.

Реляционная модель позволяет связывать сущность саму с собой — задавать *рекурсивную связь*. Такой, на первый взгляд, несколько необычный способ взаимодействия оказывается весьма полезным при "выращивании" древообразных конструкций, например, описывающих организационную структуру предприятия, завода или, скажем, университета. Здесь четко просматривается иерархия, в которой одна и та же сущность может быть главной по отношению к подчиненным элементам и одновременно находиться в подчиненном состоянии к более высокому по рангу узлу. Так, дочерними элементами по отношению к деканату факультета выступают кафедры этого факультета, в свою очередь деканат подчиняется ректорату учебного заведения.

Реляционная таблица

Пользователь рассматривает всю информацию в реляционной БД как совокупность взаимосвязанных двумерных таблиц (отношений), в каждой из которых хранятся данные о строго определенном типе сущностей.

Реляционная таблица состоит из строк и столбцов. Каждая строка содержит информацию о конкретной сущности, так в таблице "Авторы" из библиотечной базы данных мы увидим строки с данными о А. С. Пушкине, М. Ю. Лермонтове, Ф. М. Достоевском и других известных (и не очень) поэтах и писателях. Но в таблице "Авторы" не должно оказаться ни единой строки о клиентах библиотеки — читателях, ведь это другой тип сущности. Каждый столбец реляционной таблицы предназначен для хранения определенного атрибута сущности. Для атрибута "Фамилия" разработчик таблицы применит текстовый тип данных размерностью в 15–20 символов. А для атрибута "Номер телефона" (которому найдется место в таблице читателей) будут наложены дополнительные доменные ограничения, запрещающие передавать в столбец нецифровые символы.

ЗАМЕЧАНИЕ

Терминология реляционных баз данных складывалась из нескольких областей знаний, поэтому в качестве синонимов к понятию "строка" можно услышать слова "ряд" и "кортеж". У термина "столбец" также есть два тождественных понятия: "атрибут" и "поле". Ведя речь о таблице, допустимо применять термин "отношение" (рис. 4.2).

Атрибут
(столбец, поле)

Таблица READER

Имя поля

PEOPLES_KEY	SNAME	FNAME	PHONENUM	BDAY
1	Петров	Василий	34-74-454	1/01/1970
2	Акулов	Алексей	76-15-011	11/12/1978
3	Козлова	Татьяна	83-13-116	21/05/1980
4	Козлов	Олег	83-13-116	5/10/1977
5	Миронов	Николай	22-57-962	11/12/1969

Рис. 4.2. Реляционная таблица

Для того чтобы таблица (отношение) получила высокую честь называться "реляционной", необходимо ее соответствие ряду строгих требований:

- ◆ уникальность имен столбцов внутри таблицы;
- ◆ каждая строка хранит информацию об отдельной сущности, строки таблицы отличаются друг от друга хотя бы единственным значением, что позволяет однозначно идентифицировать любую строку такой таблицы;