**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»**

# МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

## Б1.Б.1.32 Разработка и эксплуатация защищенных автоматизированных систем

**Специальность** 10.05.03 Информационная безопасность автоматизированных систем

**Специализация** Информационная безопасность автоматизированных систем критически важных объектов

**Форма обучения** очная

**СОДЕРЖАНИЕ**

# 1. КОНСПЕКТ ЛЕКЦИЙ

## 1. 1 Лекция №1-3 (2 часа).

**Тема:** Введение в предмет.

### 1.1.1 Вопросы лекции:

Цели и задачи курса «Разработка и эксплуатация защищенных автоматизированных систем». Предмет и содержание курса в целом, его роль и место в подготовке специалистов по комплексной защите информации.

### 1.1.2 Краткое содержание вопросов:

Проектирование и создание автоматизированных систем в защищенном исполнении

Автоматизированная система (АС) является организационно-технической системой, которая позволяет вырабатывать решения на основе автоматизации информационных процессов в различных сферах деятельности.

Процесс создания АС в защищенном исполнении заключается в выполнении совокупности мероприятий, направленных на разработку и/или практическое применение информационной технологии, реализующей функции по защите информации согласно требованиям стандартов и нормативных документов по информационной безопасности.

Цель создания АС в защищенном исполнении — исключение или существенное затруднение получения защищаемой информации злоумышленниками о самой АС в процессе ее создания, защищаемой информации, обрабатываемой в АС или являющейся ее продуктом, а также исключение или существенное затруднение несанкционированного доступа и/или непреднамеренного воздействия на защищаемую информацию и ее носители.

К средствам защиты информации, используемым в составе АС в защищенном исполнении, относятся:

Система аутентификации и авторизации — сетевая регистрация автоматизированных рабочих мест (АРМ) и пользователей АРМ с использованием удаленных средств идентификации и аутентификации;

Система контроля целостности и аутентичности — контроль целостности критичных файлов ОС и функционального программного обеспечения АРМ как на этапе загрузки, так и в процессе их функционирования;

Система межсетевого экранирования — защита АС и ее отдельных узлов от несанкционированного доступа (НСД) и сетевых воздействий с целью вывода из строя отдельных функций защиты информации, узлов сети или нарушения функционирования АС в целом;

Система контекстного анализа трафика — защита данных АС при взаимодействии со смежными информационными комплексами и системами от вредоносных программ, утечки информации, рекламной информации, а также анализ формата и структуры данных, передаваемых изнутри и из внешних сетей относительно защищаемой АС;

Система обнаружения вторжений — анализ сетевого трафика и передача сведений о возможном нападении на централизованную консоль управления;

Система антивирусного контроля — защита ресурсов сети, сервера и АРМ от проникновения компьютерных вирусов и разрушительного воздействия вредоносных программ;

Система выявления уязвимостей в средствах защиты информации — анализ настроек и оценка эффективности функционирования средств защиты информации (СЗИ) с предоставлением информации о сбоях в работе СЗИ и наличии узких мест, которые могут быть использованы злоумышленником для получения НСД к данным АС;

Система защиты удаленного доступа — защищенный обмен данными с удаленными АРМ за счет применения средств аутентификации и авторизации, средств контроля целостности, а также посредством обеспечения защиты информации при обмене с АРМ удаленных абонентских пунктов по открытым каналам связи;

Система аудита — регистрация критичных событий, связанных с безопасностью как в пределах локальных ресурсов, так и системы в целом;

Система управления конфигурацией средств защиты информации — централизованное управление конфигурацией всех служб и сервисов системы безопасности;

Система защиты серверов и АРМ — ограничение доступа к данным на локальных рабочих местах, а также к локальной консоли серверов, межсетевых экранов и активного серверного оборудования;

Система экстренного уничтожения информации — экстренное и полное уничтожение данных, представляющих стратегический интерес для злоумышленников в случае возникновения реальной угрозы захвата информационных ресурсов АС;

Система блокировки загрузки с отчуждаемых носителей — защита серверов и АРМ от несанкционированного изменения конфигурации и режим функционирования путем загрузки с посторонних носителей информации.

**1. 2 Лекция №4-5 (4 часа).**

**Тема: Понятия, виды и структура автоматизированных систем**

1.2.1 Вопросы лекции:

Защищенные компьютерные системы. Свойства защищенных компьютерных систем. Угрозы безопасности. Подходы к созданию безопасных систем обработки информации.

1.2.2 Краткое содержание вопросов:

Система (system – целое, составленное из частей; греч.) – это совокупность элементов, взаимодействующих друг с другом, образующих определенную целостность, единство.

Архитектура системы – совокупность свойств системы, существенных для пользователя.

Элемент системы – часть системы, имеющая определенное функциональное назначение. Элементы, состоящие из простых взаимосвязанных элементов, часто называют *подсистемами.*

Организация системы – внутренняя упорядоченность, согласованность взаимодействия элементов системы, проявляющаяся, в частности, в ограничении разнообразия состояния элементов в рамках системы.

Структура системы – состав, порядок и принципы взаимодействия элементов системы, определяющие основные свойства системы. Если отдельные элементы системы разнесены по разным уровням и характеризуются внутренними связями, то говорят об иерархической структуре системы.

Добавление к понятию *система* слова *информационная* отражает цель ее создания и функционирования. Информационные системы обеспечивают сбор, хранение, обработку, поиск, выдачу информации, необходимой в процессе принятия решений задач из любой области. Они помогают анализировать проблемы и создавать новые информационные продукты.

Информационная система— это взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации компьютера. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление.

Необходимо понимать разницу между компьютерами и информационными системами. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система немыслима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

В нормативно-правовом смысле *информационная система* определяется как «организационно упорядоченная совокупность документов (массив документов) и информационных технологий, в том числе и с использованием средств вычислительной техники и связи, реализующих информационные процессы» [Закон РФ «Об информации, информатизации и защите информации» от 20.02.1995, № 24-ФЗ].

Процессы, протекающие в информационных системах

Информационный процесс – «процесс создания, сбора, обработки, накопления, хранения, поиска, распространения и потребления информации» [Закон РФ «Об участии в информационном обмене» от 04.07.1996, № 85-ФЗ].

Информационный ресурс – это отдельные документы и отдельные массивы документов, документы и массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, других видах информационных систем) [Закон РФ «Об участии в информационном обмене»].

В нормативно-правовом аспекте документ определяется как зафиксированная на материальном носителе информация с реквизитами, позволяющими ее идентифицировать. Процесс *документирования* превращает информацию в информационные ресурсы.

Процессы, обеспечивающие работу информационной системы любого назначения, условно можно представить состоящими из следующих блоков:

ввод информации из внешних или внутренних источников;

обработка входной информации и представление ее в удобном виде;

вывод информации для представления потребителям или передачи в другую систему;

обратная связь — это информация, переработанная людьми данной организации для коррекции входной информации.

Информационные процессы реализуются с помощью *информационных процедур*, реализующих тот или иной механизм переработки входной информации в конкретный результат.

Различают следующие типы информационных процедур:

Полностью *формализуемые*, при выполнении которых алгоритм переработки информации остается неизменным и полностью определен (поиск, учет, хранение, передача информации, печать документов, расчет на моделях).

*Неформализуемые* информационные процедуры, при выполнении которых создается новая уникальная информация, причем алгоритм переработки исходной информации неизвестен (формирование множества альтернатив выбора, выбор одного варианта из полученного множества).

*Плохо формализованные* информационные процедуры, при выполнении которых алгоритм переработки информации может изменяться и полностью не определен (задача планирования, оценка эффективности вариантов экономической политики).

*Функции информационных подразделений*, создающих и поддерживающих информационные системы (служба администратора): оповещение и обработка запросов; поддержание целостности и сохранности информации; периодическая ревизия информации; автоматизация индексирования информации.

В целом информационные системы определяется следующими свойствами:

любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем;

информационная система является динамичной и развивающейся;

при построении информационной системы необходимо использовать системный подход;

выходной продукцией информационной системы является информация, на основе которой принимаются решения;

информационную систему следует воспринимать как человеко-машинную систему обработки информации.

Внедрение информационных систем может способствовать:

получению более рациональных вариантов решения управленческих задач за счет внедрения математических методов;

освобождению работников от рутинной работы за счет ее автоматизации;

обеспечению достоверности информации;

совершенствованию структуры информационных потоков (включая систему документооборота);

предоставлению потребителям уникальных услуг;

уменьшению затрат на производство продуктов и услуг (включая информационные).

Этапы развития информационных систем

Этапы развития информационных систем и цели их использования представлены в таблице 1.1.

Первые информационные системы появились в пятидесятых годах. Они были предназначены для обработки счетов и расчета зарплаты, а реализовывались на электромеханических бухгалтерских счетных машинах. Это приводило к некоторому сокращению затрат и времени на подготовку бумажных документов.

Таблица 1.1. Этапы развития информационных систем

| *Период времени* | *Концепция использования информации* | *Вид информационных систем* | *Цель использования* |
|---|---|---|---|
| 1950-1960 годы | Бумажный поток расчетных документов | Электромеханические бухгалтерские машины | Упрощение проце-дуры обработки счетов и расчета зарплаты |
| 1960-1970 годы | Помощь в подготовке отчетов | Управленческие инфор-мационные системы для производственной информации | Ускорение процесса подготовки отчетности |
| 1970-1980 годы | Управленческий контроль процессов | Системы поддержки принятия решений | Выработка рациональных решений |
| с 1980 года по н/в | Информация — страте-гический ресурс, обеспе-чивающий конкурентное преимущество | Стратегические инфор-мационные системы. Автоматизированные офисы | Выживание и процветание организации |

Шестидесятые годы знаменуются изменением отношения к информационным системам. Информация, полученная из них, стала применяться для периодической отчетности по многим параметрам. Для этого организациям требовалось компьютерное оборудование широкого назначения, способное обслуживать множество функций, а не только обрабатывать счета и считать зарплату.

В семидесятых – начале восьмидесятых годов информационные системы начинают широко использоваться в качестве средства управленческого контроля, поддерживающего и ускоряющего процесс принятия решений.

К концу восьмидесятых годов концепция использования информационных систем вновь изменяется. Они становятся стратегическим источником информации и используются на всех уровнях организации любого профиля. Информационные системы этого периода, предоставляя вовремя нужную информацию, помогают организации достичь успеха в своей деятельности, создавать новые товары и услуги, находить новые рынки сбыта, обеспечивать себе достойных партнеров, организовывать выпуск продукции по низкой цене и многое другое.

**1. 3 Лекция №6-7 (4 часа).**

**Тема: Жизненный цикл АС.**

1.3.1 Вопросы лекции:

Разработка программно-информационного ряда АИС на основе систем управления базами данных.База данных информационной системы. Состав и содержание работ на стадии технорабочего проектирования

1.3.2 Краткое содержание вопросов:

Разработка больших проектов связанные с работой коллективов в несколько 10-100 человек из нескольких организаций, возможно при наличии нормативно-методических документов, регламентирующих различные аспекты процессов деятельности разработчиков. Комплекс таких документов называется нормативно-методическое обеспечение. Эти документы регламентируют:

Порядок разработки, внедрения, сопровождения АС (Устав);

Общие требования к составу АС, связям между ее компонентами, а также к ее качеству (ТЗ);

Виды, состав и содержание проектной и рабочей документации (Стандарт).

Все документы НМО классифицируются по следующим признакам:

Виды регламентаций (Стандарт, РД, положение, инструкция и т. д.);

Статус регламентирующего документа (международный, отраслевой, предприятия);

Области действия документов (заказчик, подрядчик);

Объекту регламентации или методического обеспечения (АС, бизнес-процесс).

Нормативной базой НМО являются:

Международные стандарты ISO/IEC (International organization of standardization/international electrotechnical commision);

Стандарты РФ (ГОСТ Р)

Стандарты организаций (СТ/П) – стандарт предприятия

Процессы создания АС регламентированы стандартами ГОСТ 34.601-90 – «Информационная технология. Комплекс стандартов на АС. АС. Стадии создания».

***Жизненный цикл АС (ЖЦ АС) определяется как период времени, который начинается с момента принятия решения о необходимости создания АС и заканчивается в момент ее полного изъятия из эксплуатации.***

Основным нормативным документом, регламентирующим состав процессов ЖЦ АС, является международный стандарт ISO/IEC 12207:1995 Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания АС (его российский аналог ГОСТ Р ИСО/МЭК 12207-99 введен в действие в июле 2000 г.). В данном стандарте *процесс*определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами.

Каждый процесс разделен на набор действий, каждое действие — на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последо вательностей выполнения (естественно, при сохранении связей по входным данным).

**Все процессы ЖЦ АС разделены на 3-и группы:**

Основные процессы (приобретение, поставка, разработка, эксплуатация, сопровождение);

Вспомогательные процессы (документирование, управление конфигурацией, обеспечение качеством, верификация, аттестация, совместная оценка, аудит, разрешение проблем);

Организационные процессы (управление, инфраструктура, усовершенствование, обучение).

**1. 4 Лекция № 8-9 (4 часа).**

**Тема: Порядок создание изделий ИТ, удовлетворяющим требованиям безопасности.**

1.4.1 Вопросы лекции:

Жизненный цикл изделий ИТ. Виды требований безопасности ИТ.

1.4.2 Краткое содержание вопросов:

При проектировании системы защиты информации в автоматизированной системе (АС) на основе существующих компонентов перед разработчиками возникает ряд проблем, которые можно подразделить на два уровня: защита средств вычислительной техники (СВТ) и защита АС.

Уровень защиты СВТ подразумевает реализацию базовых средств защиты, обычно на уровне операционных систем (ОС). Уровень же защиты АС охватывает всю АС в целом, включая средства защиты СВТ, защиту на уровне функционального ПО, средства их взаимодействия и т.п.

Распространенным подходом при проектировании АС является применение в качестве базового компонента уже существующей ОС общего назначения, например, Microsoft Windows, Linux, QNX и т.п. Подобный подход порождает при проектировании средств защиты СВТ ряд проблем, таких как:

- недостаточная защищенность ОС;
- недоступность исходных текстов ОС;
- недоступность или скудность низкоуровневой документации;
- несоответствие базовых (атомарных) операций;
- неориентированность на защиту низкоуровневых механизмов;
- неориентированность на защиту высокоуровневых (прикладных) механизмов.

Далее рассмотрим перечисленные проблемы подробнее.

Руководящие документы Гостехкомиссии РФ [1,2], регламентирующие защиту информации от несанкционированного доступа (НСД), формализуют условия защищенности СВТ и АС. Они определяют необходимый набор подсистем и механизмов защиты и их характеристики. При этом многие из требуемых средств и механизмов редко реализованы в ОС общего назначения. Например, механизмы дискреционного разграничения доступа к файлам и каталогам, идентификации и аутентификации пользователей в том или ином виде существуют в большинстве современных ОС; в то же время такие механизмы, как мандатное разграничение доступа или контроль целостности информации, имеются лишь в некоторых системах.

Другой важной проблемой при использовании ОС общего назначения является недоступность для разработчиков средств защиты исходных текстов ОС (типичный пример – ОС Microsoft Windows). В результате этого о внутренней структуре и механизмах ОС можно судить либо по косвенным признакам, либо осуществляя дизассемблирование исполняемого кода. Как следствие, невозможно судить с достаточной уверенностью о корректности реализации различных алгоритмов и механизмов ОС, об отсутствии или наличии различного рода уязвимостей и программных закладок. Таким образом, мы не можем полагаться на уже имеющиеся в ОС средства защиты и вынуждены реализовывать свои механизмы максимально независимыми от встроенных в ОС.

Учитывая сказанное, можно сделать вывод, что для возможности использования ОС общего назначения в качестве базовой в разрабатываемой АС возникает необходимость доработки этой ОС для дублирования или замены существующих механизмов защиты, а также реализации недостающих механизмов. При подобной доработке возникает ряд проблем. Одной из наиболее серьезных является недоступность или скудность низкоуровневой документации по ОС, ее механизмам и структурам данных. Это часто не позволяет разработчикам средств защиты максимально корректно встраивать в ОС дополнительные механизмы защиты и устранять известные или потенциальные уязвимые системы. А поскольку большинство механизмов (например разграничение доступа, аудит и т.п.) должно реализовываться на нижнем уровне ОС, данная проблема получает особую актуальность.

При проектировании средств защиты информации для АС необходимо также учитывать такую проблему, как несоответствие базовых (атомарных) операций в используемых при проектировании моделях защиты и реальных ОС. Так, в большинстве моделей разграничения доступа используются обычно такие элементарные операции, как чтение, запись, реже исполнение, создание, переименование, удаление. В то же время ОС может атомарно реализовывать более сложные операции. Пример такой операции – пересоздание файла (подразумевает создание нового файла либо очистку содержимого существующего файла). Следовательно, подобным операциям должно ставиться в соответствие некое множество (или последовательность) более простых операций. При этом возникает вопрос, как будет корректнее рассматривать данную операцию в случае существования файла: как запись в файл пустого набора данных (итоговый размер файла становится равен 0) или как удаление существующего файла и последующее создание нового, пустого файла. Выбор конкретного варианта определяет реализацию и механизма разграничения доступа к файлам, и механизма регистрации событий.

Рассмотрим проблему неориентированности на защиту низкоуровневых механизмов базовой ОС. Организация и алгоритмы функционирования ряда подобных механизмов ОС не предусматривают возможность корректного встраивания дополнительных механизмов их защиты. Так, возможность дискреционного разграничения доступа к файлам и каталогам заложена в идеологию большинства современных ОС, соответственно существуют встроенные в ОС механизмы разграничения. В то же время организация разграничения доступа, например к портам ввода/вывода (ПВВ) или каналам связи, является задачей неоднозначной и нетривиальной. Здесь существуют два основных аспекта. Рассмотрим их на примере ПВВ.

Во-первых, ПВВ функционально является каналом связи ЭВМ с каким-либо устройством ввода/вывода (а возможно, с другой ЭВМ). Следовательно, должна осуществляться некая идентификация подключенного к порту устройства для сопоставления ему какого-либо уровня секретно- сти и т.п. Однако далеко не все устройства можно идентифицировать, например, по причине элементарного отсутствия у них подобных функций, а также из-за того, что получение подобных данных может быть не предусмотрено протоколом взаимодействия через данный порт либо с данным типом устройства. Эта проблема актуальна в основном при работе со старой техникой, поскольку современная аппаратура обычно поддерживает технологию Plug&Play, частично этот вопрос решающую.

Другим аспектом защиты ПВВ является использование при работе с ними множества различных протоколов высокого (например прикладного) уровня. Например,

LPT-порт может использоваться для подключения принтера, сканера, электронного ключа, другого компьютера; в каждом случае будет использоваться свой протокол взаимодействия через порт. В таком случае возникает вопрос, на каком уровне (на уровне какого протокола) необходимо осуществлять разграничение доступа. Наиболее универсальным выглядит разграничение на самом нижнем уровне – уровне непосредственной передачи данных. Однако при этом следует учитывать двунаправленность потоков данных при использовании протоколов более высокого уровня («проблема удаленного чтения» – для чтения данных необходимо предварительно послать запрос на эти данные [3]). Разграничение же на более высоком уровне автоматически ограничивает доступные для использования протоколы набором, для которого это разграничение реализовано. Эти ограничения особенно актуальны для универсальных ПВВ типа COM, LPT, USB, IEEE1394.

Проблема неприспособленности к функционированию в условиях защиты информации актуальна также и для высокоуровневых (прикладных) механизмов. Так, при проектировании АС часто выявляются такие тривиальные и распространенные задачи, как редактирование текстов и электронных таблиц, просмотр web-страниц и т.п. Разработка специального ПО для решения таких задач в рамках проектируемой АС часто бывает избыточной, поскольку требует порой значительных затрат труда, в то время как существующее прикладное ПО общего назначения прекрасно с такими задачами справляется.

### 1. 5 Лекция № 10-11 (4 часа).

**Тема: Технология доступа к базам данных ADO, BDE, ODBC, COM, CORBA.**

1.1.1 Вопросы лекции:

1.1.2 Организация взаимоействия клиент-сервер.

1.1.3 Перенос персональных данных на сервер.

1.1.4 Краткое содержание вопросов:

На заре эпохи баз данных разработчикам достаточно было знать только те базы данных, которые они использовали. Но базы данных и их технологии развивались довольно быстро — от реляционных баз данных к нереляционным информационным хранилищам, таким, как электронная почта и файловые системы. Развитие баз данных сейчас идет в ногу со стремительными изменениями в технике. А с появлением клиент-серверных и многоуровневых архитектур разработчикам уже приходится разбираться во всем многообразии технологий баз данных. Большинство разработчиков потратили годы на изучение ODBC, DAO, RDO, OLE DB, ADO и RDS. К настоящему моменту Microsoft представила .NET Framework и вместе с ней новую технологию баз данных ADO.NET.

Как только мы начинаем углубляться в какую-то новую технологию, мы забываем, как она развивалась и что рационального за ней стоит. Проследив развитие технологий баз данных от ODBC до ADO.NET, проще выбрать подходящую технологию и оптимизировать ее для своих целей.

ODBC

В большинстве систем проектирования баз данных приложения основываются на одном типе баз данных. В таких простых схемах разработчик приложения может программировать напрямую, используя системный интерфейс базы данных. Хотя

подобный подход обеспечивает быстрый и эффективный доступ к данным, могут возникать проблемы, когда задача расширяется, и разработчику приходится дорабатывать программу. При данном подходе это означает, что каждая готовая программа должна иметь различные версии с поддержкой всевозможных типов баз данных. Если компании расширяются или объединяются одна с другой, приложение должно получить доступ к базам данных, основанным на различных платформах.

Технология ODBC обеспечивает общий интерфейс для доступа к разнородным базам данных стандарта SQL. ODBC использует язык SQL как стандарт для доступа к данным. На Рисунке 1 показана архитектура ODBC. Этот интерфейс очень удобен: одно приложение может обращаться к различным базам данных SQL через общий набор команд. Таким образом, разработчик может создавать и распространять приложения, не привязываясь к конкретной базе данных. Можно также добавить драйвер базы данных, чтобы приложение могло работать с базой данных по выбору пользователя. Как показано на Рисунке 1, менеджер драйверов является промежуточным звеном между приложением и базами данных. Интерфейс ODBC содержит набор функций, который управляет каждым инструментом базы данных. Если приложению нужно сменить используемую базу, разработчик просто заменяет один драйвер другим, и приложение может работать как обычно, без необходимости модификации кода программы.

OLE DB

Спустя несколько лет ODBC становится стандартом для клиент-серверного доступа к базам данных. ODBC обеспечивает стандартный интерфейс, который требует функций SQL и оптимизирован под методы SQL. Однако что произойдет, если нужно будет обратиться к нереляционной базе данных, в которой не используются принципы SQL (например, Microsoft Exchange Server, хранилище которого не содержит данные реляционно).

Рассмотрим OLE DB. Технология OLE DB построена на ODBC и расширяет ее до компонентной архитектуры, которая обеспечивает высокоуровневый интерфейс доступа к данным. Эта архитектура предоставляет постоянный доступ к SQL-данным, не SQL-данным и неструктурированным источникам данных по локальным сетям и Internet. В действительности для доступа к SQL-данным OLE DB использует ODBC, потому что это самая подходящая архитектура для работы с SQL. На Рисунке 5 показано, что OLE DB состоит из трех компонентов: потребителя данных (например, приложения); поставщика (провайдера) данных, который содержит и предоставляет данные; служебного компонента, который обрабатывает и транспортирует данные (в частности, процессоры запросов, процессоры курсоров). OLE DB — это единый API, который обрабатывает как совместимые с SQL источники данных, так и несовместимые, такие, как почта и каталоги.

ADO

OLE DB обеспечивает связывание для программистов на C и C++, а также программистов, использующих языки с C-подобными вызовами функций. Такие языки, как VB и VBScript, не поддерживают тип данных «указатель» (адресных переменных). Следовательно, они не могут использовать связывание в стиле C и прямое обращение к OLE DB.

Вероятно, для большей путаницы разработчики Microsoft ввели еще одну объектную модель доступа к данным: ADO. ADO работает с объектами DAO и RDO, а также поддерживает более простые модели, чем DAO и RDO (хотя с избыточной функциональностью, так что можно выполнить операцию несколькими способами).

Объектная иерархия в ADO более однородная, чем в DAO. ADO содержит несколько встроенных объектов, которые упрощают доступ к данным из информационных хранилищ.

На Рисунке 6 показано несколько способов, с помощью которых приложение связывается с базой данных. Например, VB-программист может использовать ADO для соединения приложения с провайдером OLE DB. Если база данных не поддерживает OLE DB, приложение может задействовать ODBC. Программист на Visual C++ может применять ADO или соединяться напрямую через OLE DB.

**1. 6 Лекция № 12-13 (4 часа).**

**Тема: Клиенты удаленного доступа и построение запросов к СУБД**

1.1.1 Вопросы лекции:

1.1.2 Хранимые процедуры и триггеры.

1.1.3 Достоинства хранимых процедур.

1.1.4 Краткое содержание вопросов:

В технологиях распределенных информационных систем в настоящее время существуют следующие направления:

- технологии ***Клиент-сервер***;
- технологии ***реплицирования (тиражирования)***;
- технологии ***объектного связывания***.

Реальные распределенные информационные системы, как правило, построены на основе сочетания всех трех технологий.

Технологии и модели ***Клиент-сервер.***

В основе клиент-серверных технологий лежат две основные идеи:

- общие для всех пользователей данные на одном или нескольких серверах;
- много пользователей (клиентов) на различных вычислительных установках, совместно (параллельно и одновременно) обрабатывающих общие данные.

Под ***сервером*** в широком смысле понимается любая система, процесс, компьютер, владеющие каким-либо вычислительным ресурсом (памятью, временем, производительностью процессора и т.д.).

***Клиентом*** называется также любая система, процесс, компьютер, пользователь, запрашивающие у сервера какой-либо ресурс, пользующиеся каким-либо ресурсом или обслуживаемые сервером иным способом.

***В структуре СУБД выделяют три компонента***:

- компонент представления, реализующий функции ввода и отображения данных;
- прикладной компонент, включающий набор запросов, событий, правил, процедур и др. вычислительных функций;
- компонент доступа к данным, реализующий функции хранения, извлечения, физического обновления и изменения данных.

Исходя из особенностей реализации и распределения в системе этих компонентов, различают четыре ***модели технологий ╚Клиент-сервер╝***:

- модель файлового сервера (FS);
- модель удаленного доступа к данным (RDA);
- модель сервера базы данных (DBS);
- модель сервера приложений (AS).

***Модель файлового сервера.***

Один из компьютеров сети определяется файловым сервером (общим хранилищем данных), а все основные компоненты СУБД размещаются на клиентских установках. При обращении к данным ядро СУБД обращается с запросами на ввод-вывод данных к

файловой системе. В оперативную память клиентской установки на время сеанса работы полностью или частично копируется файл базы данных. Достоинства модели: простота и отсутствие высоких требований к производительности сервера. Недостатки: высокий сетевой трафик, отсутствие специальных механизмов СУБД по обеспечению безопасности данных.

***Модель удаленного доступа к данным.***

Эта модель основана на учете специфики хранения и физической обработки данных во внешней памяти для реляционных СУБД. В данной модели компонент доступа к данным реализуется в виде самостоятельной программной части СУБД, называемой SQL-сервером, и размещается на сервере. SQL-сервер выполняет низкоуровневые операции по организации, размещению, хранению и манипулированию данными. На сервере размещаются также файлы БД и системный каталог БД. На клиентских установках размещаются программы, реализующие интерфейсные и прикладные функции СУБД. Прикладной компонент клиента формирует необходимые SQL-инструкции и направляет их SQL-серверу, который принимает, интерпретирует, выполняет, проверяет эти инструкции, обеспечивает выполнение ограничений целостности и безопасности данных и направляет клиентам результаты обработки SQL-инструкций (наборы данных).

Достоинства. В результате реализации такого подхода резко уменьшается загрузка сети. RDA-модель позволяет также унифицировать интерфейс взаимодействия прикладных компонентов СУБД с общими данными. Такое взаимодействие стандартизовано в рамках языка SQL специальным протоколом ODBC, играющим важную роль в обеспечении независимости от типа СУБД на клиентских установках. Это позволяет интегрировать уже существующие локальные БД в создаваемые распределенные информационные системы независимо от типов СУБД клиентов и сервера. Недостатки. Высокие требования к клиентским вычислительным установкам, так как на них выполняются прикладные программы обработки данных. Значительный трафик сети, поскольку с сервера направляются клиентам наборы данных, которые могут иметь существенный объем.

***Модель сервера базы данных.***

На клиентских установках в DBS-модели размещается только интерфейсный компонент, а все остальные компоненты СУБД размещаются на сервере. От клиентов на сервер направляются только вызовы необходимых процедур, запросов и других функций по обработке данных. Все операции по обработке данных выполняются на сервере, а пользователю направляются лишь результаты обработки (а не наборы данных, как в RDA-модели).

Достоинства. Разгрузка сети. Активная роль сервера, что позволяет более эффективно настраивать информационную систему на особенности предметной области, а также более надежно обеспечивать согласованность состояния и изменения данных, что повышает надежность хранения и обработки данных.

***Модель сервера приложений.***

Чтобы разнести требования к вычислительным ресурсам сервера в отношении быстродействия и памяти по разным вычислительным установкам, используется модель сервера приложений. Суть данной модели состоит в переносе прикладного компонента СУБД на специализированный в отношении повышенных по быстродействию ресурсов дополнительный сервер системы √ сервер приложений. На клиентских установках располагается интерфейсная часть СУБД, откуда вызовы функций обработки данных направляются на сервер приложений. За выполнением низкоуровневых операций по доступу к данным сервер приложений обращается к SQL-серверу, направляя ему вызовы SQL-процедур и получая от него наборы данных. Таким образом, сервер приложений управляет формированием транзакций, которые выполняет SQL-сервер. Поэтому прикладной компонент СУБД, расположенный на сервере приложений, называют ***монитором транзакций***. AS-модель, сохраняя достоинства DBS-модели,

позволяет более оптимально построить вычислительную схему информационной системы, однако при этом увеличивается трафик сети.

RDA- и DBS-модели называют двухзвенными (двухуровневыми), AS-модель - трехзвенной (трехуровневой).

На практике используются ***смешанные модели***, когда простые прикладные функции и обеспечение ограничений целостности данных поддерживаются процедурами, хранимыми на сервере, а более сложные функции бизнес-правила реализуются программами, расположенными на клиентских установках или на сервере приложений.

**1. 7 Лекция № 14-15 (4 часа).**

**Тема: Объекты для работы с данными**

1.1.1 Вопросы лекции: Объекты для управления работой приложений и оформления интерфейса. Объекты – контейнеры. Объекты OLE

1.1.2 Краткое содержание вопросов:

БД может содержать разные типы объектов. Каждая СУБД может реализовывать свои типы объектов.

**Таблицы** – основные объекты любой БД, в которых хранятся все данные, имеющиеся в базе, и хранится сама структура базы (поля, их типы и свойства).

**Отчеты** – предназначены для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, время создания отчета и другое).

**Страницы** или страницы доступа к данным – специальные объекты БД, выполненные в коде HTML , размещаемые на web -странице и передаваемые клиенту вместе с ней. Сам по себе объект не является БД, посетитель может с ее помощью просматривать записи базы в полях страницы доступа. Т.о., страницы – интерфейс между клиентом, сервером и базой данных, размещенным на сервере.

**Макросы и модули** – предназначены для автоматизации повторяющихся операций при работе с системой управления БД, так и для создания новых функций путем программирования. Макросы состоят из последовательности внутренних команд СУБД и являются одним из средств автоматизации работы с базой. Модули создаются средствами внешнего языка программирования. Это одно из средств, с помощью которых разработчик БД может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, уровень ее защищенности.

- **Запросы и формы .**

**Запросы** – служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С их помощью выполняют отбор данных, их сортировку и фильтрацию. Можно выполнить преобразование данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое заполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и многое другое.

Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную *результирующую таблицу* (*моментальный снимок* ) – образ отобранных из базовых таблиц полей и записей. Работа с образом происходит быстрее и эффективнее, нежели с таблицами, хранящимися на жестком диске.

Обновление БД тоже можно осуществить посредством запроса. В базовые таблицы все данные вносятся в порядке поступления, т.е. они не упорядочены. Но по соответствующему запросу можно получить отсортированные и отфильтрованные нужным образом данные.

**Формы** – средства для ввода данных, предоставляющие пользователю необходимые для заполнения поля. В них можно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и прочее) для автоматизации ввода. Пример, заполнение определенных полей бланка. При выводе данных с помощью форм можно применять специальные средства их оформления.

- **Интегрированные системы. Понятие интегрированной системы** (ИС).
- **Системы программирования. Понятие системы программирования .**

**Система программирования** – инструментальное ПО, предназначенное для поддержки разработки программных систем на этапах программирования и отладки. Каждая система программирования должна иметь некоторый встроенный в нее язык программирования, предназначенный для общения с человеком – разработчиком программной системы.

В самом общем случае для создания программы на выбранном языке программирования нужно иметь следующие *компоненты* **.**

1. **Текстовый редактор** . Т.к. текст программы записывается с помощью слов, происходящих от английского языка, и символов для записи всевозможных операций, то формировать файл с *исходным текстом программы* можно в любом редакторе. Специализированные редакторы ориентированы на конкретный язык программирования. Подобные редакторы созданы для всех популярных языков программирования и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.

2. **Компилятор** . Исходный текст с помощью **программы-компилятора** переводится в машинный код.

На этом этапе создается промежуточный объектный код (двоичный файл, стандартное решение . OBJ ).

3. **Редактор связей** и **библиотеки функций** . Исходный текст большой программы, как правило, состоит из нескольких модулей. Каждый модуль состоит отдельный файл с объектным кодом, которые затем нужно объединить в единое целое. К ним нужно добавить машинный код подпрограмм, реализующих различные стандартные функции (например, вычисление sin и cos ). Такие функции содержатся в библиотеках (файлы с расширением . LIB ), которые поставляются вместе с компилятором.

Объектный код обрабатывается специальной программой – **редактором связей** или **сборщиком** , который выполняет связывание объектных модулей и машинного кода стандартных функций, находя их в библиотеках, и формирует на выходе работоспособное приложение – **исполнимый код** для конкретной платформы.

Если по каким-то причинам объектный модуль или нужная библиотека не найдены, то сборщик сообщает об ошибке и готовой программы не получается.

4. Исполнимый код – это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение . EXE или . COM .

**1. 8 Лекция № 16-17 (4 часа).**

**Тема: Администрирование и эксплуатация защищенных КС, эксплуатационная документация защищенных КС.**

1.1.1 Вопросы лекции: Модель канала утечки. Методы достижения условия защищенности. Обзор систем контроля защищенности.

1.1.2 Краткое содержание вопросов:

Общие положения по эксплуатации КС Общие положения по эксплуатации изделий, комплексов, средств деятельности. Понятие эксплуатации и системы эксплуатации изделий. Организационные мероприятия по эксплуатации (планирование эксплуатации, контроль технического состояния, анализ показателей надежности и функционирования, рекламационная и претензионная работа, категорирование, списание), их содержание и общая характеристика. 12 Технические мероприятия по эксплуатации (применение по назначению, техническое обслуживание, ремонт, [хранение, сбережение, транспортирование, консервация]). Понятие, содержание и виды технического обслуживания (регламентных работ). Виды ремонтов и особенности организации и проведения ремонтных работ.

Основы организации хранения изделий и комплексов. Особенности эксплуатации автоматизированных информационных систем и изделий ИТ как комплекса технических средств обработки информации (ТСОИ – СВТ, коммуникационное оборудование, линии связи), программного обеспечения (ПО), средств информационного обеспечения (информационная база – БД) и средств организационного обеспечения (коллектива пользователей). Составляющие эксплуатации АС и изделий ИТ – работы, мероприятия и процедуры, характерные для эксплуатации технических средств и изделий (ТСОИ); специальные работы по обеспечению функционирования ПО (развертывание, настройка, устранение сбоев и восстановление после них ПО, авторское сопровождение ПО, включая внесение изменений и доработок в ПО, обеспечение требований по авторскому праву на ПО); специальные работы по обеспечению целостности и сохранности информационной базы (устранение нарушений целостности, внесение изменений/доработок в логическую структуру, в настройки, словарно- классификационную базу, резервирование, архивирование, восстановление данных после сбоев); администрирование работы пользователей (регистрация и установление полномочий, ролей и т.д., обучение пользователей, контроль за выполнением пользователями правил эксплуатации и работы и т.д.). Снятие с эксплуатации защищенных АС и изделий ИТ - архивирование ресурсов информационной базы АС (для последующего возможного использования, в т.ч. функционально- ориентированных данных с соблюдением юридических аспектов); очистка носителей информации (стирание данных и надежное удаление данных); физическое уничтожение носителей данных (в установленных нормативными предписаниями случаях); списание ТСОИ и их утилизация (по требованиям, установленным эксплуатационной документацией, ведомственной и/или локальной нормативной базой, в частности, в отношении компонент, содержащих драгметаллы, ядовитые, опасные вещества и материалы).

Администрирование и эксплуатация защищенных КС

Особенности эксплуатации защищенных КС. Угрозы безопасности на стадии эксплуатации и сопровождения КС. Органы системы управления эксплуатацией защищенных КС (АС), функции и компетенции инженерно-технических, ИТ и обеспечивающих подразделений, подразделений по защите информации. Комиссионные органы. Планирование эксплуатации. Программа обеспечения безопасности при эксплуатации изделия ИТ. Планирование и организация работы пользователей (планирование структуры информационных ресурсов и коллектива пользователей, регистрация пользователей, установление полномочий, обучение).

Управление конфигурацией ПО, ТСОИ. Обеспечение целостности и сохранности ( в т.ч. восстановление при разрушениях) информационной базы. Антивирусная защита ПО. Организационные и программно-технологические меры. Управление функционированием средств защиты информации (реализация политики безопасности в настройках, параметрах и процедурах функционирования КС и СЗИ, устранение изъянов в системах безопасности). Технологические процедуры парольной политики, использования других средств идентификации и аутентификации, криптографических средств.

Организация и обеспечение безопасного содержания и законодательно установленного порядка использования дистрибутивов ПО, модификации, адаптации и восстановления ПО. 13 Мониторинг, контроль, аудит безопасности в КС. Организационные и программно- технические меры. Обеспечение безопасного содержания ТСОИ и защиты от воздействия факторов внешней среды. Техническое обслуживание и ремонт ТСОИ (в т.ч. профилактика носителей информации, обеспечение расходными и пополняемыми комплектующими материалами, носителями данных), ведение эксплуатационной документации, рекламационная и претензионная работа. Сопровождение ПО. Организационные аспекты администрирования. Администраторы сетей, КС, безопасности. Нормативное обеспечение администрирования, Положение об администраторе информационной безопасности. 4.2.3.3 Эксплуатационная документация защищенных КС Конструкторские эксплуатационные документы (руководства/инструкции по эксплуатации и эксплуатационные документы на средства по видам обеспечения – на ТСОИ, ПО). Структура Руководства пользователя АС (по РД 50-34.698-99). Структура руководства пользователя изделия ИТ (по ГОСТ Р ИСО/МЭК 15408-2002 Ч.3. Класс AGD). Структура Руководства администратора системы ИТ (по ГОСТ Р ИСО/МЭК 15408-2002 Ч.3. Класс AGD). Конструкторская эксплуатационная документация на ТСОИ и ПО (по ГОСТ 19.101 и ГОСТ 2.601-95).

## 2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

### 2.1 Лабораторная работа № 1-3 (6 часов).

**Тема:** «Понятие, виды и структура автоматизированных систем»

**2.1.1 Цель работы:** Определить, что является автоматизированной системой. Выделить виды и структуры автоматизированных систем.

**2.1.2 Задачи работы:**

1. Дать понятие автоматизированных систем.

2. Определить виды автоматизированных систем.

3. Выделить структуру автоматизированных систем.

**2.1.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительная машина.

**2.1.4 Описание (ход) работы:**

1. В классической теории баз данных, модель данных есть
формальная теория представления и обработки данных в системе управления базами данных(СУБД), которая включает, по меньшей мере, три аспекта:

- аспект структуры: методы описания типов и логических структур данных в базе данных;
- аспект манипуляции: методы манипулирования данными;
- аспект целостности: методы описания и поддержки целостности базы данных.

Аспект структуры определяет, что из себя логически представляет база данных, аспект манипуляции определяет способы перехода между состояниямибазы данных (то есть способы модификации данных) и способы извлечения данных из базы данных, аспект целостности определяет средства описанийкорректных состояний базы данных.

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных[1].

Каждая БД и СУБД строится на основе некоторой явной или неявной модели данных. Все СУБД, построенные на одной и той же модели данных, относят к одному типу. Например, основой реляционных СУБД является реляционная модель данных, сетевых СУБД — сетевая модель данных, иерархических СУБД — иерархическая модель данных и т. д.

2. Иерархическая модель данных — это модель данных, где используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами (в программировании применительно к структуре данных дерево устоялось название братья).

Базы данных с иерархической моделью одни из самых старых, и стали первыми системами управления базами данных для мейнфреймов. Разрабатывались в 1950-х и 1960-х, например, InformationManagementSystem (IMS) фирмы IBM.

**2.2 Лабораторная работа №4-6 ( 6 часов).**

**Тема:** «Жизненный цикл АС»

**2.2.1 Цель работы:** Рассмотреть жизненный цикл АС.

**2.2.2 Задачи работы:**

1. Рассмотреть жизненный цикл АС

**2.2.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительная машина

**2.2.4 Описание (ход) работы:**

Понятие *домена* более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Отношение - это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой.

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения.

**2.3 Лабораторная работа №7-9 ( 6 часов).**

**Тема:** «Порядок создания изделий ИТ, удовлетворяющих требованиям безопасности»

**2.3.1 Цель работы:** определить порядок создания изделий ИТ, удовлетворяющих требованиям безопасности

**2.3.2 Задачи работы:**

1. Стандарты требований безопасности.

2. Создание изделий ИТ

**2.3.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительна машина.

**2.3.4 Описание (ход) работы:**

***Схемой отношения*** называется перечень имен атрибутов данного отношения с указанием домена, к которому они относятся:

$$S_R = (A_1, A_2, A_n),\ A_i \subseteq D_i.$$

Если атрибуты принимают значения из одного и того же домена, то они называются $\theta$-сравнимыми, где $\theta$ – множество допустимых операций сравнений, заданных для данного домена. Например, если домен содержит числовые данные, то для него допустимы все операции сравнения, тогда $\theta = \{=, <>, >=, <=, <, >\}$. Однако, и для доменов, содержащих символьные данные, могут быть заданы не только операции сравнения по равенству и неравенству значений. Если для данного домена задано лексикографическое упорядочение, то он имеет также полный спектр операций сравнения.

Схемы двух отношений называются ***эквивалентными***, если они имеют одинаковую степень и возможно такое упорядочение имен атрибутов в схемах, что на одинаковых местах будут находиться сравнимые атрибуты, то есть атрибуты, принимающие значения из одного домена:

Пусть $S_{R_1} = (A_1, A_2, \ldots, A_n)$ – схема отношения $R_1$. $S_{R_2} = (B_{i1}, B_{i2}, \ldots, B_{in})$ – схема отношения $R_2$ после упорядочения имен атрибутов. Тогда

$$S_{R_1} \sim S_{R_2} \Leftrightarrow \begin{cases} 1.\ n = m, \\ 2.\ A_j, B_{ij} \subseteq D_j. \end{cases}$$

Таким образом, для эквивалентных отношений выполняются следующие условия:

- Таблицы имеют одинаковое количество столбцов.
- Таблицы содержат столбцы с одинаковыми наименованиями.
- Столбцы с одинаковыми наименованиями содержат данные из одних и тех же доменов.
- Таблицы имеют одинаковые строки с учетом того, что порядок столбцов может различаться.

Все такие таблицы есть различные *изображения* одного и того же отношения.

### 2.4 Лабораторная работа №10-11 ( 4 часа).

**Тема:** «Технология доступа к базам данных ADO, BDE, ODBC, COM, CORBA»

**2.4.1 Цель работы:** рассмотреть технологию доступа к базам данных ADO, BDE, ODBC, COM, CORBA

**2.4.2 Задачи работы:**

1. Доступ к базам данных

**2.4.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительная машина

**2.4.4 Описание (ход) работы:**

1.Первичные и внешние ключи

Каждая **реляционная таблица** должна обладать следующими свойствами:

• один элемент таблицы — один элемент данных;

• все столбцы таблицы содержат однородные по типу данные (целочисленный, числовой, текстовый, и т.д.);

• каждый столбец имеет уникальное имя;

• число столбцов задается при создании таблицы;

• порядок записей в отношении может быть произвольным;

• записи не должны повторяться;

• количество записей в отношении не ограничено.

Объекты, их взаимосвязи и отношения представлены в виде таблиц. Формальное построение таблиц связано с фундаментальным понятием отношение (термин реляционная исходит от английского слова relation — отношение).

В реляционной таблице каждый столбец есть домен (его альтернативное название поле), а совокупность элементов каждой строки — кортеж (или запись).

Строка заголовков называется схемой отношения. В отношении каждый конкретный экземпляр сущности представляется строкой, которая называется кортежем (или записью).

Первичным ключом отношения называется поле или группа полей, однозначно определяющие запись. На практике обычно в качестве ключевого выбирают поле, в котором совпадения заведомо исключены.

**Свойства первичного ключа:**
• уникальность — в таблице может быть назначен только один первичный ключ, у составного ключа поля могут повторяться, но не все;

• неизбыточность — не должно быть полей, которые, будучи удаленными из первичного ключа, не нарушат его уникальность;

• в состав первичного ключа не должны входить поля типа, комментарий и графическое.

Чтобы избежать повторяющихся записей, приходят к связыванию таблиц.

Для связи реляционных таблиц необходимо ввести в обе таблицы одинаковые по типу поля, по которым определится связь между записями обеих таблиц. Связи бывают нескольких типов «один к одному», «один ко многим», «многие ко многим». В вышеприведенном примере была установлена связь «один ко многим».

### 2.5 Лабораторная работа №12-13 ( 4 часа).

**Тема: «**Клиенты удаленного доступа и построение запросов к СУБД**»**

**2.5.1 Цель работы:** рассмотреть пример клиента удаленного доступа и построить запрос к СУБД

**2.5.2 Задачи работы:**

1. рассмотреть пример клиента удаленного доступа

2. построить запрос к СУБД

**2.5.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительная машина

**2.5.4 Описание (ход) работы:**
1. Особенности языков описания и манипулирования данными в реляционной модели языки запросов, основанные на реляционном исчислении.
структурный язык запросов SQL.

Язык для взаимодействия с БД SQL появился в середине 70-х и был разработан в рамках проекта экспериментальной реляционной СУБД System R. Исходное название языка SEQUEL (StructeredEnglishQueryLanguage) только частично отражает суть этого языка. Конечно, язык был ориентирован главным образом на удобную и понятную пользователям формулировку запросов к реляционной БД, но на самом деле уже являлся полным языком БД, содержащим помимо операторов формулирования запросов и манипулирования БД средства определения и манипулирования схемой БД; определения ограничений целостности и триггеров; представлений БД; возможности определения структур физического уровня, поддерживающих эффективное выполнение запросов; авторизации доступа к отношениям и их полям; точек сохранения транзакции и откатов. В языке отсутствовали средства синхронизации доступа к объектам БД со стороны параллельно выполняемых транзакций: с самого начала предполагалось, что необходимую синхронизацию неявно выполняет СУБД.

Рассмотрим эти свойства языка немного более подробно.

### 13.1.1. Запросы и операторы манипулирования данными

Как известно, двумя фундаментальными языками запросов к реляционным БД являются языки реляционной алгебры и реляционного исчисления. При всей своей строгости и теоретической обоснованности эти языки редко используются в современных реляционных СУБД в качестве средств пользовательского интерфейса. Запросы на этих языках трудно формулировать и понимать. SQL представляет собой некоторую комбинацию реляционного исчисления кортежей и реляционной алгебры, причем до сих пор нет общего согласия, к какому из классических языков он ближе. При этом возможности SQL шире, чем у этих базовых реляционных языков, в частности, в общем случае невозможна трансляция запроса, сформулированного на SQL, в выражение реляционной алгебры, требуется некоторое ее расширение.

Существенными свойствами подъязыка запросов SQL являются возможность простого формулирования запросов с соединениями нескольких отношений и использование вложенных подзапросов в предикатах выборки. Вообще говоря, одновременное наличие обоих средств избыточно, но это дает пользователю при формулировании запроса возможность выбора более понятного ему варианта.

В предикатах со вложенными подзапросами в SQL System R можно употреблять теретико-множественные операторы сравнения, что позволяет формулировать квантифицированные запросы (эти возможности обычно труднее всего понимаются пользователями и поэтому в дальнейшем в SQL появились явно квантифицируемые предикаты).

Существенной особенностью SQL является возможность указания в запросе потребности группирования отношения-результата по указанным полям с поддержкой условий выборки на всю группу целиком. Такие условия выборки могут содержать агрегатные функции, вычисляемые на группе. Эта возможность SQL главным образом отличает этот язык от языков реляционной алгебры и реляционного исчисления, не содержащих аналогичных средств.

Еще одним отличием SQL является необязательное удаление кортежей-дубликатов в окончательном или промежуточных отношениях-результатах. Строго говоря, результатом

оператора выборки в языке SQL является не отношение, а мультимножество кортежей. В тех случаях, когда семантика запроса требует наличия отношения, уничтожение дубликатов производится неявно.

Самый общий вид запроса на языке SQL представляет теоретико-множественное алгебраическое выражение, составленное из элементарных запросов. В SQL System R допускались все базовые теретико-множественные операции (UNION, INTERSECT и MINUS).

Работа с неопределенными значениями в SQL System R до конца продумана не была, хотя неявно предполагалось использование трехзначной логики при вычислении логических выражений.

Операторы манипулирования данными UPDATE и DELETE построены на тех же принципах, что и оператор выборки данных SELECT. Набор кортежей указанного отношения, подлежащих модификации или удалению, определяется входящим всоответствующий оператор логическим выражением, которое может включать сложные предикаты, в том числе и с вложенными подзапросами.

В операторе вставки кортежа(ей) в указанное отношение заносимый кортеж может задаваться как в литеральной форме, так и с помощью внутреннего подоператора выборки.

13.1.2. Операторы определения и манипулирования схемой БД

В число операторов определения схемы БД SQL System R входили операторы создания и уничтожения постоянных и временных хранимых отношений (CREATE TABLE и DROP TABLE) и создания и уничтожения представляемых отношений (CREATE VIEW и DROP VIEW). В языке и в реализации System R не запрещалось использовать операторы определения схемы в пределах транзакции, содержащей операторы выборки и манипулирования данными. Допускалось, например, использование операторов выборки и манипулирования данными, в которых указываются отношения, не существующие в БД к моменту компиляции оператора. Конечно, эта возможность существенно усложняла реализацию и требовалась по существу очень редко.

Оператор манипулирования схемой БД ALTER TABLE позволял добавлять указываемые поля к существующим отношениям. В описании языка определялось, что выполнение этого оператора не должно приводить к недействительности ранее откомпилированных операторов над отношением, схема которого изменяется, и что значения вновь определенных полей в существующих кортежах отношения становятся неопределенными.

### 2.6 Лабораторная работа №14-15 (4 часа).

**Тема:** «Объекты для работы с данными»

**2.6.1 Цель работы:** выделить объекты для работы с данными

**2.6.2 Задачи работы:**

1. Дать понятие объекту

2. Выделить объекты для работы с данными

**2.6.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительная машина

**2.6.4 Описание (ход) работы:**

Язык SQL System R включал очень мощные средства контроля и поддержания целостности БД. Средства контроля базировались на аппарате ограничений целостности (ASSERTIONS). Фактически, ограничение целостности - это логическое выражение, вычисляемое над текущим состоянием БД, ложность которого соответствует нецелостному состоянию БД. Логическое выражение ограничения целостности могло содержать любой допустимый в языке предикат.

Более точно, ограничения целостности делились на два класса: проверяемые после выполнения оператора манипулирования данными и проверяемые при завершении транзакции или при выполнении специального оператора INFORCE INTEGRITY. Типы предикатов, которые можно использовать в операторах определения ограничений целостности разных классов, различаются. В операторах первого класса проверяется, фактически, текущий кортеж, с которым производится манипулирование. Во втором случае проверяются указанные в ограничении целостности отношения, т.е. все их кортежи. Различается и определяемая в языке реакция системы на нарушения ограничений целостности разных классов. В первом случае нарушение ограничения целостности приводит к откату транзакции в точку, непосредственно предшествующую операции манипулирования данными, выполнение которого вызвало нарушение ограничения целостности. Во втором случае ограничение приводит к полному откату транзакции к ее началу.

Очень важным механизмом, определенным в языке SQL System R, является механизм триггеров. В контексте System R этот механизм рассматривался главным образом как средство автоматического поддержания целостности БД. При определении триггера указывалось условие проверки его применимости (имя отношения и тип операции манипулирования данными), условие применимости триггера (логическое выражение, построенное по правилам, близким к правилам для ограничений целостности первого класса) и действие, которое должно быть выполнено над БД в случае истинности условия применимости. Такое действие могло быть выражено с помощью произвольного оператора манипулирования данными. Во время выполнения действия могли срабатывать другие триггеры и т.д.

Механизмы ограничений целостности и триггеров System R являлись очень мощными и общими, но реализация их очень трудна и накладна (как уже отмечалось, триггеры так и не были реализованы в System R). Дополнительную сложность в реализации создавал тот факт, что допускалось (по крайней мере не запрещалось языком) определение ограничений целостности и триггеров в пределах той же транзакции, в которой выполняются операторы манипулирования данными. При наиболее полной реализации требовалось бы большое число дополнительных действий во время выполнения транзакции. Кроме того, в ряде случаев отсутствие зафиксированной семантики соответствующих конструкций языка приводило к неоднозначному пониманию выполнения транзакций.

**2.7 Лабораторная работа №16-17 (4 часа).**

**Тема: «**Администрирование и эксплуатация защищенных КС, эксплуатационная документация защищенных КС**»**

**2.7.1 Цель работы:** Администрирование и эксплуатация защищенных КС, эксплуатационная документация защищенных КС

**2.7.2 Задачи работы:**

1. Администрирование защищенных КС
2. Эксплуатация защищенных КС
3. Эксплуатационная документация защищенных КС

**2.7.3 Перечень приборов, материалов, используемых в лабораторной работе:**

1. Электронно-вычислительная машина

**2.7.4 Описание (ход) работы:**

В языке допускалось использование хранимых отношений БД и представляемых отношений. Наиболее удачным решением было использование для определения представлений общего аппарата операторов выборки. Любой оператор выборки может быть использован для определения представления.

В языке отсутствуют какие-либо ограничения по поводу использования представлений: в любом операторе SQL, в котором допускается использование имени хранимого отношения, допускается и использование имени представления. В SQL System R ничего не говорится о рекомендуемом способе реализации доступа к представлениям, но при любом способе эффект должен быть таким, как если бы выполнить полную материализацию представления до выполнения оператора.

Массу проблем, исследований и предложений породила потенциальная возможность выполнения операторов манипулирования данными над представлениями. Понятно, что эта возможность легко реализуема для простых представлений, но в более сложных случаях не только реализация, но и семантика операций становится нетривиальной. Кстати, в System R операторы манипулирования данными допускались только над простыми представлениями.

13.1.5. Определение управляющих структур

Внесение в реляционный язык, каким является SQL, явных операторов порождения и уничтожения структур физического уровня, поддерживающих эффективное выполнение запросов к БД, явилось в SQL System R чисто прагматическим решением, обеспечивающим возможность всех видов работ с БД с помощью одного языка.

В SQL System R упоминаются два вида таких структур: индексы и связи (links). Индекс в его абстрактном языковом представлении - это инвертированный файл, обеспечивающий доступ к кортежам соответствующего отношения на основе заданных значений одного или нескольких столбцов, составляющих ключ индекса. Операторы языка позволяли создавать и уничтожать индексы, но никаким образом не давали возможности явно указать на необходимость использования существующего индекса при выполнении оператора выборки, решение об этом возлагалось на реализацию.

С помощью оператора определения индекса можно было выразить два дополнительных утверждения, касающихся логической схемы отношения и физической структуры его хранения. Использование при определении индекса ключевого слова UNIQUE означало, что ключ этого индекса является возможным ключом соответствующего отношения.

Фактически это означает наличие дополнительного механизма определения ограничения целостности отношения. Один из индексов для данного отношения мог быть определен с ключевым словом CLUSTERING. Это означает требование физической кластеризации во внешней памяти кортежей отношения с равными или близкими значениями ключа индекса.