Кафедра прикладной математики и информационных технологий

Дисциплина
«БАЗЫ ДАННЫХ»
направление подготовки
220100.62 «Системный анализ и управление»

Тема № 3 «Локальные базы данных»
Занятие № 3.1 «Системы управления базами данных»

Учебные цели

Дидактические

- сформировать у обучающихся твердые знания в области программных средств работы с базами данных;

Учебные

- изложить основы систем управления базами данных;
- раскрыть сущность объектов современных систем управления базами данных.

Воспитательные

- воспитывать у обучающихся стремление к углубленному изучению учебного материала.

Содержание и порядок проведения занятия	Время, мин
ВСТУПИТЕЛЬНАЯ ЧАСТЬ	5
ОСНОВНАЯ ЧАСТЬ	80
Учебные вопросы:	
1. Классификация систем управления базами данных (СУБД)	25
2. Основные функции СУБД	25
3. Характеристика типовой СУБД для построения локальных баз данных	15
4. Основные объекты базы данных, создаваемые в среде СУБД	15
ЗАКЛЮЧИТЕЛЬНАЯ ЧАСТЬ	5

Литература

Основная:

1. Иванов, Александр Юрьевич. Базы данных: учебное пособие: [гриф МЧС] / А.Ю. Иванов; МЧС России. – СПб.: СПбУ ГПС МЧС России, 2010. – 204 с.

Дополнительная:

1. Кузин, Александр Владимирович. Базы данных: учебное пособие: [гриф УМО] / А.В. Кузин, С.В. Левонисова. – 4-е изд., стер. – М. : Академия, 2010. – 320 с.
2. Мамаев Е.В. SQL Server 2000. – СПб.: БХВ-Петербург, 2004. – 1280 с.
3. Грэй П. Логика, алгебра и базы данных / Пер. с англ.- М.: Машиностроение, 1989. – 368 с.

Нормативно-правовая:

Доклад «О долгосрочных перспективах развития системы МЧС России (МЧС России - 2030) Доклад Министра РФ по делам гражданской обороны, чрезвычайным ситуациям и ликвидации последствий стихийных бедствий. М.: МЧС России, 2012».

Государственный доклад «О состоянии защиты населения и территорий Российской Федерации от чрезвычайных ситуаций природного и техногенного характера в 2012 году».

Основы единой государственной политики РФ в области ГО на период 2020 года (утверждена Президентом РФ от 03.09.2011, № ПР-2613).

Стратегия инновационного развития РФ на период до 2020 года (утверждена распоряжением Правительства РФ от 08.12.2011 года, №2227-р).

Федеральный закон от 22.07.2008 г. №123 – ФЗ (ред.от 10.07.2012) «Технический регламент о требованиях пожарной безопасности».

Закон РФ от 29 декабря 2012 года №273-ФЗ «Об образовании в Российской Федерации» с изменениями и дополнениями на 2013 год.

Организационно-методические указания по подготовке территориальных органов, спасательных воинских формирований, подразделений федеральной противопожарной службы, военизированных горноспасательных частей, образовательных учреждений и организаций МЧС России в области гражданской обороны, предупреждения и ликвидации чрезвычайных ситуаций, обеспечения пожарной безопасности и безопасности людей на водных объектах на 2014-2016 годы.

Федеральный закон № 149 «Об информации, информационных технологиях и о защите информации» от 27 июля 2006 г.

1. Классификация систем управления базами данных (СУБД)

Система управления базами данных определяется как система программного обеспечения, имеющая средства, которые позволяют обрабатывать обращения к базе данных от прикладных программ и/или пользователей и поддерживать целостность базы. СУБД обеспечивает связь между прикладными программами и базой данных. Любой доступ к данным осуществляется посредством СУБД.

Многообразие уже созданных к настоящему времени СУРБД предполагает их группирование с целью всесторонней характеристики.

Различают **персональные** («легкие») и **корпоративные** («тяжелые») СУБД.

По модели данных:

Иерархические

Сетевые

Реляционные

Объектно-ориентированные

Объектно-реляционные

По степени распределённости

Локальные СУБД (все части локальной СУБД размещаются на одном компьютере)

Распределённые СУБД (части СУБД могут размещаться на двух и более компьютерах).

По способу доступа к БД

Файл-серверные

Клиент-Серверные

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервереВ файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок.

Преимуществом этой архитектуры является низкая

Недостатки: потенциально высокая загрузка локальной сети; затруднённость или невозможность централизованногоНедостатки: потенциально высокая загрузка локальной сети; затруднённость или невозможность централизованного управленияНедостатки: потенциально высокая загрузка локальной сети; затруднённость или невозможность централизованного управления; затруднённость или невозможность обеспечения таких важных характеристик как высокая надёжностьНедостатки: потенциально высокая загрузка локальной сети; затруднённость или невозможность централизованного управления; затруднённость или невозможность обеспечения таких важных характеристик как высокая надёжность, высокая

Клиент-серверные

Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно.

Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу.

Достоинства: потенциально более низкая загрузка локальной сети; удобство централизованного управления; удобство обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность.

Примеры: OracleПримеры: Oracle, FirebirdПримеры: Orac le, Firebird, InterbaseПримеры: Oracle, Firebird, Interbase, I

Встраиваемые

Встраиваемая СУБД — СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установкиВстраиваемая СУБД — СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установки. Встраиваемая СУБД предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически встраиваемая СУБД чаще всего реализована в виде подключаемой библиотекиВстраиваемая

Персональная реляционная СУБД Access фирмы Microsoft характеризуется тесной интеграцией с другими приложениями этой фирмы. Существенный элемент ее – комплекс специальных драйверов открытого интерфейса ODBC (Open DataBase Connectivity) — обеспечивает связь с реляционными СУБД других производителей, взаимодействие между различными форматами баз данных. Access располагает своим процедурным языком, реализует запросы на примере (QBE Query by Example, запрос по образцу) и запросы SQL («язык структурированных запросов») в сочетании со всеми преимуществами Windows в графике, эффективно работает в сети, активно использует графические объекты. Система включает функции обмена данными и обработки файлов, формирования интерфейса пользователя и управления периферийными устройствами.

Персональная СУБД *DBase* фирмы *Borland* поддерживает языки запросов *QBE* и *SQL*, а также включающие языки *Си, Паскаль*, ассемблеры. В состав языка *DBase* входит более 300 функций и операторов. Запросы *SQL* автоматически преобразуются в последовательность команд *DBase*. К дополнительным средствам языка относятся шаблоны для связи с автоматическим генератором кода, утилиты создания приложений, средства тестирования многопользовательских баз данных в сети и модуль запуска приложений. Язык поддерживает автоматическое сохранение информации, многоуровневую парольную защиту и кодирование данных.

Во многом подобна *Access* СУБД *Paradox* на основе язык *PAL* (*Paradox Application Language*) фирмы *Borland*, длительное время удерживавшая лидерство на рынке СУБД для персональных компьютеров

В *Access* программа может распространяться только с интегрированной средой. В отличие от нее, в *FoxPro* (*Microsoft*) и *Clipper* (*Computer Associates*) готовые программы распространяются в виде исполнительных модулей или динамически загружаемых библиотек. *FoxPro* имеет быстродействие, на порядок превышающее *Paradox* и в несколько десятков раз — *DBase*, а приложения, созданные на *FoxPro* под *MS-DOS*, можно адаптировать под *Windows*. Набор команд и функций, предлагаемых разработчикам программных продуктов в среде *FoxPro*, по мощи и гибкости отвечает самым современным требованиям к представлению и обработке данных. Он позволяет организовать максимально удобный пользовательский *Windows*-интерфейс. Система обеспечивает многоуровневый доступ к файлам, управление цветом, настройку принтеров, генерацию экранных форм и отчетов, поддержку языка *SQL* и функционирование в сети.

К наиболее распространенным корпоративным СУБД относятся *Infomix, Ingres, Sybase, Oracle, SQL Server*.

Реляционная система *Ingres* для средних и крупных компаний включает средства автоматизации разработки приложений, систему управления знаниями, библиотеку приложений на национальных языках. В СУБД реализована прямая адресация запросов конкретным клиентам с назначением приоритетов, работает контекстное переключение между различными базами данных. Многотомные таблицы позволяют распределять базы данных по различным дискам, хранить в базах не только данные, но и коммутируемые процедуры в виде распределенных объектов с доступом из разных приложений. Оптимизатор запросов *Ingress* учитывает множество факторов, таких как размер таблиц, тип данных, статистическое распределение данных в таблицах и индексах. При оптимизации все запросы приводятся к нормализованной форме, не зависящей от исходной записи. Специальный механизм правил позволяет программировать обработку ситуаций при включении, обновлении, исключении строк и изменении записей.

Более 10 лет представлен на мировом рынке пакет *Oracle*. Долгое время каждая третья продаваемая в мире СУБД работала под *Oracle*. Она наиболее часто используется в операционной среде *Unix*, хотя может работать на множестве других системных платформ. На *Oracle* разработано значительное число прикладных систем для банков, промышленных предприятий, энергетических объектов, учреждений здравоохранения. Она обеспечивает целостность баз данных при выполнении распределенных запросов, автономию узлов базы высокую производительность. Система поддерживает открытую архитектуру: в едином приложении могут согласованно работать компоненты СУБД различных фирм, файлы операционной системы, аппаратура (промышленные контроллеры, кассовые аппараты). Инструментарий *Oracle* позволяет создавать графический интерфейс пользователя со сложной логикой обработки данных. Постепенно реляционная СУБД *Oracle* преобразуется в объектно-ориентированную систему на основе языка *SQL++*, хранящую данные в виде объектов вместо таблиц.

Под управлением СУБД *Pick* сотни пользователей могут совместно использовать один компьютер. Приложения *Pick* совместимы с *MS-DOS* и *Unix*, благодаря чему около полумиллиона предприятий в мире используют *Pick* в качестве программного обеспечения административных и экономических систем.

Профессиональная СУБД *SQL Server* (*Microsoft*) – это еще одна развитая реляционная система, решающая задачи как в режиме непосредственной обработки сложных очередей, так и в качестве поддержки информационных систем принятия решения. Она имеет надежную защиту от несанкционированного доступа и сбоев и обладает широкими возможностями администрирования в режиме удаленного доступа.

К числу непроцедурных многопользовательских СУБД относятся продукты фирмы *Gupta* — одного из мировых лидеров в области профессиональных распределенных СУБД. Эти программные средства установлены более чем в 500 компаниях. Одних только копий *SQL Base* насчитывается свыше 20 тысяч в разных странах. Неограниченное число ее пользователей работает на платформах *MS-DOS, OS/2, NetWare, SUN, Unix. SQL. Gupta* поддерживает многозадачность, обеспечивает целостность данных, обрабатывает типы данных любого размера в национальных алфавитах, обладает возможностями оперативной архивации и автоматического восстановления данных после сбоя. *Gupta* предоставляет разработчикам полностью объектно-ориентированную среду. Она поддерживает DDE и OLE, MDI и множество библиотек. А сетевой набор программных продуктов обеспечивает доступ к данным различных баз данных в сети.

Интегрированные системы программирования, включающие генераторы кодов и процедурные языки, называют CASE-инструментами (*Computer Aided Software Engineering*). В таких комплексах среда проектирования не отделена от прикладной системы. Примерами CASE-инструментов являются системы *BPWin* и *ERWin* компании *LogicWorks*, *Designer/2000* компании *Oracle*, *SilverRun* компании *SilverRun Technologies*, *Rational Rose* и др. В частности, в *Oracle CASE*. Для создания конкретной прикладной системы, например банковской, проектировщик представляет свои знания о работе конкретного банка в системный словарь. Настройка проектируемой системы на технологию работы банка закладывается уже на первоначальных стадиях проектирования средствами конструктора.

При любых изменениях технологии меняется лишь представление знаний в системном словаре. Затем выполняется генерация сразу же готовой системы. Для повышения производительности *Oracle CASE* оснащена моделью-прототипом системы автоматизации деятельности коммерческого банка. Проектирование по прототипу снижает трудоемкость реализации и модификации системы, гарантирует целостность и непротиворечивость данных, полную документируемость, переносимость, «прозрачность» работы в различных сетевых средах.

2. Основные функции СУБД

К числу функций СУБД принято относить следующие:

1) непосредственное управление данными во внешней памяти;

2) управление буферами оперативной памяти;

3) управление транзакциями;

4) журнализация;

5) поддержка языков БД.

1. Непосредственное управление данными во внешней памяти

Эта функция включает обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для ускорения доступа к данным в некоторых случаях (обычно для этого используются индексы). В некоторых реализациях СУБД активно используются возможности существующих файловых систем, в других работа производится вплоть до уровня устройств внешней памяти. Но подчеркнем, что в развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, и если использует, то как организованы файлы. В частности, СУБД поддерживает собственную систему именования объектов БД.

2. Управление буферами оперативной памяти

СУБД обычно работают с БД значительного размера; по крайней мере, этот размер обычно существенно больше доступного объема оперативной памяти. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. При этом, даже если операционная система производит общесистемную буферизацию (как в случае ОС UNIX), этого недостаточно для целей СУБД, которая располагает гораздо большей информацией о полезности буферизации той или иной части БД. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

Существует отдельное направление СУБД, которое ориентировано на постоянное присутствие в оперативной памяти всей БД. Это направление основывается на предположении, что в будущем объем оперативной памяти компьютеров будет настолько велик, что позволит не беспокоиться о буферизации. Пока эти работы находятся в стадии исследований.

3. Управление транзакциями

Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует (COMMIT) изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД. Поддержание механизма транзакций является обязательным условием даже однопользовательских СУБД (если, конечно, такая система заслуживает названия СУБД). Но понятие транзакции гораздо более важно в многопользовательских СУБД.

То свойство, что каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к БД. При соответствующем управлении параллельно выполняющимися транзакциями со стороны СУБД каждый из пользователей может в принципе ощущать себя единственным пользователем СУБД (на самом деле, это несколько идеализированное представление, поскольку в некоторых случаях пользователи многопользовательских СУБД могут ощутить присутствие своих коллег).

С управлением транзакциями в многопользовательской СУБД связаны важные понятия *сериализации транзакций* и *сериального плана выполнения смеси транзакций*. Под сериализаций параллельно выполняющихся транзакций понимается такой порядок планирования их работы, при котором суммарный эффект смеси транзакций эквивалентен эффекту их некоторого последовательного выполнения. Сериальный план выполнения смеси транзакций – это такой план, который приводит к сериализации транзакций. Понятно, что если удается добиться действительно сериального выполнения смеси транзакций, то для каждого пользователя, по инициативе которого образована транзакция, присутствие других транзакций будет незаметно (если не считать некоторого замедления работы по сравнению с однопользовательским режимом).

Существует несколько базовых алгоритмов сериализации транзакций. В централизованных СУБД наиболее распространены алгоритмы, основанные на синхронизационных захватах объектов БД. При использовании любого алгоритма сериализации возможны ситуации конфликтов между двумя или более транзакциями по доступу к объектам БД. В этом случае для поддержания сериализации необходимо выполнить откат (ликвидировать все изменения, произведенные в БД) одной или более транзакций. Это один из случаев, когда пользователь многопользовательской СУБД может реально (и достаточно неприятно) ощутить присутствие в системе транзакций других пользователей.

4. Журнализация

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть: аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя)

или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции.

Понятно, что в любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД. В разных СУБД изменения БД журнализуются на разных уровнях: иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), иногда – минимальной внутренней операции модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода.

Во всех случаях придерживаются стратегии «упреждающей» записи в журнал (так называемого протокола *Write Ahead Log* – WAL). Грубо говоря, эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. Известно, что если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Самая простая ситуация восстановления – индивидуальный откат транзакции. Строго говоря, для этого не требуется общесистемный журнал изменений БД.

Достаточно для каждой транзакции поддерживать локальный журнал операций модификации БД, выполненных в этой транзакции, и производить откат транзакции путем выполнения обратных операций, следуя от конца локального журнала. В некоторых СУБД так и делают, но в большинстве систем локальные журналы не поддерживают, а индивидуальный откат транзакции выполняют по общесистемному журналу, для чего все записи от одной транзакции связывают обратным списком (от конца к началу).

При мягком сбое во внешней памяти основной части БД могут находиться объекты, модифицированные транзакциями, не закончившимися к моменту сбоя, и могут отсутствовать объекты, модифицированные транзакциями, которые к моменту сбоя успешно завершились (по причине использования буферов оперативной памяти, содержимое которых при мягком сбое пропадает). При соблюдении протокола WAL во внешней памяти журнала должны гарантированно находиться записи, относящиеся к операциям модификации обоих видов объектов. Целью процесса восстановления после мягкого сбоя является состояние внешней памяти основной части БД, которое возникло бы при фиксации во внешней памяти изменений всех завершившихся транзакций и которое не содержало бы никаких следов незаконченных транзакций.

Для того чтобы этого добиться, сначала производят откат незавершенных транзакций (*undo*), а потом повторно воспроизводят (*redo*) те операции завершенных транзакций, результаты которых не отображены во внешней памяти. Этот процесс содержит много тонкостей, связанных с общей организацией управления буферами и журналом. Для восстановления БД после жесткого сбоя используют журнал и архивную копию БД.

Это полная копия БД к моменту начала заполнения журнала (имеется много вариантов более гибкой трактовки смысла архивной копии). Конечно, для нормального восстановления БД после жесткого сбоя необходимо, чтобы журнал не пропал. Как уже отмечалось, к сохранности журнала во внешней памяти в СУБД предъявляются особо повышенные требования. Тогда восстановление БД состоит в том, что исходя из архивной копии, по журналу воспроизводится работа всех транзакций, которые закончились к моменту сбоя. В принципе, можно даже воспроизвести работу незавершенных транзакций и продолжить их работу после завершения восстановления. Однако в реальных системах это обычно не делается, поскольку процесс восстановления после жесткого сбоя является достаточно длительным.

5. Поддержка языков БД

Для работы с базами данных используются специальные языки, в целом называемые *языками баз данных*. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка – *язык определения схемы* БД *(SDL – Schema Definition Language)* и *язык манипулирования данными (DML – Data Manipulation Language).* SDL служил главным образом для определения логической структуры БД, т.е. той структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные. Мы рассмотрим более подробно языки ранних СУБД в следующей лекции.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (*Structured Query Language*). В следующих лекциях язык SQL будет рассматриваться подробнее.

3. Характеристика типовой СУБД для построения локальных баз данных

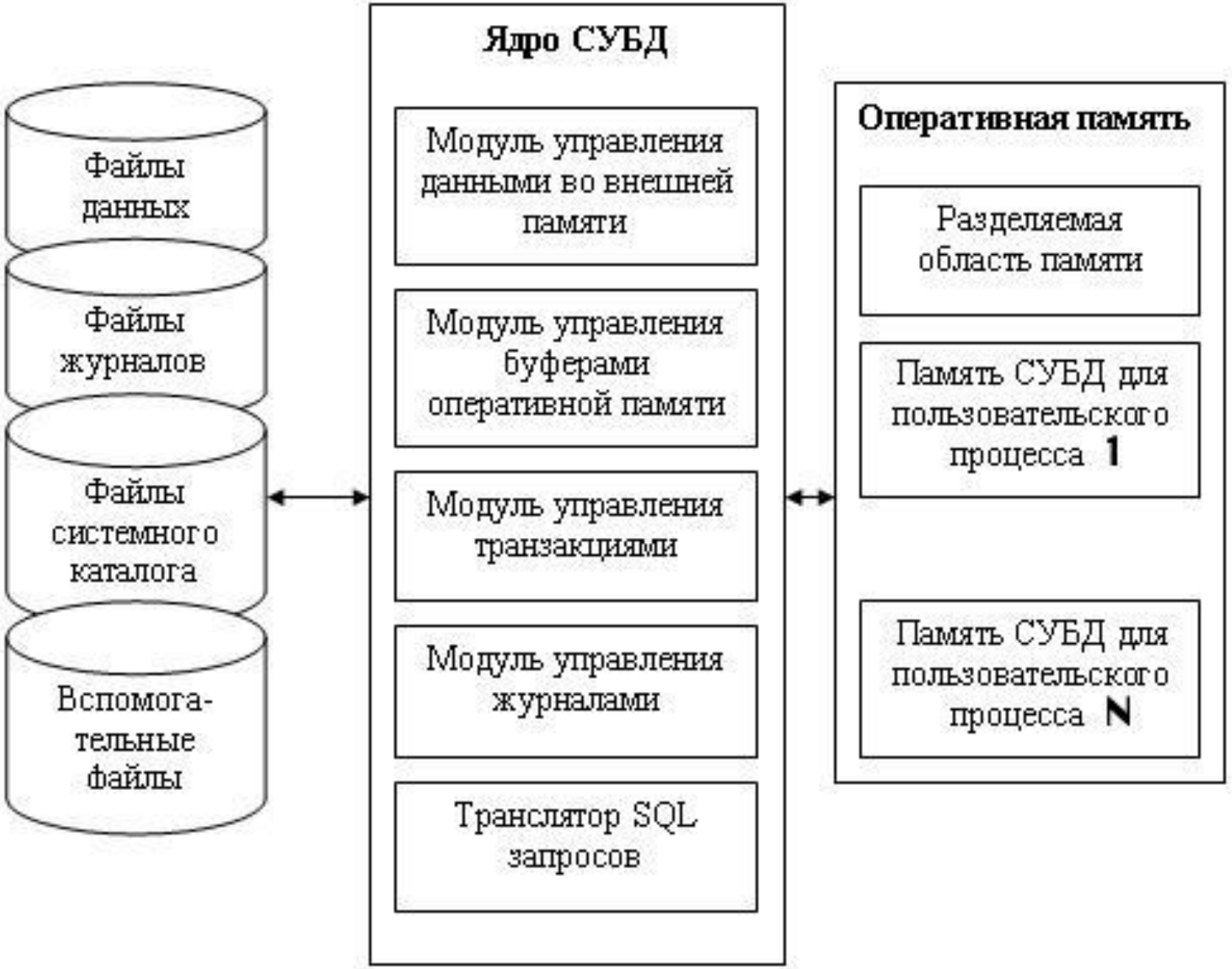
а) *Организация современной СУБД*

Организация типичной СУБД и состав ее компонентов соответствуют рассмотренному набору функций.

СУБД должна управлять внешней памятью, в которой расположены файлы с данными, файлы журналов и файлы системного каталога.

СУБД управляет и оперативной памятью, в которой располагаются буфера с данными, буфера журналов, данные системного каталога, которые необходимы для поддержки целостности и проверки привилегии пользователей. Кроме того, в оперативной памяти во время работы СУБД располагается информация, которая соответствует текущему состоянию обработки запросов,

Логически в современной реляционной СУБД можно выделить внутреннюю часть – ядро СУБД (*Data Base Engine*), транслятор языка БД (обычно SQL), подсистему поддержки времени выполнения, набор утилит. В некоторых системах эти части выделяются явно, в других – нет, но логически такое разделение можно провести во всех СУБД.


Ядро СУБД
Файлы данных
Файлы журналов
Файлы системного каталога
Вспомога-тельные файлы
Модуль управления данными во внешней памяти
Модуль управления буферами оперативной памяти
Модуль управления транзакциями
Модуль управления журналами
Транслятор SQL запросов
Оперативная память
Разделяемая область памяти
Память СУБД для пользовательского процесса 1
Память СУБД для пользовательского процесса N

Ядро СУБД отвечает за управление данными во внешней памяти, управление буферами оперативной памяти, управление транзакциями и журнализацию. Соответственно, можно выделить такие компоненты ядра (по крайней мере, логически, хотя в некоторых системах эти компоненты выделяются явно), как модуль управления данными, модуль управления буферами оперативной памяти, модуль управления транзакциями и модуль управления журналами. Функции этих компонентов взаимосвязаны, и для обеспечения корректной работы СУБД все компоненты должны взаимодействовать по определенным протоколам. Ядро СУБД обладает собственным интерфейсом, не доступным пользователям напрямую и используемым в программах, производимых компилятором SQL (или в подсистеме поддержки выполнения таких программ) и утилитах БД. Ядро СУБД

Модуль управления внешней памятью обеспечивает создание необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным в некоторых случаях (обычно для этого используются индексы). Как мы рассматривали ранее, в некоторых реализациях СУБД активно используются возможности существующих файловых систем, в других работа производится вплоть до уровня устройств внешней памяти. Но подчеркнем, что в развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, и если использует, то, как организованы файлы. В частности, СУБД поддерживает собственную систему именования объектов БД.

Модуль управления буферами оперативной памяти предназначен для решения задач эффективной буферизации, которая используется практически для выполнения всех остальных функций СУБД.

Условно оперативную память, которой управляет СУБД, можно представить как совокупность буферов, хранящих страницы данных, буферов, хранящих страницы журналов транзакций и область совместно используемого пула (рис.2). Последняя область содержит фрагменты системного каталога, которые необходимо постоянно держать в оперативной памяти, чтобы ускорить обработку запросов пользователей, и область операторов SQL с курсорами. Фрагменты системного каталога в некоторых реализациях называются словарем данных. В стандарте SQL2 определены общие требования к системному каталогу.


Оперативная память
Совместно используемый пул
Кэш словаря данных (системного каталога)
Область SQL с курсорами
Кэш буферов данных
Управляющая информация СУБД
Кэш буферов журналов

Системный каталог в реляционных СУБД представляет собой совокупность специальных таблиц, которыми владеет сама СУБД. Таблицы системного каталога создаются автоматически при установке программного обеспечения сервера БД. Все системные таблицы обычно объединяются некоторым специальным «системным идентификатором пользователя». При обработке SQL-запросов СУБД постоянно обращается к этим таблицам. В некоторых СУБД разрешен ограниченный доступ пользователей к ряду системных таблиц, однако, только в режиме чтения. Только системный администратор имеет некоторые права на модификацию данных в некоторых системных таблицах.

Основной функцией *транслятора языка БД* является компиляция операторов языка БД в некоторую выполняемую программу. Проблемой реляционных СУБД является то, что языки этих систем (а это, как правило, SQL) являются непроцедурными, т.е. в операторе такого языка специфицируется некоторое действие над БД, но эта спецификация не является процедурой, а лишь описывает в некоторой форме условия совершения желаемого действия. Поэтому компилятор должен решить, каким образом выполнять оператор языка прежде, чем произвести программу. Применяются достаточно сложные методы оптимизации операторов. Результатом компиляции является выполняемая программа, представляемая в некоторых системах в машинных кодах, но более часто в выполняемом внутреннем машинно-независимом коде.

В последнем случае реальное выполнение оператора производится с привлечением подсистемы поддержки времени выполнения, представляющей собой, по сути дела, интерпретатор этого внутреннего языка.

б) *Общие сведения о типовой СУБД для локальных баз данных*

Наиболее многочисленными и распространенными являются реляционные СУБД, которые принято подразделить на SQL-ориентированные, являющиеся программной основой построения серверов баз данных и использующие для интерфейса язык SQL, и «настольные», обладающие более дружественным интерфейсом и предназначенные дли создания пользовательских баз данных на персональных компьютерах. К числу первых можно отнести такие СУБД, как *DB2*, *ORACLE*, *INFORMIX*, *MS SQL Server* и др. Примерами вторых служат *FoxPro*, *dBASE*, *MS Access*, *Paradox*.

Настольные реляционные СУБД достаточно просты и понятны пользователю, при этом эффективны для построении БД сравнительно небольшого объема.

Изучение принципов работы настольных СУБД важно, как минимум, по двум причинам. С одной стороны, оно предоставляют возможность быстро автоматизировать решение новых информационных задач с помощью ПК, с другой – позволяет уяснить общие принципы работы реляционных СУБД как таковых и служит начальной точкой отсчета в освоении более мощных SQL-ориентированных СУБД.

В настоящее время среди пользователей ПК наиболее популярна СУБД, *Microsoft Access* (*MS Access, Access*), входящая в состав многофункционального программного пакета *MS Office*.

Характеристики СУБД Access (па примере *Access 97*)

Характеристики	*Значения характеристик*
Количество баз данных	32
Количество таблиц в базе данных	32767
Количество столбцов в таблице	255
Количество строк в таблице	Не ограничено
Количество индексов в таблице	32
Количество пользователей	255
Максимальная длина строки	2 Кб
Максимальный объем базы данных	1 Гб

4. Основные объекты базы данных, создаваемые в среде СУБД

Каждая база данных *Access* представляет собой единый файл с расширением *.mdb*, содержащий до шести категорий объектов базы данных: таблицы, запросы, формы, отчеты, макросы и модули с уникальными именами. Базы данных могут объединяться в рабочие группы, регистрируемые в файле *System.mdw*. В зависимости от прав, предоставляемых пользователям, различают следующие группы функций, реализуемых в *Access*:

доступ к данным, то есть представление информации,

управление транзакциями, то есть добавление, удаление и правка записей,

программирование, то есть создание и правка структур баз

Источником данных для объектов базы данных всех категорий являются **таблицы**. Число таблиц задается на стадии создания базы данных и может достигать 32768 с одновременным открытием до 254 таблиц. Их содержание читается и при необходимости обновляется прикладными пользователями в ходе работы с базой данных. Таблица содержит информацию о сущностях предметной области. Описание каждой сущности размещается в отдельной записи, причем каждый атрибут в этом описании занимает отдельное поле.

Информацию о содержимом таблиц дают **запросы** *Access*. Запрос помогает пользователю в проведении поиска и выборки данных, позволяет ему комбинировать сведения из нескольких таблиц, производить вычисления, вставлять, изменять, удалять, группировать, сравнивать данные. В запрос *Access* можно включать статистические и специальные операторы, арифметические и логические выражения. Запросы сохраняются для последующего использования и построения новых запросов. *Access* позволяет делать запрос одновременно к 16 таблицам, включая в него до 255 полей. К одной и той же таблице могут обращаться самые разнообразные пользователи, которых в разной степени интересуют различные поля таблиц. Поэтому таблицу или запрос часто приходится представлять различными экранными образами, называемыми формами.

Форма *Access* – это категория объектов базы данных, служащая основным инструментом доступа к данным и выполнения транзакций. В форме отражают столько данных из таблиц или запросов, сколько требуется пользователю, и в таком формате, в каком он предпочитает их видеть. Обычно форма предоставляет возможность редактирования записей в таблицах, дополнения таблиц новыми записями и освобождения от лишней информации. Через форму можно получить графическую и полнотекстовую информацию, хранящуюся в базе данных. Форма снабжается навигационными средствами перехода от записи к записи, а также со страницы на страницу (если не помещается на одной странице). Формы масштабируются, выводятся в печать, дополняются элементами графического оформления и управления по событиям.

Отчет *Access* – это категория объектов базы данных, обеспечивающая подготовку печатного документа, раскрывающего содержание таблицы или запроса. В отчете можно сортировать и группировать записи, проводить необходимые вычисления, а также упорядочивать данные и представлять их в произвольном формате. На основе разных таблиц и запросов создаются сложные отчеты, а также надписи для конвертов, этикетки, визитные карточки, письма.

Наконец, программы в виде таких категорий объектов базы данных, как **макросы** на процедурном макроязыке *Access* и **модули** на встроенном в СУБД языке *Visual Basic for Applications (VBA),* автоматизируют выполнение сложных и регулярно повторяющихся операций над сущностями баз данных. Программы могут открывать и закрывать таблицы, отображать формы, конструировать отчеты и формировать запросы. С помощью программ часто создают интерфейс пользователя, системы меню, организуют контекстную помощь, ведут учет и контроль информационных потоков.

В *Access* входит ряд специализированных программ, решающих задачи обучения работе и построения баз данных, формирования запросов и их оформления. Это **мастера**, шаг за шагом проводящие разработчика по процедуре создания объектов базы данных с предложением ему ряда типовых решений. *Access* также включает в себя механизмы, необходимые для работы с базами данных различных форматов. Специальные драйверы ODBC обеспечивают связь пользователей с реляционными СУБД других производителей.

Они позволяют напрямую обращаться к системам *DBase, Paradox, Btrieve*, а также управлять такими платформами, как *Oracle*, или связывать их с *Access*. *Access* располагает мощной справочной системой, поисковыми средствами, конверторами текстов и таблиц.

Объекты СУБД Access. СУБД Access при построении баз данных использует следующие объекты*: таблицы, запросы, формы, отчеты, макросы и модули.*

Таблица содержит данные в формате таблицы и состоит из строк (записей) и столбцов (полей). Все строки имеют одинаковую длину и неизменный порядок следования столбцов. Строка представляет собой основную единицу манипулирования данными в реляционной СУБД. Реляционная таблица не должна содержать двух одинаковых строк. Реляционная СУБД автоматически удаляет дубликаты. Каждый столбец таблицы имеет уникальное имя.

Запрос выбирает данные из нескольких таблиц на основе заданного условия и представляется тоже в формате таблицы.

Форма отображает данные одной записи из нескольких таблиц и запросов на основании описанного пользователем формата. Форма в виде некоторого бланка позволяет вводить, просматривать, редактировать и печатать данные.

Отчет отображает и печатает данные нескольких записей из нескольких таблиц или запросов на основании описанного пользователем формата.

Макрос автоматизирует стандартные действия на основе выбранных пользователем команд и событий.

Программный модуль автоматизирует сложные операции, которые нельзя описать с помощью макрокоманд макроса. Программный модуль — это набор процедур, написанных на языке программирования Visual Basic.

Задание на самоподготовку

Отработать материал лекции в плане освоения функций СУБД и объектов, создаваемых в составе баз данных.