Титульный лист методических рекомендаций и указаний

Форма Ф СО ПГУ 7.18.3/40

Министерство образования и науки Республики Казахстан

Павлодарский государственный университет им. С. Торайгырова

Кафедра Информатики и информационных систем

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ И УКАЗАНИЯ

к изучению дисциплины Базы данных в ИС
для студентов специальности 5В070300 «Информационные системы»

Павлодар

Лист утверждения методических рекомендаций и указаний

Форма
Ф СО ПГУ 7.18.3/41

УТВЕРЖДАЮ
Проректор по УР
_____ Пфейфер Н.Э.
(подпись) (Ф.И.О.)
«___»_____20__г.

Составитель: старший преподаватель Т
Кафедра Информатика и информа
(наименование каф

Методические рекомендации и указания

к изучению дисциплины Базы данных в ИС

для студентов специальности 5В070300 «Информационные системы»

Рекомендовано на заседании кафедры
«___»_____20__г., протокол №__

Заведующий кафедрой _____ Оспанова Н.Н. «____» _____20__г.
(подпись) (Ф.И.О.)

Одобрено УМС Факультета ФМиИТ
(наименование факультета)
«___»_____20__г., протокол №____

Председатель УМС _____ Искакова А.Б. «____» _____20__г.
(подпись) (Ф.И.О.)

ОДОБРЕНО:
Начальник УМО _____ Жуманкулова Е.Н. «____» _____20__г.
(подпись) (Ф.И.О.)

Одобрена учебно-методическим советом университета
«___»_____20__г. Протокол №____

Тема 1 Базы данных и субд

Введение в базы данных

Базами данных (БД) называют электронные хранилища информации, доступ к которым осуществляется с одного или нескольких компьютеров. Обычно БД создается для хранения и доступа к данным, содержащим сведения о некоторой предметной области, то есть некоторой области человеческой деятельности или области реального мира.

Типы СУБД

Системы управления базами данных (СУБД) — это программные средства, предназначенные для создания, наполнения, обновления и удаления баз данных. Различают три основных вида СУБД: промышленные универсального назначения, промышленные специального назначения и разрабатываемые для конкретного заказчика. Специализированные СУБД создаются для управления базами данных конкретного назначения — бухгалтерские, складские, банковские и т. д. Универсальные СУБД не имеют четко очерченных рамок применения, они рассчитаны «на все случаи жизни» и, как следствие, достаточно сложны и требуют от пользователя специальных знаний. Как специализированные, так и универсальные промышленные СУБД относительно дешевы, достаточно надежны (отлажены) и готовы к немедленной работе, в то время как заказные СУБД требуют существенных затрат, а их подготовка к работе и отладка занимают значительный период времени (от нескольких месяцев до нескольких лет). Однако в отличие от промышленных заказные СУБД в максимальной степени учитывают специфику работы заказчика (того или иного предприятия), их интерфейс обычно интуитивно понятен пользователям и не требует от них специальных знаний.

По своей *архитектуре* СУБД делятся на одно-, двух- и трехзвенные (рис. 1.1). В однозвенной архитектуре используется единственное звено (клиент), обеспечивающее необходимую логику управления данными и их визуализацию. В двухзвенной архитектуре значительную часть логики управления данными берет на себя сервер БД, в то время как клиент в основном занят отображением данных в удобном для пользователя виде. В трехзвенных СУБД используется промежуточное звено — сервер приложений, являющееся посредником между клиентом и сервером БД. Сервер приложений призван полностью избавить клиента от каких бы то ни было забот по управлению данными и обеспечению связи с сервером БД.

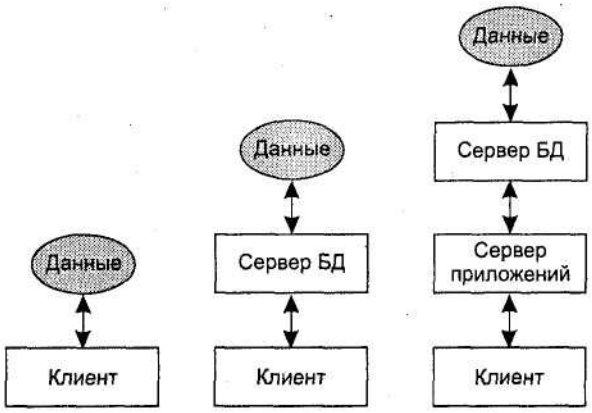


Рис. 1 Архитектура СУБД: однозвенная (слева); двухзвенная (в центре); трехзвенная (справа)

В зависимости от местоположения отдельных частей СУБД различают *локальные* и *сетевые* СУБД.

Все части локальной СУБД размещаются на компьютере пользователя базы данных. Чтобы с одной и той же БД одновременно могло работать несколько пользователей, каждый пользовательский компьютер должен иметь свою копию локальной БД. Существенной проблемой СУБД такого типа является синхронизация копий данных,

именно поэтому для решения задач, требующих совместной работы нескольких пользователей, локальные СУБД фактически не применяются.

К сетевым относятся *файл-серверные, клиент-серверные* и *распределенные* СУБД. Непременным атрибутом этих систем является сеть, обеспечивающая аппаратную связь компьютеров и делающая возможной корпоративную работу множества пользователей с одними и теми же данными.

В файл-серверных СУБД все данные обычно размещаются в одном или нескольких каталогах достаточно мощной машины, специально выделенной для этих целей и постоянно подключенной к сети. Такой компьютер называется *файл-сервером* — отсюда название СУБД. Безусловным достоинством СУБД этого типа является относительная простота ее создания и обслуживания — фактически все сводится лишь к развертыванию локальной сети и установке на подключенных к ней компьютерах сетевых операционных систем. По счастью, Delphi «умеет» использовать сетевые средства самой популярной в мире ОС — Windows для создания соответствующих *клиентских мест*, то есть специального программного обеспечения компьютеров пользователей. Нетрудно заметить, что между локальными и файл-серверными вариантами СУБД нет особых различий, так как в них все части собственно СУБД (кроме данных) находятся на компьютере клиента. По архитектуре они обычно являются однозвенными, но в некоторых случаях могут использовать сервер приложений. Недостатком файл-серверных систем является значительная нагрузка на сеть.

Если, например, клиенту нужно отыскать сведения об одной из фирм-партнеров, по сети вначале передается весь файл, содержащий сведения о многих сотнях партнеров, и лишь затем в созданной таким образом локальной копии данных отыскивается нужная запись. Ясно, что при интенсивной работе с данными уже нескольких десятков клиентов пропускная способность сети может оказаться недостаточной, и пользователя будут раздражать значительные задержки в реакции СУБД на его требования. Файл-серверные СУБД могут успешно использоваться в относительно небольших фирмах с количеством клиентских мест до нескольких десятков.

Клиент-серверные (двухзвенные) системы значительно снижают нагрузку на сеть, так как клиент общается с данными через специализированного посредника — *сервер базы данных*, который размещается на машине с данными. Сервер БД принимает запрос от клиента, отыскивает в данных нужную запись и передает ее клиенту. Таким образом, по сети передаются относительно короткий запрос и единственная нужная запись, даже если соответствующий файл с данными содержит сотни тысяч записей. Запрос к серверу формируется на специальном языке структурированных запросов (Structured Query Language, SQL), поэтому часто серверы БД называются SQL-серверами. Серверы БД представляют собой относительно сложные программы, разрабатываемые различными фирмами. К ним относятся, например, Microsoft SQL Server производства корпорации Microsoft, Sybase SQL Server корпорации Sybase, Oracle производства одноименной корпорации, DB2 корпорации IBM и т. д. SQL-сервером является также и сервер InterBase корпорации Borland, который поставляется вместе с Delphi 7 Studio в комплектациях Enterprise и Architect. Клиент-серверные СУБД масштабируются до сотен и тысяч клиентских мест.

Распределенные СУБД могут содержать несколько десятков и сотен серверов БД. Количество клиентских мест в них может достигать десятков и сотен тысяч. Обычно такие СУБД работают на предприятиях государственного масштаба, отдельные подразделения которых разнесены на значительной территории. К таковым, например, относятся подразделения Министерства обороны и Министерства внутренних дел. В распределенных СУБД некоторые серверы могут дублировать друг друга с целью достижения предельно малой вероятности отказов и сбоев, которые могут исказить жизненно важную информацию. Они используют собственные региональные средства связи. Интерес к распределенным СУБД возрос в связи со стремительным развитием

Интернета. Опираясь на возможности Интернета, распределенные системы строят не только предприятия государственного масштаба, но и относительно небольшие коммерческие предприятия, обеспечивая своим сотрудникам работу с корпоративными данными на дому и в командировках.

ADO, существенно ниже, чем при использовании BDE, а технология InterBase Express может работать только с серверами InterBase версии 5.5 и выше. Технология dbExpress позволяет обращаться непосредственно к SQL-серверам InterBase, MySQL, Oracle, DB2.

Под курсором набора данных понимается указатель текущей записи в конкретном НД. Текущая запись — это та запись, над которой в данный момент времени можно выполнять какие-либо операции (удаление, изменение, чтение значений, содержащихся в полях записи).

Рекомендуемая литература: [1, 2,3]

Тема 2 Модели данных

Таблицы

Компонент TTable (таблица) является одним из наследников класса TDBDataSet (набор данных.

В этом разделе описываются наиболее важные специфические для класса TTable свойства и методы (его события унаследованы от класса TDBDataSet и здесь не рассматриваются). Более полную информацию вы найдете в приложении.

Возможность менять текущий индекс и таким способом влиять на сортировку табличных данных является отличительной особенностью компонента TTable. В двух других компонентах-наборах программист не может явно указать индекс, а сортировка данных осуществляется в секции ORDER BY SQL-запроса. Для смены текущего индекса предназначены свойства IndexName и Index-FieldNames. В первое нужно поместить имя индекса, во второе — список индексных полей, входящий в состав индекса. Эти свойства взаимоисключающие: установка значения в одно из них очищает другое. Если НД открыт, смена текущего индекса приводит к немедленному изменению порядка следования записей в таблице.

Добавление нового индекса

Добавление нового индекса происходит в режиме эксклюзивного доступа к таблице БД (свойство Exclusive = True) и осуществляется следующим методом:

procedure Addlndex**(const** Name, Fields: **String;**

Options: TIndexOptions)

Здесь параметр Name определяет имя индекса, а параметр Fields — список индексных полей. В случае нескольких полей соседние имена в списке разделяет точка с запятой. Должны указываться только поля, объявленные в структуре таблице БД, в противном случае будет возбуждена исключительная ситуация. Параметр Options является множеством, которое содержит значения, определяющие свойства индекса:

ixCaseInsensitive — индекс чувствителен к регистру букв;

ixDescending — индексные поля сортируются по убыванию значений;

ixPrimary — создается первичный ключ;

ixUnique — значения полей в индексе должны однозначно определять запись.

Удаление индекса

Удаление существующего индекса происходит в режиме эксклюзивного доступа к таблице БД и осуществляется следующим методом:

procedure Deletelndex**(const** Name: **String)**

Здесь параметр Name определяет имя удаляемого индекса. При попытке удаления несуществующего индекса возбуждается исключительная ситуация.

Рекомендуемая литература: [1, 2,3,5,7]

Тема 3 Создание баз данных в современной СУБД

Составные индексы

Составными называются индексы, построенные по значениям двух и более полей одновременно. Такие индексы широко используются на практике для ускорения поиска нужной записи или группы записей методами FindKey и FindNearest (см. ниже).

Если в таблице построен индекс только по полю Field1, а требуется отыскать записи, содержащие нужные значения в полях Field1и Field2, то методы FindKey и FindNearest смогут установить курсор лишь в начало целой группы записей, поля Field1 в которых имеют нужные значения, но отличаются значениями остальных полей. В этом случае для поиска записей с нужными значениями поля Field2 придется последовательно просматривать таблицу, начиная с найденной записи. Если построен составной индекс по полям Field1 и Field2, методы FindKey и FindNearest сразу позиционируют курсор на нужную запись и для таблиц большого размера могут существенно сократить время поиска. Например, в моей практике был случай, когда в таблице, насчитывающей более 800 000 записей, нужно было отыскать группу записей с заданными значениями трех полей. И хотя по каждому полю в отдельности был построен индекс, поиск проходил с заметными задержками (5-6 с), которые вызывали у пользователей вполне оправданное раздражение. После построения составного индекса время поиска сократилось до долей секунды, и пользователи просто перестали замечать задержки.

Для создания составного индекса в серверных таблицах и таблицах типа Paradox с помощью программы SQL Explorer используется SQL-запрос типа

CREATE INDEX IndexName ON TableName(Fieldl, Field2,...)

В широко распространенных таблицах типа dBASE нельзя создать составной индекс и, следовательно, реализовать быстрый индексный поиск при участии нескольких полей. В базирующихся на них средствах разработки (dBASE III+, dBASE IV, FoxPro, Clipper) для этих целей обычно используется индекс, построенный по индексному выражению, с приведением нужных полей к строковому типу. Например:

STR(FINCODE, 5)+LEFT(FINTYPE, I)+DTOS(FINDATE)

Это индексное выражение формирует строку из 5 символов преобразования целого значения поля FINCODE, первого символа значения строкового поля FINTYPE и преобразованного к строке значения даты из поля FINDATE.

Увы! Delphi не имеет средств поиска по индексным выражениям. Таким образом, использование таблиц dBASE в приложениях Delphi менее удобно, чем таблиц Paradox: если вы собираетесь разработать приложение для пока еще не созданной БД и выбрали традиционную архитектуру файл-серверных таблиц, обратите на это внимание; если вы создаете приложение для уже существующей и наполненной БД, в которой используются таблицы dBASE, имеет смысл подумать о передаче содержащихся в них данных в таблицы Paradox с помощью компонентов TBatchMove.

. **Рекомендуемая литература: [2,3]**

Тема 4 Проектирование баз данных

Кэширование изменений

Суть кэширования изменений заключается в том, что в каталоге запуска программы создается локальная копия данных и все последующие изменения относятся не к реальным данным таблиц БД, а к хранящейся в буфере (кэше) их локальной копии. После

изменения данных в кэше они могут быть либо перенесены в реальные таблицы БД (подтверждение изменений), либо кэш ликвидируется без запоминания изменений (откат изменений).

Любой НД может быть переведен в режим кэширования изменений, если в его свойство CachedUpdates поместить значение True. Подтверждение изменений происходит путем обращения к методу ApplyUpdates набора данных, откат — обращением к его методу CancelUpdates.

Помимо очевидных преимуществ кэширования изменений (снижение нагрузки на сеть и отсутствие блокировок данных в условиях многопользовательского доступа к ним), этот прием позволяет реализовать разного рода диалоговые окна (формы), в которых пользователь может вначале изменить данные, но потом передумать и отказаться от сделанных изменений.

В принципе программа может одновременно кэшировать сколько угодно записей и наборов данных. Однако для локальных (файл-серверных) таблиц типа dBASE при одновременном подтверждении изменений в более чем 100 записях НД должен быть переведен в режим эксклюзивного доступа (для таблиц Paradox это необходимо сделать при подтверждении более 255 записей).

Метод ApplyUpdates компонента TDatabase

Подключение компонентов-наборов к таблицам БД желательно выполнять с помощью компонента TDatabase. Этот компонент не только создает локальный псевдоним БД, но обладает также целым рядом полезных свойств и методов, которые управляют сразу всеми подключенными к нему НД. Одним из таких методов является метод ApplyUpdates, с помощью которого программа может подтвердить кэшированные изменения сразу в нескольких НД (список имен НД передается методу в виде конструктора массива; соседние имена разделяются запятыми).

Например:

DM. DB. ApplyUpdates(['Nakls', 'Books', 'Firms']);

Видимость измененных записей и выборочный откат изменений

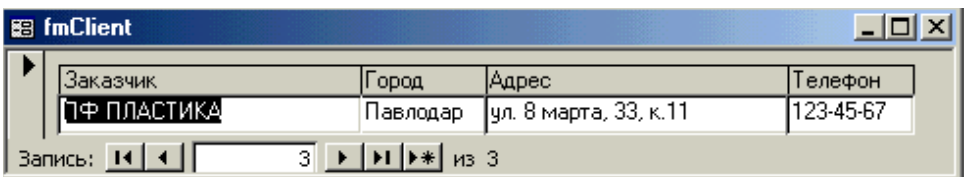
С помощью следующего свойства программист может указать, какие записи будут «видны» в кэше:

type TUpdateRecordTypes = **set** of (rtModified, rtlnserted,

rtDeleted, rtUnmodified); **property** UpdateRecordTypes: TUpdateRecordTypes;

Здесь rtModified —измененные; rtlnserted — вставленные; rtDeleted —удаленные; rtUnmodified — неизмененные записи. По умолчанию в множество. включены значения rtModified, rtlnserted и rtUnmodified, то есть в кэше программе доступны измененные, новые и неизмененные записи. Если в свойство поместить значение rtDeleted, программа получит доступ и к удаленным записям. Однако в этом случае свойство НД RecordCount покажет количество записей за вычетом удаленных. Следующее свойство возвращает статус текущей записи в кэше:

Остальной лекционный материал находится на прилагаемом к УМКД электронном носителе (компакт-диске) в файле «Базы данных в Delphi 7.doc» в следующем соответсвии между лекциями и разделами файла



Рекомендуемая литература: [1, 2,7]

Список литературы

Основная:

1 Старков В. В. Архитектура персонального компьютера: организация, устройство, работа : учеб. пособие для вузов по спец. "Информатика и вычислительная техника" - М. : Горячая линия-Телеком, 2009. - 536 с.

2 Жмакин А. П. Архитектура ЭВМ : учеб. Пособие - СПб. : БХВ-Петербург, 2008. - 306 с.

3 Ручкин В. Н., Фулин, В. А. Информационный маркетинг - М. : ДИАЛОГ-МИФИ, 2008. - 240 с.

4 Дюсембаев А. Е Архитектура компьютеров : учеб. пособие по Cjmputer Science. Образовательная программа Европейского Союза TEMPUS-TACIS (Contract # CD Jer-22077-2001) - Алматы : Print-S, 2009. - 111 с.

Дополнительная:

5 Брей Б. Применение микроконтроллеров PIC18: архитектура, программирование и построение интерфейсов с применением С и ассемблера [Электронный ресурс] - Электрон. прикладная прогр. (126 Мб.). - [б. м.] : [б. и.], 2008 1 эл. опт. диск (CD-ROM).

6 Молчанов А. Ю Системное программное обеспечение : учебник для студ. вузов СПб. : Питер, 2006. - 395 с.

7 Мюллер С. Модернизация и ремонт ПК- 17-е изд - М. : Вильямс, 2008. - 1489 с.