Учреждение образования
«Белорусский государственный университет культуры и искусств»
Факультет информационно-документных коммуникаций
Кафедра информационных ресурсов и коммуникаций

СОГЛАСОВАНО
Заведующий кафедрой
_____Ж.Л. Романова
«19» 12 2024 г.

СОГЛАСОВАНО
Декан факультета
_____Ю.Н. Галковская
«19» 12 2024 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

ТЕХНОЛОГИЯ СОЗДАНИЯ БАЗ ДАННЫХ В БИБЛИОТЕКАХ

для специальности
1-23 01 11 Библиотечно-информационная деятельность
(по направлениям)

Составитель:
Т.С. Юхновец, старший преподаватель

Рассмотрено и утверждено
на заседании Совета факультета информационно-документных коммуникаций 19.12.2024 г.
протокол № 4

Составитель:

Юхновец Т.С., старший преподаватель кафедры информационных ресурсов и коммуникаций учреждения образования «Белорусский государственный университет культуры искусств»

Рецензенты:

Ученый совет государственного учреждения ««Белорусская сельскохозяйственная библиотека им. И.С. Лупиновича» Национальной академии наук Беларуси;

Н.Ю. Вайцехович, заведующий кафедрой информационно-аналитической деятельности учреждения образования «Белорусский государственный университет культуры искусств»

Рассмотрен и рекомендован к утверждению:

Кафедрой информационных ресурсов и коммуникаций учреждения образования «Белорусский государственный университет культуры и искусств» (протокол от 13.11.2024 г. № 3);
Советом факультета информационно-документных коммуникаций учреждения образования «Белорусский государственный университет культуры и искусств» (протокол от 19.12. 2024 г. № 4).

СОДЕРЖАНИЕ

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В современном мире любой вид деятельности, как правило, связан с большим объемом информации. В свою очередь, существует необходимость хранения столь значительных информационных массивов и быстрого поиска в них необходимых сведений. Существенно облегчить эти задачи позволяют базы данных – специальным образом организованная совокупность материалов, использование которой позволяет осуществлять поиск, обработку и извлечение данных, необходимых для удовлетворения информационных потребностей и запросов. Для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации созданы специальные программы – системы управления базами данных.

В деятельности традиционных библиотек как социальных институтов, предназначенных для сбора, обработки, хранения и предоставления информационных ресурсов, базы данных занимают одно из приоритетных мест и имеют стратегическое значение, поскольку оказывают определяющее влияние на ее эффективность. Каждая библиотека уже столкнулась или в самое ближайшее время столкнется с проблемами разработки, формирования и использования баз данных. От уровня знаний баз данных и технологий создания и поддержки их функциональности, которыми владеют библиотечные специалисты, зависит качество информационного обслуживания в библиотеках. В связи с этим особую актуальность приобретает освоение библиотечными специалистами принципов создания и эффективного применения баз данных и систем управления базами данных, средств администрирования и защиты баз данных.

Учебная дисциплина «Технология создания баз данных в библиотеках» является одной из основных специальных дисциплин при подготовке студентов по специальности 1-23 01 11 «Библиотечно-информационная деятельность», которая входит в обязательный компонент учебного плана.

Учебная дисциплина «Технология создания баз данных в библиотеках» является одним их компонентов комплекса учебных дисциплин, направленных на подготовку высококвалифицированных библиотечно-информационных специалистов в области формирования и использования информационных ресурсов.

Данный учебно-методический комплекс (далее – УМК) представляет собой систему дидактических средств обучения по учебной дисциплине «Технология создания баз данных в библиотеках». Он является структурно-логической моделью процесса формирования профессионально значимых

компетенций библиотекаря-библиографа, необходимых для понимания сущности, классификации, функций и особенностей баз данных и систем управления базами данных; проектирования, создания и использования баз данных библиотек.

Цель УМК по учебной дисциплине «Технология создания баз данных в библиотеках» – это систематизация учебно-методических материалов, необходимых при изучении теоретических, организационных, технологических вопросов создания, функционирования и использования баз данных в условиях деятельности библиотек; учебно-методическая поддержка студентов в освоении учебного материала; повышение эффективности организации учебного процесса и самостоятельной работы студентов на основе компетентностного подхода.

Задачи УМК:

– систематизация содержания учебной дисциплины «Технология создания баз данных в библиотеках»;

– упорядочение процесса изучения учебной дисциплины с учетом достижений науки и практики;

– обеспечение организации самостоятельной учебной работы и контроля знаний студентов;

– оказание студентам методической помощи в освоении учебного материала;

– оказание преподавателям методической помощи, необходимой для качественного преподавания данной учебной дисциплины.

Структура УМК соответствует рекомендациям «Положения об учебно-методическом комплексе в учреждении образования «Белорусский государственный университет культуры и искусств» (утверждено Приказом ректора университета 26.04.2017, № 69) и включает в себя следующие разделы:

Теоретический раздел. В соответствии с учебной программой по курсу все разделы и темы представлены краткими текстами лекций в данном УМК.

Практический раздел – содержит рабочие материалы, задания к лабораторным занятиям, методические рекомендации к их выполнению в объеме, определенном учебной программой, которые будут способствовать усвоению, формированию умений и навыков в области цифровизации библиотечно-информационной деятельности, закреплению теоретических знаний и их проверке.

Раздел контроля знаний – включает тестовые задания по темам учебной дисциплины, перечень вопросов к зачету, перечень рекомендованных средств диагностики результатов учебной деятельности студентов.

Вспомогательный раздел включает учебную программу по учебной дисциплине, учебно-методическую карту учебной дисциплины, списки основной и дополнительной литературы.

2. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Конспект лекций

Тема 1. Основы теории баз данных. Системы управления базами данных

План:

1. Основные понятия теории баз данных.
2. База данных как интегрированный ресурс.
3. Категории пользователей баз данных.
4. Основные подходы к определению понятия «система управления базами данных». Возможности и функции систем управления базами данных.
5. Классификация систем управления базами данных.

1. Основные понятия теории баз данных.

Одним из важнейших понятий в теории баз данных является понятие *информации.* Под *информацией* понимаются любые сведения о каком-либо событии, процессе, объекте.

Данные – это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные – это информация в дискретном, фиксированном виде, удобная для хранения, обработки на компьютерах, а также для передачи по каналам связи.

База данных (БД) – именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД – это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

Отличительной чертой баз данных следует считать то, что данные хранятся совместно с их описанием, а в прикладных программах описание данных не содержится. Независимые от программ пользователя данные обычно называются *метаданными*. В ряде современных систем метаданные, содержащие также информацию о пользователях, форматы отображения,

статистику обращения к данным и другие сведения, хранятся в *словаре базы данных*.

Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Для облегчения обработки информации создаются *информационные системы (ИС).* Большинство существующих ИС являются автоматизированными.

Автоматизированная информационная система (АИС) – это система, реализующая автоматизированный сбор, обработку, манипулирование данными, функционирующая на основе компьютеров и других технических средств и включающая соответствующее программное обеспечение (ПО) и персонал. В этом качестве для краткости может использоваться термин «*информационная система»* (ИС), который подразумевает понятие «автоматизированная информационная система».

В широком понимании под определение ИС подпадает любая система обработки информации. По области применения ИС можно разделить на системы, используемые в производстве, образовании, здравоохранении, науке, военном деле, социальной сфере, торговле и других отраслях. По целевой функции ИС можно условно разделить на следующие основные категории: управляющие, информационно-справочные, поддержки принятия решений.

Иногда используется более узкая трактовка понятия ИС как совокупности аппаратно-программных средств, задействованных для решения некоторой прикладной задачи.

Информационные системы имеют следующие особенности:

- для обеспечения их работы нужны сравнительно низкие вычислительные мощности;
- данные, которые они используют, имеют сложную структуру;
- необходимы средства сохранения данных между последовательными запусками системы.

Каждая ИС в зависимости от ее назначения имеет дело с той или иной частью реального мира, которую принято называть *предметной областью* (ПрО) *системы.* Выявление ПрО – это необходимый начальный этап разработки любой ИС. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных.

Банк данных (БнД) является разновидностью ИС. БНД – это система специальным образом организованных данных: баз данных, программных,

технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

Отдельные программы или комплекс программ, реализующие автоматизацию решения прикладных задач обработки данных, называются *приложениями.* Приложения, созданные средствами СУБД, относят к *приложениям СУБД.* Приложения, созданные вне среды СУБД с помощью систем программирования, использующих средства доступа к БД, например, Delphi или Visual Studio, называют *внешними приложениями.* Для работы с БД зачастую достаточно средств СУБД и не нужно использовать приложения, создание которых обычно требует программирования. Приложения разрабатывают главным образом в случаях, когда требуется сделать работу пользователей более удобной или автоматизировать рутинные операции с БД.

2. База данных как интегрированный ресурс.

Исторически понятие базы данных возникло как альтернатива файловой организации данных при хранении с помощью ЭВМ (на магнитных носителях). Такая организация данных была характерна для прикладного программного обеспечения на начальном этапе распространения вычислительной техники. Файловая организация предполагала хранение данных в виде совокупности файлов, ориентированных на использование какой-либо одной прикладной программы, предназначенной для решения некоторой специфической задачи. Такая неуниверсальность в организации информации привела к большой избыточности (дублированию) при хранении, противоречивости данных, хранящихся в различных системах.

Понятие «база данных» возникло в результате стандартизации и унификации данных, универсально организованных и хранящихся с помощью ЭВМ с целью использования для многих приложений. При этом описание данных уже не скрыто в программах, а явным образом декларируется и хранится в самой базе. База данных может быть определена как структурная совокупность данных, поддерживаемых в актуальном состоянии (в соответствии объектам некоторой предметной области) и служащая для удовлетворения информационных потребностей многих пользователей. Для поддержания актуальности данных, хранящихся в базе, получения сведений об информационных запросах, перехода к данным и программам пользователей служат СУБД. Более подробно СУБД будут рассматриваться в следующей лекции.

Согласно ГОСТ 20886–85 «Организация данных в системах обработки данных. Термины и определения» БД – это «совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ».

В ГОСТ Р ИСО/МЭК ТО 10032–2007 «Эталонная модель управления данными» БД определяется как «совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных».

А.Б. Антопольский определяет БД как «объективную форму представления и организации совокупности данных (статей и др.), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны ЭВМ».

Согласно статье 1260 Гражданского кодекса Российской Федерации «базой данных является представленная в объективной форме совокупность самостоятельных материалов (статей, расчетов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ)»

В законе Республики Беларусь «Об информации, информатизации и защите информации» БД определяется как «совокупность структурированной и взаимосвязанной информации, организованной по определенным правилам на материальных носителях».

В БД информация должна быть организована так, чтобы обеспечить минимальную долю ее избыточности. Частичная избыточность информации необходима, но она должна быть минимизирована, так как чрезмерная избыточность данных влечет за собой ряд негативных последствий. Главные из них:

- увеличение объема информации, а значит, потребность в дополнительных ресурсах для хранения и обработки дополнительных объемов данных;
- появление ошибок при вводе дублирующей информации, нарушающих целостность БД и создающих противоречивые данные.

В БД должны храниться данные, логически связанные между собой. Для того, чтобы данные можно было связать между собой, и связать так, чтобы эти связи соответствовали реально существующим в данной предметной области, последнюю подвергают детальному анализу, выделяя сущности или объекты. Сущность или объект – это то, о чем необходимо хранить информацию. Сущности имеют некоторые характеристики, называемые

атрибутами, которые тоже необходимо сохранять в БД. Атрибуты по своей внутренней структуре могут быть простыми, а могут быть сложными. Простые атрибуты могут быть представлены простыми типами данных. Различного рода графические изображения, являющиеся атрибутами сущностей, – это пример сложного атрибута. Определив сущности и их атрибуты, необходимо перейти к выявлению связей, которые могут существовать между некоторыми сущностями. Связь – это то, что объединяет две или более сущностей. Связи между сущностями также являются частью данных, и они также должны храниться в базе данных.

Перечень важнейших *требований*, которым должны удовлетворять современные БД:

- высокое быстродействие (малое время отклика на запрос);
- простота обновления данных;
- независимость данных;
- интегрированность данных;
- целостность данных;
- динамичность данных и способность к расширению;
- гибкость и адаптивность структуры БД;
- совместное использование данных многими пользователями;
- безопасность данных – защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения.
- стандартизация построения и эксплуатации БД (фактически СУБД).
- адекватность отображения данных соответствующей предметной области;
- возможность поиска по многим ключам.

Важнейшими требованиями являются первые два противоречивых требования: повышение быстродействия требует упрощения структуры БД, что, в свою очередь, затрудняет процедуру обновления данных, увеличивает их избыточность.

Независимость данных – возможность изменения логической и физической структуры БД без изменения представлений пользователей. Независимость данных предполагает инвариантность к характеру хранения данных, программному обеспечению и техническим средствам. Она обеспечивает минимальные изменения структуры БД при изменениях стратегии доступа к данным и структуры самих исходных данных.

Безопасность данных включает их целостность и защиту. Целостность данных – устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей. Она предполагает:

– отсутствие неточно введенных данных или двух одинаковых записей об одном и том же факте;

– защиту от ошибок при обновлении БД;

– невозможность удаления (или каскадное удаление) связанных данных разных таблиц;

– отсутствие искажения данных при работе в многопользовательском режиме и в распределенных базах данных;

– сохранность данных при сбоях техники (восстановление данных).

Целостность обеспечивается триггерами целостности – специальными приложениями-программами, работающими при определенных условиях.

Защита данных от несанкционированного доступа предполагает ограничение доступа к конфиденциальным данным и может достигаться:

– введением системы паролей;

– получением разрешений от администратора базы данных (АБД);

– запретом от АБД на доступ к данным;

Стандартизация обеспечивает преемственность поколений СУБД, упрощает взаимодействие БД одного поколения СУБД с одинаковыми и различными моделями данных. Стандартизация (ANSI/SPARC) осуществлена в значительной степени в части интерфейса пользователя СУБД и языка SQL. Это позволило успешно решить задачу взаимодействия различных реляционных СУБД с помощью языка SQL.

На уровне информационных систем БД рассматривается как компонент, представляющий собой информационную модель предметной области. Здесь наиболее важной является проблема логической структуры БД.

При рассмотрении БД на уровне информационных ресурсов БД трактуется как элемент мировых информационных ресурсов. Основной характеристикой здесь является содержание БД, хотя и структуры данных также немаловажны.

Классификация БД по основным признакам приведена на рис. 1.1.

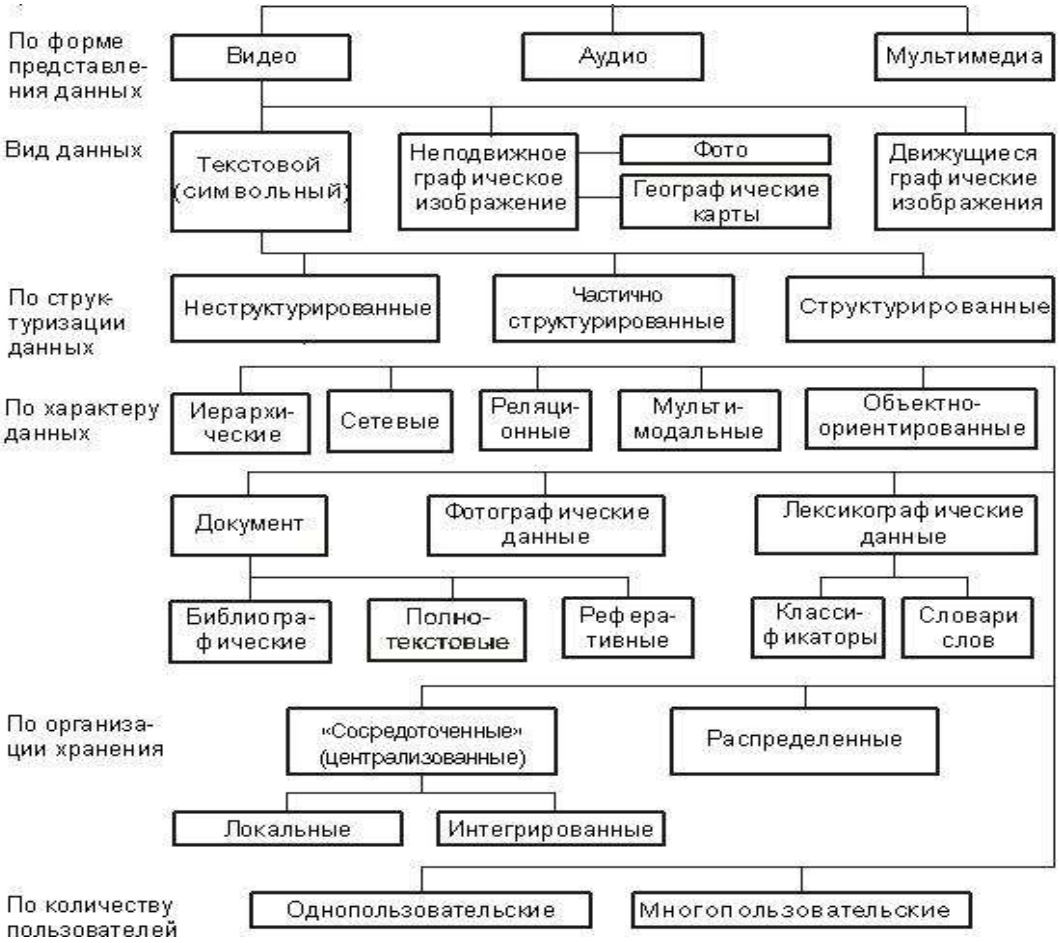

Рис. 1.1. Классификация баз данных по основным признакам

Администраторы БД также являются их специфическими пользователями. Обычно они обращаются к БД не за информацией о предметной области, а к метаинформации, а также используют ресурсы БД для выполнения своих функций. В зависимости от сложности и объема БД данных, от особенностей используемой СУБД служба администрации БД данных может различаться как по составу и квалификации специалистов, так и по количеству работающих в этой службе.

В составе группы администраторов БД можно выделить различные подгруппы в зависимости от выполняемых ими функций. Численность группы администрации, выполняемые ими функции будут в некоторой степени зависеть от масштабов БД, специфики хранимой в них информации, особенностей используемых программных средств и некоторых других факторов. В составе администрации БД должны быть системные аналитики, проектировщики структур данных и внешнего по отношению к банку данных информационного обеспечения, проектировщики технологических процессов

обработки данных, системные и прикладные программисты, операторы, специалисты по техническому обслуживанию.

Третьей категорией пользователей БД являются *разработчики и администраторы приложений*. Эта группа пользователей функционирует во время проектирования, создания и реорганизации БД. Администраторы приложений координируют работу разработчиков при разработке конкретного приложения или группы приложений, объединенных в функциональную подсистему. Разработчики конкретных приложений работают с той частью информации из БД, которая требуется для конкретного приложения.

Не для каждого БД могут быть выделены все категории пользователей. При разработке БД с использованием настольных СУБД администратор БД, администратор приложений и разработчик часто существовали в одном лице. Однако при построении современных сложных корпоративных БД данных могут существовать и группы администраторов приложений, и отделы разработчиков. Наиболее сложные обязанности возложены на группу администратора БД.

Администраторы БД выполняют большой круг разнообразных функции.

Связи администратора банка данных. В процессе своей деятельности администратор БД взаимодействует с другими категориями пользователей банка данных, а также и с «внешними» специалистами, не являющимися пользователями БД (рис. 1.2).

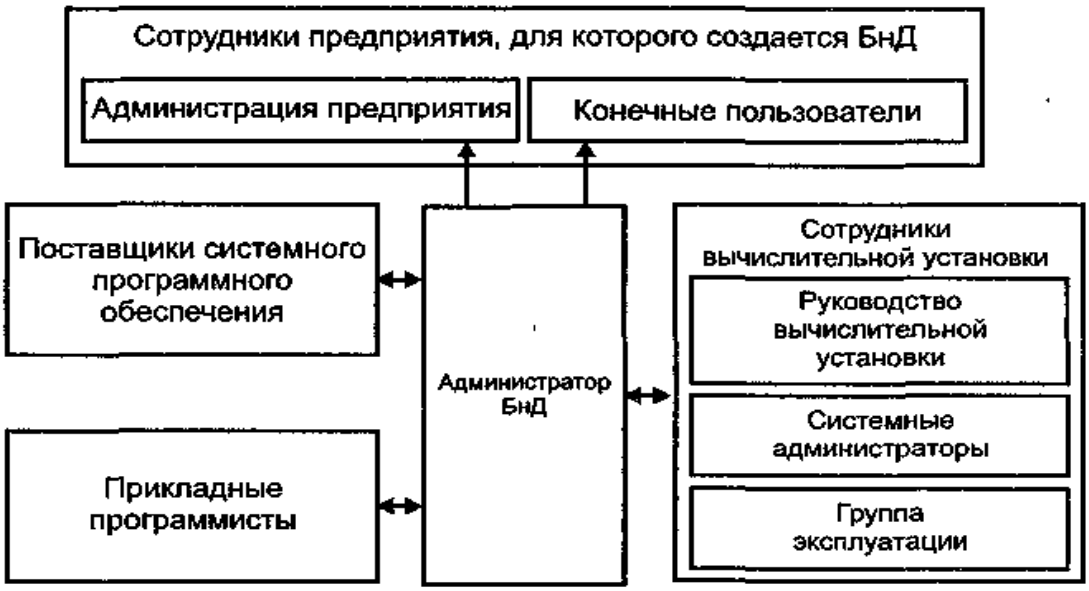

Рисунок 1.2. Взаимодействие администратора БД с другими категориями пользователей БД

Прежде всего, если БД создается для информационного обслуживания какого-либо предприятия или организации, необходимы контакты с администрацией этой организации. Внедрение БД приводит к большим изменениям не только системы обработки данных, но и всей системы управления организацией. Руководство организации должно быть ознакомлено с возможностями, предоставляемыми БД, проинформировано об их преимуществах и недостатках, а также о проблемах, вызываемых созданием и функционированием БД.

Поскольку БД является динамическим информационным отображением предметной области, то желательно, чтобы администратор БД, в свою очередь, был своевременно информирован о перспективах развития объекта, для которого создается БД.

Руководством организации и администратором БД должны быть согласованы цели, основные направления и сроки создания БД и его развития, очередность подключения пользователей.

Очень тесная связь у администратора БД на всех этапах жизненного цикла БД наблюдается с конечными пользователями. Это взаимодействие возникает на начальных стадиях проектирования системы, когда изучаются потребности пользователей, уточняются особенности предметной области, и постоянно поддерживается в процессе проектирования и функционирования системы.

Следует отметить, что в последнее время наблюдается активное перераспределение функций между конечными пользователями и администратором БД. Это, прежде всего, связано с развитием языковых и программных средств, ориентированных на конечных пользователей. К ним относятся простые и одновременно мощные языки запросов, а также средства автоматизации проектирования.

Если БД функционирует в составе какой-либо автоматизированной информационной системы, то администратор БД должен работать в контакте со специалистами по обработке данных в этой системе.

Администраторы БД взаимодействуют также с внешними по отношению к нему группами специалистов, и, прежде всего, поставщиками СУБД и прикладными программистами, администраторами других БД.

4. Основные подходы к определению понятия «система управления базами данных». Возможности и функции систем управления базами данных.

Определяя базы данных как среду хранения данных, отделенную от

процедур, следует подразумевать наличие специальных программных средств доступа к данным – систем управления базами данных (СУБД). СУБД определяется по-разному. Приведем некоторые определения:

1. Согласно ГОСТ 20886–85 «Организация данных в системах обработки данных. Термины и определения» СУБД – это «совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами».

2. Согласно ГОСТ Р ИСО/МЭК ТО 10032–2007 «Эталонная модель управления данными» СУБД – это «Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных».

3. С.Д. Кузнецов определяет СУБД как «комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать)».

За время развития БД и систем их управления было создано множество различных СУБД. Многие из них уже устарели технически и заменены новыми продуктами. Тем не менее, можно разделить все СУБД на две глобальные категории: *СУБД общего назначения и специализированные СУБД*. Как исходит из названия, при разработке СУБД общего назначения не делается акцент на какой-либо стороне работы СУБД, так же как программа не создается только для некоего узкого круга БД. СУБД общего назначения являются многофункциональными продуктами, которые уже в ходе создания либо эксплуатации БД могут быть модифицированы под нужды пользователя либо администратора системы. В связи с широкими возможностями СУБД общего назначения, часто это коммерческие продукты, поставляемые пользователям, которые в дальнейшем могут использовать в полной мере те специфические особенности данной СУБД, которые им нужны. Такие СУБД обладают средствами настройки на работу с конкретной БД. Использование СУБД общего назначения в качестве инструментального средства для создания автоматизированных информационных систем, позволяет существенно сокращать сроки разработки, экономить трудовые ресурсы. Этим СУБД присущи развитые функциональные возможности.

Специализированные СУБД создаются в редких случаях при невозможности или нецелесообразности использования СУБД общего назначения.

СУБД общего назначения – это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией БД.

Рынок программного обеспечения персональных компьютеров располагает большим числом разнообразных по своим функциональным возможностям коммерческих СУБД общего назначения, а также средствами их окружения практически для всех массовых моделей машин и для различных операционных систем.

Используемые в настоящее время СУБД обладают средствами обеспечения целостности данных и надежной безопасности, что дает возможность разработчикам гарантировать полную безопасность данных при меньших затратах сил на низкоуровневое программирование. Продукты, функционирующие в среде WINDOWS, выгодно отличаются удобством пользовательского интерфейса и встроенными средствами повышения производительности.

Приведем сравнение СУБД ACCESS, MySQL и Oracle по некоторым параметрам:

1. Объем памяти на жестком диске, необходимый для самой СУБД: ACCESS (OfficeXP) – 530 Мб, Oracle – > 1 Гб, для работы с MySQL + PHP через Интернет-сервер необходим только браузер (например Internet Explorer – 14,7 Мб), а для работы локально нужен еще сервер, поддерживающий MySQL и PHP (например Apache – 8 Мб).

2. Размер БД в формате, соответствующем каждой СУБД: ACCESS – 1,73 Мб, MySQL – 113 Кб, Oracle – размер определяется не содержанием самой базы, а задаваемым табличным пространством.

3. Оперативная память, используемая СУБД при работе с БД: ACCESS – 4528 Кб, сервер Apache + Internet Explorer – 28 612 Кб (из них Internet Explorer – 11 660 Кб).

4. Быстродействие: при работе локально разница между временем выполнения запроса в ACCESS и временем выполнения аналогичного запроса в MySQL (Internet Explorer – Apache – Internet Explorer) практически неощутима (десятые доли секунды); при работе же с MySQL через Internet скорость зависит от таких параметров, как трафик сети, удаленность и быстродействие сервера и пр.

5. Простота использования: ACCESS, как и все продукты из MS OFFICE, очень наглядна, содержит хорошую систему помощи и опции так называемых «мастеров» создания и заполнения. Это все в совокупности позволяет даже неопытному пользователю, не имеющему навыков работы с какими-либо СУБД, довольно быстро научиться создавать и управлять своими БД; MySQL – несмотря на то, что приходится прописывать все в ручную, особых трудностей не вызывает, особенно если пользователь обладает хоть какими-то навыками программирования и работы с БД. Oracle

– это СУБД несколько иного уровня, и поэтому требует изучения в течение большего, по сравнению с ACCESS и MySQL, времени.

Производительность СУБД оценивается:

– временем выполнения запросов;

– скоростью поиска информации в неиндексированных полях;

– временем выполнения операций импортирования базы данных из других форматов;

– скоростью создания индексов и выполнения таких массовых операций, как обновление, вставка, удаление данных;

– максимальным числом параллельных обращений к данным в многопользовательском режиме;

– временем генерации отчета.

На производительность СУБД оказывают влияние два фактора:

1. СУБД, которые следят за соблюдением целостности данных, несут дополнительную нагрузку, которую не испытывают другие программы.

2. Производительность собственных прикладных программ сильно зависит от правильного проектирования и построения БД.

В процессе реализации своих функций СУБД постоянно взаимодействует с БД и с другими прикладными программными продуктами пользователя, предназначенными для работы с данной БД и называемыми приложениями.

Для того, чтобы СУБД успешно справлялась со своими задачами, она должна обладать определенными возможностями.

Обобщенная характеристика *возможностей* современных СУБД:

1. СУБД включает язык определения данных, с помощью которого можно определить БД, ее структуру, типы данных, а также средства задания ограничений для хранимой информации. В многопользовательском варианте СУБД этот язык позволяет формировать представления как некоторое подмножество БД, с поддержкой которых пользователь может создавать свой взгляд на хранимые данные, обеспечивать дополнительный уровень безопасности данных и многое другое.

2. СУБД позволяет вставлять, удалять, обновлять и извлекать информацию из БД посредством языка управления данными.

3. Большинство СУБД могут работать на компьютерах с разной архитектурой и под разными операционными системами, причем на работу пользователя при доступе к данным практически тип платформы влияния не оказывает.

4. Многопользовательские СУБД имеют достаточно развитые средства администрирования БД.

5. СУБД предоставляет контролируемый доступ к БД с помощью:

– системы обеспечения безопасности, предотвращающей несанкционированный доступ к информации БД;

– системы поддержки целостности БД, обеспечивающей непротиворечивое состояние хранимых данных;

– системы управления параллельной работой приложений, контролирующей процессы их совместного доступа к БД;

– системы восстановления, позволяющей восстановить БД до предыдущего непротиворечивого состояния, нарушенного в результате аппаратного или программного обеспечения.

Функции СУБД:

1. Управление данными во внешней памяти. Данная функция предоставляет пользователям возможности выполнения самых основных операций, которые осуществляются с данными, – сохранение, извлечение и обновление информации. Она включает в себя обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным.

2. Управление транзакциями. Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция представляет собой набор действий, выполняемых с целью доступа или изменения содержимого БД. Примерами простых транзакций может служить добавление, обновление или удаление в БД сведений о некоем объекте. Сложная же транзакция образуется в том случае, когда в БД требуется внести сразу несколько изменений. Инициализация транзакции может быть вызвана отдельным пользователем или прикладной программой.

3. Восстановление БД. 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: 1) мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания); 2) жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. 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.

4. Поддержка языков БД. Для работы с БД используются специальные языки, называемые языками БД. 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 БД. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language – язык структурированных запросов). Язык SQL позволяет определять схему реляционной БД и манипулировать данными.

5. Поддержка словаря данных. Наличие интегрированного системного каталога с данными о схемах, пользователях, приложениях и т. д. Системный каталог, который еще называют словарем данных, является, таким образом, хранилищем информации, описывающей данные в БД. Предполагается, что каталог доступен как пользователям, так и функциям СУБД. Обычно в словаре данных содержится следующая информация: имена, типы и размеры элементов данных; имена связей; накладываемые на данные ограничения поддержки целостности; имена пользователей, которым предоставлено право доступа к данным; внешняя, концептуальная и внутренняя схемы и отображения между ними; статистические данные, например частота транзакций и счетчики обращений к объектам БД.

6. Управление параллельным доступом. Одна из основных целей создания и использования СУБД заключается в том, чтобы множество пользователей могло осуществлять параллельный доступ к совместно обрабатываемым данным. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных, поскольку в этом случае они не могут помешать друг другу. Однако когда два или больше пользователей одновременно получают доступ к БД, конфликт с нежелательными последствиями легко может возникнуть, например, если хотя бы один из них попытается обновить данные. СУБД должна гарантировать, что при одновременном доступе к БД многих пользователей подобных конфликтов не произойдет.

7. Управление буферами оперативной памяти. СУБД обычно работают с БД значительного размера. 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. В развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

8. Контроль доступа к данным. СУБД должна иметь механизм, гарантирующий возможность доступа к БД только санкционированных пользователей и защищающий ее от любого несанкционированного доступа. В современных СУБД поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных: избирательный подход или обязательный подход. В большинстве современных систем предусматривается избирательный подход, при котором некий пользователь обладает различными правами при работе с разными объектами. Значительно реже применяется альтернативный, обязательный подход, где каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

9. Поддержка целостности данных. Термин «Целостность» используется для описания корректности и непротиворечивости хранимых в БД данных. Реализация поддержки целостности данных предполагает, что СУБД должна содержать сведения о тех правилах, которые нельзя нарушать при работе с данными, и обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам.

СУБД обладают как преимуществами по сравнению с файловыми системами, так и недостатками.

*Преимущества СУБД**:***

1. Контроль за избыточностью данных. При использовании БД предпринимается попытка исключить избыточность данных за счет интеграции файлов, чтобы избежать хранения нескольких копий одного и того же элемента информации. Однако полностью избыточность информации в БД не исключается, а лишь контролируется ее степень.

2. Непротиворечивость данных. Если элемент данных хранится в БД только в одном экземпляре, то для изменения его значения потребуется выполнить только одну операцию обновления. Если элемент данных хранится в БД в нескольких экземплярах, то такая система сможет следить за тем, чтобы копии не противоречили друг другу.

3. Совместное использование данных. БД принадлежит группам пользователей или всей организации в целом и может совместно использоваться всеми зарегистрированными пользователями. При этом можно создавать новые приложения на основе уже существующей в БД информации и добавлять в нее только те данные, которые в настоящий момент еще не хранятся в ней, а не определять вновь требования ко всем данным, необходимым новому приложению.

4. Поддержка целостности данных. Целостность БД означает

корректность и непротиворечивость хранимых в ней данных. Целостность обычно описывается с помощью ограничений, т.е. правил поддержки непротиворечивости, которые не должны нарушаться в БД. Ограничения можно применять к элементам данных внутри одной записи или к связям между записями.

5. Повышенная безопасность. Безопасность БД заключается в защите данных от несанкционированного доступа со стороны пользователей. Без привлечения соответствующих мер безопасности интегрированные данные становятся уязвимыми. СУБД приводит в действие систему безопасности БД. Система обеспечения безопасности может быть выражена в форме учетных имен и паролей для идентификации пользователей, которые зарегистрированы в БД.

6. Повышение эффективности с ростом масштабов системы. Комбинируя все рабочие данные в одной БД и создавая набор приложений, которые работают с одним источником данных, можно добиться существенной экономии средств.

7. Повышение доступности данных и их готовности к работе. Данные в результате интеграции становятся непосредственно доступными конечным пользователям. Во многих СУБД предусмотрены языки запросов или инструменты для создания отчетов, которые позволяют пользователям задавать непредусмотренные заранее вопросы и почти немедленно получать требуемую информацию на своих терминалах, не прибегая к помощи программиста.

8. Улучшение показателей производительности. На базовом уровне СУБД обеспечивает все низкоуровневые процедуры работы с файлами, которую обычно выполняют приложения. Во многих СУБД также предусмотрена среда разработки четвертого поколения с инструментами, упрощающими создание приложений БД. Результатом является повышение производительности работы программистов и сокращение времени разработки новых приложений.

Недостатки СУБД:

1. Сложность. Обеспечение функциональности, которой должна обладать каждая СУБД, сопровождается ее значительным усложнением. Чтобы воспользоваться всеми преимуществами СУБД, проектировщики и разработчики БД, администраторы БД, а также конечные пользователи должны хорошо понимать функциональные возможности СУБД.

2. Размер программного обеспечения. Сложность и широта функциональных возможностей приводит к тому, что СУБД становится программным продуктом, который может занимать много места на диске и

требовать большого объема оперативной памяти для эффективной работы.

3. Стоимость СУБД. В зависимости от имеющейся вычислительной среды и требуемых функциональных возможностей, стоимость СУБД может варьироваться в очень широких пределах – от нескольких сот до нескольких сот тысяч долларов. Кроме того, следует учесть ежегодные расходы на сопровождение системы, которые составляют некоторый процент от ее общей стоимости.

4. Дополнительные затраты на аппаратное обеспечение. Для удовлетворения требований, предъявляемых к дисковым накопителям со стороны СУБД и базы данных, может понадобиться приобрести дополнительные устройства хранения информации. Более того, для достижения требуемой производительности может понадобиться более мощный компьютер, который, возможно, будет работать только с СУБД.

5. Затраты на преобразование приложений. В некоторых ситуациях стоимость СУБД и дополнительного аппаратного обеспечения может оказаться несущественной по сравнению со стоимостью преобразования существующих приложений для работы с новой СУБД и новым аппаратным обеспечением. Эти затраты также включают стоимость подготовки персонала для работы с новой системой, а также оплату услуг специалистов, которые будут оказывать помощь в преобразовании и запуске новой системы.

6. Производительность. СУБД предназначены для решения общих задач и обслуживания нескольких приложений, а не одного из них. В результате многие приложения в новой среде будут работать не так быстро, как прежде.

7. Серьезные последствия при выходе системы из строя. Централизация ресурсов повышает уязвимость системы. Поскольку работа всех пользователей и приложений зависит от готовности к работе СУБД, выход из строя одного из ее компонентов может привести к полному прекращению всей работы СУБД.

5. Классификация систем управления базами данных.

СУБД классифицируют по различным признакам. Важнейшим классификационным признаком СУБД является *тип модели данных*, поддерживаемый СУБД. По этому признаку СУБД делятся на:

– иерархические;
– сетевые;
– реляционные;
– постреляционные;
– многомерные;

– объектно-ориентированные.

Иерархические СУБД поддерживают иерархическую модель БД, которая основана на древовидной структуре хранения информации. В этом смысле иерархические БД напоминают файловую систему компьютера.

В сетевой структуре каждый элемент может быть связан с любым другим элементом. Сетевые БД подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию.

Реляционная СУБД – это СУБД, управляющая реляционными БД. В постреляционной модели СУБД сняты ограничения неделимости данных, хранящихся в записях таблиц, т.е. допускается наличие многозначных полей В многомерной модели СУБД данные рассматриваются как кубы, которые являются обобщением электронных таблиц на любое число измерений.

Объектно-ориентированная СУБД реализует объектно-ориентированный подход. Эта система управления обрабатывает данные как абстрактные объекты, наделенные свойствами, в виде неструктурированных данных, и использующие методы взаимодействия с другими объектами окружающего мира. Объектно-ориентированные СУБД: Itaska (IBEX), Jasmine (Computer Associates), Matisse (ODB), Object Store (ODI), Ontos (Ontos), O2 (Ardent Software), Poet (Poet Software), Versant (Versant Technologies).

По *архитектуре организации хранения данных* СУБД делятся на:

– централизованные СУБД (все части СУБД размещаются на одном компьютере): dBase-подобные системы, DB2, Paradox, Access, FoxPro, Oracle, MS SQL Server;

– распределенные СУБД (части СУБД могут размещаться на двух и более компьютерах): Informix On-Line фирмы Informix Software; Ingres Intelligent Database фирмы Ingres Corp; Oracle (version 7) фирмы Oracle Corp; Sybase System 10 фирмы Sybase Inc.

СУБД классифицируются по *способу доступа к БД* на: файл-серверные, клиент-серверные и встраиваемые СУБД.

В 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 нагрузка на процессор файлового сервера. Недостатки файл-серверных СУБД: потенциально высокая загрузка локальной сети; затруднённость или невозможность централизованного управления; затруднённость или

невозможность обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность. Применяются чаще всего в системах с низкой интенсивностью обработки данных и низкими нагрузками на БД.

На данный момент файл-серверная технология считается устаревшей, а её использование в крупных информационных системах – недостатком.

Примеры файл-серверных СУБД: Microsoft Access, Paradox, dBase, FoxPro, Visual FoxPro.

Клиент-серверная СУБД позволяет обмениваться клиенту и серверу минимально необходимыми объёмами информации. При этом основная вычислительная нагрузка ложится на сервер. Клиент может выполнять функции предварительной обработки перед передачей информации серверу, но в основном его функции заключаются в организации доступа пользователя к серверу.

В большинстве случаев клиент-серверная СУБД гораздо менее требовательна к пропускной способности компьютерной сети, чем файл-серверная СУБД, особенно при выполнении операции поиска в базе данных по заданным пользователем параметрам, т.к. для поиска нет необходимости получать на клиент весь массив данных: клиент передаёт параметры запроса серверу, а сервер производит поиск по полученному запросу в локальной базе данных. Результат выполнения запроса, который обычно на несколько порядков меньше по объёму, чем весь массив данных, возвращается клиенту, который обеспечивает отображение результата пользователю.

Примеры клиент-серверных СУБД: Oracle, MySQL, MS SQL Server, IBM DB2, Informix, ЛИНТЕР.

Встраиваемые СУБД являются составной частью некоторого программного продукта и не требуют процедуры самостоятельной установки. Встраиваемая СУБД предназначена ля локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически такая СУБД чаще всего реализована в виде подключаемой библиотеки. Доступ к данным со стороны приложения может происходить через SQL, либо через специальные программные интерфейсы.

Примерами встраиваемых СУБД являются OpenEdge, SQLite, BerkeleyDB, Microsoft SQL Server Compact и др.

По *степени универсальности* различаются два класса СУБД – системы общего назначения и специализированные системы. СУБД общего назначения не ориентированы на какую-либо конкретную предметную область или на информационные потребности конкретной группы пользователей. СУБД общего назначения обладает

средствами настройки на работу с конкретной БД в условиях конкретного применения. В некоторых ситуациях СУБД общего назначения не позволяют добиться требуемых проектных и эксплуатационных характеристик (производительность, занимаемый объем памяти и прочее). Тем не менее, создание специализированных СУБД весьма трудоемкий процесс и для того, чтобы его реализовать, нужны очень веские основания.

По *языкам общения* СУБД делятся на открытые, замкнутые и смешанные. Открытые системы – это системы, в которых для обращения к БД используются универсальные языки программирования. Замкнутые системы имеют собственные языки общения с пользователями БД. Открытые системы в настоящее время используются редко.

По *выполняемым функциям* СУБД делятся на информационные и операционные. Информационные СУБД позволяют организовать хранение информации и доступ к ней. Операционные СУБД выполняют достаточно сложную обработку, например, автоматически позволяют получать агрегированные показатели, не хранящиеся непосредственно в БД, могут изменять алгоритмы обработки и т.д.

По *сфере возможного применения* различают универсальные и специализированные СУБД.

Системы управления базами данных поддерживают разные типы данных. *Набор типов данных,* допустимых в разных СУБД, различен. Ряд СУБД позволяет разработчику (прикладному программисту или администратору БД) добавлять новые типы данных и новые операции над этими данными. Такие системы называются расширяемыми системами баз данных (РСБД). Дальнейшим развитием концепции РСБД являются объектно-ориентированные системы баз данных, обладающие достаточно мощными возможностями, чтобы непосредственно моделировать сложные объекты.

По «*мощности*» СУБД делятся на «*настольные»* и «*корпоративные*». Характерными чертами «настольных» СУБД являются сравнительно невысокие требования к техническим средствам, ориентация на конечного пользователя, низкая стоимость. «Корпоративные» СУБД обеспечивают работу в распределенной среде, высокую производительность, поддержку коллективной работы при проектировании систем, имеют развитые средства администрирования и более широкие возможности поддержания целостности. В связи с этим очевидно, что корпоративные СУБД сложны, дороги, требуют значительных вычислительных ресурсов.

Наиболее известными из «корпоративных» СУБД являются Oracle, Informix, Sybase, MS SQL Server, Progress, DB2 и некоторые другие.

По *ориентации на преобладающую категорию пользователей* можно выделить СУБД для разработчиков и для конечных пользователей. Системы, относящиеся к первому классу, должны иметь качественные компиляторы и позволять создавать «отчуждаемые» программные продукты, обладать развитыми средствами отладки и обладать другими возможностями, позволяющими создавать эффективные сложные системы. Основными требованиями, предъявляемыми к системам, ориентированным на конечного пользователя, являются: удобство интерфейса, высокий уровень языковых средств, наличие интеллектуальных модулей подсказок, повышенная защита от непреднамеренных ошибок и т.п.

По *характеру использования* СУБД делят на персональные и многопользовательские. Персональные СУБД обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. К персональным СУБД, например, относятся Visual FoxPro, Paradox, Clipper, dBase, Access и др. Многопользовательские СУБД включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (с разными типами компьютеров и разными операционными системами). К многопользовательским СУБД относятся, например, СУБД Oracle и Informix.

По *своей архитектуре* СУБД делятся на одно-, двух- и трехзвенные (рис. 1.3). В однозвенной архитектуре используется единственное звено (клиент), обеспечивающее необходимую логику управления данными и их визуализацию. В двухзвенной архитектуре значительную часть логики управления данными реализует сервер баз данных (сервер БД), в то время как клиентское звено в основном занято отображением данных в удобном для пользователя виде. В трехзвенных СУБД используется промежуточное звено – сервер приложений. Сервер приложений позволяет полностью избавить клиента от функций по управлению данными и обеспечению связи с сервером БД.

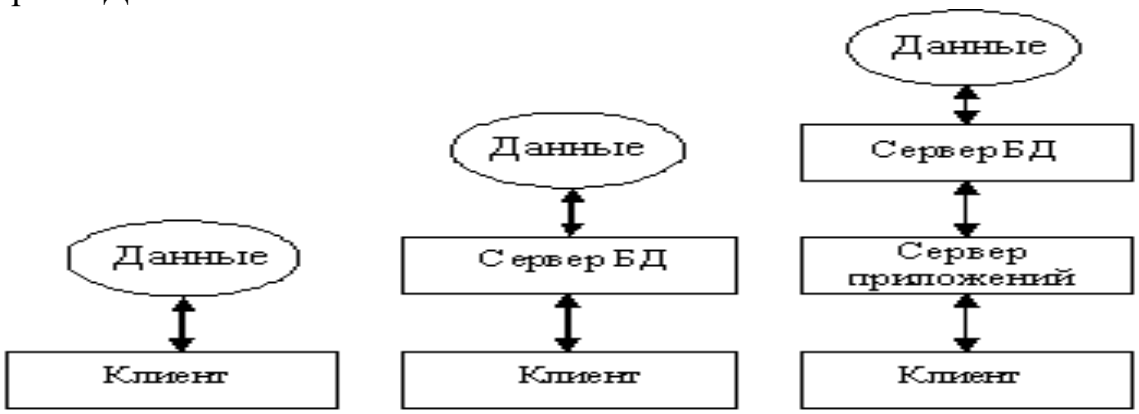

Рисунок 1.3. Классификация СУБД по архитектуре

В качестве классификационных признаков СУБД можно также рассматривать:

– среду функционирования СУБД (платформу) – класс компьютеров и операционных систем, под управлением которых работает СУБД;

– наличие диалоговых и инструментальных средств конструирования объектов БД;

– возможности встроенного языка СУБД;

– использование OLE-технологии – взаимодействие объектов БД с объектами других приложений: табличных и текстовых процессоров, графических редакторов и др.;

– обеспечение интеграции данных из баз, созданных в разных СУБД.

Тема 2. Реляционные базы данных. Базисные средства манипулирования реляционными данными

План:

1. Реляционная модель данных: основные понятия, форма представления, достоинства и недостатки.
2. Базисные средства манипулирования реляционными данными.

1. Реляционная модель данных: основные понятия, форма представления, достоинства и недостатки.

Реляционная модель данных в настоящее время используется в большинстве коммерческих СУБД.

Сотрудник фирмы IBM доктор Э.Ф. Кодд, математик по образованию, в 1970 году предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как *отношение* (relation).

Положив *теорию отношений* в основу реляционной модели, Э.Ф Кодд обосновал реляционную замкнутость отношений и ряда некоторых специальных операций, которые применяются сразу ко всему множеству строк отношения, а не к отдельной строке. Указанная реляционная замкнутость означает, что результатом выполнения операций над отношениями является также отношение, над которым в свою очередь можно осуществить некоторую операцию. Из этого следует, что в данной модели можно оперировать реляционными выражениями, а не только отдельными

операндами в виде простых имен таблиц.

Предложенная Э.Ф. Коддом реляционная модель позволила пользователю не заботиться о физической структуре данных и не интересоваться ею. Кодд ввел два языка манипулирования данными – *реляционная алгебра* и *реляционное исчисление*, которые предлагали более эффективные средства доступа к данным и их обработки и в настоящее время являются основой коммерческих СУБД.

Представление данных в виде множественной совокупности таблиц позволило избежать многих недостатков ранних СУБД и создать системы с упрощенным интерфейсным управлением. Многие серверы баз данных основаны на принципах работы, связанных с реляционной моделью.

К числу достоинств *реляционного подхода* можно отнести:

– наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;

– наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации БД;

– возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

Популяризатор идей Э.Ф. Кодда Кристофер Дейт, который воспроизводит ее (с различными уточнениями) практически во всех своих книгах (например, К. Дейт. Введение в системы баз данных. – 6-е изд. – М.; СПб.: Вильямс, 2000), в реляционной модели выделяет три части, описывающие разные аспекты реляционного подхода:

1) структурная часть;

2) манипуляционная часть;

3) целостная часть.

Структурную часть реляционной модели составляют следующие компоненты:

– *отношения неопределенного порядка*, концептуально представленные таблицами;

– *атрибуты* – атомарные данные, характеризующие отношения и представленные столбцами таблицы;

– *домены* – множества допустимых значений атрибутов;

– *кортежи* – совокупности значений всех атрибутов отношения, взятых

по одному для каждого атрибута, представленные строками таблицы;

– *возможные ключи* – множество атрибутов, однозначно определяющее кортеж в отношении;

– *первичные ключи* – для каждого отношения это один из возможных ключей.

Манипуляционная часть состоит из операторов выбора, проекции, соединения и т.п., которые преобразуют отношения в отношения.

Целостная часть состоит из правил: правил целостности на уровне *объектов* и правил целостности на уровне *ссылок*. В любой реализации можно определить ограничения, которые определят меньшее множество возможных непротиворечивых значений.

Логическая структура данных представляется набором связанных таблиц. Модель поддерживает связи «один к одному» и «один ко многим». Связь «многие ко многим» реализуется с помощью декомпозиции, то есть легко разрешается введением дополнительного (производного) отношения, которое является промежуточным межу исходными отношениями и соединено с ними двумя связями «один ко многим».

Основные элементы реляционной модели данных представлены в рисунке 1.4.

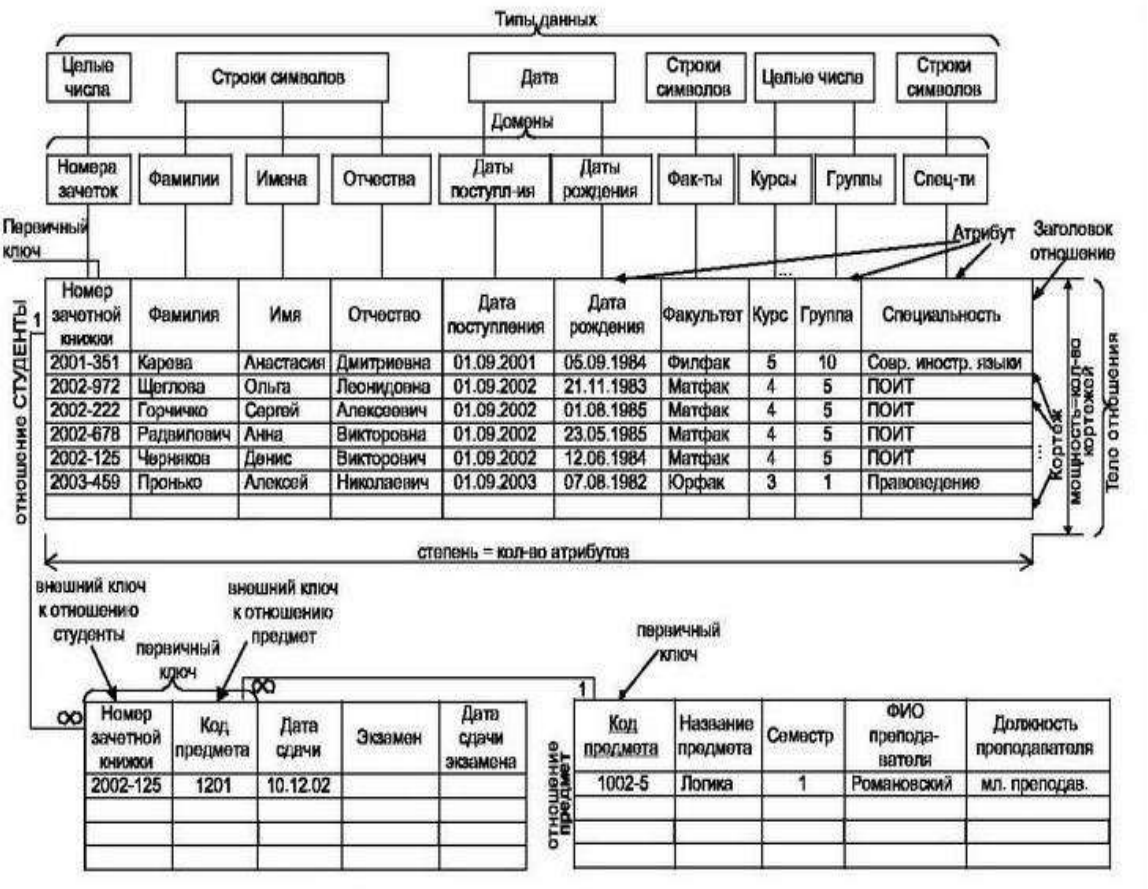

Рисунок 1.4. Элементы реляционной модели данных

Краткие описания основных понятий реляционной модели приведены в таблице 1.1.

Таблица 1.1. Основные понятия реляционной модели

Термин реляционной модели	Описание
База данных (БД)	Набор таблиц и других объектов, необходимых для абстрактного представления выбранной предметной области
Схема БД	Набор заголовков таблиц, взаимосвязанных друг с другом
Отношение	Таблица – совокупность объектов реального мира, которые характеризуются общими свойствами и характеристиками (поля таблицы)
Заголовок отношения	Заголовок таблицы – названия полей (столбцов) таблицы
Тело отношения	Тело таблицы – совокупность значений для всех объектов реального мира, которая представима в виде записей таблицы (строки таблицы)
Схема отношения	Строка заголовков столбцов таблицы («шапка» таблицы)
Атрибут отношения	Наименование столбца таблицы (поле таблицы)
Кортеж отношения	Строка таблицы (запись) – однозначное представление объекта реального мира, созданное с использованием значений полей таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значение поля в записи (кортеже)
Первичный ключ	Один или несколько (в случае составного ключа) атрибутов, которые единственным образом определяют значение кортежа (значение строки таблицы)
Внешний ключ	Атрибут таблицы, значения которого соответствуют значениям первичного ключа в другой связанной (родительской, первичной) таблице. Внешний ключ может состоять как из одного, так и из

	нескольких атрибутов (составной внешний ключ). В случае если число атрибутов внешнего ключа меньше, чем количество атрибутов соответствующего первичного ключа, то он принято называть усеченным (частичным) внешним ключом
Степень (арность) отношения	Количество столбцов таблицы
Мощность отношения	Количество строк (кортежей) таблицы
Экземпляр отношения	Множество записей (кортежей) для данной таблицы (отношения). С течением времени экземпляр может изменяться. Поскольку обычная БД в текущий момент времени работает только с одной версией отношения, то такой экземпляр отношения принято называть текущим
Тип данных	Тип значений элементов таблицы
Базовое отношение	Отношение, содержащие один или несколько столбцов, характеризующих свойства объекта, а также первичный ключ
Производное отношение	Не является базовым отношением, т.е. не характеризует свойства объекта и используется для обеспечения связей между другими таблицами, может не содержать первичного ключа. В случае если первичный ключ задан, то он состоит из внешних ключей, связанных с первичными ключами базового отношения
Связь	Устанавливает взаимосвязь между совпадающими значениями в ключевых полях – первичным ключом одной таблицы и внешним ключом другой
Связь «один-к-одному» (1:1)	При использовании этого вида связи запись в одной таблице может иметь не более одной связанной с нею записи в другой таблице. В обеих таблицах ключевые поля должны быть первичными. Используется для разделения таблиц с многочисленными полями или по требованию защиты данных
Связь «один-ко-многим» (1:M)	При использовании этого вида связи каждой записи одной таблицы может соответствовать несколько записей второй, но каждой записи второй таблицы соответствует лишь одна запись первой таблицы. В первой таблицы обязательно должен быть задан первичный ключ, во второй – внешний

Связь «многие-ко-многим» (N:M)	При данном типе связи одной записи в первой таблице может соответствовать несколько записей второй таблицы, но и одной записи второй таблицы может соответствовать несколько записей первой. Уникальность ключей для таких таблиц не требуется. В процессе проектирования схемы БД такие связи преобразуют. Для этого крайне важно ввести вспомогательное отношение, позволяющее заменить связь «многие-ко-многим» на две связи типа «один-ко-многим»

Э.Ф. Кодд приводит критерии, по которым СУБД можно отнести к реляционным. Эти критерии следующие:

– СУБД должна поддерживать таблицы без видимых пользователю навигационных связей;

– язык манипулирования данными должен обеспечивать минимальную возможность реляционной обработки, то есть включать операторы выбора, проекции и соединения.

Если СУБД не удовлетворяет второму критерию, ее называют *табличной (полуреляционной)*. Реляционные СУБД с минимальной возможностью реляционной обработки называются *минимально реляционными*, если же в СУБД в полной мере реализованы две последние составляющие реляционной модели,– это *полностью реляционные* СУБД. СУБД, реализующие полный набор реляционных операторов, называются *реляционно полными*.

Пусть имеется отношение r. Схемой отношения r называется конечное множество имен атрибутов R = {A1, A2, ... An}. Заголовки столбцов отношения содержат имена его атрибутов и, следовательно, все вместе отражают его схему. Схема отношения ПРЕПОДАВАТЕЛЬ может быть представлена следующим образом:

{Таб_ном_преп, Фамилия, Должность}

Или

ПРЕПОДАВАТЕЛЬ
Таб_ном_преп
Фамилия
Должность

Тогда заголовок отношения ПРЕПОДАВАТЕЛЬ примет вид

Таб_ном_преп	Фамилия	Должность

Отношение строится с учетом ряда факторов. Каждому имени атрибута Ai, 1 >= i <= n ставится в соответствие множество допустимых для соответствующего столбца значений. Это множество Di, называется доменом данного имени атрибута. Каждая строка отношения является множеством значений, взятых по одному из домена каждого имени атрибута. Домены являются произвольными непустыми конечными или счетными множествами и образуют множество: D = D1 ∪ D2 ∪ ... ∪ Dn. Отношение r со схемой R – это конечное множество отображений {t1, t2, ... tp) из R в D. Причем каждое отображение t∈ r должно удовлетворять следующему ограничению: t(Ai) принадлежит Di где 1 <= i <= n. Эти отображения называются кортежами. Каждый кортеж отношения отображает экземпляр сущности, а атрибут отношения отображает атрибут сущности. Множество кортежей называется телом отношения. Тело отношения отражает состояние сущности, поэтому во времени оно постоянно меняется. Тело отношения характеризуется кардинальным числом, которое равно количеству содержащихся в нем кортежей. Одной из главных характеристик отношения является его степень. Степень отношения определяется количеством атрибутов, которое в нем присутствует. Эта характеристика отношения имеет еще названия: ранг и арность. Отношение с одним атрибутом называется унарным, с двумя атрибутами – бинарным, с тремя – тернарным, с n атрибутами n-арным. Определение степени отношения осуществляется по заголовку отношения.

Все названные характеристики отношения обозначены на рисунке 1.5.

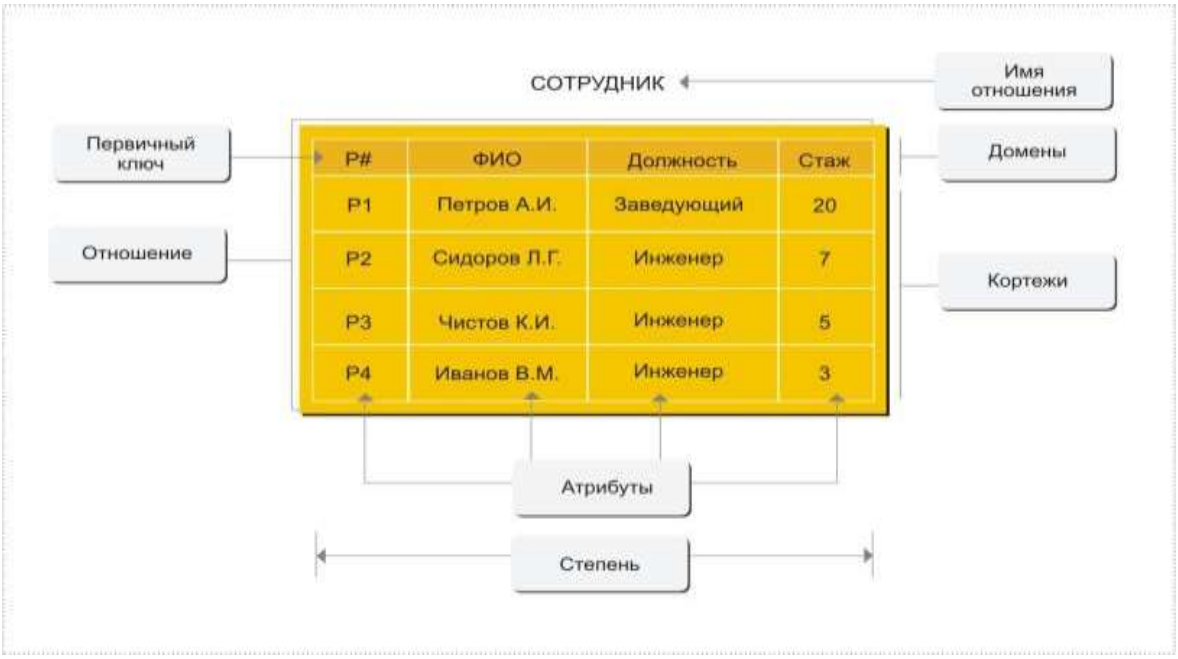

Рисунок 1.5. Отношение СОТРУДНИК

Отношение по структуре подобно таблице, но таблице, обладающей определенными свойствами. Свойства отношения следующие:

1. Отношение имеет имя, которое отличается от имен всех других отношений.

2. Отношение представляется в виде табличной структуры.

3. Каждый атрибут имеет уникальное имя, его значения берутся из одного и того же домена.

4. Каждый компонент кортежа является простым, атомарным значением, не состоящим из группы значений.

5. Упорядочение атрибутов теоретически несущественно, однако оно может влиять на эффективность доступа к кортежам.

6. Все строки (кортежи) должны быть различны.

7. Теоретически порядок следования кортежей не имеет значения.

В отношении могут существовать несколько одиночных или составных атрибутов, которые однозначно идентифицируют кортеж отношения. Это – потенциальные ключи.

Говорят, что множество атрибутов К = {Ai, Aj, ..., Ak} отношения r является потенциальным ключом r тогда и только тогда, когда удовлетворяются два независимых от времени условия:

– уникальность: в произвольный заданный момент времени никакие два различных кортежа r не имеют одного и того же значения для Ai, Aj, ..., Ак;

– минимальность: ни один из атрибутов Ai, Aj, ... Ак не может быть исключен из К без нарушения уникальности.

Отношение может иметь несколько потенциальных ключей. Ключ, содержащий два и более атрибута, называется *составным ключом.* Каждое отношение обладает хотя бы одним возможным ключом, поскольку в отношении не может быть одинаковых кортежей, а это значит, что, по меньшей мере, комбинация всех его атрибутов удовлетворяет условию уникальности. Потенциальные ключи, позволяя гарантированно выделить точно один кортеж, обеспечивают основной механизм адресации на уровне кортежей реляционной модели.

Один из возможных ключей (выбранный произвольным образом) принимается за его *первичный ключ*. Обычно первичным ключом назначается тот возможный ключ, которым проще всего пользоваться при повседневной работе. Остальные возможные ключи, если они есть, называются *альтернативными ключами.* Для индикации связи между отношениями используются *внешние ключи.*

Внешний ключ – это набор атрибутов одного отношения, являющийся

потенциальным ключом другого отношения.

Благодаря наличию связок между потенциальными и внешними ключами обеспечивается взаимосвязь кортежей определенных отношений.

Отношение, содержащее внешний ключ, называется *дочерним или ссылающимся отношением*. А отношение, содержащее связанный с внешним ключом потенциальный ключ, — *родительским или целевым отношением.*

Отношения не могут рассматриваться как статические объекты, так как они предназначены для отражения некоторой части реального мира, а эта часть реального мира может изменяться во времени. Поэтому и отношения изменяются во времени: кортежи могут добавляться, удаляться или модифицироваться. Тем не менее, предполагается, что сама схема отношения инвариантна во времени. Отношение должно восприниматься как множество возможных состояний, которые может принимать отношение.

Пример

Пусть рассматривается концептуальная модель, приведенная на рисунке 1.6. Она относится к предметной области, которую можно назвать «Преподавательская деятельность». Данная модель содержит две сущности: ЛЕКТОР и ПРЕДМЕТ, между которыми установлена связь ЧИТАЕТ типа «многие ко многим». Характеристики сущностей представлены изображенными на рисунке атрибутами. Связь ЧИТАЕТ не имеет собственных атрибутов.

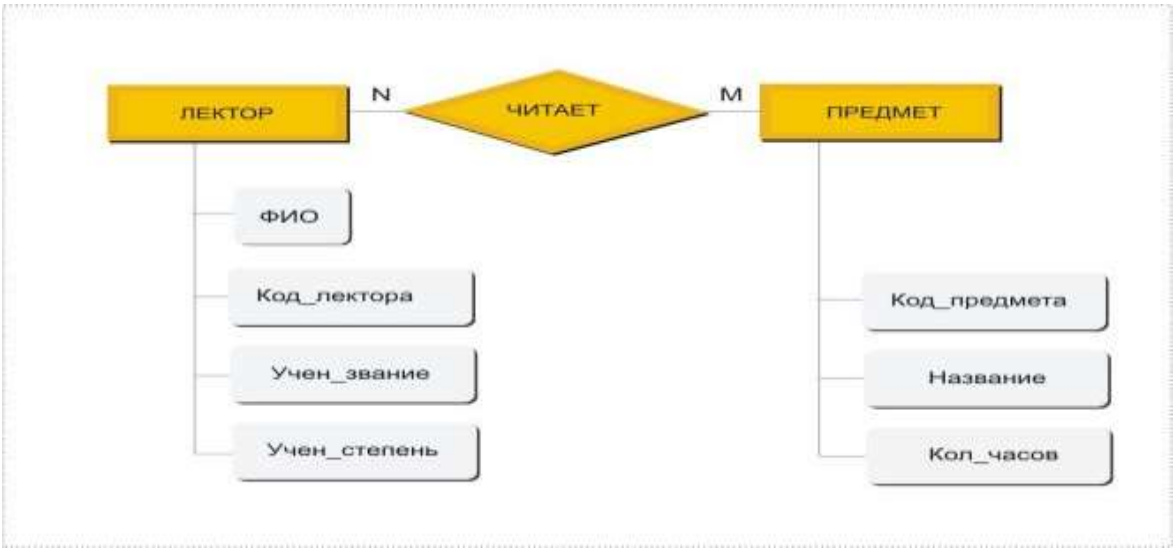

Рисунок 1.6. Концептуальная модель

Реляционная схема, соответствующая указанной концептуальной модели включает в себя три отношения: ЛЕКТОР, ПРЕДМЕТ, ЧИТАЕТ. Схемы отношений и связи между ними изображены на рисунке 1.7.

Рисунок 1.7. Реляционная схема базы данных

На рисунке 1.8 даны соответствующие отношения, заполненные кортежами. Эти отношения имеют следующие характеристики:

ЛЕКТОР – 4-арное отношение с первичным ключом Код_лек с кардинальным числом, равным четырем; атрибуты определены на следующих доменах: Код_лек – {целые: 1...4}, ФИО – {возможные фамилии и инициалы}, Уч_степень – {к.т.н., д.т.н., нет степени}, Уч_звание – {Доцент, Профессор, Нет_звания};

ПРЕДМЕТ – тернарное отношение с первичным ключом Код_предм. с кардинальным числом, равным шести; атрибуты определены на следующих доменах: Код_предм – {символьный}. Назв_предм – {Информатика, Программирование, Физика, ООП, Базы данных, Базы данных}, Кол_во_час — {целые: 54, 102, 36};

ЧИТАЕТ – бинарное отношение с составным первичным ключом Код_лек, Код_предм, с кардинальным числом, равным шести, в котором присутствуют первичные ключи только читающих лекторов и первичные ключи только читаемых предметов.

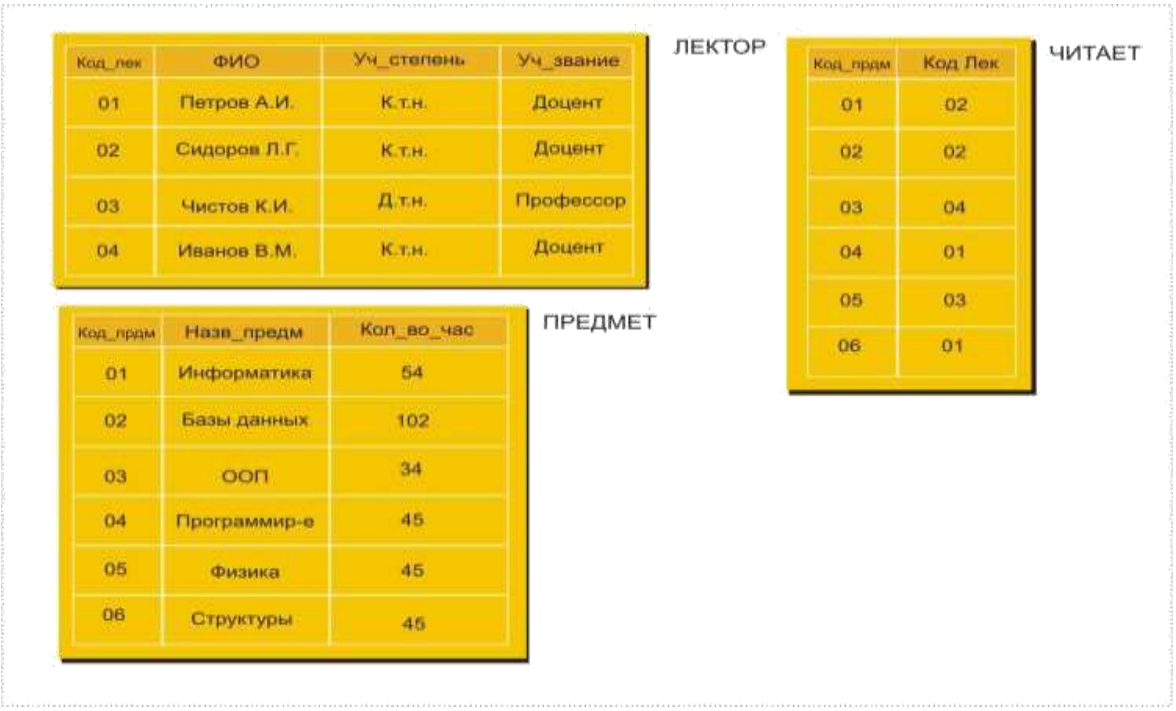

Рисунок 1.8. Реляционные отношения модели

Все приведенные отношения являются нормализованными, поскольку атрибуты всех отношений имеют атомарные значения. Во всех трех отношениях отсутствуют дублирующие кортежи.

Для обновления отношений необходимо иметь возможность выполнять следующие операции:

– добавление кортежа;

– удаление кортежа;

– изменение кортежа.

Операция добавления для отношения r со схемой (A_1, A_2, ..., A_n) имеет вид:

ADD (r: A_l=d_l, A_2=d_2, ..., A_n=d_n).

Если порядок атрибутов фиксирован, возможна более короткая запись:

ADD (r: d_l, d_2, …, d_n).

Выполнение этой операции может стать невозможным в ряде случаев:

– добавляемый кортеж не соответствует схеме определенного отношения;

– некоторые значения кортежей не принадлежат соответствующим доменам;

– описанный кортеж совпадает по ключу с кортежем, уже находящимся в отношении.

Операция удаления предназначена для удаления кортежей. Она может быть записана следующим образом:

DEL (r; $A_1=d_1$, $A_2=d_2$,..., $A_n=d_n$),

или для упорядоченных атрибутов:

DEL (r; d_1, d_2,..., d_n) .

Для удаления некоторого кортежа часто достаточно указать значение некоторого ключа:

DEL (r; ключ).

Если указанный кортеж в отношении отсутствует, то отношение остается неизменным.

Операция изменения предназначена для модификации части кортежа. Для отношения r ее можно при $\{C_1, C_2, ... ,C_p\} \in \{A_1, A_2, ..., A_n\}$ определить так:

CH (r; $A_1=d_1$, $A_2=d_2$, ..., $A_n=d_n$; $C_1=e_1$, $C_2=e_2$, ..., $C_p=e_p$).

Если $K = \{B_1, B_2, ..., B_k\}$ является ключом, то запись данной операции может быть сокращена:

CH (r; $B_1=d_1$, $B_2=d_2$, ..., $B_k=d_k$; $C_1=e_1$, $C_2=e_2$, ..., $C_p=e_p$).

Возможные ошибки в данном случае те же, что и у предыдущих операций:

– указанный кортеж не существует;

– изменения имеют неправильный формат;

– используемые значения не принадлежат соответствующим доменам

Достоинствами реляционной модели являются простота, наглядность, независимость от данных. К тому же, в отличие от сетевых и иерархических моделей, реляционные модели для организации связей между записями применяют не внутренние указатели, а фактические значения атрибута, используя общий атрибут в каждой из записей. Недостатки реляционной модели связаны с однородностью структуры данных, семантической перегруженностью модели, ограниченным набором операций.

Примерами реляционных СУБД являются следующие:

– dBaseIII Plus и dBase IY (фирма Ashton-Tate);

– DB2 (IBM);

– FoxPro ранних версий и FoxBase (Fox Software);

– Paradox и dBASE for Windows (Borland);

– Visual FoxPro и Access (Microsoft);

– Oracle (Oracle).

2. Базисные средства манипулирования реляционными данными.

Манипуляционная составляющая определяет два базовых механизма манипулирования реляционными данными:

– основанная на теории множеств *реляционная алгебра;*

– базирующееся на математической логике *реляционное исчисление*.

Алгебра и исчисление обладают большой выразительной мощностью: очень сложные запросы к базе данных могут быть выражены с помощью одного выражения реляционной алгебры или одной формулы реляционного исчисления. Именно по этой причине именно эти механизмы включены в реляционную модель данных.

Конкретный язык манипулирования реляционными БД называется реляционно полным, если любой запрос, выражаемый с помощью одного выражения реляционной алгебры или одной формулы реляционного исчисления, может быть выражен с помощью одного оператора этого языка.

Известно, что механизмы реляционной алгебры и реляционного исчисления эквивалентны, т.е. для любого допустимого выражения реляционной алгебры можно построить эквивалентную (т.е. производящую такой же результат) формулу реляционного исчисления и наоборот. Однако они различаются уровнем процедурности. Запрос, представленный на языке реляционной алгебры, может быть вычислен на основе выполнения элементарных алгебраических операций с учетом их приоритетности и возможного наличия скобок. Для формулы реляционного исчисления однозначная вычислительная интерпретация отсутствует. Формула только ставит условия, которым должны удовлетворять кортежи результирующего отношения. Поэтому языки реляционного исчисления являются в большей степени непроцедурными или декларативными.

Все эти механизмы обладают одним важным свойством: они замкнуты относительно понятия отношения. Это означает, что выражения реляционной алгебры и формулы реляционного исчисления определяются над отношениями реляционных БД и результатом вычисления также являются отношения. В результате любое выражение или формула могут интерпретироваться как отношения, что позволяет использовать их в других выражениях или формулах.

Поскольку механизмы реляционной алгебры и реляционного исчисления эквивалентны, то в конкретной ситуации для проверки степени реляционности некоторого языка БД можно пользоваться любым из этих механизмов.

Реляционная алгебра представляет собой теоретический язык операций, которые на основе одного или нескольких отношений позволяют создавать другое отношение.

Все операции реляционной алгебры можно разделить на *операции обработки данных, которые включают операции над строками (кортежами) таблиц-отношений, и операции над отношениями, осуществляющие обработку данных нескольких отношений.*

Операциями, выполняемыми на уровне строк (кортежей) отношений, являются:

• включение – в таблицу добавляется новая строка;

• удаление – из таблицы удаляется строка;

• обновление – осуществляется изменение значений атрибутов в строках.

При выполнении операций над отношениями таблица-отношение обрабатывается как единый объект. При этом результатом обработки всегда является новая таблица-отношение, которая также может быть обработана.

Основными операциями над отношениями реляционной модели данных являются:

• основные операции над множествами:

– объединение,

– пересечение,

– разность (вычитание),

– декартово произведение,

• специальные операции:

– выборка,

– проекция,

– соединение,

– деление множеств.

Совокупность этих операций образует полную алгебру отношений.

Объединение (union) – операция выполняется над двумя совместимыми отношениями R1 и R2 (с идентичной структурой). В результате операции объединения строится новое отношение R=R1 U R2 . Отношение R имеет тот же состав атрибутов и совокупность кортежей исходных отношений. Причем в эту совокупность не включаются дубликаты.

Пример.

Приведены исходные отношения: R1 «Клиенты банка А» (таблица 1.2), R2 «Клиенты банка В» (таблица 1.3) и результат объединения R (таблица 1.4).

Таблица 1.2. R1 «Клиенты банка А»

	Город	Фамилия
К11	Москва	Петров
К12	Санкт-Петербург	Смирнов
К13	Воронеж	Соколов

Таблица 1.3. R2 «Клиенты банка В»

	Город	Фамилия
К21	Самара	Петров
К22	Москва	Петров
К23	Тверь	Семенов

Таблица 1.4. R «Клиенты»

	Город	Фамилия
К11	Москва	Петров
К12	Санкт-Петербург	Смирнов
К13	Воронеж	Соколов
К21	Самара	Петров
К23	Тверь	Семенов

В новое отношение R не вошел кортеж К22, так как он дублирует кортеж К11.

Пересечение (intersection) – операция выполняется над двумя совместимыми отношениями R1 и R2. Результирующее отношение RP= R1 ∩ R2 содержит одинаковые кортежи, которые есть в каждом из двух исходных. Результат пересечения имеет тот же состав атрибутов, как и в исходных.

Пример.

Пересечение двух отношений R1 «Клиенты банка А» (таблица 1.2) и R2 «Клиенты банка В» (таблица 1.3) дает отношение RP «Клиент» (таблица 1.5).

Таблица 1.5. RP Пересечение отношений

Город	Фамилия	
Москва	Петров	К11 (К22)

Вычитание, разность (set difference) – операция выполняется над двумя совместимыми отношениями R1 и R2 с идентичным набором атрибутов. В результате операции вычитания строится новое отношение RV= R1 – R2 с идентичным набором атрибутов, содержащее только те кортежи первого отношения R1 которые не повторяются в другом отношении R2.

Пример.

Вычитание двух отношений R1 «Клиенты банка А» (таблица 1.2) и R2 «Клиенты банка В» (таблица 1.3) дает отношение RV «Клиенты только банка А» (таблица 1.6).

Таблица 1.6. RV «Клиенты только банка А»

	Город	Фамилия
К12	Санкт-Петербург	Смирнов
К13	Воронеж	Соколов

Декартово произведение (cartesian product) выполняется над двумя отношениями R1 и R2, имеющими в общем случае разный состав атрибутов. В результате операции декартова произведения образуется новое отношение RD=R1* R2, которое включает все атрибуты исходных отношений. Результирующее отношение состоит из всевозможных сочетаний кортежей исходных отношениями R1 и R2. Число кортежей декартова произведения равно произведению количеств кортежей в исходных отношениях.

Пример.

Декартово произведение двух отношений R1 «Студент» (таблица 1.6) и R2 «Предмет» (таблица 1.8) дает новое отношение RD «Экзаменационная

ведомость» (таблица 1.9), которое содержит все атрибуты исходных отношений.

Таблица 1.7. R1 «Студент»

	Номер	Фамилия
К11		Иванов
К12		Петров
К13		Сидоров

Таблица 1.8. R2 «Предмет»

	Код	Наименование
К21	П1	Математика
К22	П2	Информатика

Таблица 1.9. RD «Экзаменационная ведомость»

		Номер	Фамилия	Код	Наименование	Оценка
К11	К21		Иванов	П1	Математика	
К11	К22		Петров	П1	Математика	
К12	К21		Сидоров	П1	Математика	
К12	К22		Иванов	П2	Информатика	
К13	К21		Петров	П2	Информатика	
К13	К22		Сидоров	П2	Информатика	

В полученное отношение целесообразно добавить атрибут «Оценка» для записи результатов экзамена.

Рассмотренные выше операции в той или иной мере реализуются в средствах обеспечивающих обработку реляционных таблиц. К таким средствам относятся средства запросов и другие языковые конструкции.

Развитие реляционного подхода привело к созданию реляционных языков. Например, язык SQL, реализованный в большинстве СУБД, является более чем реляционно-полным, так как кроме операций реляционной алгебры он содержит полный набор операторов над строками — «включить», «удалить», «обновить», а также реализует арифметические операции и операции сравнения.

С практической точки зрения, *специальные реляционные операции* имеют большее практическое значение по сравнению с теоретико-множественными.

Выборкой (ограничением, селекцией или фильтрацией; selection) на отношении *A,* с условием *C* называется отношение с тем же заголовком, что и у отношения *A*, и телом, состоящем из кортежей, значения атрибутов которых при подстановке в условие *C* дают значение ИСТИНА. *C* – логическое выражение, в которое могут входить атрибуты отношения *A* и (или) скалярные выражения.

В простейшем случае условие С имеет вид *X*Q*Y*, где Q – один из операторов сравнения (=, ¹, <, >, £, ³ и т.д.), а *X* и *Y* – атрибуты отношения А или скалярные значения. Такие выборки называются Q – *выборки (тэта-выборки)* или Q – *селекция*, Q – *ограничения*.

Синтаксис операции выборки: A WHERE C, где C – условие выборки, или XQY.

Пусть дано отношение *A* с информацией о сотрудниках (таблица 1.10), необходимо выбрать всех сотрудников с зарплатой менее 3000, в этом случае выполняем выборку *A WHERE Зарплата* < 3000, результат выборки в таблице 1.11.

Таблица 1.10. Отношение А

Табельный номер	Фамилия	Зарплата
	Иванов	
	Петров	
	Сидоров	

Таблица 1.11. Результат операции A WHERE Зарплата < 3000

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000

Смысл операции выборки очевиден – выбрать кортежи отношения, удовлетворяющие некоторому условию. Таким образом, операция выборки дает «*горизонтальный срез*» отношения по некоторому условию.

Проекцией (projection) отношения *A* по атрибутам *(X,Y,…,Z)*, где каждый из атрибутов принадлежит отношению *A*, называется отношение с заголовком *(X,Y,…,Z)* и телом, содержащим кортежи соответствующих атрибутов

Синтаксис проекции: A[X,Y,…,Z].

Для отношения *A* (таблица 1.10) результатом проекции *A* [Фамилия, Зарплата] таблица 1.12.

Таблица 1.12. Результат операции A [Фамилия, Зарплата]

Фамилия	**Зарплата**
Иванов	1000
Петров	2000
Сидоров	3000

Видно, что операция проекции выполняет «*вертикальный срез*» отношения, в котором будут удалены все возникшие при таком срезе дубликаты кортежей.

Соединение (join). Операция соединения отношений, наряду с операциями выборки и проекции, является одной из наиболее важных реляционных операций.

Обычно рассматривается несколько разновидностей операции соединения:

– общая операция соединения;

– Q-соединение (тэта-соединение);

– экви-соединение;

– естественное соединение.

Наиболее важным из этих частных случаев является операция естественного соединения. Все разновидности соединения являются частными случаями общей операции соединения.

Соединением отношений *A* и *B* по условию *C* называется отношение образованное последовательностью операций декартова произведения и выборки:

(A Ä B) WHERE C,

где *C* представляет собой логическое выражение, в которое могут входить атрибуты отношений *A* и *B* и (или) скалярные выражения.

Если в отношениях *A* и *B* имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать.

Тэта-соединение. Пусть отношение *A* содержит атрибут *X*, отношение *B* содержит атрибут *Y*, а Q – один из операторов сравнения (=, ¹, <, >, £, ³ и т.д.). Тогда Q-***соединением*** отношения *A* по атрибуту *X* с отношением *B* по атрибуту *Y* называют отношение *(A Ä B) WHERE XQY*

Это частный случай операции общего соединения. Иногда, для операции соединения применяют более короткий синтаксис *A[XQY]B.*

Экви-соединение является наиболее важным частным случаем тэта-соединения, когда тэта является просто равенством и имеет следующий синтаксис: *A[X=Y]B* или *(A Ä B) WHERE X=Y.*

Пусть даны два отношения *A* и *B*. Отношение *A* (таблица 1.13) – данные о товарах, отношение *B* (таблица 1.14) – данные о продаже товаров. Необходимо определить, когда и в каком количестве отпускались товары со склада.

Таблица 1.13. Отношение А «Товары»

Код товара	Товар	Единица	Цена единицы
1	Сахар	кг	16р.
2	Макароны	кг	14р.

Таблица 1.14. Отношение В «Отпуск товаров»

Код товара	Дата продажи	Количество
1	12.07.02	10
2	12.07.02	15
2	12.07.02	3

Таблица 1.15 представляет собой декартово произведение двух отношений, в котором темным выделены кортежи, для которых не выполнится условие выборки *A. Код товара = B. Код тов.,* следовательно, они будут вычеркнуты из окончательного результата (таблица 1.16).

Таблица 1.15. Соединение *(A Ä B) WHERE A. Код товара = B. Код тов.*

Код товара	Товар	Единица	Цена единицы	Код тов.	Дата продажи	Количест-во
1	Сахар	кг	16р.	*1*	12.07.02	10
1	Сахар	кг	16р.	*2*	12.07.02	15
1	Сахар	кг	16р.	*2*	12.07.02	3
2	Макароны	кг	16р.	*1*	12.07.02	10
2	Макароны	кг	16р.	*2*	12.07.02	15
2	Макароны	кг	16р.	*2*	12.07.02	3

Таблица 1.16. Окончательный результат соединения *(A Ä B) WHERE A.Код товара = B.Код тов.*

Код товара	Товар	Единица	Цена единицы	Код тов.	Дата продажи	Количест-во
1	Сахар	кг	16р.	*1*	12.07.02	10
2	Макароны	кг	16р.	*2*	12.07.02	15
2	Макароны	кг	16р.	*2*	12.07.02	3

Естественное соединение (join). Пусть даны отношения $A(A_1, A_2,\ldots,An, X_1, X_2,\ldots,X_p)$ и $B(X_1, X_2,\ldots,X_p, B_1, B_2,\ldots,Bm)$, имеющие одинаковые атрибуты $X_1, X_2,\ldots,X_p$ (т.е. атрибуты с одинаковыми именами и определенные на одинаковых доменах).

Тогда *естественным соединением* отношений *A* и *B* называется отношение с заголовком $(A_1, A_2,\ldots,An, X_1, X_2,\ldots,X_p, B_1, B_2,\ldots,Bm)$, и телом, содержащим множество соответствующих кортежей.

Естественное соединение настолько важно, что для него используют специальный синтаксис: A JOIN B.

В синтаксисе естественного соединения не указываются, по каким атрибутам производится соединение. Естественное соединение производится по всем одинаковым атрибутам.

Естественное соединение эквивалентно следующей последовательности реляционных операций:

1. Переименовать одинаковые атрибуты в отношениях
2. Выполнить декартово произведение отношений
3. Выполнить выборку по совпадающим значениям атрибутов, имевших одинаковые имена
4. Выполнить проекцию, удалив повторяющиеся атрибуты

5. Переименовать атрибуты, вернув им первоначальные имена

Можно выполнять последовательное естественное соединение нескольких отношений. Естественное соединение (как и соединение общего вида) обладает свойством *ассоциативности*, т.е. *(A JOIN B) JOIN C = AJOIN (B JOIN C),* поэтому его можно записать, опуская скобки *A JOIN B JOIN C.*

Применяя естественное соединение, результат, полученный в таблице 1.16, можно было получить операцией *A JOINB*, но с одним условием, атрибут отношения *B* используемый для связи с отношением *A* должен иметь имя совпадающее с атрибутом связи отношения *A* (т.е. Код товара).

Деление (division). Пусть даны отношения $A(X_1,X_2,\ldots,Xn,Y1,Y2,\ldots,Y_m)$ и $B(Y1,Y2,\ldots,Y_m)$, причем атрибуты $(Y1,Y2,\ldots,Y_m)$ – общие для двух отношений. *Делением отношений A* на *B* называется отношение с заголовком $(X_1,X_2,\ldots,Xn)$ и телом, содержащим множество кортежей $(x_1,x_2,\ldots,xn)$, только таких, для которых найдутся *все* кортежи $(y1,y2,\ldots,y_m)\hat{I}B,$ в отношении *A*.

Синтаксис операции деления: A DEVIDBY B (A : B)

Типичные запросы, реализуемые с помощью операции деления, обычно в своей формулировке имеют слово «все» – «какие поставщики поставляют *все* детали?».

Таблицы 1.15 и 1.16 нуждаются в логическом дополнении, т.е. нужна таблица, связывающая поставляемые товары и поставщиков (по их кодам). Введем такую таблицу и на ее примере рассмотрим операцию деления.

Таблица 1.17. Отношение X «Поставщики-Детали».

Номер поставщика	Номер детали
1	1
1	2
1	3
2	1
2	2
3	1

Требуется узнать, какой поставщик поставляет все детали. Отношение X возьмем в качестве делимого, а проекцию таблицы 1.17 «детали» –Y=B [Номер детали] (таблица 1.18).

Таблица 1.18. Отношение Y = B [Номер детали].

Номер детали
1
2
3

Деление *X DEVIDBY Y* дает список номеров поставщиков, поставляющих *все* детали (таблица 1.19):

Таблица 1.19. Результирующее отношение *(A : B)*.

Номер поставщика
1

Тема 3. Основы проектирования баз данных

План:

1. Проектирование базы данных: понятие, требования, задачи. Этапы жизненного цикла базы данных.
2. Этапы проектирования базы данных и их процедуры.

1. Проектирование базы данных: понятие, требования, задачи. Этапы жизненного цикла базы данных.

Процесс *проектирования БД* заключается в достижении компромиссов между функциональными, информационными, аппаратными, архитектурными и технологическими требованиями к базе данных и строится на информированном принятии решений по структуре базы данных.

Проектирование БД – это поиск способов удовлетворения функциональных требований средствами имеющейся компьютерной технологии с учетом заданных ограничений.

Процесс *проектирования БД* охватывает несколько основных сфер:

– *проектирование объектов базы данных* (таблицы, представления, индексы, триггеры, хранимые процедуры, функции, пакеты) для представления данных предметной области в базе данных;

– *проектирование интерфейса* взаимодействия с БД (формы, отчеты и т.д.), т.е. проектирование приложений, которые будут сопровождать данные в БД и реализовывать вопросно-ответные отношения на этих данных;

– *проектирование БД* под конкретную вычислительную среду или информационную технологию (архитектура «клиент-сервер», параллельные архитектуры, распределенная вычислительная среда);

– *проектирование БД* под назначение системы (интеллектуальный анализ данных, OLAP, OLTP и т.д.).

Зачастую вычислительная среда задается в качестве входных условий проектирования, но иногда проектирование следует проводить с учетом возможного перехода в будущем на другую аппаратную платформу или технологию.

Проектирование БД – это многоэтапный процесс принятия обоснованных решений в процессе анализа информационной модели предметной области, требований к данным со стороны прикладных программистов и пользователей, синтеза логических и физических структур данных, анализа и обоснования выбора программных и аппаратных средств.

В результате проектирования БД должны быть определены: содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными.

Проектируемая БД должна обеспечивать:

– хранение всей необходимой информации;

– возможность получения данных по всем необходимым запросам;

– сокращение избыточности и дублирования данных;

– целостность данных (правильность их содержания):

– исключение противоречий в содержании данных, исключение их потери и т.д.

Основными *задачами* при проектировании БД являются:

1. Определение границ проектируемой предметной области (постановка задания), построение структуры данных предметной области – концептуальное проектирование (инфологическая модель).

2. Выбор оптимальной СУБД для заданной предметной области. Описание БД на языке СУБД – логическое проектирование (даталогическая модель).

3. Определение типов используемых данных и методов доступа к ним (запросы, отчёты, формы) – физическое проектирование (физическая модель).

Жизненный цикл БД включает в себя следующие основные этапы (рисунок 1.9):

1. Планирование разработки БД.
2. Определение требований к системе.
3. Сбор и анализ требований пользователей.
4. Проектирование БД:
   – концептуальное проектирование БД;
   – логическое проектирование БД;
   – физическое проектирование БД.
5. Разработка приложений:
   – проектирование транзакций;
   – проектирование пользовательского интерфейса;
6. Реализация.
7. Загрузка данных.
8. Тестирование.
9. Эксплуатация и сопровождение:
   – анализ функционирования и поддержка исходного варианта БД;
   – адаптация, модернизация и поддержка переработанных вариантов.

Рисунок 1.9. Этапы жизненного цикла БД

Планирование разработки БД. Содержание данного этапа – разработка стратегического плана, в процессе которой осуществляется предварительное планирование конкретной системы управления БД.

Планирование разработки БД состоит в определении трех основных компонентов: объема работ, ресурсов и стоимости проекта.

Важной частью разработки стратегического плана является проверка осуществимости проекта, состоящая из нескольких частей.

Первая часть – проверка технологической осуществимости. Она состоит в выяснении вопроса, существует ли оборудование и программное обеспечение, удовлетворяющее информационным потребностям фирмы.

Вторая часть – проверка операционной осуществимости — выяснение наличия экспертов и персонала, необходимых для работы БД.

Третья часть – проверка экономической целесообразности осуществления проекта. При исследовании этой проблемы весьма важно дать оценку ряду факторов, в том числе и таким:

– целесообразность совместного использования данных разными отделами;

– величина риска, связанного с реализацией системы БД;

– ожидаемая выгода от внедрения подлежащих созданию приложений;

– время окупаемости внедренной БД;

– влияние системы управления БД на реализацию долговременных планов организации.

Определение требований к системе. На данном этапе необходимо определить диапазон действия приложения БД, состав его пользователей и области применения. Определение требований включает выбор целей БД, выяснение информационных потребностей различных отделов и руководителей фирмы и требований к оборудованию и программному обеспечению.

Сбор и анализ требований пользователей. На данном этапе необходимо создать для себя модель движения важных материальных объектов и уяснить процесс документооборота. По каждому документу необходимо установить периодичность использования, определить данные, необходимые для выполнения выделенных функций (анализируя существующую и планируемую документацию, выясняют, как получается каждый элемент данных, кем получается, где в дальнейшем используется, кем контролируется). Собранная информация о каждой важной области применения приложения и пользовательской группе должна включать следующие компоненты: исходную и генерируемую документацию, подробные сведения о выполняемых транзакциях, а также список требований с указанием их приоритетов.

Формализация собранной на этом этапе информации может быть повышена с помощью методов составления спецификаций требований, к числу которых относятся, например, технология структурного анализа и проектирования, диаграммы потоков данных и графики «вход — процесс — выход».

Проектирование БД. Полный цикл разработки БД включает *концептуальное, логическое и физическое ее проектирование.*

Первая фаза процесса проектирования БД заключается в создании для анализируемой части предприятия *концептуальной модели данных.*

Проектирование сложных БД с большим количеством атрибутов осуществляется использованием, так называемого, нисходящего подхода.

Этот подход начинается с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

Нисходящий подход демонстрируется в концепции модели «сущность – связь» (Entity-Relationship model – ER-модель) – самой популярной технологии высокоуровневого моделирования данных, предложенной П. Ченом.

Модель «сущность – связь» относится к семантическим моделям. *Семантическое моделирование* данных, связанное со смысловым содержанием данных, независимо от их представления.

В построении *общей концептуальной модели данных* выделяют ряд этапов:

1. Выделение локальных представлений, соответствующих обычно относительно независимым данным. Каждое такое представление проектируется как подзадача.

2. Формулирование сущностей, описывающих локальную предметную область проектируемой БД, и описание атрибутов, составляющих структуру каждой сущности.

3. Выделение ключевых атрибутов.

4. Спецификация связей между сущностями. Удаление избыточных связей.

5. Анализ и добавление неключевых атрибутов.

6. Объединение локальных представлений.

Созданная концептуальная модель данных является источником информации для фазы логического проектирования БД.

Цель второй фазы проектирования БД состоит в создании логической модели данных. Логическая модель, отражающая особенности представления о функционировании организациии одновременно многих типов пользователей, называется *глобальной логической моделью данных.*

Процесс проектирования БД должен опираться на определенную модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД.

Концептуальное и логическое проектирование — это итеративные процессы, которые включают в себя ряд уточнений, продолжающиеся до тех пор, пока не будет получен наиболее соответствующий структуре организации продукт.

Целью третьей фазы проектирования БД является создание описания СУБД ориентированной модели БД.

Действия, выполняемые на этом этапе, слишком специфичны для различных моделей данных, поэтому их сложно обобщить. Остановимся на реляционной модели данных. В этом случае под физическим проектированием подразумевается:

– создание описания набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;

– определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность системы с базой данных;

– разработка средств зашиты создаваемой системы.

Разработка приложений. Параллельно с проектированием системы базы данных выполняется разработка приложений. Главные составляющие данного процесса – это проектирование транзакций и пользовательского интерфейса.

Проектирование транзакций. Транзакции представляют некоторые события реального мира. Транзакция может состоять из нескольких операций, однако с точки зрения пользователя эти операции представляют собой единое целое, переводящее БД из одного непротиворечивого состояния в другое. Реализация транзакций опирается на тот факт, что СУБД способна обеспечивать сохранность внесенных во время транзакции изменений в БД и непротиворечивость БД даже в случае возникновения сбоя. Проектирование транзакций заключается в определении:

– данных, которые используются транзакцией;

– функциональных характеристик транзакции;

– выходных данных, формируемых транзакцией;

– степени важности и интенсивности использования транзакции.

Проектирование пользовательского интерфейса. Интерфейс должен быть удобным и обеспечивать все функциональные возможности, предусмотренные в спецификациях требований пользователей.

Специалисты рекомендуют при проектировании пользовательского интерфейса использовать следующие основные элементы и их характеристики:

– содержательное название;

– ясные и понятные инструкции;

– логически обоснованные группировки и последовательности полей;

– визуально привлекательный вид окна формы или поля отчета;

– легко узнаваемые названия полей;

– согласованную терминологию и сокращения;

– согласованное использование цветов;

– визуальное выделение пространства и границ полей ввода данных;

– удобные средства перемещения курсора;

– средства исправления отдельных ошибочных символов и целых полей;

– средства вывода сообщений об ошибках при вводе недопустимых значений;

– особое выделение необязательных для ввода полей;

– средства вывода пояснительных сообщений с описанием полей;

– средства вывода сообщения об окончании заполнения формы.

Реализация. На данном этапе осуществляется физическая реализация БД и разработанных приложений, позволяющих пользователю формулировать требуемые запросы к БД и манипулировать данными в БД.

БД описывается на языке определения данных выбранной СУБД. В результате компиляции его команд и их выполнения создаются схемы и пустые файлы БД. На этом же этапе определяются и все специфические пользовательские представления.

Прикладные программы реализуются с помощью языков третьего или четвертого поколений. Кроме того, на этом этапе создаются другие компоненты проекта приложения, например, экраны меню, формы ввода данных и отчеты.

Реализация этого, а также и более ранних этапов проектирования БД может осуществляться с помощью инструментов автоматизированного проектирования и создания программ, которые принято называть CASE-инструментами (Computer-Aided Software Engineering).

Загрузка данных. На этом этапе созданные в соответствии со схемой БД пустые файлы, предназначенные для хранения информации, должны быть заполнены данными. Наполнение БД может протекать по-разному, в зависимости от того, создается ли БД вновь или новая БД предназначена для замены старой.

Тестирование. Для оценки законченности и корректности выполнения приложения БД может использоваться несколько различных стратегий тестирования:

– нисходящее тестирование;

–

– восходящее тестирование;

– тестирование потоков;

– интенсивное тестирование.

Нисходящее тестирование начинается на уровне подсистем с модулями, которые представлены заглушками, т.е. простыми компонентами, имеющими такой же интерфейс, как модуль, но без функционального кода. Каждый модуль низкого уровня представляется заглушкой. Постепенно все программные компоненты заменяются фактическим кодом и после каждой замены снова тестируются.

Восходящее тестирование выполняется в противоположном направлении по отношению к нисходящему. Оно начинается с тестирования модулей на самых низких уровнях иерархии системы, продолжается на более высоких уровнях и заканчивается на самом высоком уровне.

Тестирование потоков осуществляется при тестировании работающих в реальном масштабе времени систем, которые обычно состоят из большого количества взаимодействующих процессов, управляемых с помощью прерываний. Стратегия тестирования потоков направлена на слежение за отдельными процессами.

Стратегия *интенсивного тестирования* часто включает серию тестов постепенно возрастающей нагрузкой и продолжается до тех пор, пока система не выйдет из строя.

Эксплуатация и сопровождение. Основные действия, связанные с этим этапом сводятся к наблюдению за созданной системой и поддержке ее нормального функционирования по окончании развертывания.

Поддержка БД предполагает разрешение проблем, возникающих в процессе эксплуатации БД и связанных как с ошибками реализации БД, так и с изменениями в самой предметной области, созданием дополнительных программных компонент или модернизацией самой БД.

2. Этапы проектирования базы данных и их процедуры.

Процесс проектирования БД состоит из последовательности преобразований модели данных одного уровня в модель данных другого уровня.

Последовательность преобразований модели данных:

− систематизация объектов реального мира;

− создание информационных структур, описывающих систему объектов реального мира;

− создание структуры данных, которая используется для представления информационных структур в БД и прикладных программах;

− представление структуры памяти, используемой для хранения структур данных.

Процесс проектирования базы данных включает три этапа.

Первый этап – анализ предметной области или этап концептуального проектирования. На этом этапе осуществляется сбор, анализ и редактирование требований к данным. Анализ предметной области заканчивается построением информационной структуры (концептуальной схемы). На данном этапе анализируются запросы пользователей, выбираются информационные объекты и их характеристики, которые предопределяют содержание проектируемой БД. На основе проведенного анализа структурируется предметная область. Анализ предметной области не зависит от программной и технической сред, в которых будет реализовываться БД.

Анализ предметной области целесообразно разбить на три фазы:

1) анализ требований и информационных потребностей;

2) выявление информационных объектов и связей между ними;

3) построение модели предметной области и проектирование схемы БД.

Во время анализа концептуальных требований и информационных потребностей необходимо выполнить:

– анализ требований пользователей к БД (концептуальных требований);

– выявление имеющихся задач по обработке информации, которая должна быть представлена в БД (анализ приложений);

– выявление перспективных задач (перспективных приложений);

– документирование результатов анализа.

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требования к вводу, обновлению и корректировке информации. Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных задач.

Фаза выявления информационных объектов и связей между ними заключается:

– в выборе информационных объектов;

– в задании необходимых свойств для каждого объекта;

– в выявлении связей между объектами;

– в определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов.

При выборе информационных объектов следует ответить на следующие вопросы:

1. На какие классы можно разбить данные, подлежащие хранению в БД?

2. Какое имя можно присвоить каждому классу данных?

3. Какие наиболее интересные характеристики (с точки зрения пользователя) каждого класса данных можно выделить?

4. Какие имена можно присвоить выбранным наборам характеристик?

В ходе выявления связей между информационными объектами следует ответить на следующие вопросы:

1. Какие типы связей между информационными объектами?

2. Какое имя можно присвоить каждому типу связей?

3. Каковы возможные типы связей, которые могут быть использованы впоследствии?

4. Имеют ли смысл какие-нибудь комбинации типов связей?

Далее проектировщик пытается задать ограничения на объекты и их характеристики. Под ограничением целостности обычно понимают логические ограничения, накладываемые на данные.

При выявлении условий ограничения целостности проектировщик пытается ответить на следующие вопросы:

1. Какова область значений для числовых характеристик?

2. Каковы функциональные зависимости между характеристиками одного информационного объекта?

3. Какой тип отображения соответствует каждому типу связей?

Заключительная фаза анализа предметной области состоит в проектировании ее информационной структуры или концептуальной модели. Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных. Концептуальная модель применяется для структурирования предметной области с учетом информационных интересов пользователей системы. Она дает возможность систематизировать информационное содержание предметной области, позволяет как бы «подняться вверх» над ПО и увидеть ее отдельные элементы. При этом уровень детализации зависит от выбранной модели.

Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений. Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель остается неизменной или увеличивается с

целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель «сущность-связь». Наиболее известной моделью данного типа, названной по фамилии автора, является модель П. Чена или ER-модели. Основными конструкциями данной модели являются сущности и связи. Под сущностью понимают основное содержание объекта предметной области, о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление. Экземпляр сущности – конкретный объект. Например: сущность (объект) – студент, экземпляр сущности – Иванов А. В.; сущность (объект) – колледж, экземпляр сущности – КемПК. Сущность принято определять атрибутами – поименованными характеристиками. Например: сущность – студент, атрибуты – ФИО, год рождения, адрес, номер группы и т.д. Чтобы задать атрибут в модели, ему надо присвоить имя и определить область допустимых значений. Одно из назначений атрибута - идентифицировать сущность.

Второй этап – этап логического проектирования, т.е. моделирование построенной информационной системы и проектирование её отдельных составляющих в форме, соответствующей реальной БД. В процессе логического проектирования требования к данным преобразуются в структуры используемой СУБД. На этом этапе достаточно ответственным является выбор СУБД. Это обусловлено тем что, с одной стороны, число СУБД достаточно велико, а с другой – проектировщику необходимо оценить СУБД по множеству характеристик. Однако основным критерием отбора остается оценка того, насколько эффективно внутренняя модель данных, поддерживаемая системой, способна описать построенную концептуальную схему.

Третий этап – этап физического проектирования (тесно связан с этапом реализации) – решаются вопросы, связанные с производительностью системы, определяются структуры хранения данных и методы доступа. Процесс проектирования БД не может быть сделан автоматическим, так как для решения многих проблем участие человека является обязательным.

Задача разработчика БД состоит в структуризации данных таким образом, чтобы устранить ненужное дублирование и обеспечить быстрый путь поиска необходимой информации. При проектировании БД могут появиться нежелательные свойства, такие как избыточность, аномалии обновления, аномалии включения, аномалии удаления и др. Для уменьшения нежелательных характеристик БД к схемам отношений применяют процедуры нормализации.

Нормализация – это разбиение таблицы на две или более, обладающих

лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта БД, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Нормализация – это процесс разделения информации на структурные единицы, т.е. таблицы. Нормализация БД должна быть выполнена с учётом следующего правила: таблицы, которые содержат повторяющуюся информацию, для устранения дублирования значений должны быть разделены на отдельные таблицы, что приводит к сокращению размеров БД. Если определенным образом ограничить наличие зависимостей записей в схеме данных, то получатся нормальные формы отношения: первая нормальная форма (1НФ), вторая нормальная форма (2НФ), третья нормальная форма (ЗНФ), нормальная форма Бойса-Кодда (НФБК).

Теория нормализации основана на концепции нормальных форм. Таблица находится в данной нормальной форме, если она удовлетворяет определенному набору требований. Теоретически существует пять нормальных форм, но на практике обычно используются только первые три. Первые две нормальные формы являются промежуточными шагами для приведения базы данных к третьей нормальной форме.

Первая нормальная форма. Реляционная таблица находится в первой нормальной форме (1НФ), если все значения ее полей атомарны, т.е. неделимы и все записи уникальны и значения элементов в домене не являются ни списками, ни множествами.

Схема БД находится в 1НФ, если каждая схема отношения находится в 1НФ.

Пример. Есть отношение: R рождение (ФИО, дата рождения). Перевод отношения R рождение в 1НФ: R рождение (фамилия, имя, отчество, день рождения, месяц, год рождения).

Вторая нормальная форма. Реляционная таблица находится во второй нормальной форме (2НФ), если она находится в первой нормальной форме и ее неключевые поля полностью зависят от всего первичного ключа.

Чтобы перейти от первой нормальной формы ко второй, нужно выполнить следующую последовательность действий:

1. Определить, на какие части можно разбить первичный ключ, так чтобы некоторые из не ключевых полей зависели от одной из этих частей (эти части не обязаны состоять из одной колонки).

2. Создать новую таблицу для каждой такой части ключа и группы,

зависящих от нее полей и переместить их в эту таблицу. Часть бывшего первичного ключа станет при этом первичным ключом новой таблицы.

3. Удалить из исходной таблицы поля, перемещенные в другие таблицы, кроме тех, которые станут внешними ключами.

Схема БД находится в 2НФ, если схема каждого отношения БД находится в 2НФ.

Пример. Есть отношение R учеба (факультет, группа, дисциплина, вид занятий, преподаватель, квалификация преподавателя). Данное отношение не находится во 2НФ, так как неключевые атрибуты «факультет» и «квалификация преподавателя» функционально неполно зависят от ключа, например, неключевой атрибут «квалификация преподавателя» зависит только от части ключа – от «преподавателя».

Приведение отношение R учеба ко 2НФ выполняется путем его декомпозиции с помощью проекции исходного отношения: R1 (группа, дисциплина, вид занятий, преподаватель), R2 (дисциплина, вид занятий, квалификация преподавателя), R3 (группа, факультет), R4 (преподаватель, квалификация преподавателя) Каждое из отношений Rl, R2, R3, R4 находится во 2НФ. Исходное отношение R учеба может быть восстановлено путем эквисоединения полученных отношений R1, R2, R3, R4.

Третья нормальная форма. Приведение отношений к ЗНФ требует дальнейшего сокращения возможных функциональных зависимостей между атрибутами. Переход к ЗНФ происходит через проекцию исходного отношения. Переход может быть не единственным. Можно говорить об оптимальной ЗНФ, если число отношений в ЗНФ будет минимально. Третья нормальная форма не допускает функциональных зависимостей между неосновными атрибутами отношения, т.е. транзитивная зависимость неосновных атрибутов от ключа исключается. Реляционная таблица находится в третьей нормальной форме, если она находится во второй нормальной форме, и ее некпючевые поля полностью зависят только от первичного ключа, т.е. не взаимозависимы. Чтобы перейти от второй нормальной формы к третьей, нужно выполнить следующую последовательность действий:

1. Определить все поля, от которых зависят другие поля.

2. Создать новую таблицу для каждого такого поля или группы полей и группы зависящих от него полей и переместить их в эту таблицу. Поле или группа полей, от которого зависят перемещенные поля, станет при этом первичным ключом новой таблицы.

3. Удалить из исходной таблицы поля, перемещенные в другие таблицы, кроме тех, которые станут внешними ключами.

В целом этапы проектирования БД представлены схемой на рисунке 1.10.

Рисунок 1.10. Обобщенная схема этапов проектирования БД

Таким образом, процесс проектирования БД заключается в достижении согласованных решений между функциональными, информационными, аппаратными, архитектурными и технологическими требованиями к БД и строится на информированном принятии решений по структуре БД на каждом из его этапов.

Хороший проект БД может помочь сэкономить место на диске за счет уменьшения избыточности данных, а также обеспечить точность и надежность хранения данных. Более того, хорошо спроектированную БД проще использовать и обслуживать.

Тема 4. Основы структурированного языка запросов SQL

План:

1. Алфавит и лексемы языка SQL.
2. Типы данных языка SQL.
3. Операторы языка SQL.
4. Операции языка SQL.
5. Функции языка SQL.
6. Создание, модификация и удаление таблиц.
7. Выбор информации из базы данных.

1. Алфавит и лексемы языка SQL.

Язык SQL (Structured Query Language – язык структурированных запросов) является непроцедурным языком и ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц. Формулируя запросы на языке SQL, можно создавать и модифицировать различные объекты БД и, оперируя группами строк, вставлять, выбирать, обновлять и удалять данные из таблиц. Кроме того, он позволяет управлять доступом к объектам БД и обеспечивать непротиворечивость и целостность данных, хранящихся в базе.

Алфавит языка включает следующие символы:

1) буквы: A..Z, a..z;

2) цифры: 0..9;

3) символы: + – * / ! @ $ # = < > ^ ' “ ( ) | _ ; , .

Идентификаторы и комментарии. Идентификаторы, длина которых может достигать 30 символов, обычно начинаются с буквы и могут включать в себя также цифры, символы $ и #, символ подчеркивания. Исключение составляют имена БД (до восьми символов) и удаленные имена. В некоторых версиях системы Oracle допускается использование русских букв. Имя любого объекта может состоять из нескольких частей: [схема.]объект[@dblink]. Схема представляет собой набор объектов разной структуры, принадлежащих конкретному пользователю, и идентифицируется его именем. Среди объектов схемы могут быть таблицы, представления (виртуальные таблицы), индексы, последовательности, триггеры, процедуры и функции. @dblink – это удаленное имя таблицы или представления БД.

Допускается использование однострочных и многострочных комментариев. Однострочные комментарии представляют собой следующую конструкцию:

-- текст комментария

Многострочные комментарии имеют следующий вид:

/* текст комментария */

Литералы. Символьные литералы определяются как тип CHAR и записываются в одинарных кавычках: 'test'. При необходимости присутствия одинарной кавычки внутри символьного литерала она удваивается.

Числовые литералы определяются как тип NUMBER и представляют собой целые или действительные значения со знаком или без знака, при этом действительные значения могут быть записаны в формате с десятичной точкой или в экспоненциальной форме.

Пустые значения. В языке SQL имеется специальное предопределенное значение NULL, которое расценивается как неопределенное значение. Оно не эквивалентно понятию «пустая строка» для символьных типов и не эквивалентно нулевому значению для числовых типов. Если в некотором столбце таблицы данные отсутствуют, говорят, что его значение NULL. Столбец с данными любого типа может содержать значение NULL, если только он специально не описан как NOT NULL.

Псевдостолбцы. Это формируемые системой столбцы, имеющие стандартные имена. Их значения можно только просматривать и использовать, но корректировать (добавлять, удалять, изменять) нельзя.

К ним относятся: ROWID, ROWNUM, LEVEL, CURRVAL, NEXTVAL.

Псевдостолбец ROWID содержит уникальные для всей БД физические адреса строк таблицы. Значение псевдостолбца ROWID определяется при вставке строки в таблицу и не изменяется, пока строка присутствует в таблице.

Псевдостолбец ROWNUM определяет порядковый номер строки, выбранной при выполнении запроса. Он обычно используется для ограничения числа строк, выбираемых из таблицы.

Псевдостолбец LEVEL возвращает уровень вложенности данных, позволяя тем самым строить запросы для получения информации об иерархии данных.

Для работы с последовательностями генерируемых значений, используемых в качестве уникальных ключей, имеются псевдостолбцы:

1) имя_последовательности. CURRVAL – возвращает текущее значение из указанной последовательности генерируемых значений;

2) имя_последовательности.NEXTVAL – возвращает следующее значение из указанной последовательности генерируемых значений.

Предварительно последовательность с именем имя_последовательности должна быть создана с помощью оператора CREATE SEQUENCE.

2. Типы данных языка SQL.

Наиболее часто используются следующие типы данных: символьные, числовые, тип DATE, двоичные и большие объекты.

Символьные типы данных представлены типами CHAR (длина), VARCHAR2 (длина) и LONG.

Тип данных CHAR представляет собой символьные строки фиксированной длины. Минимальная длина равна 1, максимальная – 2000 байт. Если значение, помещаемое в столбец данного типа, превосходит

указанный размер, то выводится сообщение об ошибке; если длина помещаемого значения меньше указанной длины, то значение дополняется пробелами справа.

Тип данных VARCHAR2 представляет собой символьные строки переменной длины. Максимальный размер строки 4000 байт, минимальный – 1 байт. При помещении текста в столбец большего размера дополнение пробелами не производится.

Строки этих двух типов сравниваются по-разному. Строки типа CHAR – посимвольно с дополнением пробелами строки с меньшей длиной до размера строки с большей длиной. Строки VARCHAR2 – без дополнения пробелами до большей длины. Поэтому для двух в принципе одинаковых строк могут быть получены различные результаты при их сравнении.

П р и м е р. Для двух строк 'AB' и 'AB ' (вторая строка содержит пробел, а первая нет) типа CHAR получим, что 'AB' = 'AB '. Для этих же строк типа VARCHAR2 результатом сравнения будет 'AB' < 'AB '.

Тип данных LONG представляет собой символьные данные переменной длины, величина которой может достигать 2 Гб. На использование переменных этого типа накладывается ряд ограничений: столбец такого типа должен быть единственным в таблице, его нельзя индексировать, использовать в качестве ключа упорядочения и в операциях группирования, а также для построения условий.

Числовые типы данных представлены типом NUMBER, который позволяет определить три различных типа данных:

NUMBER, NUMBER(p), NUMBER(p, s).

В первом случае определяются действительные числа, диапазон которых от 1.0*10 –130 до 1.0*10 126 –1, мантисса содержит 38 знаков. Во втором случае считается, что определяется диапазон целых чисел, где p – количество цифр в числе (от 1 до 38). В третьем случае описываются числа с фиксированной точкой; p – общее количество цифр (от 1 до 38), s – масштаб (от –84 до 127) – определяет количество цифр после запятой. Если s > 0, то число округляется до указанного числа знаков справа от десятичной точки, если s < 0, то число округляется до указанного числа знаков слева от десятичной точки.

П р и м е р . Значение 123.89, помещенное в переменную с типом NUMBER(5,1) будет округлено до значения 123.9, а при помещении в переменную с типом NUMBER(5, –1) будет округлено до 120.

Следует отметить, что язык SQL поддерживает типы данных стандарта ANSI SQL. Если такой тип данных (INTEGER, SMALLINT, DECIMAL, FLOAT и REAL и др.) встречается при определении типа столбца таблицы,

то имя типа сохраняется, но сами данные хранятся в виде, определяемом одним из типов БД Oracle.

Тип данных DATE представляет собой специальным образом организованный тип. Он хранит столетие, год, месяц, день, часы, минуты, секунды. Для выборки текущего дня и времени в стандартном формате используется функция SYSDATE. Стандартный формат даты, автоматически преобразуемый во внутреннее представление, записывается символьной строкой следующего вида: 'DD–MM–YY', где DD – день месяца, MM – значение месяца, а YY – значение года.

П р и м е р. Строка символов '11–01–04' будет представлять дату 11 января 2004 г.

Над переменными типа DATE можно выполнить арифметическое действие – вычитание. Результат операции определяет количество дней между этими двумя датами. К дате можно прибавить или отнять числовую константу, рассматриваемую как количество дней или часть дня.

П р и м е р. Для добавления месяца к текущей дате необходимо записать SYSDATE+30, а для добавления часа – SYSDATE+1/24.

Двоичные типы данных используются для хранения двоичных неструктурированных данных (звуковые файлы, файлы изображений и др.), обработка которых не поддерживается языком. К ним относятся типы RAW(длина) и LONGRAW. Максимальный размер строки типа RAW 2000 байт, минимальный – 1 байт. Длина переменных типа LONGRAW может достигать 2 Гб.

Большие объекты (LOB-объекты) представлены типами CLOB, BLOB и BFILE и предназначены для хранения неструктурированных данных большого объема – до 4 Гб. Тип данных CLOB хранит данные символьного типа, типы данных BLOB и BFILE используют для хранения двоичных данных. Столбцы типа LOB содержат не сами данные, а указатели на их местоположение. При этом типы CLOB и BLOB хранятся в специальных сегментах БД, а тип BFILE, являющийся внешним двоичным файлом, хранится как обычный файл. На их использование наложен ряд ограничений.

3. Операторы языка SQL.

Все операторы SQL делятся на группы:

1) операторы языка описания данных – DDL;
2) операторы языка манипулирования данными – DML;
3) операторы управления транзакциями;
4) операторы управления сеансом;

5) операторы управления системой.

К операторам DDL относятся следующие:

CREATE, ALTER, DROP, GRANT, REVOKE.

Оператор CREATE используется для создания объектов БД. К ним относятся:

1) TABLE – таблица базы данных;

2) VIEW – представление или вид; представляет собой виртуальную таблицу, которая строится на основе выбранной в результате выполнения запроса информации из одной или нескольких таблиц;

3) SEQUENCE – последовательность, последовательно выбираемые значения которой можно использовать для задания уникальных значений ключа;

4) INDEX – индекс, используемый для обеспечения более быстрого доступа к данным таблицы;

5) TRIGGER – хранимая в БД программная единица, запускаемая автоматически при наступлении определенных событий;

6) FUNCTION – хранимая в БД программная единица, вызываемая пользователем или другими программными единицами для выполнения;

7) PROCEDURE – хранимая в БД программная единица, вызываемая пользователем или другими программными единицами для выполнения;

8) USER – имя пользователя, имеющего доступ к информации БД;

9) ROLE – совокупность определенных привилегий, обеспечивающих возможность создания, удаления и модификации объектов БД.

Оператор ALTER используется для изменения объектов БД. Применяется по отношению ко всем перечисленным выше объектам.

Оператор DROP применяется для удаления всех вышеперечисленных объектов из БД.

Оператор GRANT позволяет наделить роль или пользователя различного вида привилегиями или ролями.

Оператор REVOKE отменяет предоставленные пользователям или ролям привилегии и роли.

К группе операторов DML относятся:

INSERT, DELETE, UPDATE, SELECT.

Первые три оператора позволяют осуществить соответственно вставку, удаление и модификацию строк таблиц. Оператор SELECT предназначен для построения запросов, в результате выполнения которых из таблиц выбирается вся необходимая информация. Запрос на выборку информации, включенный в запись некоторого другого оператора, образует подзапрос.

В группу операторов управления транзакциями входят следующие

операторы:

1) COMMIT – фиксация текущей транзакции;

2) ROLLBACK – откат текущей транзакции.

Транзакция представляет собой неделимую с точки зрения системы единицу работы, выполняемую системой. В случае успешного выполнения всех входящих в транзакцию действий ее результаты фиксируются (COMMIT). В противном случае состояние БД можно вернуть в исходное состояние, отменив транзакцию (ROLLBACK).

Операторы управления сеансом работы изменяют установки, задаваемые для сеанса работы в БД (ALTER SESSION, SET ROLE).

Операторы управления системой изменяют установки всей БД (ALTER SYSTEM).

4. Операции языка SQL.

Операции языка SQL делятся на ряд групп.

Арифметические операции представлены в свою очередь двумя группами операций:

1) унарные +, –

2) бинарные +, –, *, / .

Арифметические операции используются в выражениях для изменения знака операнда, сложения, вычитания, умножения и деления числовых величин. Унарные операции оперируют только с одним операндом, бинарные требуют при своем использовании два операнда.

В группе *операций над строками* имеется только одна операция – операция сцепления строк. Для ее обозначения используется комбинация двух символов вертикальная черта (||).

Операции сравнения применяются в основном в операторах DML при построении простых условий проверки для сравнения значения одного выражения со значением другого выражения. Результатом сравнения может быть либо TRUE, либо FALSE, либо UNKNOWN. Значение UNKNOWN может появиться в результате сравнения значений двух выражений, если одно из них или оба имели значение NULL. Над значениями двух выражений X и Y могут быть выполнены следующие операции сравнения:

1) X = Y – проверка значений выражений X и Y на равенство; результат равен TRUE, если указанное соотношение выполняется;

2) X != Y, X<>Y, X ^= Y – проверка значений выражений X и Y на неравенство; результат равен TRUE, если указанное соотношение выполняется;

3) X < Y, X > Y, X >= Y, X <= Y – проверка значений выражений X и Y на соотношение «меньше, чем», «больше, чем», «больше или равно», «меньше или равно»; результат равен TRUE, если указанное соотношение выполняется;

4) X [NOT] BETWEEN A AND B – проверка, (не) находится ли значение выражения X в указанном диапазоне, определяемом значениями выражений A и B; результат равен TRUE, если указанное соотношение выполняется;

5) X IN (список выражений | подзапрос) – проверка значения выражения X на равенство некоторому элементу из списка значений выражений или множества значений, возвращенных подзапросом; результат равен TRUE, если указанное соотношение выполняется хотя бы для одного элемента списка выражений или множества значений, возвращенных подзапросом;

6) X NOT IN (список выражений | подзапрос) – проверка значения выражения X на неравенство ни одному элементу из списка значений выражений или множества значений, возвращенных подзапросом; результат равен TRUE, если указанное соотношение выполняется для всех элементов списка выражений или множества значений, возвращенных подзапросом;

7) X LIKE Z – проверка значения выражения X на подобие; результат проверки равен TRUE, если X совпадает с шаблоном Z. Шаблон представляет собой символьную строку, внутри которой символ '%' используется для сопоставления с любой строкой из нуля или более символов, кроме NULL – строки, а символ подчеркивания (_) сопоставляется с любым одиночным символом;

8) X IS [NOT] NULL – проверка значения выражения X на (не) пустое значение NULL; результат равен TRUE, если указанное соотношение выполняется;

9) операция сравнения с квантором ANY позволяет сравнивать проверяемое значение со всеми элементами из заданного списка значений выражений или множества значений, возвращенных подзапросом; результат проверки равен TRUE, если указанная операция сравнения (=, !=, >,<, >=, <=) выполняется хотя бы для одного элемента списка выражений или множества значений, возвращенных подзапросом;

10) операция сравнения с квантором ALL позволяет сравнивать проверяемое значение со всеми элементами из заданного списка значений выражений или множества значений, возвращенных подзапросом; результат проверки равен TRUE, если указанная операция сравнения (=, !=, >,<, >=, <=) выполняется для всех элементов списка выражений или множества значений, возвращенных подзапросом;

11) операция сравнения EXISTS проверяет результат выполнения

подзапроса; результат проверки равен TRUE, если подзапрос возвращает не пустое множество значений.

Логические операции представлены стандартными логическими операциями: NOT, AND, OR, используемыми при построении сложных условий проверки, в которых простые условия объединяются в более сложное условие с помощью логических операций.

Логические операции выполняются в трехзначной логике, которая задается следующими таблицами истинности:

OR	True	False	Unknown
False	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

NOT	True	False	Unknown
	False	True	Unknown

Операции над множествами позволяют выполнить определенные действия над выбираемыми в результате выполнения одного или нескольких запросов группами строк. Естественно, что структуры этих строк должны совпадать по количеству, порядку расположения и типу данных входящих в них элементов. К ним относятся следующие операции:

1) UNION ALL – объединяет все строки, извлеченные одним или несколькими запросами, включая повторяющиеся;

2) UNION – объединяет все строки, извлеченные одним или несколькими запросами, с устранением дублирующих строк;

3) INTERSECT – объединяет только те строки, которые присутствуют в результатах выполнения каждого из запросов, с устранением дублирующих строк;

4) MINUS – объединяет все неповторяющиеся строки, извлеченные первым запросом, но не извлеченные вторым.

Класс *других операций* содержит две операции: операцию внешнего соединения (+) и специальную операцию PRIOR. Операция внешнего соединения используется при выборе информации из нескольких таблиц в

том случае, если из одной таблицы необходимо выбрать все строки, а из остальных таблиц только те строки, для которых выполняются определенные условия.

Операция PRIOR устанавливает взаимосвязь между родительскими и дочерними строками при построении иерархических запросов.

5. Функции языка SQL.

Наиболее часто используемые группы функций языка SQL:

1. *Числовые функции*. Они предназначены для вычисления степени числа, абсолютного значения, округления и усечения числа с заданной точностью, вычисления тригонометрических значений. К числовым функциям относятся:

– Функция ABS(n) возвращает абсолютное значение аргумента n, имеющего числовой тип.

– Функция ROUND(n, [r]) осуществляет округление значения аргумента n, имеющего числовой тип, с точностью до количества указанных знаков r. При этом если значение r положительно, то округление производится до указанного количества знаков после запятой, если значение r отрицательно, то округление производится до указанного количества знаков до запятой. При r = 0 функция возвращает округленную целую часть аргумента n.

– Функция MOD(m, n) возвращает остаток от деления целочисленного аргумента m на целочисленный аргумент n.

– Функция POWER(m, n) возвращает аргумент m, имеющий числовой тип, возведенный в степень, заданную аргументом n, имеющим также числовой тип.

– Функция SQRT(m) возвращает квадратный корень из аргумента n, имеющего числовой тип.

Символьные функции предназначены для работы со строками. Они могут возвращать либо строку, либо целое значение. Ниже приводится описание некоторых символьных функций.

1. Функция UPPER(str) возвращает строку str, все символы которой преобразованы в верхний регистр.

2. Функция LENGTH(str) возвращает длину строки str в символах.

3. Функция SUBSTR(str, n, m) выделяет из строки str подстроку длины n, начиная с символа в позиции m.

4. Функция LPAD(str, n, chr) возвращает строку str, дополненную слева указанным символом chr до указанной длины n.

5. Функция RPAD(str, n, chr) возвращает строку str, дополненную справа

указанным символом chr до указанной длины n.

Функции работы с датами предназначены для работы с данными типа DATE. Ниже описываются некоторые функции этой группы.

1. Функция ADD_MONTHS(data, n) добавляет к указанной дате (аргумент data) или вычитает из нее n месяцев.

2. Функция MONTHS_BETWEEN(data1, data2) возвращает количество месяцев, находящихся между указанными датами data1 и data2.

3. Функция LAST_DAY(data) возвращает последний день месяца, указанного датой data.

Функции преобразования типа в основном используются для преобразования данных символьного типа в числовой или в тип DATE и, наоборот, для преобразования данных числового типа или типа DATE в символьный тип. Преобразование осуществляется в соответствии с задаваемым форматом. Формат преобразования имеет вид символьной строки, где каждый символ или группа символов имеет определенное назначение.

1. Функция TO_CHAR(d1, [fmt]) преобразует значение d1 типа DATE в значение типа VARCHAR2 по формату fmt.

2. Функция TO_CHAR(n1, [fmt]) преобразует значение n1 типа NUMBER в значение типа VARCHAR2 по формату fmt.

3. Функция TO_DATE(char, [fmt]) преобразует значение char типа CHAR или VARCHAR2 в значение типа DATE по формату fmt.

4. Функция TO_NUMBER(char, [fmt]) преобразует значение char типа CHAR или VARCHAR2 в значение типа NUMBER по формату fmt.

В форматах для даты используются следующие группы символов:

1) DD – задает номер дня месяца в диапазоне от 1 до 31;

2) DAY – задает полное название дня недели;

3) MON – задает краткое название месяца;

4) MONTH – задает полное название месяца;

5) YY – задает две последние цифры номера календарного года;

6) YYYY – задает полный номер календарного года.

В форматах для чисел используются следующие символы:

1) цифра 9 задает цифру;

2) символ точка (.) задает десятичную точку;

3) цифра 0 задает обязательный нуль;

4) буква s задает обязательное наличие знака {+; –};

5) символ $ задает знак доллара, проставляемый в начале числа.

Групповые функции выполняют операции над группами строк.

1. Функция COUNT({*}) – возвращает количество строк в группе.

2. Функция COUNT([DISTINCT] выражение) – возвращает количество строк в группе, игнорируя значение NULL.

3. Функция SUM([DISTINCT] выражение) – возвращает сумму значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

4. Функция AVG([DISTINCT] выражение) – возвращает среднее значение указанного выражения для группы строк или списка значений, игнорируя значение NULL.

5. Функция MIN([DISTINCT] выражение) – возвращает минимальное из значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

6. Функция MAX([DISTINCT] выражение) – возвращает максимальное из значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

7. Фраза DISTINCT предписывает групповым функциям рассматривать только различные значения выражения.

Выполняются также *другие функции.* Относящаяся к этой группе функция NVL(выражение1, выражение2) обрабатывает пустое значение NULL. Если значение выражения1 равно NULL, то функция возвращает значение выражения2; если же значение выражения1 не равно NULL, то функция возвращает значение выражения1.

6. Создание, модификация и удаление таблиц.

Рассмотрим основные операции, выполняемые над таблицами.

Создание таблицы выполняется с помощью оператора CREATE, упрощенный синтаксис которого имеет следующий вид:

```
CREATE TABLE имя_таблицы
{({ имя_столбца тип_данных [DEFAULT значение]
[ограничения_столбца] | ограничение_таблицы}
[,{ имя_столбца тип_данных [DEFAULT значение]
[ограничения_столбца] | ограничение_таблицы}]...) |
AS подзапрос}
```

Таблица может быть создана либо стандартным образом через описание ее компонентов, либо в результате выполнения некоторого подзапроса. Подзапрос – это обычный запрос на выборку информации, реализуемый оператором SELECT. При создании таблицы задаваемые имена таблиц и имена столбцов должны удовлетворять правилам, предписываемым идентификаторам. При этом естественно, что имена, присваиваемые

таблицам, должны быть уникальными в схеме пользователя, а имена столбцов – в рамках одной таблицы. Для каждого столбца указывается тип данных и, если необходимо, значение, вставляемое в столбец по умолчанию (DEFAULT значение). Важным элементом при создании таблиц является задание *ограничений целостности* данных, которые позволяют отслеживать правильность модификации имеющихся данных или вставляемых в таблицу новых данных. Ограничения целостности данных делятся на два типа: ограничения столбца и ограничения таблицы. Ограничения столбца позволяют определить условия, которым должны удовлетворять значения соответствующего столбца; ограничения целостности данных, накладываемые на таблицу, позволяют проверить правильность всех добавляемых или модифицируемых строк таблицы. Ограничение может быть именованным или безымянным. Удобнее использовать именованные ограничения, поскольку при выдаче информации, связанной с возникшим нарушением одного из ограничений, выдается и имя этого ограничения, что весьма полезно для исправления ошибок в дальнейшем. Задание ограничений на столбец или ограничений на таблицу осуществляется по следующему синтаксису:

[CONSTRAINT имя_ограничения] тип_ограничения

Имеются следующие типы ограничений, накладываемых на столбец:

1) PRIMARY KEY – это ограничение требует, чтобы вводимые в столбец значения были уникальными и отличными от пустых, поскольку они будут использоваться в качестве первичного ключа, однозначно идентифицирующего запись; первичный ключ определяется для таблицы только единожды;

2) UNIQUE – это ограничение требует, чтобы вводимые в столбец значения в рамках одной таблицы были уникальными;

3) NOT NULL – это ограничение требует обязательного присутствия в столбце некоторого значения;

4) CHECK(выражение) – это ограничение позволяет подвергнуть определенной проверке вставляемое в столбец значение; если условия, наложенные на вставляемые значения, не выполняются, то значения в столбец не помещаются;

5) REFERENCES – это ограничение позволяет установить взаимосвязь значений данного столбца со значениями столбца другой таблицы. Взаимосвязь обеспечивается использованием следующей конструкции:

REFERENCES имя_таблицы[(имя_столбца)] [ON DELETE CASCADE]

При внесении значения в столбец создаваемой таблицы система будет автоматически проверять наличие аналогичного значения в указанном

столбце указанной таблицы. При этом естественно, что для обеспечения однозначности устанавливаемой взаимосвязи все значения, находящиеся в указанном столбце, на которые производится ссылка, должны иметь ограничение UNIQUE или PRIMARY KEY. Если в качестве имени столбца указанной таблицы используется первичный ключ, то имя столбца можно не указывать. Таблица, на чей столбец ссылается другая таблица, называется главной, а таблица, ссылающаяся на нее, – подчиненной. Конструкция ON DELETE CASCADE указывает, что при удалении строк в главной таблице автоматически осуществляется удаление соответствующих строк и в подчиненной таблице.

Ограничения на таблицу во многом напоминают ограничения столбца, но при этом обычно задействуют, как правило, несколько столбцов. Например, можно задать ограничение PRIMARY KEY, указав список имен столбцов, тем самым определив составной первичный ключ. При этом для столбцов, указываемых в списке, должны быть заданы ограничения UNIQUE и NOT NULL.

Используя следующую форму записи, можно определить *составной внешний ключ* для таблицы:

FOREIGN KEY (список_имен_столбцов)

REFERENCES имя_таблицы(список_имен_столбцов)

[ON DELETE CASCADE]

Естественно, что в случае составного внешнего ключа перечень столбцов в подчиненной таблице и перечень столбцов в главной таблице должны совпадать по количеству, типу данных и порядку следования. Конструкция ON DELETE CASCADE позволяет обеспечить целостность и непротиворечивость данных при изменениях, которые затрагивают значения столбцов главной таблицы, являющихся внешним ключом по отношению к подчиненной таблице.

Если ограничение CHECK затрагивает значения нескольких столбцов, увязывая их в некоторое достаточно сложное условие, то такое ограничение также удобно определить как ограничение на таблицу. Для генерации и вставки в столбец таблицы уникальных значений можно создать специально предназначенный для этих целей объект – последовательность. Создание последовательности выполняется с помощью оператора CREATE по следующему упрощенному синтаксису:

CREATE SEQUENCE имя_последовательности

[START WITH начальное_значение] [INCREMENT BY шаг];.

В самом простейшем случае генерируется последовательность целых чисел от 1 до 1027 с шагом единица. Конструкция INCREMENT BY

позволяет указать шаг изменения значений последовательности. Конструкция START WITH позволяет задать начальное значение генерируемой последовательности, которое при ее отсутствии устанавливается равным единице. Для вставки в столбец текущего значения последовательности нужно указать имя_последовательности. CURRVAL, а для вставки в столбец измененного по правилам формирования последовательности следующего значения используется имя_последовательности.NEXTVAL. Последовательности являются самостоятельными объектами БД, одна и та же последовательность может быть использована для задания уникальных значений столбцов нескольких таблиц; при удалении или модификации последовательности значения, созданные ею, сохраняются в таблицах БД.

Вставка строк в таблицу осуществляется с помощью оператора INSERT, который имеет следующий синтаксис:

INSERT INTO имя_таблицы [(список_столбцов)]

{VALUES (значение1 [, значение2] ...) | подзапрос};

Если список столбцов не указывается, то список значений в конструкции VALUES должен содержать значения для всех столбцов таблицы, причем порядок их следования должен однозначно соответствовать порядку их следования в строке. Использование подзапроса позволяет перенести строки из некоторой таблицы в создаваемую таблицу.

Удаление строк из таблицы осуществляется с помощью оператора DELETE, который имеет следующий синтаксис:

DELETE [FROM] имя_таблицы [WHERE условие];

При отсутствии ключевого слова WHERE из таблицы удаляются все строки, но сама таблица остается.

Модификация строк таблицы реализуется с помощью оператора UPDATE, форма записи которого приведена ниже. При отсутствии условия модифицируются все строки таблицы для указанного списка столбцов:

UPDATE имя_таблицы

SET {(имя_столбца1 [, имя_столбца2] ...) = (подзапрос) |

имя_столбца1=значение1, имя_столбца2={значение2 | (подзапрос)}}

[WHERE условие];

Изменение структуры таблицы выполняется оператором ALTER TABLE, с помощью которого можно осуществить добавление одного или нескольких новых столбцов в таблицу, изменение характеристик у одного или нескольких столбцов, добавление ограничения столбца или таблицы или удаление ограничений столбца или таблицы. Примеры его использования приведены ниже.

Удаление таблицы можно выполнить с помощью следующего оператора:

DROP TABLE имя_таблицы [CASCADE CONSTRAINTS];

При наличии конструкции CASCADE CONSTRAINTS вместе с удалением таблицы уничтожаются ограничения внешнего ключа в других таблицах.

П р и м е р ы

Постоянно работающий магазин напрямую контактирует с издательствами и имеет постоянный штат продавцов. Директор магазина должен иметь сведения:

1) о поступивших книгах: название книги, фамилия автора, цена, издательство, жанр;

2) о распределении книг среди продавцов: название книги, фамилия продавца, количество экземпляров, дата поступления.

1. Необходимо создать таблицы: BOOKS, BOOKS_DELIVERY, указав все необходимые ограничения целостности данных.

Перечень, названия и тип данных столбцов таблицы BOOKS:

Код книги	CODE_BOOK	Number(5)
Название книги	TITLE	Varchar2(25)
ФИО автора	AUTHOR	Varchar2(20)
Цена книги	PRICE	Number(7)
Издательство	PUBLISH_HOUSE	Varchar2(15)
Жанр	GENRE	Varchar2(15)

Информация таблицы BOOKS:

Код кн.	Назв. книги	ФИО автора	Цена	Изд-во	Жанр
1	Гибель Богов	Перумов Н.	345	Аст	Фантастика
2	Казаки	Толстой Л.	5568	Нова	Роман
3	Ярость	Перумов Н.	1385	Аст	Детектив
4	Дюна	Герберт Ф.	2668	Нова	Фантастика
5	Гибель Титана	Кристи А.	2345	Аст	Роман
6	Дети Дюны	Герберт Ф.	2500	Аст	Фантастика

Перечень, названия и тип данных столбцов таблицы BOOKS_DELIVERY:

Код операции	CODE_OPERATION	Number(10)
Код книги	CODE_BOOK	Number(5)
ФИО продавца	SALESMAN	Varchar2(20)
Количество единиц	QUANTITY	Number(4)
Дата поставки	DATE_DELIVERY	Date

Информация таблицы BOOKS_DELIVERY:

Код опер.	Код книги	ФИО Продавца	Кол-во ед.	Дата поставки
1	1	Иванов И. И.	5	20-01-2006
2	3	Иванов И. И.	5	10-02-2006
3	2	Петров П. П.	4	25-01-2006
4	4	Петров П . П.	4	20-02-2006
5	4	Иванов И. И.	4	20-02-2006

Для создания таблицы BOOKS воспользуемся оператором CREATE TABLE. Столбец CODE_BOOK, содержащий уникальный код книги, является ключевым, и по отношению к его значениям устанавливаем ограничение PRIMARY KEY. За значениями для этого столбца пользователь должен следить сам. Ограничение именуется как PK_BOOKS. Поскольку столбец TITLE не может иметь пустые значения, на него накладываем ограничение NOT NULL. На значения столбца PRICE накладываем ограничение, связанное со стоимостью книги, она должна быть не менее 100 руб. Ограничение получает имя PRICE_BOOKS. Если таблица BOOKS была уже создана ранее, то перед повторным созданием ее следует удалить следующим оператором:

```
DROP TABLE BOOKS;
```

Следующий оператор CREATE языка SQL создает таблицу BOOKS с необходимыми ограничениями целостности данных:

```
CREATE TABLE BOOKS (CODE_BOOK NUMBER(5)
CONSTRAINT PK_BOOKS PRIMARY KEY,
TITLE VARCHAR2(25) CONSTRAINT TITLE_BOOKS NOT NULL,
AUTHOR VARCHAR2(20),
PRICE NUMBER(7) CONSTRAINT PRICE_BOOKS
CHECK(PRICE >100),
PUBLISH_HOUSE VARCHAR2(15), GENRE VARCHAR2(15));
```

Вставка строк в таблицу BOOKS осуществляется следующей совокупностью операторов:

```
INSERT INTO BOOKS VALUES(1,'Гибель Богов','Перумов Н.', 345,
'Аст', 'Фантастика');
INSERT INTO BOOKS VALUES(2,'Казаки','Толстой Л.', 5568, 'Нова',
'Роман');
INSERT INTO BOOKS VALUES(3,'Ярость','Перумов Н.',1385,'Аст',
'Детектив');
INSERT INTO BOOKS VALUES(4,'Дюна', 'Герберт Ф.', 2668, 'Нова',
'Фантастика');
INSERT INTO BOOKS VALUES(5,'Гибель Титана', 'Кристи А.', 2345,
'Аст', 'Роман');
INSERT INTO BOOKS VALUES(6,'Дети Дюны', 'Герберт Ф.', 2500,
'Аст', 'Фантастика');
```

Просмотр введенных значений можно выполнить с помощью следующего оператора SQL:

```
SELECT * FROM BOOKS;
```

Для создания таблицы BOOKS_DELIVERY также воспользуемся оператором CREATE TABLE. Столбец CODE_OPERATION, содержащий уникальный код операции, является ключевым, и по отношению к его значениям устанавливаем ограничение PRIMARY KEY. Это ограничение получает имя DELIVERY_PR. Для генерации уникальных значений этого столбца используем предварительно созданную с помощью оператора CREATE SEQUENCE последовательность CODE_OP:

```
CREATE SEQUENCE CODE_OP;
```

Поскольку столбец CODE_BOOK должен содержать только те значения, которые присутствуют в соответствующем столбце таблицы BOOKS, необходимо задать соответствующее ограничение на значения столбца CODE_BOOK. Это ограничение можно сделать как ограничением столбца, так и ограничением таблицы, задав ему имя DELIVERY_FK. Конструкция ON DELETE CASCADE обеспечит при удалении из таблицы BOOKS строк, содержащих значения внешнего ключа, автоматическое удаление строк и из таблицы BOOKS_DELIVERY. Для столбца DATE_DELIVERY значением по умолчанию устанавливаем текущую дату (SYSDATE). Если таблица BOOKS_DELIVERY была уже создана ранее, то перед повторным созданием ее следует удалить следующим оператором:

```
DROP TABLE BOOKS_DELIVERY;
```

Следующий оператор создает таблицу BOOKS_DELIVERY с необходимыми ограничениями целостности данных:

```
CREATE TABLE BOOKS_DELIVERY (
CODE_OPERATION NUMBER(10)
```

```
CONSTRAINT DELIVERY_PR PRIMARY KEY,
CODE_BOOK NUMBER(5), SALESMAN VARCHAR2(20),
QUANTITY NUMBER(4),
DATE_DELIVERY DATE DEFAULT SYSDATE,
CONSTRAINT DELIVERY_FK FOREIGN KEY(CODE_BOOK)
REFERENCES BOOKS(CODE_BOOK) ON DELETE CASCADE);
```

Вставка строк в таблицу BOOKS_DELIVERY осуществляется использованием следующей совокупности операторов INSERT:

```
INSERT INTO BOOKS_DELIVERY VALUES(CODE_OP.NEXTVAL,
1, 'Иванов И. И.', 5, '20-01-2006');
INSERT INTO BOOKS_DELIVERY VALUES(CODE_OP.NEXTVAL,
3, 'Иванов И. И.', 5, '10-02-2006');
INSERT INTO BOOKS_DELIVERY VALUES(CODE_OP.NEXTVAL,
2, 'Петров П. П.', 4, '25-01-06');
INSERT INTO BOOKS_DELIVERY VALUES(CODE_OP.NEXTVAL,
4, 'Петров П. П.', 4, '20-02-06');
INSERT INTO BOOKS_DELIVERY VALUES(CODE_OP.NEXTVAL,
4, 'Иванов И. И.', 4, '20-02-06');
```

Просмотр введенных значений можно выполнить с помощью следующего оператора SQL:

```
SELECT * FROM BOOKS_DELIVERY;
```

2. Создать таблицу BOOKS1, структура которой идентична структуре таблицы BOOKS, и перенести в нее строки с информацией о книгах жанра «Фантастика» и «Роман» из таблицы BOOKS.

Следующий набор операторов SQL создает таблицу BOOKS1 и переносит в нее строки из таблицы BOOKS:

```
CREATE TABLE BOOKS1 (CODE_BOOK NUMBER(5)
CONSTRAINT PK_BOOKS1 PRIMARY KEY,
TITLE VARCHAR2(25) CONSTRAINT TITLE_BOOKS1
NOT NULL, AUTHOR VARCHAR2(15),
PRICE NUMBER(7) CONSTRAINT PRICE_BOOKS1
CHECK(PRICE >100), PUBLISH_HOUSE VARCHAR2(15),
GENRE VARCHAR2(15));
INSERT INTO BOOKS1 SELECT CODE_BOOK, TITLE, AUTHOR,
PRICE, PUBLISH_HOUSE, GENRE FROM BOOKS WHERE
GENRE = 'Фантастика' OR GENRE = 'Роман';
```

Аналогичные действия выполняются следующим оператором:

```
CREATE TABLE BOOKS1 AS
SELECT CODE_BOOK, TITLE, AUTHOR, PRICE,
```

```
PUBLISH_HOUSE, GENRE FROM BOOKS WHERE
GENRE = 'Фантастика' OR GENRE = 'Роман';
```

Просмотр введенных значений можно выполнить с помощью следующего оператора SQL:

```
SELECT * FROM BOOKS1;
```

3. Выполнить с помощью оператора UPDATE следующие изменения в таблице BOOKS:

а) заменить в таблице BOOKS в строке с фамилией автора «Кристи А.» в столбце AUTHOR фамилию «Кристи А.» на «Агата Кристи» и в столбце GENRE жанр «Роман» на жанр «Детектив»;

б) заменить в таблице BOOKS цену книги «Гибель Титана» на цену книги «Казаки».

Следующий набор операторов выполнит указанные действия:

```
UPDATE BOOKS SET GENRE='Детектив', AUTHOR='Агата Кристи'
WHERE AUTHOR='Кристи А.';
UPDATE BOOKS SET PRICE=
(SELECT PRICE FROM BOOKS WHERE TITLE='Казаки')
WHERE TITLE='Гибель Титана';
```

4. Удалить из таблицы BOOKS все строки, содержащие информацию о книгах в жанре «Роман». Используем следующий оператор:

```
DELETE BOOKS WHERE GENRE = 'Роман';
```

5. Добавить в таблицу BOOKS новый столбец PRICE_У_Е с типом данных NUMBER(7,2) и пересчитать цену книг в условных единицах. Следующие операторы выполнят сначала добавление нового столбца в таблицу, а затем заполнят его соответствующими значениями:

```
ALTER TABLE BOOKS ADD PRICE_У_Е NUMBER(7,2);
UPDATE BOOKS SET PRICE_У_Е=PRICE/2155;
```

Отметим, что при добавлении нескольких новых столбцов в таблицу они заключаются в скобки:

```
ALTER TABLE BOOKS ADD (YEAR_PUBLISH CHAR(4),
TOWN_PUBLISH CHAR(15));
```

Для изменения характеристик столбца указывается ключевое слово MODIFY, имя столбца и новые характеристики:

```
ALTER TABLE BOOKS MODIFY (YEAR_PUBLISH NUMBER(4),
TOWN_PUBLISH CHAR(25));
```

Для добавления ограничения целостности данных необходимо указать ключевое слово CONSTRAINT, имя ограничения и само ограничение. При этом необходимо помнить, что добавляемое ограничение целостности данных не должно противоречить данным, находящимся в таблице:

ALTER TABLE BOOKS ADD CONSTRAINT ZZ CHECK(PRICE>200);

Для временного отключения проверки ограничения целостности с именем ZZ необходимо использовать команду:

ALTER TABLE BOOKS DISABLE CONSTRAINT ZZ;

Для включения проверки этого ограничения целостности можно использовать команду:

ALTER TABLE BOOKS ENABLE CONSTRAINT ZZ;

Для удаления ограничения целостности с именем ZZ необходимо указать ключевые слова DROP и CONSTRAINT и имя удаляемого ограничения

ALTER TABLE BOOKS DROP CONSTRAINT ZZ;

7. Выбор информации из базы данных.

Выбор информации из одной или нескольких таблиц БД осуществляется при помощи оператора SELECT, упрощенный синтаксис которого имеет следующий вид:

SELECT [DISTINCT] {* | {[имя_таблицы.] имя_столбца | выражение }
[псевдоним] [,{[имя_таблицы.] имя_столбца | выражение} [псевдоним]]...}
FROM {имя_таблицы | имя_представления | подзапрос} [псевдоним]
[,{имя_таблицы | имя_представления | подзапрос} [псевдоним]…]
[WHERE условие]
[GROUP BY выражение1 [, выражение2...] [HAVING условие]]
[{UNION | UNION ALL | INTERSECT | MINUS} SELECT оператор]
[ORDER BY выражение1 [ASC | DESC]
[, выражение2 [ASC | DESC]] ...];

Построение списка выбора. В операторе SELECT при формировании списка выбора, состоящего из имен столбцов и выражений, для построения выражений могут использоваться имена столбцов таблиц и представлений, литералы, функции, соединяемые знаками арифметических действий. Если необходимо указать имена столбцов, которые имеют одинаковые идентификаторы в двух разных таблицах, то к этим именам через точку необходимо приписать имя таблицы. При необходимости сложному выражению можно присвоить псевдоним, который будет использован в дальнейшем тексте оператора. Все имена таблиц, имена столбцов которых использовались для построения выражений в списке выбора, обязательно должны быть перечислены в тексте оператора после ключевого слова FROM.

Упорядочение строк. Строки, возвращаемые SELECT-запросом, могут быть упорядочены по возрастанию или убыванию значений определенных выражений. В качестве выражения может использоваться имя столбца. Для реализации этой процедуры используется конструкция ORDER BY, в которой указывается перечень, состоящий из одного или нескольких выражений, разделенных запятыми, по значениям которых и осуществляется упорядочение. По умолчанию извлекаемые строки упорядочиваются по возрастанию значений указанных в перечне выражений. Для того чтобы задать другой вариант упорядочения строк, после каждого выражения или группы выражений в перечне указывается либо ASC (по возрастанию), либо DESC (по убыванию значений), а перечисление этих групп осуществляется через запятую.

Условие выбора строк. В конструкции WHERE при построении условия, которому должны удовлетворять выбираемые строки, можно использовать весь имеющийся в языке SQL набор операций сравнения и логических операций. Для того чтобы задать конкретное значение в условии, можно воспользоваться переменной подстановки, которая позволяет ввести необходимое значение в момент выполнения запроса. Переменная подстановки определяется наличием символа & в начале ее имени.

Подзапросы. Подзапросы, или вложенные запросы, применяются для возврата группы строк или множества значений, которые будут использованы родительским запросом. В зависимости от формы построения подзапрос может выполняться либо один раз для родительского запроса, либо один раз для каждой строки, извлеченной родительским запросом. В последнем случае такой подзапрос называется коррелированным подзапросом. Характерным признаком коррелированного подзапроса является наличие в его фразе WHERE ссылок на столбцы родительского запроса.

Объединение строк. В многокомпонентном запросе можно определенным образом объединить в единое целое группы строк, извлекаемые отдельно выполняемыми запросами. Чтобы задать порядок объединения, используется одно из ключевых слов конструкции UNION | UNION ALL | INTERSECT | MINUS.

П р и м е р ы

1. Выбрать из таблицы BOOKS всю информацию о книгах:

```
SELECT * FROM BOOKS;
```

2. Выбрать из таблицы BOOKS всю информацию о первых трех книгах:

```
SELECT * FROM BOOKS WHERE ROWNUM < 4;
```

3. Выбрать из таблицы BOOKS информацию о книгах с указанием фамилии автора, названия и цены и упорядочить ее по возрастанию значений столбца AUTHOR и убыванию значений столбца PRICE; фамилии авторов и названия вывести заглавными буквами:

```
SELECT UPPER(AUTHOR), UPPER(TITLE), PRICE FROM BOOKS
ORDER BY AUTHOR ASC, PRICE DESC;
```

4. Выбрать из таблицы BOOKS информацию (фамилия автора, название) о книгах в жанре «Роман»;

```
SELECT AUTHOR, TITLE FROM BOOKS WHERE GENRE = 'Роман';
```

5. Выбрать из таблицы BOOKS информацию (фамилия автора, название, жанр) о книгах в жанре не «Роман»;

а)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS WHERE
GENRE != 'Роман';
```

б)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS WHERE
GENRE <> 'Роман';
```

в)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS
MINUS
SELECT AUTHOR, TITLE, GENRE FROM BOOKS
WHERE GENRE = 'Роман';
```

6. Выбрать из таблицы BOOKS информацию (фамилия автора, название, цена) о книгах стоимостью больше 260 и меньше 1000;

```
SELECT AUTHOR, TITLE, PRICE FROM BOOKS WHERE PRICE
BETWEEN 260 AND 1000;
```

7. Выбрать из таблицы BOOKS информацию (фамилия автора, название, жанр) о книгах в жанрах «Роман» и «Детектив»:

а)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS
WHERE GENRE = 'Роман' OR GENRE = 'Детектив';
```

б)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS
WHERE GENRE IN ('Роман', 'Детектив');
```

в)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS
WHERE GENRE = 'Роман'
UNION
SELECT AUTHOR, TITLE, GENRE FROM BOOKS
WHERE GENRE = 'Детектив';
```

8. Выбрать из таблицы BOOKS информацию (фамилия автора, название, жанр) о книгах жанров «Роман», «Детектив» издательства «Аст»:

а)
```
SELECT AUTHOR, TITLE, GENRE FROM BOOKS WHERE
  (GENRE = 'Роман' OR GENRE = 'Детектив') AND
  PUBLISH_HOUSE = 'Аст';
```

б) SELECT AUTHOR, TITLE, GENRE FROM BOOKS WHERE
GENRE IN ('Роман', 'Детектив') AND PUBLISH_HOUSE = 'Аст';

9. Выбрать из таблицы BOOKS информацию (фамилия автора, название) о книгах, название которых начинается со слова «Гибель»:

а) SELECT AUTHOR, TITLE FROM BOOKS
WHERE TITLE LIKE 'Гибель%';

б) SELECT AUTHOR, TITLE FROM BOOKS
WHERE SUBSTR(TITLE, 1, 6) = 'Гибель';

10. Выбрать из таблицы BOOKS информацию (фамилия автора, название, жанр) о книгах, относящихся к указанному жанру. Необходимое значение задать, используя переменную подстановки:

SELECT AUTHOR, TITLE, GENRE FROM BOOKS
WHERE GENRE = '&GANR' ORDER BY AUTHOR;

В ответ на запрос, выдаваемый системой, набрать одно из значений столбца GENRE (Роман, Фантастика, Детектив).

11. Выбрать из таблицы BOOKS информацию о книгах, имеющихся в других издательствах, того же жанра, что и в издательстве «Нова»:

а) SELECT * FROM BOOKS WHERE GENRE IN
(SELECT DISTINCT GENRE FROM BOOKS WHERE
PUBLISH_HOUSE = 'Нова') AND PUBLISH_HOUSE <> 'Нова';

б) SELECT * FROM BOOKS WHERE GENRE = ANY
(SELECT DISTINCT GENRE FROM BOOKS WHERE
PUBLISH_HOUSE = 'Нова') AND PUBLISH_HOUSE <> 'Нова';

12. Выбрать из таблицы BOOKS список фамилий авторов, чьи книги имеются в каждом из издательств:

SELECT AUTHOR FROM BOOKS WHERE PUBLISH_HOUSE = 'Аст'
INTERSECT
SELECT AUTHOR FROM BOOKS WHERE PUBLISH_HOUSE = 'Нова';

Группирование строк. Строки, возвращаемые SELECT-запросом, могут быть объединены в группы на основе значений определенного выражения для каждой строки. Примером такого группирования может служить объединение в группы книг одного жанра, информация о которых имеется в таблице BOOKS. Так как в таблице присутствуют книги трех жанров, то будут сформированы только три группы строк. Применив к каждой группе функцию SUM для столбца, содержащего значение цены, можно получить суммарную величину стоимости книг по каждому жанру. Для осуществления группирования используется конструкция GROUP BY оператора SELECT, в которой указывается перечень, состоящий из одного

или нескольких выражений, разделенных запятыми, по значениям которых и осуществляется группирование. Если оператор SELECT содержит пункт GROUP BY, то список извлекаемых значений ограничен. Он может содержать константы, групповые функции, функцию SYSDATE и выражения, идентичные указанным в пункте GROUP BY. На формирование результирующих строк могут быть наложены определенные условия. Чтобы задать такое условие, используется ключевое слово HAVING.

П р и м е р ы

1. Выбрать из таблицы BOOKS информацию о количестве различных жанров:

```
SELECT COUNT(DISTINCT GENRE) FROM BOOKS;
```

2. Выбрать из таблицы BOOKS информацию о количестве, суммарной стоимости и максимальной стоимости имеющихся книг:

```
SELECT COUNT(CODE_BOOK), SUM(PRICE), MAX(PRICE)
FROM BOOKS;
```

3. Выбрать из таблицы BOOKS по каждому издательству информацию о количестве и суммарной стоимости изданных им книг, сгруппировав ее по жанрам:

```
SELECT PUBLISH_HOUSE, GENRE, COUNT(CODE_BOOK),
SUM(PRICE) FROM BOOKS GROUP BY PUBLISH_HOUSE,
GENRE;
```

4. Выбрать из таблицы BOOKS информацию о количестве и средней стоимости (округлив значение средней стоимости до двух знаков после запятой) книг в тех жанрах, где количество различных названий книг не менее 2:

```
SELECT GENRE, COUNT(DISTINCT TITLE),
ROUND(AVG(PRICE), 2) FROM BOOKS GROUP BY GENRE
HAVING COUNT(DISTINCT TITLE) >= 2;
```

5. Выбрать из таблицы BOOKS информацию о минимальной стоимости книг в жанре «Роман»:

```
SELECT MIN(PRICE) FROM BOOKS WHERE GENRE = 'Роман';
```

6. Выбрать из таблицы BOOKS информацию (фамилия автора, название, цена) о самой дешевой книге в жанре «Роман»:

```
SELECT AUTHOR, TITLE, PRICE FROM BOOKS
WHERE PRICE = (SELECT MIN(PRICE) FROM BOOKS
WHERE GENRE = 'Роман') AND GENRE = 'Роман';
```

7. Выбрать из таблицы BOOKS информацию (фамилия автора, название, жанр, цена) о книгах, имеющих максимальную стоимость в своем жанре:

а) SELECT AUTHOR, TITLE, GENRE, PRICE FROM BOOKS
WHERE (PRICE, GENRE) IN (SELECT MAX(PRICE), GENRE
FROM BOOKS GROUP BY GENRE);

б) SELECT AUTHOR, TITLE, BOOKS.GENRE, PRICE FROM
BOOKS, (SELECT GENRE, MAX(PRICE) МАКС FROM BOOKS
GROUP BY GENRE) P1
WHERE PRICE = МАКС AND BOOKS.GENRE = P1.GENRE;

в) SELECT AUTHOR, TITLE, GENRE, PRICE FROM BOOKS P1
WHERE PRICE = (SELECT MAX(PRICE) FROM BOOKS
WHERE BOOKS.GENRE = P1.GENRE);

Третий запрос содержит коррелированный подзапрос. Поскольку в своем условии подзапрос содержит ссылку на столбец родительского запроса, он будет выполняться один раз для каждой строки, извлекаемой родительским запросом. В первом и втором вариантах подзапрос не является коррелированным, он выполняется только один раз для родительского запроса.

8. Выбрать из таблицы BOOKS информацию (фамилия автора, название, цена) о книгах стоимостью больше средней стоимости книг:

```
SELECT AUTHOR, TITLE, PRICE FROM BOOKS
WHERE PRICE > (SELECT AVG(PRICE) FROM BOOKS);
```

9. Выбрать из таблицы BOOKS список жанров, по которым имеется наибольшее количество различных книг, с указанием количества книг:

```
SELECT GENRE, COUNT(DISTINCT TITLE) FROM BOOKS
GROUP BY GENRE HAVING COUNT(DISTINCT TITLE) =
(SELECT MAX(COUNT(DISTINCT TITLE)) FROM BOOKS
GROUP BY GENRE);
```

10. Выбрать из таблицы BOOKS информацию о книгах (фамилия автора, название), относящихся к жанрам, по которым имеется наибольшее количество различных книг:

```
SELECT AUTHOR, TITLE FROM BOOKS WHERE GENRE IN
(SELECT GENRE FROM BOOKS GROUP BY GENRE HAVING
COUNT(DISTINCT TITLE) =
(SELECT MAX(COUNT(DISTINCT TITLE)) FROM BOOKS
GROUP BY GENRE));
```

Выбор информации из нескольких таблиц (соединение). Соединение – это SELECT-запрос, который выбирает строки из двух или более таблиц. При этом запрос может извлекать любые столбцы из любой таблицы. Если хотя бы две из этих таблиц имеют одинаково названные столбцы, то имена таких столбцов должны уточняться именами таблиц, записываемых перед

именами столбцов через точку. Большинство SELECT-запросов с соединениями содержат условия, в которых сравниваются значения столбцов из разных таблиц. Такие условия называются условиями соединения. Эквисоединение – это соединение с использованием в условии соединения операции равенства. Таким образом, эквисоединение извлекает строки с эквивалентными значениями в указанных столбцах.

Декартово произведение таблиц строится при отсутствии в запросе условия соединения. В этом случае к каждой строке первой таблицы приписывается каждая строка второй таблицы.

Самосоединение соединяет таблицу саму с собой. При этом таблица появляется в списке FROM дважды и должна иметь дополнительное имя (псевдоним), чтобы можно было однозначно идентифицировать столбцы в условии соединения.

Внешнее соединение выдает все строки, которые удовлетворяют условию соединения, а также строки одной из таблиц, которые не удовлетворяют условию соединения. Чтобы записать запрос, который выполняет внешнее соединение таблиц A и B и выдает все строки из таблицы A, применим операцию внешнего соединения (+) ко всем столбцам из таблицы B в условиях соединения. Тогда для всех строк из таблицы A, для которых нет соответствующих строк в таблице B, система предоставит строку, содержащую NULL во всех выражениях в списке столбцов, которые содержат столбцы из таблицы B.

П р и м е р ы

1. Выбрать из таблиц BOOKS и BOOKS_DELIVERY информацию о книгах, поставленных продавцам за период с 24.01.2006 по 12.02.2006, указав для выводимого значения даты специальный формат вывода:

```
SELECT SALESMAN, AUTHOR, TITLE,
TO_CHAR(DATE_DELIVERY, ‘DD MONTH YYYY’)
FROM BOOKS, BOOKS_DELIVERY WHERE
BOOKS.CODE_BOOK = BOOKS_DELIVERY.CODE_BOOK AND
DATE_DELIVERY BETWEEN '24-01-06' AND '12-02-06'
ORDER BY SALESMAN, AUTHOR;
```

2. Выбрать из таблиц BOOKS и BOOKS_DELIVERY по указанному продавцу перечень издательств и жанров имеющихся у него книг без повторения; чтобы задать фамилию продавца, использовать переменную подстановки:

```
SELECT DISTINCT SALESMAN, PUBLISH_HOUSE, GENRE
FROM BOOKS, BOOKS_DELIVERY
```

```
WHERE  BOOKS_DELIVERY.CODE_BOOK = BOOKS.CODE_BOOK
AND  SALESMAN = '&SALESMAN'
ORDER  BY  PUBLISH_HOUSE, GENRE;
```

3. Выбрать из таблиц BOOKS и BOOKS_DELIVERY список продавцов, у которых в наличии не менее 10 книг, указав данные об общем количестве и суммарной стоимости имеющихся у них книг:

```
SELECT  SALESMAN, SUM(QUANTITY), SUM(PRICE*QUANTITY)
FROM  BOOKS, BOOKS_DELIVERY
WHERE  BOOKS_DELIVERY.CODE_BOOK = BOOKS.CODE_BOOK
GROUP  BY  SALESMAN  HAVING  SUM(QUANTITY) >= 10;
```

4. Выбрать из таблиц BOOKS и BOOKS_DELIVERY по каждому издательству информацию об общем количестве и суммарной стоимости поставленных ими книг каждому продавцу:

```
SELECT  PUBLISH_HOUSE, SALESMAN, COUNT(QUANTITY),
SUM(PRICE*QUANTITY)  FROM  BOOKS, BOOKS_DELIVERY
WHERE  BOOKS.CODE_BOOK = BOOKS_DELIVERY.CODE_BOOK
GROUP  BY  PUBLISH_HOUSE, SALESMAN;
```

5. Выбрать из таблиц BOOKS и BOOKS_DELIVERY по каждой книге, сведения о которой имеются в таблице BOOKS, информацию о количестве продавцов, которым она была поставлена:

```
SELECT  TITLE, AUTHOR, COUNT(SALESMAN)  FROM BOOKS,
BOOKS_DELIVERY  WHERE BOOKS_DELIVERY.CODE_BOOK
(+)= BOOKS.CODE_BOOK  GROUP  BY  TITLE, AUTHOR;
```

6. Выбрать из таблиц BOOKS и BOOKS_DELIVERY по каждому продавцу информацию об отсутствующих у них книгах, общий перечень которых находится в таблице BOOKS:

```
SELECT  SALESMAN, TITLE, PRICE  FROM  BOOKS,
BOOKS_DELIVERY
MINUS
SELECT  SALESMAN, TITLE, PRICE  FROM  BOOKS,
BOOKS_DELIVERY  WHERE  BOOKS_DELIVERY.CODE_BOOK =
BOOKS.CODE_BOOK;
```

7. Выбрать из таблицы BOOKS информацию (название, цена) о трех самых дешевых книгах; предполагается, что в перечне книг не более трех различных книг с минимальной ценой:

```
SELECT  A.TITLE, A.PRICE  FROM  BOOKS  A, BOOKS  B WHERE
A.PRICE >= B.PRICE
GROUP  BY  A.TITLE, A.PRICE  HAVING COUNT(B.TITLE) <= 3
ORDER  BY  A.TITLE;
```

8. Выбрать из таблиц BOOKS и BOOKS_DELIVERY информацию (название, фамилия автора) о книгах, которые не были поставлены в магазин для продажи:

```
SELECT TITLE, AUTHOR FROM BOOKS WHERE
NOT EXISTS
(SELECT * FROM BOOKS_DELIVERY WHERE
BOOKS_DELIVERY.CODE_BOOK = BOOKS.CODE_BOOK);
```

3. ПРАКТИЧЕСКИЙ РАЗДЕЛ

3.1 Методические указания к лабораторным занятиям

Лабораторные работы представляют одну из форм освоения теоретического материала с одновременным формированием практических навыков в изучаемой учебной дисциплине. Их назначение – углубление проработки теоретического материала, формирование практических навыков путем регулярной и планомерной самостоятельной работы студентов на протяжении всего курса. Процесс подготовки к лабораторным работам включает изучение нормативных документов, обязательной и дополнительной литературы по рассматриваемому вопросу.

Непосредственное проведение лабораторной работы предполагает: изучение теоретического материала по теме лабораторной работы (по вопросам изучаемой темы); выполнение необходимых заданий; оформление отчета с выводами о выполненных заданиях; по каждой лабораторной работе проводится контроль: проверяется содержание отчета, усвоение материала.

3.2 Тематика лабораторных занятий

Лабораторная работа № 1. Моделирование предметной области на этапе концептуального проектирования базы данных

Задание:

1. Создать ER-модель предметной области.

Методика выполнения:

2. Ознакомиться с предметной областью, описанной в проекте.

Вариант 1. Проект ПОСТАВКА ТОВАРОВ

Завод «Прогресс» поставляет товары (изделие А, изделие В, изделие С и др.) заказчикам по договорам. Для каждого товара определены планы поставок.

Необходимо спроектировать базу данных ПОСТАВКА ТОВАРОВ, информация которой будет использоваться для анализа выполнения заводом планов поставок.

В БД должна храниться информация:

– о ТОВАРАХ: код товара, наименование товара, цена товара (тыс. руб.);

– о ЗАКАЗАХ на поставку товаров: код заказа, наименование заказчика, адрес заказчика, телефон, номер договора, дата заключения договора, наименование товара, плановая поставка (шт.);

– о фактических ОТГРУЗКАХ товаров: код отгрузки, код заказа, дата отгрузки, отгружено товара (шт.).

При проектировании БД необходимо учитывать следующее:

– товар имеет несколько заказов на поставку. Заказ соответствует одному товару;

– товару могут соответствовать несколько отгрузок. В отгрузке могут участвовать несколько товаров.

Кроме того, следует учесть:

– товар не обязательно имеет заказ. Каждому заказу обязательно соответствует товар;

– товар не обязательно отгружается заказчику. Каждая отгрузка обязательно соответствует некоторому товару.

Вариант 2. Проект РОЗНИЧНАЯ ТОРГОВЛЯ

Магазин розничной торговли продает персональные компьютеры, средства связи и периферийное оборудование: принтеры, накопители CD-RW и др.

Необходимо спроектировать базу данных РОЗНИЧНАЯ ТОРГОВЛЯ, информация которой будет использоваться для анализа продаж в магазине.

В БД должна храниться информация:

– о ТОВАРАХ: код товара, наименование товара, дата поступления в магазин, количество товара, цена закупки (руб.);

– о ПОСТАВЩИКАХ товаров: код поставщика, наименование поставщика, адрес, телефон, к кому обращаться;

– о ПРОДАЖАХ товаров в магазине: код продажи, код товара, дата продажи, количество проданного товара (шт.), цена розничная (руб.).

При проектировании БД необходимо учитывать следующее:

– поставщик поставляет несколько товаров. Товар поступает на склад магазина от нескольких поставщиков;

– товар имеет несколько продаж. Продажа относится к одному товару.

Кроме того, следует учесть:

– поставщик не обязательно поставляет товар (может временно не работать). Каждый товар обязательно поставляется;

– товар не обязательно продается. Каждая продажа обязательно связана с товаром.

Вариант 3. Проект ТУРАГЕНСТВО

Работники турагентства продают путевки путешествий по разным странам. В каждую страну организуются несколько маршрутов. По каждому маршруту указывается цель путешествия (отдых, экскурсия, лечение, шоп-тур, обучение и др.).

Необходимо спроектировать базу данных ТУРАГЕНСТВО, информация которой позволит определять наиболее популярные маршруты за текущий год, отслеживать обращения клиентов и др.

В БД должна храниться информация:

– о СТРАНАХ: код страны, название страны, стоимость визы (руб.);

– о МАРШРУТАХ: код страны, код маршрута, наименование маршрута;

– о ПРОДАЖАХ: код маршрута, цель путешествия, цена путевки (руб.), количество проданных путевок по маршруту, дата продажи.

При проектировании БД необходимо учитывать следующее:

– в каждую страну организуются несколько маршрутов. Маршрут имеет отношение только к одной стране;

– маршрут участвует в нескольких продажах. Продажа связана только с одним маршрутом.

Кроме того, следует учесть:

– по каждой стране обязательно организуется маршрут. Каждый маршрут обязательно имеет отношение к некоторой стране;

– маршрут не обязательно может участвовать в продаже (может быть невостребован). Каждая продажа *обязательно* связана с одним маршрутом.

Вариант 4. Проект ПОДПИСНЫЕ ИЗДАНИЯ

Отделение Белпочты каждое полугодие осуществляет подписку граждан (в дальнейшем получателей) на различные издания (газеты, журналы) на один, три или шесть месяцев.

Необходимо спроектировать базу данных ПОДПИСНЫЕ ИЗДАНИЯ, информация которой будет использоваться для учета получателей и выписанных ими изданий.

В БД должна храниться информация:

– об ИЗДАНИЯХ, на которые можно оформить подписку: индекс издания, вид издания (газета, журнал), название издания, стоимость подписки на издание на 1 месяц (руб.);

– о ПОЛУЧАТЕЛЯХ: код получателя, Ф.И.О. получателя, адрес получателя (улица, дом, квартира);

– о ПОДПИСКАХ, осуществленных получателями: код получателя, индекс издания, срок подписки (в месяцах), месяц начала доставки издания, год начала доставки издания.

При проектировании БД необходимо учитывать следующее:

– получатель может осуществить подписку несколько раз (подписаться на несколько изданий). Каждая подписка осуществляется одним получателем;

– издание может быть использовано для нескольких подписок (на издание могут подписаться несколько получателей). Каждая подписка соответствует одному изданию.

Кроме того, следует учесть:

– каждый получатель обязательно осуществляет хотя бы одну подписку. Каждая подписка обязательно соответствует получателю;

– на издание не обязательно может быть подписка (оно может быть не востребовано). Подписка обязательно соответствует некоторому изданию.

3. Выделить сущности и обозначить их графически прямоугольниками. Каждую сущность описать набором атрибутов в виде схемы.

4. Выделить связи между сущностями и представить их графически ромбами.

5. Определить тип каждой связи и представить ее графически в виде ER-диаграммы.

6. Определить класс принадлежности каждой сущности и отразить его графически на ER-диаграмме связи.

7. Построить графическую ER-модель предметной области с использованием специальных программных сервисов Draw.io или Diagrams.net.

Литература

1. Лобанов, А. Создание ER-диаграммы в Drow.io [Электронный ресурс] / Александр Лобанов. – Режим доступа : https://www.youtube.com/watch?v=uKImrwjOKTU. – Дата доступа : 23.08.2023.

2. Попова-Коварцева, Д. А. Основы проектирования баз данных [Электронный ресурс] : учеб. пособие / Д. А. Попова-Коварцева, Е. В. Сопченко. – Самара : Изд-во Самарского университета, 2019. – С. 51–62. – Режим доступа : http://repo.ssau.ru/bitstream/Uchebnye-izdaniya/Osnovy-proektirovaniya-baz-dannyh-Elektronnyi-resurs-ucheb-posobie-80353/1/Попова-Коварцева%20Д.А.%2C%20Сопченко%20Е.В.%20Основы%20проектирования%20баз%20данных%202019.pdf. – Дата доступа : 27.08.2023.

Лабораторная работа № 2. Создание запросов на языке SQL

Задание:

1. Создать запросы на языке SQL

Методика выполнения:

Вариант 1

1. Рассмотреть ситуацию поступления в высшие учебные заведения, которая была характерна для периода, когда были разрешены так называемые репетиционные вступительные экзамены, которые сдавались раньше основных вступительных экзаменов в высшее учебное заведение.

Исходными являются три отношения R1, R2 и R3. Все они имеют эквивалентные схемы:

– R1= (ФИО, Паспорт, Школа);

– R2= (ФИО, Паспорт, Школа);

– R3= (ФИО, Паспорт, Школа).

Отношение R1 содержит список абитуриентов, сдававших репетиционные экзамены. Отношение R2 содержит список абитуриентов, сдававших экзамены на общих условиях. Отношение R3 содержит список абитуриентов, принятых в высшее учебное заведение. Считается, что при неудачной сдаче репетиционных экзаменов абитуриент мог делать вторую попытку и сдавать экзамены в общем потоке, поэтому некоторые абитуриенты могут присутствовать как в первом, так и во втором отношении.

2. Определить список абитуриентов, которые поступали два раза и не поступили в высшее учебное заведение.

3. Определить список абитуриентов, которые поступили в высшее учебное заведение с первого раза, то есть они сдавали экзамены только один раз и сдали их так хорошо, что сразу были зачислены в высшее учебное заведение.

4. Определить список абитуриентов, которые поступили в высшее учебное заведение только со второго раза.

5. Определить список абитуриентов, которые поступали только один раз в высшее учебное заведение и не поступили.

Вариант 2

1. Рассмотреть ситуацию сдачи экзамена студентами одной специальности.

Исходными являются три отношения R1, R2 и R3. Все они имеют эквивалентные схемы:

– R1= (ФИО, Номер_зач);

– R2= (Дисциплина);

– R3= (Номер_зач, Дисциплина).

Отношение R1 содержит список студентов, которые должны сдавать экзамены. Отношение R2 содержит список всех дисциплин, по которым студенты должны сдавать экзамены. Отношение R3 содержит список студентов, сдавших экзамены.

2. Определить список фамилий студентов, которые сдавали экзамены.

3. Определить список фамилий студентов, которые не сдали хотя бы один экзамен.

4. Определить список фамилий студентов, сдавших все экзамены.

5. Определить список фамилий студентов, сдавших все экзамены на 8 и 9.

Вариант 3

1. Рассмотреть ситуацию сдачи сессии студентами некоторого учебного заведения.

Исходными являются три отношения R1, R2 и R3. Все они имеют эквивалентные схемы:

– R1= (ФИО, Дисциплина, Оценка);

– R2= (ФИО, Группа);

– R3= (Группы, Дисциплина).

R1 – информация о попытках (как успешных, так и неуспешных) сдачи экзаменов студентами. R2 – состав групп. R3 – список дисциплин, которые надо сдавать каждой группе.

2. Определить список фамилий студентов, которые сдали экзамен по дисциплине «Базы данных» на 9 баллов.

3. Определить список тех студентов, которые должны были сдавать экзамен по дисциплине «Базы данных», но пока еще не сдавали.

4. Определить список тех студентов, которые имеют несколько двоек.

5. Определить список отличников.

Литература

1. Дунаев, В. В. Базы данных. Язык SQL [Электронный ресурс]. – Санкт-Петербург : БХВ-Петербург, 2006. – 288 с. – Режим доступа : https://docviewer.yandex.by/view/0/?page=4&*=2xvwegq5%2FhYyInf8gMxjD4mA0fF7InVybCI6Imh0dHBzOi8vYm9va3MuNG5tdi5ydS9ib29rcy9iYXp5X2Rhbm55a2hfeWF6eWtfc3FsXzM2NDI3NzEucGRmIiwidGl0bGUiOiJiYXp5X2Rhbm55a2hfeWF6eWtfc3FsXzM2NDI3NzEucGRmIiwibm9pZnJhbWUiOnRydWUsInVpZCI6IjAiLCJ0cyI6MTcxNTc3NzA5MDk3MCwieXUiOiI3NDQ4Njg1OTYxNjk1NzIzMTc0Iiwic2VycFBhcmFtcyI6InRtPTE3MTU3NzY5MTImdGxkPWJ5Jmxhbmc9bWlzJm5hbWU9YmF6eV9kYW5ueWtoX3lhenlrX3NxbF8zNjQyNzcxLnBkZiZ0ZXh0PSVEMSU4RiVEMCVCNyVEMSU4QiVEMCVCQStzcWwrJUQxJTgzJUQxJTg3JUQwJUI1JUQwJUIxJUQwJUJEJUQwJUJFJUQwJUI1KyVEMCVCRiVEMCVCRSVEMSU4MSVEMCVCRSVEMCVCMSVEMCVCOCVEMCVCNSZ1cmw9aHR0cHMlM0EvL2Jvb2tzLjRubXYucnUvYm9va3MvYmF6eV9kYW5ueWtoX3lhenlrX3NxbF8zNjQyNzcxLnBkZiZscj0xNTcmbWltZT1wZGYmbDEwbj1ydSZzaWduPTkzYmE1OThiMTVmOGRjZmNiODkxYWIzZTlkNWMxZjQ2JmtleW5vPTAifQ%3D%3D. – Дата доступа : 28.08.2023.

2. Дьяков, И. А. Базы данных. Язык SQL [Электронный ресурс] : учеб. пособие / И. А. Дьяков ; М-во образования и науки Рос. Федерации, Федер.

гос. бюджетное образоват. учрежд. высш. профес. образования «Тамбовский государственный технический университет». – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2012. – 82 с. – Режим доступа : http://biblioclub.ru/index.php?page=book&id=277628. – Дата доступа : 28.08.2023.

3. СУБД: язык SQL в примерах и задачах [Электронный ресурс] : учеб. пособие для вузов / [И.Ф. Астахова и др.]. – Москва : Физматлит, 2009. – 165 с. – Режим доступа : https://biblioclub.ru/index.php?page=book&id=76768. – Дата доступа : 28.08.2023.

4. РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

4.1 Вопросы к зачету

ПЕРЕЧЕНЬ ТЕОРЕТИЧЕСКИХ ВОПРОСОВ

для проведения зачета

1. Понятие банка данных (БнД). Классификация БнД.
2. Понятие базы данных (БД). Классификация БД.
3. Сетевые БД, архитектура «файл-сервер», «клиент-сервер».
4. Пользователи БД. Требования к БД со стороны внешних пользователей.
5. Понятие системы управления базами данных (СУБД). Требования к СУБД.
6. Компоненты СУБД.
7. Классификация СУБД.
8. Функции СУБД.
9. Языковые средства современных СУБД (DDL, DML, DCL). Общая характеристика.
10. Понятие моделей данных. Общая классификация моделей данных и их характеристика.
11. Иерархическая модель представления данных и ее общая характеристика.
12. Сетевая модель представления данных и ее общая характеристика.
13. Реляционная модель представления данных. Основныепонятия и характеристика. Виды связей.
14. Постреляционная модель представления данных и ее общая характеристика.
15. Объектно-ориентированная модель представления данных и ее общая характеристика.
16. Инфологическая модель данных. Элементы. Способ построения.
17. Даталогическая модель данных.
18. Физическая модель данных.
19. Базовые понятия реляционных БД: тип данных, домен, атрибут, кортеж, ключ, отношение, схема отношений.
20. Понятие и цели нормализации БД.
21. Нормальные формы 1НФ, 2НФ, 3НФ, НФБК, нормальные формы более высокогопорядка.
22. Жизненный цикл БД. Этапы жизненного цикла БД.
23. Этапы проектирования БД и их процедуры.

24. Язык SQL: понятие, назначение, стандарты и виды.

25. Основные категории команд языка SQL: DDL, DML, DQL, DCL, TPL, CCL.

26. Структура команды языка SQL. Основные предложения языка SQL: FROM, WHERE, INTO, GROUPE BY, HAVING, ORDER BY.

27. Типы данных в языке SQL.

28. Функции языка SQL.

4.2 Тестовые задания

Вариант 1

1. Как называется организованная совокупность данных, предназначенная для длительного хранения во внешней памяти ЭВМ и постоянного применения?

а) банк данных
б) база данных
в) информационная система
г) реляционная таблица
д) СУБД

2. Какая база данных содержит обширную информацию самого разного типа: текстовую, графическую, звуковую, мультимедийную?

а) документальная
б) сетевая
в) реляционная
г) фактографическая
д) распределенная

3. Что такое реляционная база данных?

а) база данных, разные части которой хранятся на различных ЭВМ компьютерной сети.
б) базы данных с табличной формой организации
в) комплекс аппаратно-программных средств для хранения, изменения и поиска информации, для взаимодействия с пользователем
г) база, которая содержит краткие сведения об описываемых объектах, представленные в строго определённом формате.
д) организованная совокупность данных, предназначенная для длительного хранения во внешней памяти ЭВМ и постоянного применения.

4. Как классифицируются базы данных по типу хранимой информации?

а) документальные БД
б) сетевые БД
в) распределенные БД
г) иерархические БД
д) фактографические БД

5. Указать основные понятия реляционной базы данных?

а) таблица
б) тип данных
в) поле

г) домен

д) первичный ключ

6. Что такое запись в реляционной БД?

а) это информация об одном объекте той реальной системы, которая представлена в таблице реляционной базы данных.

б) база данных, разные части которой хранятся на различных ЭВМ компьютерной сети

в) строка прямоугольной таблицы реляционной базы данных

г) столбец прямоугольной таблицы реляционной базы данных

д) совокупность данных, предназначенная для длительного хранения во внешней памяти ЭВМ и постоянного применения

7. Какие действия можно выполнить с помощью СУБД?

а) создание структуры базы данных

б) заполнение базы данных информацией

в) изменение (редактирование) структуры и содержания БД

г) поиск информации в базе данных и сортировка данных

д) защита данных и проверка целостности БД

8. Лицо или группа лиц, отвечающих за выработку требований к БД, ее проектирование, создание, эффективное использование и сопровождение – это

а) Администратор базы данных

б) Диспетчер базы данных

в) Программист базы данных

г) Пользователь базы данных

д) Технический специалист

9. Модель представления данных – это

а) Логическая структура данных, хранимых в базе данных

б) Физическая структура данных, хранимых в базе данных

в) Иерархическая структура данных

г) Сетевая структура данных

10. Если каждой записи в таблице А могут соответствовать несколько записей в таблице В, а запись в таблице В не может иметь более одной соответствующей ей записи в таблице А, то это связь...

а) один-к-одному

б) многие-к-себе

в) один-ко-многим

г) многие-ко-многим

11. 6. Объект Access, определяемый и используемый для хранения данных:

а) таблицы

б) запросы

в) формы

г) отчеты

12. Различные приложения пользователей, которые формируют запросы к серверу, проверяют допустимость данных и получают ответы – это

а) Сервер базы данных

б) Клиенты

в) Сеть

г) Коммуникационное программное обеспечение

13. По разделению функций между клиентом и сервером выделяют следующие модели архитектуры клиент-сервер:

а) модель доступа к удаленным данным

б) модель сервера баз данных

в) модель сервера приложений

г) модель сервера клиентов

14. Укажите характеристики, относящиеся к СУБД централизованной архитектуры:

а) СУБД, база данных и прикладные программы, которые работают с базой данных, функционируют на центральном компьютере

б) процессы, связанные с обработкой данных, производятся на центральном компьютере

в) централизованная архитектура СУБД подразумевает доступ из одного узла локальной сети к ресурсам, находящимся на в других узлах

г) рабочая станция предназначена для работы пользователя и обладает ресурсами, соответствующими потребностям пользователя

д) сервер локальной сети предоставляет ресурсы рабочим станциям и другим серверам.

15. Если на компьютере-клиенте располагаются части приложения, реализующие только функции представления, а прикладные функции размещены на стороне сервера, то говорят о модели ...

а) доступа к удаленным данным

б) сервера баз данных

в) сервера приложений

г) сервера клиентов

16. Внутренний уровень архитектуры СУБД

а) Наиболее близок к физическому, описывает способ размещения данных на устройствах хранения информации

б) Наиболее близок к пользователю, описывает способ размещения данных на устройствах хранения информации

в) Наиболее близок к пользователю, описывает обобщенное представление данных

г) Наиболее близок к физическому, описывает способ размещения данных в логической структуре базы данных

17. Внешний уровень архитектуры СУБД

а) Наиболее близок к физическому, описывает способ размещения данных на устройствах хранения информации

б) Наиболее близок к пользователю, предоставляет возможность манипуляции данными в СУБД с помощью языка запросов или языка специального назначения

в) Для множества пользователей, описывает обобщенное представление данных

г) Наиболее близок к физическому, описывает способ размещения данных в логической структуре базы данных

18. Концептуальный уровень архитектуры СУБД

а) Наиболее близок к физическому, описывает способ размещения данных на устройствах хранения информации

б) Наиболее близок к пользователю, описывает способ размещения данных на устройствах хранения информации

в) Наиболее близок к пользователю, предоставляет возможность манипуляции с данными

г) Переходный от внутреннего к внешнему, описывает обобщенное представление данных для множества пользователей

19. Проектированием БД занимается

а) Администратор БД

б) Программист БД

в) Пользователь БД

г) Проектировщик БД

20. Основная роль администратора баз данных заключается в:

а) определении требований к системе

б) выборе целевой СУБД

в) планировании разработки БД

г) разработке приложений

Вариант 2

1. База данных – это...

а) совокупность данных, организованных по определенным правилам

б) совокупность программ для хранения и обработки больших массивов информации

в) интерфейс, поддерживающий наполнение и манипулирование данными

г) определенная совокупность информации

2. Набор программных и аппаратных средств, обеспечивающих выполнение действий по определению данных, их обработке и управлению называется...

а) база данных

б) система управления базой данных

в) компьютерная база данных

г) система управления данными

3. Какая база содержит краткие сведения об описываемых объектах, представленные в строго определённом формате.

а) иерархическая

б) сетевая

в) реляционная

г) фактографическая

д) документальная

4. Как называется база данных, разные части которой хранятся на различных ЭВМ компьютерной сети?

а) локальная

б) распределенная

в) сетевая

г) иерархическая

д) реляционная

5. Как классифицируются базы данных по структуре организации данных?

а) иерархические БД

б) сетевые БД

в) реляционные БД

г) операционные БД

д) справочно-информационные БД

6. Если в модели каждый сегмент связан с одним или несколькими на более низком уровне, и только с одним на более высоком уровне,

то такая модель называется ...

а) иерархической моделью

b) реляционной моделью

c) сетевой моделью

7. Наиболее точным аналогом реляционной базы данных может служить:

1) неупорядоченное множество данных
2) генеалогическое древо
3) двумерная таблица
4) трехмерный куб

8. Каково назначение СУБД MS Access?

а) СУБД MS Access позволяет создание базы данных

б) СУБД MS Access позволяет редактирование БД

в) СУБД MS Access позволят манипулирование данными

г) СУБД MS Access ориентирована на пользователя

д) СУБД MS Access – это система программирования со своим специализированным языком программирования для создания программ обработки данных

9. В каком режиме создания таблиц в СУБД Access предоставляется набор таблиц, из которых можно создавать таблицы по своему вкусу. Некоторые таблицы могут полностью подойти для вашего приложения. Тип данных и другие свойства полей уже определены.

а) режим таблицы

б) конструктор таблиц

в) мастер таблиц

г) импорт таблиц

д) связь с таблицами

10. Если запись в таблице А может иметь не более одной связанной записи в таблице В и наоборот, то это связь...

а) многие-ко-многим

б) многие-к-себе

в) один-ко-многим

г) один-к-одному

11. К пользователям БД относятся следующие категории пользователей:

а) системный программист, прикладной программист

б) инженер электронной техники

в) администратор системы

г) оператор компьютера, конечный пользователь

д) прикладные программы, СУБД

12. Какую структуру имеет схема данных?

а) реляционную структуру

б) сетевую структуру

в) иерархическую структуру

г) распределённую структуру

д) линейную структуру

13. Сетевая модель представления данных - данные представлены с помощью

а) Таблиц

б) Списков

в) Упорядоченного графа

г) Произвольного графа

д) Файлов

14. Если на компьютере-клиенте располагаются части приложения, реализующие только функции представления, а прикладные функции размещены на стороне сервера, то говорят о модели ...

а) доступа к удаленным данным

b) сервера баз данных

c) сервера приложений

d) сервера клиентов

15. Укажите принципы системной архитектуры "клиент–сервер":

а) система разбивается на две части – клиентскую и серверную

б) в качестве основного интерфейса между клиентской и серверной частью выступает язык БД SQL

в) на рабочих станциях–клиентах работает MS Access

г) клиентская часть системы при потребности обращается к серверной части

д) SQL-сервер выполняет обработку данных

16. Внутренний уровень архитектуры СУБД

а) Наиболее близок к физическому, описывает способ размещения данных на устройствах хранения информации

б) Наиболее близок к пользователю, описывает способ размещения данных на устройствах хранения информации

в) Наиболее близок к пользователю, описывает обобщенное представление данных

г) Наиболее близок к физическому, описывает способ размещения данных в логической структуре базы данных

17. Внешний уровень архитектуры СУБД

а) Наиболее близок к физическому, описывает способ размещения данных на устройствах хранения информации

б) Наиболее близок к пользователю, предоставляет возможность манипуляции данными в СУБД с помощью языка запросов или языка специального назначения

в) Для множества пользователей, описывает обобщенное представление данных

г) Наиболее близок к физическому, описывает способ размещения данных в логической структуре базы данных

18. Концептуальный уровень архитектуры СУБД

а) Наиболее близок к физическому, описывает способ размещения данных на устройствах хранения информации

б) Наиболее близок к пользователю, описывает способ размещения данных на устройствах хранения информации

в) Наиболее близок к пользователю, предоставляет возможность манипуляции с данными

г) Переходный от внутреннего к внешнему, описывает обобщенное представление данных для множества пользователей

19. В группу администраторов БД входят:

а) Эксперт по языкам запросов

б) эксперт по прикладным программам

в) конечные пользователи

г) администратор данных

20. Основная роль администратора баз данных заключается в:

а) определении требований к системе

б) выборе целевой СУБД

в) планировании разработки БД

г) разработке приложений

4.3 Перечень рекомендуемых средств диагностики результатов учебной деятельности студентов

Для выявления уровня учебных достижений студентов рекомендуется использовать следующие средства диагностики:

– устные и письменные опросы в ходе лекционных занятий;

– выполнение лабораторных заданий с использованием компьютера;

– подготовка рефератов и учебных сообщений;

– групповые дискуссии по наиболее сложным вопросам учебной дисциплины;

– выполнение индивидуальных заданий;

– выполнение тестовых заданий;

– подготовка электронных презентаций;

– подготовка электронных отчетов по результатам выполнения лабораторных заданий;

– защита самостоятельно разработанных заданий (проектов, кейсов);

– зачет.

5. ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

5.1 Учебная программа

Введение

Предмет учебной дисциплины, его цель, задачи и место в системе профессиональной подготовке специалистов библиотечно-информационной сферы.

Связь курса по выбору с другими учебными дисциплинами. Объем, структура, содержание и порядок изучения курса по выбору. Формы самостоятельной работы. Система средств диагностики. Учебно-методическое обеспечение курса по выбору.

Тема 1. Основы теории баз данных. Системы управления базами данных

Определение основных понятий: информация, данные, база данных, система управления базами данных, автоматизированная информационная система, предметная область, банк данных, приложения системы управления базами данных, внешние приложения. Децентрализованный и централизованный подход к организации данных. Преимущества и недостатки таких подходов.

Банк данных (БнД) как информационная система. Состав банка данных в узком и широком смысле этого понятия. Состав банка данных как автоматизированной системы. Компоненты банка данных. Отличительные особенности БнД. Классификация БнД.

База данных (БД) как интегрированный ресурс. Объекты и свойства БД. Отличительные особенности БД. Выполняемые операции с БД. Классификация БД. Основные категории пользователей БД.

Функции системы управления базами данных (СУБД). Требования к СУБД. Программные компоненты (модули) СУБД: процессор запросов, контроллер БД, контроллер файлов, препроцессор языка DML, компилятор языка DDL, контролер словаря. Признаки классификации СУБД: степень универсальности, функциональность, язык общения и др. Языковые средства СУБД. Язык описания данных (DDL – Data Definition Language). Язык манипулирования данными (DML – Data Manipulation Language). Язык

структурированных запросов (SQL – Structured Query Language). Язык запросов по образцу (QBE – Query By Example).

Выбор СУБД: основные подходы к выбору СУБД; показатели пригодности; технические характеристики; оценка производительности. Возможности современных СУБД. Интерфейсы, предоставляемые СУБД пользователям.

Особенности функционирования, применения и использования СУБД Microsoft Access, Microsoft FoxPro for Windows, Microsoft Visual FoxPro, Borland dBase IV, Oracle, MySQL, Microsoft SQL Server, Clipper, Informix, Линтер.

Тема 2. Реляционные базы данных. Базисные средства манипулирования реляционными данными

Общее понятие реляционного подхода к организации БД. Три основных компонента реляционной базы данных по К. Дж. Дейту: структурный, манипуляционный и целостный.

Определение основных понятий реляционных БД: тип данных, домен, атрибут, схема отношения, схема базы данных, кортеж, отношение. Фундаментальные свойства отношений: отсутствие кортежей-дубликатов; отсутствие упорядоченности кортежей; отсутствие упорядоченности атрибутов; атомарность значений атрибутов. Виды отношений: именованное отношение, базовое отношение, производное отношение, выражаемое отношение.

Таблица как способ представления отношений, ее элементы. Требования к табличной форме представления отношений: конечность; одноярусность заголовка и уникальность имени столбца; несущественность порядка строк, однотипность данных во всех столбцах.

Ключ связи как инструмент объединения данных из разных таблиц. Отношения записей типа «один к одному» (1:1) и типа «один ко многим» (1:M) в двух таблицах реляционной базы данных.

Необходимость нормализации. Понятия нормализации отношений и нормальной формы. Нормальные формы: 1) первая нормальная форма (1NF); 2) вторая нормальная форма (2NF); 3) третья нормальная форма (3NF); 4) нормальная форма Бойса-Кодда (BCNF). Приведение базы данных к нормализованному виду. Достоинства и недостатки нормализованных и ненормализованных реляционных таблиц.

Реляционная алгебра: понятие и основная цель. Традиционные (бинарные) операции над множествами: объединение; пересечение; разность; декартово произведение. Использование декартового произведения для получения информации из множества взаимосвязанных таблиц.

Специальные (унарные) реляционные операции: выборка; проекция; переименование, деление, соединение. Варианты операций соединения: операция внутреннего соединения; операции левого внешнего соединения и правого внешнего соединения; операция полного внешнего соединения. Свойства соединения.

Понятие реляционного выражения. Использование реляционных выражений. Запросы к базе данных в форме реляционных выражений.

Тема 3. Основы проектирования баз данных

Основная цель и задачи процесса проектирования БД. Требования к проекту БД. Функции и задачи Администратора БД в процессе проектирования БД. Факторы, влияющие на проектирование базы данных.

Семь этапов жизненного цикла БД: предварительное планирование; проверка осуществимости; определение требований, концептуальное проектирование, логическое проектирование, физическое проектирование, оценка работы и поддержка БД. Главные задачи каждого этапа.

Этапы проектирования БД. Формулировка и анализ требований. Концептуальное проектирование и его процедуры: определение сущностей и их документирование; определение связей между сущностями и их документирование; создание ER-модели предметной области, определение атрибутов и их документирование; определение значений атрибутов и их документирование; определение первичных ключей для сущностей и их документирование; обсуждение концептуальной модели данных с конечными пользователями. Требования, предъявляемые к концептуальной модели.

Модель «сущность-связь» (ER-модель) как средство моделирования предметной области. Основные понятия ER-диаграммы. ER-диаграмма связи 1:1. ER-диаграмма связи 1:M. ER-диаграмма связи M:N.

Логическое проектирование и его процедуры: выбор модели данных; определение набора таблиц и их документирование; нормализация таблиц; проверка логической модели данных; определение требований поддержки целостности данных и их документирование; создание окончательного варианта логической модели данных и его обсуждение.

Преобразование ER-модели в схему реляционной БД. Правила формирования набора предварительных таблиц из ER-диаграмм. Нормализация таблиц. Приведение таблиц к формам 1НФ, 2НФ, 3НФ. Нормальная форма Бойса-Кодда (НФБК), 4НФ, 5НФ.

Физическое проектирование и его процедуры: проектирование таблиц БД средствами выбранной СУБД; реализация бизнес-правил в среде выбранной

СУБД; проектирование физической организации БД; разработка стратегии защиты БД; организации мониторинга функционирования БД и ее настройка.

Взаимосвязь этапов проектирования БД. Критерии оценки проектируемой/спроектированной БД: адекватность, полнота, адаптируемость, универсальность, сложность структуры БД, степень дублирования данных в БД, объем требуемой памяти, скорость обработки информации.

Тема 4. Основы структурированного языка запросов SQL

Язык SQL: понятие, назначение, стандарты, достоинства и недостатки. Функциональные возможности языка SQL. Место языка SQL в разработке информационных систем, организованных на основе технологии клиент-сервер. Виды языка SQL: интерактивный и вложенный.

Основные категории команд языка SQL: DDL, DML, DQL, DCL, TPL, CCL.

Функции и основные команды DDL: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX. Функции и команды DML: INSERT, UPDATE, DELETE. Функции DQL и его команда SELECT. Общая структура команды SELECT. Функции и команды DCL: GRANT, REVOKE. Функции и команды TPL: BEGIN, COMMIT, ROLLBACK. Функции и команды CCL: DECLARE CURSOR, OPEN, CLOSE, FETCH INTO, DROP CURSOR.

Структура команды языка SQL: ключевое слово и предложения. Основные предложения языка SQL: FROM, WHERE, INTO, GROUPE BY, HAVING, ORDER BY.

Типы данных языка SQL, определенные стандартом: Символьный, Битовый, Точные числа, Округленные числа, Дата/время, Интервал. Специальные символы и знаки пунктуации SQL. Предикаты.

Типы запросов в языке SQL. Создание различных видов запросов в языке SQL. Оптимизация SQL-запросов.

5.2 Учебно-методическая карта учебной дисциплины

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов	
		Лекции	Лабораторные занятия
1	*2*	*3*	*4*
1	Введение	1	
2	***Тема 1***. Основы теории баз данных. Системы управления базами данных.	1	
3	***Тема 2.*** Реляционные базы данных. Базисные средства манипулирования реляционными данными	2	
4	***Тема 3.*** Основы проектирование баз данных	2	2
5	***Тема 4.*** Основы структурированного языка запросов SQL	2	2
	Всего	8	4

5.3 Основная литература

1. *Агафонов, А. А.* Основы технологий баз данных : учебное пособие / А. А. Агафонов, А. М. Белов. – Самара : Самарский университет, 2023. – 304 с. – Режим доступа : https://e.lanbook.com/book/406457.

2. *Голицына, О. Л.* Базы данных : учеб. пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва : ФОРУМ : ИНФРА-М, 2020. – 400 с. – Режим доступа : https://znanium.ru/catalog/product/1091314.

3. *Диязитдинова, А.Р.* Основы проектирования баз данных : учеб. пособие / А.Р. Диязитдинова. – Самара : ПГУТИ, 2022. – 245 с. – Режим доступа : https://e.lanbook.com/book/329933.

4. *Кореньков, В.В.* Технологии баз данных. Вводный курс = Database technology. Introductory course : учеб. пособие / В.В. Кореньков, О.В. Иванцова, И.А. Филозова. – Москва : Курс, 2020. – 173, [1] с.

5. *Осипов, Д. Л.* Технологии проектирования баз данных. – М. : ДМК Пресс, 2019. – 498 с.

6. *Скакун, В.В.* Системы управления базами данных : учеб.-метод. пособие для студентов учреждений высшего образования / В.В. Скакун. – Минск : БГУ, 2020. – 158, [1] с.

7. *Управление* данными : учебник / Ю. Ю. Громов, О. Г. Иванова, А. В. Яковлев, В. Г. Однолько ; М-во образования и науки Рос. Федерации, Федер. гос. бюджетное образоват. учрежд. высш. профес. образования «Тамбовский государственный технический университет». – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2015. – 192 с. – Режим доступа : http://biblioclub.ru/index.php?page=book&id=444642.

5.4 Дополнительная литература

1. *Большакова, Г. И.* Проектирование баз данных в среде MS Access : практическое руководство / Г. И. Большакова, Т. Я. Каморникова, Е. И. Сукач ; М-во образования Республики Беларусь, Гомельский гос. ун-т им. Ф. Скорины. – Гомель: ГГУ им. Ф. Скорины, 2016 – 40 с. – Режим доступа : https://elib.gsu.by/bitstream/123456789/2199/1/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B0%D0%BA%D0%BE%D0%B2%D0%B0%20%D0%93.%D0%98.%20%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%91%D0%94.pdf?ysclid=m4gxyyifax289923853.

2. Комаров, В. И. Путеводитель по базам данных. — М.: ДМК-Пресс 2024. – 482 с.

3. *Лазицкас, Е. А.* Базы данных и системы управления базами данных : учеб. пособие / Е. А. Лазицкас, И. Н. Загумённикова, П. Г. Гилевский. – Минск : РИПО, 2016. – 267 с. – Режим доступа : http://biblioclub.ru/index.php?page=book&id=463305.

4. *Кузнецов, С.* Введение в реляционные базы данных / С. Кузнецов. – 2-е изд., испр. – Москва : Нац. Открытый Ун-т «ИНТУИТ», 2016. – 248 с. – (Основы информационных технологий). – То же [Электронный ресурс]. – URL : http://biblioclub.ru/index.php?page=book&id=429088.