МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Маркетинг и менеджмент»

БАЗЫ ДАННЫХ

Методические рекомендации к самостоятельной работе для студентов специальности 1-28 01 02 «Электронный маркетинг» заочной формы обучения

Могилев 2022

УДК 004.43
ББК 32.973-018
Б17

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Маркетинг и менеджмент» «30» июня 2021 г., протокол № 13

Составитель канд. физ.-мат. наук, доц. С. Н. Батан

Рецензент канд. техн. наук, доц. В. М. Ковальчук

В методических рекомендациях представлены описание аудиторной контрольной работы, критерии ее оценки, краткое изложение теоретического учебного материала, а также перечень вопросов для написания аудиторной контрольной работы.

Учебно-методическое издание

БАЗЫ ДАННЫХ

Ответственный за выпуск	А. В. Александров
Корректор	Т. А. Рыжикова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Порядок выполнения аудиторной контрольной работы и критерии её оценки

Аудиторная контрольная работа (АКР) направлена на проверку подготовленности студента по теоретической части дисциплины.

АКР включает два теоретических вопроса из 16 тем.

Для получения зачета по АКР необходимо дать исчерпывающий ответ на оба теоретических вопроса.

АКР оценивается исходя из 10 (десяти) баллов. Критерии оценки представлены в таблице 1. АКР считается зачтенной, если сумма полученных баллов составляет не менее 5 (пяти) баллов.

Таблица 1 – Критерии оценки АКР

Задание	Максимальный балл
Вопрос 1	5
Вопрос 2	5
Итого по заданиям	**10**

1 Логическое проектирование

1.1 Основные понятия, лежащие в основе концепции баз данных

Понятие информационной системы.

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.

Информационный процесс – процесс создания, сбора, обработки, накопления, хранения, поиска, распространения и потребления информации.

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).

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.

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

2 *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).

3 *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 определяются 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).

1.2 Понятие базы данных и системы баз данных

База данных (БД) представляет собой совокупность структурированных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов, и их взаимосвязей в рассматриваемой предметной области.

Система управления базами данных (СУБД) – это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

Словарь данных представляет собой подсистему БД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т. п.

Информационные системы, основанные на использовании БД, обычно функционируют в архитектуре клиент-сервер. В этом случае БД размещается на компьютере-сервере, и к ней осуществляется совместный доступ.

Сервером определенного ресурса в компьютерной сети называется компьютер (программа), управляющий этим ресурсом, **клиентом** – компьютер (программа), использующий этот ресурс. В качестве ресурса компьютерной сети могут выступать, к примеру, базы данных, файлы, службы печати, почтовые службы.

Система управления базами данных (СУБД) – это система программного обеспечения, позволяющая обрабатывать обращения к базе данных, поступающие от прикладных программ конечных пользователей. Иными словами, СУБД является интерфейсом между базой данных и прикладными задачами.

Основные функции СУБД:

– управление данными во внешней памяти (на дисках);

– управление данными в оперативной памяти;

– журнализация изменений и восстановление базы данных после сбоев;

– поддержание языков БД (язык определения данных, язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты:

– ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;

– процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;

– подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД;

– сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию.

Современные СУБД дают возможность включать в них не только текстовую и графическую информацию, но и звуковые фрагменты и даже видеоклипы.

Простота использования СУБД позволяет создавать новые базы данных, не прибегая к программированию, а используя только встроенные функции.

СУБД обеспечивают правильность, полноту и непротиворечивость данных, а также удобный доступ к ним.

Для менее сложных применений вместо СУБД используются информационно-поисковые системы (ИПС), которые выполняют следующие функции:

– хранение большого объема информации;
– быстрый поиск требуемой информации;
– добавление, удаление и изменение хранимой информации;
– вывод ее в удобном для человека виде.

1.3 Жизненный цикл базы данных

Процесс проектирования, реализации и поддержания системы базы данных называется жизненным циклом базы данных (ЖЦБД). Процедура создания системы называется жизненным циклом системы (ЖЦС).

Понимание и правильный подход к ЖЦБД очень важен и требует детального рассмотрения, т. к. в его основе лежит подход, ориентированный *на данные*. Элементы данных более стабильны, чем выполняемые функции системы. Создание правильной структуры данных требует сложного анализа классов единиц данных и отношений между ними. Если построить логичную схему базы данных, то в дальнейшем можно создать любое количество функциональных систем, использующих эту схему. Функционально ориентированный подход можно применять лишь для создания временных систем, которые рассчитаны на недолгое время функционирования.

ЖЦБД состоит из следующих этапов.

1 Предварительное планирование – планирование БД, выполняемое в процессе разработки стратегического плана БД. В процессе планирования собирается следующая информация: какие прикладные программы используются и какие функции они выполняют; какие файлы связаны с каждым из этих приложений; какие новые приложения и файлы находятся в процессе работы.

Данная информация помогает определить, как используется информация приложений, определить будущие требования к системе БД.

Информация этого этапа документируется в виде обобщенной модели данных.

2 Проверка осуществимости. Здесь определяется технологическая, операционная и экономическая осуществимость плана создания БД. Технологическая

осуществимость – есть ли технология для реализации запланированной БД? Операционная осуществимость – есть ли средства и эксперты, необходимые для успешного осуществления плана создания БД? Экономическая целесообразность – можно ли определить выводы; окупится ли запланированная система; можно ли оценить издержки и выгоду?

3 Определение требований включает выбор целей БД, выяснение информационных требований к системе и требований к оборудованию и программному обеспечению. Таким образом, на данном этапе сбора данных и определения требований создаётся общая информационная модель, выражающаяся в следующих задачах.

Определяются цели системы путём анализа информационных потребностей. Здесь также обязательно указывается, какую именно БД следует создавать (распределённую, целостную) и какие коммуникационные средства необходимы. Выходной документ – комментарий, описывающий цели системы.

Определение пользовательских требований: документация в виде обобщённой информации (комментарии, отчёты, опросы, анкеты и т. д.); фиксация функций системы и определение прикладных систем, которые будут выполнять эти требования. Данные представляются в виде соответствующих документов.

Определение общих требований к оборудованию и программному обеспечению, связанных с поддержанием желаемого уровня быстродействия (выяснение количества пользователей системы, числа входных сообщений в день, количество распечаток). Данная информация используется для выбора типов компьютеров и СУБД, объёма дисков, количества принтеров. Данные этого этапа излагаются в отчёте, содержащем примерные конфигурации оборудования и программного обеспечения.

Разработка плана поэтапного создания системы, включающего выбор исходных приложений.

4 Концептуальное проектирование – создание концептуальной схемы БД. Спецификации разрабатываются в той степени, которая необходима для перехода к реализации.

Основным выходным документом является единая инфологическая модель (или схема БД на концептуальном уровне). При разработке данной модели используются информация и функции, которые должна выполнить система, определённые на этапе сбора и определения требований к системе. На данном этапе желательно также определить правила для данных, правила для процессов, правила для интерфейса.

5 Реализация – процесс превращения концептуальной модели в функциональную БД. Он включает в себя следующие этапы:

– выбор и приобретение необходимой СУБД;

– преобразование концептуальной (инфологической) модели БД в логическую и физическую модель данных: на основе инфологической модели данных построить схему данных для конкретной СУБД, при необходимости реализовать денормализацию БД с целью ускорения обработки запросов во всех критичных по времени приложениях; определить, какие прикладные процессы необходимо реализовать в схеме данных как хранимые процедуры; реализовать ограничения,

предназначенные для обеспечения целостности данных и реализации правил для данных; спроектировать и сгенерировать триггеры для реализации всех централизованно определённых правил для данных и правил целостности данных, которые не могут быть заданы как ограничения; разработать стратегию индексирования и кластеризации; выполнить оценку размеров всех таблиц, кластеров и индексов; определить уровни доступа пользователей, разработать и внедрить правила обеспечения безопасности и аудита, создать роли и синонимы для обеспечения многопользовательского доступа с согласованными уровнями полномочий доступа; разработать сетевую топологию БД и механизм бесшовного доступа к удалённым данным (реплицированная или распределённая БД);

– построение словаря данных, который определяет хранение определений структуры данных БД. Словарь данных также содержит информацию о полномочиях доступа, правилах защиты данных и контроля данных;
– заполнение базы данных;
– создание прикладных программ, контроль управления;
– обучение пользователей.

6 Оценка и усовершенствование схемы БД включает опрос пользователей с целью выяснения функциональных неучтенных потребностей. При необходимости вносятся изменения, добавление новых программ и элементов данных по мере изменения и расширения потребностей.

Таким образом, ЖЦБД включает в себя:

– изучение предметной области и представление соответствующей документации.
– построение инфологической модели.
– реализацию.
– оценку работы и поддержку БД.

1.4 Модели данных

Различают четыре *основные модели базы данных* – иерархическую, сетевую, реляционную и объектно-ориентированную. Эти модели отличаются между собой по способу установления связей между данными.

1 Иерархический подход к организации баз данных. Иерархические базы данных имеют форму деревьев с дугами-связями и узлами-элементами данных. Иерархическая структура предполагала неравноправие между данными – одни жестко подчинены другим. Подобные структуры, безусловно, четко удовлетворяют требованиям многих, но далеко не всех реальных задач.

2 Сетевая модель данных. В сетевых БД, наряду с вертикальными, реализованы горизонтальные связи. Однако унаследованы многие недостатки иерархической, и главный из них – необходимость четко определять на физическом уровне связи данных и столь же четко следовать этой структуре связей при запросах к базе.

3 Реляционная модель. Реляционная модель появилась вследствие стремления сделать базу данных как можно более гибкой. Данная модель

предоставила простой и эффективный механизм поддержания связей данных.

Во-первых, все данные в модели представляются в виде таблиц, и только таблиц. Реляционная модель единственная из всех обеспечивает единообразие представления данных. И сущности, и связи этих самых сущностей представляются в модели совершенно одинаково – таблицами. Однако такой подход усложняет понимание смысла хранящейся в базе данных информации, и, как следствие, манипулирование этой информацией.

Избежать трудностей манипулирования позволяет второй элемент модели – реляционно-полный язык (отметим, что язык является неотъемлемой частью любой модели данных, без него модель не существует). Полнота языка в приложении к реляционной модели означает, что он должен выполнять любую операцию реляционной алгебры или реляционного исчисления (полнота последних доказана математически Э. Ф. Коддом). Более того, язык должен описывать любой запрос в виде операций с таблицами, а не с их строками. Одним из таких языков является SQL.

Третий элемент реляционной модели требует от реляционной модели поддержания некоторых ограничений целостности. Одно из таких ограничений утверждает, что каждая строка в таблице должна иметь некий уникальный идентификатор, называемый первичным ключом. Второе ограничение накладывается на целостность ссылок между таблицами. Оно утверждает, что атрибуты таблицы, ссылающиеся на первичные ключи других таблиц, должны иметь одно из значений этих первичных ключей.

4 Объектно-ориентированная модель. Новые области использования вычислительной техники, такие как научные исследования, автоматизированное проектирование и автоматизация учреждений, потребовали от баз данных способности хранить и обрабатывать новые объекты – текст, аудио- и видеоинформацию, а также документы. Основные трудности объектно-ориентированного моделирования данных проистекают из того, что такого развитого математического аппарата, на который могла бы опираться общая объектно-ориентированная модель данных, не существует. В большой степени поэтому до сих пор нет базовой объектно-ориентированной модели. С другой стороны, некоторые авторы утверждают, что общая объектно-ориентированная модель данных в классическом смысле и не может быть определена по причине непригодности классического понятия модели данных к парадигме объектной ориентированности.

Иерархическая модель базы данных.

Иерархические базы данных – самая ранняя модель представления сложной структуры данных. Информация в иерархической базе организована по принципу древовидной структуры, в виде отношений «предок-потомок». Каждая запись может иметь не более одной родительской записи и несколько подчиненных. Связи записей реализуются в виде физических указателей с одной записи на другую. Основной недостаток иерархической структуры базы данных – невозможность реализовать отношения «многие-ко-многим», а также ситуации, когда запись имеет несколько предков.

Иерархические базы данных графически могут быть представлены как перевернутое дерево, состоящее из объектов различных уровней. Верхний уровень (корень дерева) занимает один объект, второй – объекты второго уровня и т. д.

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 объект-предок может не иметь потомков или иметь их несколько, тогда как объект-потомок обязательно имеет только одного предка. Объекты, имеющие общего предка, называются близнецами.

Организация данных в СУБД иерархического типа определяется в терминах: элемент, агрегат, запись (группа), групповое отношение, база данных.

Атрибут (элемент данных) – наименьшая единица структуры данных. Обычно каждому элементу при описании базы данных присваивается уникальное имя. По этому имени к нему обращаются при обработке. Элемент данных также часто называют полем.

Запись – именованная совокупность атрибутов. Использование записей позволяет за одно обращение к базе получить некоторую логически связанную совокупность данных. Именно записи изменяются, добавляются и удаляются. Тип записи определяется составом ее атрибутов. Экземпляр записи – конкретная запись с конкретным значением элементов.

Групповое отношение – иерархическое отношение между записями двух типов. Родительская запись (владелец группового отношения) называется исходной записью, а дочерние записи (члены группового отношения) – подчиненными. Иерархическая база данных может хранить только такие древовидные структуры.

Корневая запись каждого дерева обязательно должна содержать ключ с уникальным значением. Ключи некорневых записей должны иметь уникальное значение только в рамках группового отношения. Каждая запись идентифицируется полным сцепленным ключом, под которым понимается совокупность ключей всех записей от корневой, по иерархическому пути.

При графическом изображении групповые отношения изображают дугами ориентированного графа, а типы записей – вершинами (диаграмма Бахмана).

Для групповых отношений в иерархической модели обеспечивается автоматический режим включения и фиксированное членство. Это означает, что для запоминания любой некорневой записи в БД должна существовать ее родительская запись.

Сетевая модель базы данных.

Сетевая модель данных определяется в тех же терминах, что и иерархическая. Она состоит из множества записей, которые могут быть владельцами или членами групповых отношений.

Основное различие этих моделей состоит в том, что в сетевой модели запись может быть членом более чем одного группового отношения. Согласно этой модели, каждое групповое отношение именуется и проводится различие между его типом и экземпляром. Тип группового отношения задается его именем

и определяет свойства, общие для всех экземпляров данного типа. Экземпляр группового отношения представляется записью-владельцем и множеством (возможно пустым) подчиненных записей. При этом имеется следующее ограничение: экземпляр записи не может быть членом двух экземпляров групповых отношений одного типа.

Объектно-реляционные СУБД.

Разница между объектно-реляционными и объектными СУБД: первые являют собой надстройку над реляционной схемой, вторые же изначально объектно-ориентированы. Главная особенность и отличие объектно-реляционных, как и объектных, СУБД от реляционных заключается в том, что О(Р)СУБД интегрированы с объектно-ориентированным языком программирования, внутренним или внешним как C++, Java. Характерные свойства ОРСУБД – комплексные данные, наследование типа и объектное поведение.

Объектно-ориентированные СУБД.

Появление объектно-ориентированных (ОО) СУБД вызвано потребностями программистов на ОО-языках, которым были необходимы средства для хранения объектов, не помещавшихся в оперативной памяти компьютера. Также важна была задача сохранения состояния объектов между повторными запусками прикладной программы. Поэтому большинство ООСУБД представляют собой библиотеку, процедуры управления данными которой включаются в прикладную программу. Примеры реализации ООСУБД как выделенного сервера базы данных крайне редки.

Сразу же необходимо заметить, что общепринятого определения «объектно-ориентированной модели данных» не существует. Сейчас можно говорить лишь о неком «объектном» подходе к логическому представлению данных и о различных объектно-ориентированных способах его реализации.

В объектно-ориентированных базах данных, в отличие от реляционных, хранятся не записи, а объекты. ОО-подход представляет более совершенные средства для отображения реального мира, чем реляционная модель, естественное представление данных. В реляционной модели все отношения принадлежат одному уровню, именно это осложняет преобразование иерархических связей модели «сущность-связь» в реляционную модель. ОО-модель можно рассматривать послойно, на разных уровнях абстракции. Имеется возможность определения новых типов данных и операций с ними.

1.5 Операции реляционной алгебры

Реляционная алгебра – теоретический язык операций, который на основе одного или нескольких отношений позволяет создавать другое отношение без изменения самих исходных отношений.

Теоретико-множественные операции реляционной алгебры:

- объединение (union);
- пересечение (intersection);
- разность (set difference);
- декартово произведение (cartesian product).

Специальные реляционные операции:

- выборка (selection);
- проекция (projection);
- соединение (join);
- деление (division).

Операция объединения R∪S получается в результате конкатенации **R** и **S** с образованием одного отношения с тем же заголовком, что и у отношений **R** и **S** и телом, состоящим из кортежей, принадлежащих или **R**, или **S**, или обоим отношениям (с максимальным количеством кортежей), если кортежи-дубликаты исключены.

Операция пересечения R ∩ S определяет отношение, которое содержит кортежи, присутствующие как в отношении **R**, так и в отношении **S**.

Операция разности R – S определяет отношение с тем же заголовком, что и у отношений **R** и **S**, и телом, состоящим из кортежей, принадлежащих отношению **R** и не принадлежащих отношению **S**, таких, которые имеются в отношении **R**, но отсутствуют в отношении **S**.

Декартово произведение R × S определяет новое отношение, которое является результатом конкатенации (т. е. сцепления) каждого кортежа из отношения **R** с каждым кортежем из отношения **S**.

Операция выборки работает с одним отношением **R**. Определяет результирующее отношение с тем же заголовком, что и отношение **R**, и телом, состоящим из кортежей, значения атрибутов которых при подстановке в условие (предикат) дают значение истина.

Операция проекции работает с одним отношением **R**. Определяет новое отношение с заголовком **(X, …, Z)**, содержащее вертикальное подмножество отношения **R**, создаваемое посредством извлечения значений указанных атрибутов из результата строк-дубликатов.

Операция соединения – комбинация декартового произведения и выборки, эквивалентна операции выборки из декартового произведения двух операндов отношений тех кортежей, которые удовлетворяют условию, указанному в предикате соединения в качестве формулы выборки.

Типы операций соединения:

- тета-соединение;
- соединение по эквивалентности (частный случай тета-соединения);
- естественное соединение;
- внешнее соединение;
- полусоединение.

Операция деления. Пусть отношение **R** определено на множестве атрибутов **A,** отношение **S** – на множестве атрибутов **B**; **B** $\subseteq$ **A**; **C = A – B** (**C** является множеством атрибутов отношения **R**, которые не являются атрибутами отношения **S**). Результат **деления R÷S** – набор кортежей отношения **R**, определенных на множестве атрибутов **C**, которые соответствуют комбинации всех кортежей отношения **S**.

1.6 Типы связей между сущностями

Пусть имеются два множества сущностей – E1 и E2, а R – связь между ними. Тогда:

– R имеет вид «один-ко-много» в направлении от E1 к E2, если посредством R каждый элемент множества E1 может быть соединен с несколькими элементами из множества E2;

– R имеет вид «один-к-одному», если посредством R каждый элемент множества E1 может быть соединен не более чем с одним элементом из множества E2, и наоборот;

– R имеет вид «много-ко-много», если R в обоих направлениях имеет вид «один-ко-много».

1.7 Нормализация данных

Метод нормальных форм – последовательный перевод отношений из первой нормальной формы в нормальные формы более высокого порядка по определенным правилам. Каждая следующая нормальная форма ограничивает определенный тип функциональных зависимостей, устраняет аномалии при выполнении операций над отношениями БД и сохраняет свойства предшествующих нормальных форм. Выделяют следующую последовательность нормальных форм:

– первая нормальная форма (1НФ);
– вторая нормальная форма (2НФ);
– третья нормальная форма (3НФ);
– усиленная третья нормальная форма, или нормальная форма Бойса – Кодда (БКНФ);
– четвертая нормальная форма (4НФ);
– пятая нормальная форма (5НФ).

Перевод отношения в следующую нормальную форму осуществляется методом «декомпозиции без потерь», т. е. запросы к исходному отношению и к отношениям, получаемым в результате декомпозиции, дадут одинаковый результат. Основной операцией метода является операция проекции.

Первая нормальная форма. Отношение находится в 1НФ, если все его атрибуты являются простыми (имеют единственное значение). Исходное отношение строится таким образом, чтобы оно было в 1НФ.

Наиболее важные на практике нормальные формы отношений основываются на фундаментальном в теории реляционных баз данных понятии функциональной зависимости. Для дальнейшего изложения потребуются несколько определений.

Определение 1. *Функциональная зависимость.* В отношении R атрибут Y функционально зависит от атрибута X (X и Y могут быть составными) в том, и только в том случае, если каждому значению X соответствует в точности одно значение Y: $R.X \rightarrow R.Y$.

Определение 2. *Полная функциональная зависимость*. Функциональная зависимость R.X → R.Y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X.

Определение 3. *Транзитивная функциональная зависимость.* Функциональная зависимость R.X → R.Y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости R.X → R.Z и R.Z → R.Y и отсутствует функциональная зависимость R.Z → R.X. При отсутствии последнего требования мы имели бы «неинтересные» транзитивные зависимости в любом отношении, обладающем несколькими ключами.

Определение 4. *Неключевой атрибут.* Неключевым атрибутом называется любой атрибут отношения, не входящий в состав первичного ключа.

Определение 5. Взаимно независимые атрибуты. Два или более атрибута взаимно независимы, если ни один из этих атрибутов не является функционально зависимым от других.

Вторая нормальная форма. Отношение находится в 2НФ, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от первичного ключа (составного). Для устранения частичной зависимости и перевода отношения в 2НФ необходимо, используя операцию проекции, разложить его на несколько отношений следующим образом: построить проекцию без атрибутов, находящихся в частичной функциональной зависимости от первичного ключа; построить проекции на части составного первичного ключа и атрибуты, зависящие от этих частей.

Третья нормальная форма.

Определение 1. Отношение находится в 3НФ, если оно находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Определение 2. Отношение находится в 3НФ в том, и только в том случае, если все неключевые атрибуты отношения взаимно независимы и полностью зависят от первичного ключа.

На практике 3НФ схем отношений в большинстве случаев является достаточным, и приведением к ним процесс проектирования реляционной БД заканчивается. Если в отношении имеется зависимость атрибутов составного ключа, то необходимо перейти к усиленной 3НФ.

Усиленная 3НФ, или нормальная форма Бойса – Кода (БКНФ). Отношение находится в БКНФ, если оно находится в 3НФ и в нем отсутствуют зависимости ключей (атрибутов составного ключа) от неключевых атрибутов.

Четвертая нормальная форма (4НФ) требует отсутствия многозначных зависимостей между атрибутами.

Для приведения сущности к *четвертой нормальной форме* следует создать новую сущность и перенести атрибуты с многозначной зависимостью в разные сущности. Связь между новыми сущностями при этом устанавливать нельзя, поскольку в результате миграции атрибутов внешних ключей атрибуты с многозначной зависимостью вновь окажутся в одной сущности. Ссылочную целостность в этом случае следует поддерживать при помощи триггеров.

1.8 Целостность данных

Выполнение операторов модификации данных в таблицах базы данных может привести к нарушению целостности данных и их корректности, т. е. к потере их достоверности и непротиворечивости.

Чтобы информация, хранящаяся в базе данных, была однозначной и непротиворечивой, в реляционной модели устанавливаются некоторые ограничительные условия – правила, определяющие возможные значения данных и обеспечивающие логическую основу для поддержания корректных значений. Ограничения целостности позволяют свести к минимуму ошибки, возникающие при обновлении и обработке данных.

В базе данных, построенной на реляционной модели, задается ряд правил целостности, которые, по сути, являются ограничениями для всех допустимых состояний базы данных и гарантируют корректность данных. Рассмотрим следующие типы ограничений целостности данных: обязательные данные; ограничения для доменов полей; корпоративные ограничения; целостность сущностей; ссылочная целостность.

Обязательные данные.

Некоторые поля всегда должны содержать одно из допустимых значений, другими словами, эти поля не могут иметь пустого значения.

Ограничения для доменов полей.

Каждое поле имеет свой домен, представляющий собой набор его допустимых значений.

Корпоративные ограничения целостности.

Существует понятие «корпоративные ограничения целостности» как дополнительные правила поддержки целостности данных, определяемые пользователями, принятые на предприятии или администраторами баз данных. Ограничения предприятия называются бизнес-правилами.

Целостность сущностей.

Это ограничение целостности касается первичных ключей базовых таблиц. По определению, ***первичный ключ*** – минимальный идентификатор (одно или несколько полей), который используется для уникальной идентификации записей в таблице. Таким образом, никакое подмножество первичного ключа не может быть достаточным для уникальной идентификации записей.

Целостность сущностей определяет, что в базовой таблице ни одно поле первичного ключа не может содержать отсутствующих значений, обозначенных NULL.

Если допустить присутствие определителя NULL в любой части первичного ключа, то это равносильно утверждению, что не все его поля необходимы для уникальной идентификации записей, и противоречит определению первичного ключа.

Ссылочная целостность.

Указанное ограничение целостности касается внешних ключей. ***Внешний ключ*** – это поле (или множество полей) одной таблицы, являющееся ключом другой (или той же самой) таблицы. Внешние ключи используются для

установления логических связей между таблицами. Связь устанавливается путем присвоения значений внешнего ключа одной таблицы значениям ключа другой.

Между двумя или более таблицами базы данных могут существовать отношения подчиненности, которые определяют, что для каждой записи главной таблицы (называемой еще ***родительской***) может существовать одна или несколько записей в подчиненной таблице (называемой также ***дочерней***).

Часто связь между таблицами устанавливается по первичному ключу, т. е. значениям внешнего ключа одной таблицы присваиваются значения первичного ключа другой. Однако это не является обязательным – в общем случае связь может устанавливаться и с помощью вторичных ключей. Кроме того, при установлении связей между таблицами не требуется непременная уникальность ключа, обеспечивающего установление связи. Поля внешнего ключа не обязаны иметь те же имена, что и имена ключей, которым они соответствуют. Внешний ключ может ссылаться на свою собственную таблицу – в таком случае внешний ключ называется рекурсивным.

Ссылочная целостность определяет: если в таблице существует внешний ключ, то его значение должно либо соответствовать значению первичного ключа некоторой записи в базовой таблице, либо задаваться определителем NULL.

Существует несколько важных моментов, связанных с внешними ключами. Во-первых, следует проанализировать, допустимо ли использование во внешних ключах пустых значений. В общем случае, если участие дочерней таблицы в связи является обязательным, то рекомендуется запрещать применение пустых значений в соответствующем внешнем ключе. В то же время если имеет место частичное участие дочерней таблицы в связи, то помещение пустых значений в поле внешнего ключа должно быть разрешено.

Следующая проблема связана с организацией поддержки ссылочной целостности при выполнении операций модификации данных в базе. Здесь возможны следующие ситуации.

1 Вставка новой строки в дочернюю таблицу. Для обеспечения ссылочной целостности необходимо убедиться, что значение внешнего ключа новой строки дочерней таблицы равно пустому значению либо некоторому конкретному значению, присутствующему в поле первичного ключа одной из строк родительской таблицы.

2 Удаление строки из дочерней таблицы. Никаких нарушений ссылочной целостности не происходит.

3 Обновление внешнего ключа в строке дочерней таблицы. Этот случай подобен описанной выше первой ситуации. Для сохранения ссылочной целостности необходимо убедиться, что значение внешнего ключа в обновленной строке дочерней таблицы равно пустому значению либо некоторому конкретному значению, присутствующему в поле первичного ключа одной из строк родительской таблицы.

4 Вставка строки в родительскую таблицу. Такая вставка не может вызвать нарушения ссылочной целостности. Добавленная строка просто становится родительским объектом, не имеющим дочерних объектов.

5 Удаление строки из родительской таблицы. Ссылочная целостность окажется нарушенной, если в дочерней таблице будут существовать строки, ссылающиеся на удаленную строку родительской таблицы. В этом случае может использоваться одна из следующих стратегий:

– NO ACTION. Удаление строки из родительской таблицы запрещается, если в дочерней таблице существует хотя бы одна ссылающаяся на нее строка;

– CASCADE. При удалении строки из родительской таблицы автоматически удаляются все ссылающиеся на нее строки дочерней таблицы. Если любая из удаляемых строк дочерней таблицы выступает в качестве родительской стороны в какой-либо другой связи, то операция удаления применяется ко всем строкам дочерней таблицы этой связи и т. д. Другими словами, удаление строки родительской таблицы автоматически распространяется на любые дочерние таблицы;

– SET NULL. При удалении строки из родительской таблицы во всех ссылающихся на нее строках дочернего отношения в поле внешнего ключа, соответствующего первичному ключу удаленной строки, записывается пустое значение. Следовательно, удаление строк из родительской таблицы вызовет занесение пустого значения в соответствующее поле дочерней таблицы. Эта стратегия может использоваться, только когда в поле внешнего ключа дочерней таблицы разрешается помещать пустые значения;

– SET DEFAULT. При удалении строки из родительской таблицы в поле внешнего ключа всех ссылающихся на нее строк дочерней таблицы автоматически помещается значение, указанное для этого поля как значение по умолчанию. Таким образом, удаление строки из родительской таблицы вызывает помещение принимаемого по умолчанию значения в поле внешнего ключа всех строк дочерней таблицы, ссылающихся на удаленную строку. Эта стратегия применима лишь в тех случаях, когда полю внешнего ключа дочерней таблицы назначено некоторое значение, принимаемое по умолчанию;

– NO CHECK. При удалении строки из родительской таблицы никаких действий по сохранению ссылочной целостности данных не предпринимается.

6 Обновление первичного ключа в строке родительской таблицы. Если значение первичного ключа некоторой строки родительской таблицы будет обновлено, нарушение ссылочной целостности случится при том условии, что в дочернем отношении существуют строки, ссылающиеся на исходное значение первичного ключа. Для сохранения ссылочной целостности может применяться любая из описанных выше стратегий. При использовании стратегии CASCADE обновление значения первичного ключа в строке родительской таблицы будет отображено в любой строке дочерней таблицы, ссылающейся на данную строку.

Существует и другой вид целостности – смысловая (семантическая) целостность базы данных. Требование **смысловой целостности** определяет, что данные в базе данных должны изменяться таким образом, чтобы не нарушалась сложившаяся между ними смысловая связь.

Уровень поддержания целостности данных в разных системах существенно варьируется.

Идеология архитектуры клиент-сервер требует переноса максимально возможного числа правил целостности данных на сервер. К преимуществам такого подхода относятся:

– гарантия целостности базы данных, поскольку все правила сосредоточены в одном месте (в базе данных);

– автоматическое применение определенных на сервере ограничений целостности для любых приложений;

– отсутствие различных реализаций ограничений в разных клиентских приложениях, работающих с базой данных;

– быстрое срабатывание ограничений, поскольку они реализованы на сервере и, следовательно, нет необходимости посылать данные клиенту, увеличивая при этом сетевой трафик;

– доступность внесенных в ограничения на сервере изменений для всех клиентских приложений, работающих с базой данных, и отсутствие необходимости повторного распространения измененных приложений клиентов среди пользователей.

К недостаткам хранения ограничений целостности на сервере можно отнести:

– отсутствие у клиентского приложения возможности реагировать на некоторые ошибочные ситуации, возникающие на сервере при реализации тех или иных правил (например, ошибок при выполнении хранимых процедур на сервере);

– ограниченность возможностей языка SQL и языка хранимых процедур и триггеров для реализации всех возникающих потребностей определения целостности данных.

2 Физическое проектирование

2.1 Проектирование реляционных баз данных

Модельные представления, основанные на анализе семантики данных, иногда называют семантическими моделями. Одним из распространенных средств спецификации модельных представлений этого типа является так называемая «модель сущность – связь» (Entity – Relationship Model). Модель «сущность – связь» (ER-модель) была предложена П. Ченом в 1976 г. как средство «ручного» проектирования баз данных. Модель «сущность – связь» представляет собой набор концепций, используемых для описания логической структуры базы данных. Базис понятий семантического моделирования в общем случае включает: определение объекта-экземпляра; определение объекта-типа; определение связи между объектами; определение свойства объекта; определение идентифицирующего свойства объекта. Для модели «сущность – связь» базовыми являются понятия «сущность», «связь», «атрибут». Модель «сущность – связь» основана на диаграммной технике. Для представления

различных аспектов структуры данных (объектов, свойств объектов, связей между объектами, свойств связей и других) используются графические средства. За многолетнюю историю использования модель «сущность – связь» претерпела различные модификации, известны несколько ее нотаций. Наиболее популярной в настоящее время является нотация, в большей степени приближенная к реальному процессу проектирования баз данных, так называемая методология информационного моделирования IDEF1X. В ER-модели абстрактным объектам реальной действительности соответствует понятие «сущность». Сущность – это абстрактный объект, который в конкретном контексте имеет независимое существование. Различают понятия «тип сущности» и «экземпляр сущности». Тип сущности (в дальнейшем – просто сущность) представляет собой объект-тип, результат абстракции обобщения множества однородных объектов-экземпляров реальной действительности с одинаковыми свойствами. Сущность имеет семантически значимое имя, как правило, имя существительное. Экземпляры сущностей уникальны, в природе не бывает двух одинаковых объектов. Следовательно, экземпляр сущности может быть идентифицирован уникальным образом. Связь – это ассоциация сущностей или отношение между сущностями. Различают понятия «тип связи» и «экземпляр связи». Тип связи (в дальнейшем – просто связь) представляет собой отношение между типами сущностей. Связь имеет семантически значимое имя, как правило, в форме глагола. Связи обладают свойствами: вид (категориальная, идентифицирующая, неидентифицирующая); степень (унарная, бинарная, тернарная, N-арная); кардинальность; внешний ключ сущности. Атрибут – это свойство сущности или связи. Атрибутам как спецификаторам свойств сущностей или связей присваиваются семантически значимые имена, как правило, в форме существительного. С понятием атрибута связаны понятия «домен атрибута», «зависимости между атрибутами», «потенциальный ключ сущности», «детерминант» и др.

CASE-средства (Computer – Aided Software Engineering) – это методы и технологии, которые позволяют проектировать различные информационные системы (в частности, базы данных) и автоматизировать их создание.

К ключевым понятиям проектирования баз данных относятся:

– **CASE-технологии** – программная основа CASE-средств, применяемая для разработки и поддержки процессов жизненных циклов ПО, используемых в моделировании данных и генерации схем баз данных. Чаще всего программные коды в CASE-технологиях пишутся на языке SQL;

– **концептуальное проектирование** – построение обобщенной, не имеющей конкретики, модели базы данных с описанием ее объектов и связей между ними;

– **логическое проектирование** – создание схемы базы данных с учетом специфики конкретной модели данных (но не конкретной СУБД). Например, для реляционной модели данных логическая схема БД будет содержать определенный набор таблиц и связей между ними;

– **физическое проектирование** – построение схемы базы данных под конкретную СУБД. При таком проектировании учитываются ограничения на

именование объектов базы данных, ограничения на определенные типы данных, физические условия хранения данных в БД (разделение по файлам и устройствам), возможность доступа к БД.

При проектировании баз данных с помощью CASE-средств выделяются и анализируются выбранные бизнес-процессы, для которых создается БД, определяются взаимосвязи их элементов, оптимизируется их инфраструктура. CASE-средства позволяют существенно сократить время на разработку БД и уменьшить количество ошибок в них.

Для создания баз данных под наиболее распространенные СУБД чаще всего используются следующие CASE-средства:

– **ERwin (Logic Works)** – CASE-инструмент для создания концептуальных и логических схем баз данных. Он позволяет редактировать различные наборы данных, представляя их в виде электронных таблиц, разрабатывать структуры баз данных, синхронизировать модели, скрипты и БД, настраивать шаблоны, выводить рабочую информацию в виде отчетов, строить удобные и понятные диаграммы, отображающие различные процессы в системе и взаимосвязи между ними;

– **S-Designor (SDP)** – графический CASE-инструмент для проектирования структуры реляционных БД. Он создает модели баз данных в два этапа – выстраивая концептуальную модель и затем преобразуя ее в физическую, причем в данном процессе разработки возможен как прямой, так и обратный переход между моделями. Данный инструмент позволяет проектировать базы данных под различные СУБД, в том числе под Oracle и MySQL;

– **DataBase Designer (ORACLE)** – интегрированная CASE-среда, которая позволяет анализировать предметную область создания БД, выполнять программирование и проектирование, проводить оценку и тестирование, осуществлять сопровождение, обеспечивать качество, управлять конфигурацией и проектом, разрабатывать и анализировать требования к информационной системе.

В зависимости от того, на каком этапе проектирования баз данных используются CASE-средства, их относят к:

– **CASE-средствам верхнего уровня.** Их задействуют на начальных этапах проектирования, когда требуется выполнить анализ поставленной задачи, поставить цели и определить приоритеты, представить необходимую информацию в виде диаграмм и деревьев решений;

– **CASE-средствам нижнего уровня.** С помощью этих средств выполняются заключительные этапы проектирования БД, проводятся собственно проектирование, написание кода, тестирование и внедрение программного обеспечения поддержки информационных систем.

– **интегрированным CASE-средствам.** Дают возможность выполнять все этапы проектирования БД благодаря наличию функций верхнего и нижнего уровней.

2.2 Физическая организация баз данных. Системы управления базами данных

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Исторически первыми системами хранения и доступа были файловые структуры и системы управления файлами (СУФ), которые фактически являлись частью операционных систем. СУБД создавала над этими файловыми моделями свою надстройку, которая позволяла организовать всю совокупность файлов таким образом, чтобы она работала как единое целое и получала централизованное управление от СУБД. Но непосредственный доступ осуществлялся на уровне файловых команд, которые СУБД использовала при манипулировании всеми файлами, составляющими хранимые данные одной или нескольких баз данных.

Однако механизмы буферизации и управления файловыми структурами не приспособлены для решения задач собственно СУБД, эти механизмы разрабатывались просто для традиционной обработки файлов, и с ростом объемов хранимых данных они стали неэффективными для использования СУБД. Тогда постепенно произошел переход от базовых файловых структур к непосредственному управлению размещением данных на внешних носителях самой СУБД. И пространство внешней памяти уже выходило из-под владения СУФ и управлялось непосредственно СУБД. При этом механизмы, применяемые в файловых системах, перешли во многом и в новые системы организации данных во внешней памяти, называемые чаще страничными системами хранения информации. Поэтому раздел, посвященный физическим моделям данных, начинается с обзора файлов и файловых структур, используемых для организации физических моделей, применяемых в базах данных, а заканчивается ознакомлением с механизмами организации данных во внешней памяти, использующими страничный принцип организации.

Файловые структуры, используемые для хранения информации в базах данных.

В каждой СУБД по-разному организованы хранение и доступ к данным, однако существуют некоторые файловые структуры, которые имеют общепринятые способы организации и широко применяются практически во всех СУБД.

2.3 Назначение и функции СУБД

СУБД выполняют целый ряд функций, которые для конечного пользователя незаметны. Основными из них являются:

- управление словарем данных;
- управление хранением, преобразованием и представлением данных;
- обеспечение безопасности данных;
- обеспечение целостности данных СУБД;

– управление многопользовательским доступом к данным;

– управление резервным копированием и восстановлением данных;

– наличие механизмов ранжирования (репликации – обмен информации между БД, расположенными на различных серверах) данных;

– наличие возможности экспорта и импорта данных;

– наличие языков доступа к данным и интерфейсов прикладного программирования;

– наличие интерфейсов взаимодействия с БД;

– управление многопользовательским доступом к данным.

В СУБД имеют специальные средства, которые обеспечивают целостность данных, их непротиворечивость. Обеспечение целостности данных основано на анализе и контроле их связей, описание которых хранится в словаре данных, а также на использовании механизма транзакции.

Транзакция – последовательность операций над БД, рассматриваемых СУБД как единое целое. Если все операции, составляющие транзакцию, успешно выполняются, то СУБД фиксирует изменения в БД, вызванные этой транзакцией. Если хотя бы одна из операций не будет выполнена, то БД останется в том же состоянии, что и перед выполнением транзакции (все операции отменяются и выполняется так называемый откат транзакции). Все операции над данными выполняются в рамках транзакции, каждая из которых переводит БД из одного целостного состояния в другое.

СУБД создают сложные структуры, позволяющие работать с данными одновременно нескольким пользователям без нарушения целостности и непротиворечивости данных. В основе этого – механизмы транзакции и блокировок. Механизм блокировок заключается в том, что если несколько пользователей хотят изменить одни и те же данные, то только транзакции одного из них разрешается доступ к общим ресурсам. Остальные транзакции ожидают завершения ее работы и разблокирования общего ресурса, затем запускается другая транзакция и т. д. Транзакция и блокировки тесно связаны друг с другом. Блокировка является механизмом, обеспечивающим независимость транзакций друг от друга.

Использование блокировок существенно замедляет процесс обработки транзакций. Путем уменьшения объема ресурса можно уменьшить задержки, которые образуются из-за занятости данных, обрабатываемых транзакцией. Существуют различные уровни блокировки: на уровне БД (самый неприемлемый вариант), на уровне таблицы, части таблицы (страницы), строки таблицы. Очевидно, что блокировка строк обеспечивает максимальную производительность.

2.4 Администрирование баз данных. Безопасность данных

Оптимизация работы базы данных является весьма непростой задачей и включает в себя решение целого комплекса взаимосвязанных проблем. Это обеспечение приемлемого быстродействия и функциональности базы данных, удобства работы пользователей, оптимизация потребляемых ресурсов, например, по критерию минимизации затрат памяти и максимизации использования сети и др. Важнейшим аспектом оптимизации является повышение производительности БД.

Для повышения производительности базы данных можно использовать общие методы повышения быстродействия программ, такие как увеличение мощности аппаратных средств, конфигурирование операционной системы, оптимизация структуры внешних носителей и размещения базы данных на них и др. Кроме того, используются специальные средства оптимизации работы базы данных, встроенные в СУБД. В частности, большинство современных реляционных СУБД имеют в своем составе специальный компонент – **оптимизатор запросов**, позволяющий максимально быстро и эффективно обрабатывать запросы выбора и запросы манипулирования данными.

Распространенный способ оптимизации работы базы данных – это **сжатие базы данных.** Оно обеспечивает оптимизацию размещения объектов базы данных на внешних носителях и возвращение освободившегося дискового пространства для дальнейшего использования

Наиболее распространена технология сжатия на основе различий, когда некоторое значение заменяется сведениями об его отличиях от предыдущего значения.

Другой тип технологии сжатия основан на иерархическом сжатии данных.

Существует технология сжатия, основанная на кодировании Хаффмана. Суть в кодировании отдельных символов битовыми строками разной длины.

Для ускоренного доступа к данным базы по запросам пользователей используются индексирование и хеширование.

Индекс – средство ускорений операций поиска в таблице БД, а также других операций, требующих поиска: извлечения, корректировки, сортировки. Цель использования индексирования – ускорение извлечения данных за счет уменьшения количества дисковых операций ввода-вывода.

Другим распространенным способом упорядочения записей и доступа к данным является **хеширование –** технология быстрого прямого доступа к записи БД на основе заданного значения некоторого поля записи, как правило, ключевого.

2.5 Язык структурированных запросов SQL

Structured Query Language (SQL) представляет собой непроцедурный язык, используемый для управления данными реляционных СУБД. Термин «непроцедурный» означает, что на данном языке можно сформулировать, что нужно сделать с данными, но нельзя проинструктировать, как именно это следует сделать. Иными словами, в этом языке отсутствуют алгоритмические конструкции, такие как метки, операторы цикла, условные переходы и др.

Язык SQL был создан в начале 70-х гг. XX в. в результате исследовательского проекта IBM, целью которого было создание языка манипуляции реляционными данными. Первоначально он назывался SEQUEL (Structured English Query Language), затем SEQUEL/2, а затем SQL. Официальный стандарт SQL был опубликован ANSI (American National Standards Institute – Национальный институт стандартизации, США) в 1986 г. (это наиболее часто используемая ныне реализация SQL). Данный стандарт был расширен в 1989 и 1992 гг., поэтому последний стандарт SQL носит название SQL92. В настоящее время ведется работа над стандартом SQL3, содержащим некоторые объектно-ориентированные расширения.

Существует три уровня соответствия стандарту ANSI: начальный, промежуточный и полный. Многие производители серверных СУБД, такие как IBM, Informix, Microsoft, Oracle и Sybase, применяют собственные реализации SQL, основанные на стандарте ANSI (отвечающие как минимум начальному уровню соответствия стандарту) и содержащие некоторые расширения, специфические для данной СУБД.

Более подробную информацию о соответствии стандарту версии SQL, используемой в конкретной СУБД, можно найти в документации, поставляемой с этой СУБД.

Инструкция SQL состоит из нескольких частей, называемых предложениями. Каждое предложение в инструкции SQL имеет свое назначение. Некоторые предложения являются обязательными. В данном пособии не рассматривается синтаксис команд SQL.

3 Распределенные базы данных и хранилища данных

3.1 Распределенные базы данных. Тиражирование данных

Распределенная база данных (DDB – distributed database) – это совокупность логически взаимосвязанных баз данных, распределенных в компьютерной сети. Распределенная система управления базой данных определяется как программная система, которая позволяет управлять распределенной базой данных таким образом, чтобы ее распределенность была прозрачна для пользователей. В этом определении следует уточнить две отличительные архитектурные особенности. Первая из них заключается в том, что система состоит из множества (возможно, пустого) узлов приема запросов (query site)

и непустого множества узлов данных (data site). Узлы данных обладают средствами для хранения данных, а узлы приема запросов – нет. В узлах приема запросов лишь выполняются программы, реализующие пользовательский интерфейс для доступа к данным, хранящимся в узлах данных. Вторая особенность состоит в том, что узлы логически представляют собой независимые компьютеры. Следовательно, у такого узла имеется собственная основная и внешняя память, установлена собственная операционная система (может быть, одна и та же на всех узлах, а возможно, и нет) и имеется возможность выполнять приложения. Узлы связаны компьютерной сетью, а не входят в мультипроцессорную конфигурацию. Важно подчеркнуть слабую связанность процессоров, которые обладают собственными операционными системами и функционирует независимо.

База данных физически распределяется по узлам данных на основе **фрагментации** и **репликации** данных. При наличии схемы реляционной базы данных каждое отношение фрагментируется на горизонтальные или вертикальные разделы. Горизонтальная фрагментация реализуется при помощи операции селекции, которая направляет каждый кортеж отношения в один из разделов, руководствуясь предикатом фрагментации. За счет фрагментации данные приближаются к месту их наиболее интенсивного использования, что потенциально снижает затраты на пересылки; уменьшаются также размеры отношений, участвующих в пользовательских запросах.

Фрагменты данных могут также реплицироваться на основе характера доступа к ним. Это полезно, если доступ к одним и тем же данным производится из приложений, выполняющихся на разных узлах. В таком случае с точки зрения экономии затрат более эффективно дублировать данные в ряде узлов, чем непрерывно пересылать данные между узлами.

При ослаблении отличительных особенностей распределенной СУБД получается параллельная система баз данных. Не существует четкого разграничения между параллельными и распределенными СУБД.

Параллельную СУБД можно определить как СУБД, реализованную на мультипроцессорном компьютере. Такое определение подразумевает наличие множества альтернатив, спектр которых варьируется от непосредственного переноса существующих СУБД с переработкой лишь интерфейса к операционной системе до изощренных комбинаций алгоритмов параллельной обработки и функций баз данных, приводящих к новым аппаратно-программным архитектурам. Как и всегда, приходится выбирать между переносимостью (на несколько платформ) и эффективностью. Изощренные подходы направлены, главным образом, на более полное использование преимуществ конкретного мультипроцессора в ущерб переносимости.

Решение, таким образом, заключается в применении широкомасштабного параллелизма, чтобы усилить мощность отдельных компонентов путем их интеграции в целостную систему на основе соответствующего программного обеспечения параллельных баз данных. Важное значение имеет применение стандартных аппаратных компонентов для того, чтобы иметь возможность с минимальным отставанием использовать результаты постоянных техноло-

гических усовершенствований. В программном обеспечении базы данных могут быть предусмотрены три вида параллелизма, присущие приложениям интенсивной обработки данных. **Межзапросный параллелизм** предполагает одновременное выполнение множества запросов, относящихся к разным транзакциям. Под **внутризапросным параллелизмом** понимается одновременное выполнение сразу нескольких операций (например, операций выборки), относящихся к одному и тому же запросу. И внутризапросный, и межзапросный параллелизм реализуется на основе разделения данных, аналогичного горизонтальному фрагментированию. Наконец, понятие **внутриоперационного параллелизма** означает параллельное выполнение одной операции в виде набора субопераций с применением в дополнение к фрагментации данных также и фрагментации функций. Языки баз данных, ориентированные на операции над множествами, обеспечивают много возможностей для использования внутриоперационного параллелизма.

Характерные черты параллельных и распределенных СУБД.

1 Распределенная/параллельная база данных – это именно база данных, а не «коллекция» файлов, индивидуально хранимых на разных узлах сети. В этом заключается разница между DDB и распределенной файловой системой. Распределенные данные представляют собой DDB, только если они связаны в соответствии с некоторым структурным формализмом (таким как реляционная модель), а для доступа к ним имеется единый высокоуровневый интерфейс.

2 Система обладает полной функциональностью СУБД. Она не сводится по своим возможностям ни к распределенным файловым системам, ни к системам обработки транзакций. Обработка транзакций – только одна из функций, предоставляемых подобными системами. Наряду с этим, они должны также обеспечивать функции запросов и структурной организации данных, которые необязательно поддерживаются системами обработки транзакций.

3 Распределение (включая фрагментацию и репликацию) данных по множеству узлов невидимо для пользователей. Это свойство называется прозрачностью. Технология распределенных/параллельных баз данных распространяет основополагающую для управления базами данных концепцию независимости данных на среду, где данные распределены и реплицированы по множеству компьютеров, связанных сетью. Это обеспечивается за счет нескольких видов прозрачности: прозрачность сети (следовательно, прозрачность распределения), прозрачность репликации и прозрачность фрагментации. Прозрачность доступа означает, что пользователи имеют дело с единым логическим образом базы данных и осуществляют доступ к распределенным данным точно так же, как если бы они хранились централизованно. В идеале полная прозрачность подразумевает наличие языка запросов к распределенной СУБД, не отличающегося от языка для централизованной СУБД.

Вопросы прозрачности более критичны для распределенных, чем для параллельных СУБД. Для этого есть две причины. Во-первых, многопроцессорные системы, для которых реализуются параллельные СУБД, функционируют под управлением единой операционной системы. Операционная система может быть организована таким образом, чтобы брать на себя некоторые аспекты

функциональности СУБД, предоставляя тем самым определенную степень прозрачности. Во-вторых, разработки программного обеспечения на параллельных системах поддерживаются языками параллельного программирования, также обеспечивающими некоторую степень прозрачности.

В распределенных СУБД данные и приложения, которые осуществляют доступ к ним, могут быть локализованы на одном и том же узле, благодаря чему исключается (или сокращается) потребность в удаленном доступе к данным, характерная для систем телеобработки данных в режиме разделения времени. Далее, поскольку на каждом узле выполняется меньше приложений и хранится меньшая порция базы данных, можно сократить также конкуренцию при доступе к данным и ресурсам. Наконец, параллелизм, внутренне присущий распределенным системам, открывает возможности для реализации межзапросного и внутризапросного параллелизма.

Если доступ пользователей к базе данных заключается только в выполнении запросов (то есть имеет место доступ только по чтению), то реализация межзапросного и внутризапросного параллелизма подразумевает реплицирование по возможности максимальной части базы данных. Но, поскольку на практике доступ к базе данных осуществляется не только по чтению, для реализации перемежающихся операций чтения и модификации данных необходима поддержка распределенных транзакций.

Высокая производительность – одна из важнейших целей, на достижение которой направлены технологии параллельных СУБД. Как правило, она обеспечивается за счет сочетания нескольких взаимно дополняющих решений, таких как применение операционных систем, ориентированных на поддержку баз данных, параллелизм, оптимизация, балансировка нагрузки. Наличие операционной системы, «осведомленной» о специфических потребностях баз данных (например, относительно управления буферами), упрощает реализацию функций баз данных нижнего уровня и способствует снижению их стоимости. Так, затраты на передачу сообщения могут быть значительно снижены (до нескольких сот инструкций) за счет применения специализированного коммуникационного протокола. Механизмы распараллеливания способствуют повышению общей пропускной способности системы (межзапросный параллелизм), снижению времени отклика для отдельных транзакций (внутризапросный и внутриоперационный параллелизм).

Технологии распределенных и параллельных СУБД направлены также на повышение надежности, поскольку благодаря репликации данных исключаются одиночные точки отказа. Отказ одного узла или сбой на линии связи не приводит к выходу из строя всей системы. Даже если часть данных становится недоступной, при правильной организации системы пользователи могут иметь доступ к остальной части информации. Под «правильной организацией» понимается поддержка распределенных транзакций и протоколов обеспечения надежности (то есть протоколов фиксации и восстановления).

В среде параллельных и распределенных СУБД упрощается решение вопросов, связанных с возрастанием объема баз данных или потребностей обработки. При этом редко возникает необходимость в серьезной перестройке

системы; расширение возможностей обычно достигается за счет добавления процессорных мощностей или памяти.

В идеале параллельная (и, в меньшей степени, распределенная) СУБД обладает свойством **линейной масштабируемости (linear scaleup)** и **линейного ускорения (linear speedup)**. Под линейной масштабируемостью понимается сохранение того же уровня производительности при увеличении размера базы данных и одновременном пропорциональном увеличении процессорной мощности и объема памяти. Линейное ускорение означает, что с наращиванием процессорной мощности и объема памяти при сохранении прежнего размера базы данных пропорционально возрастает производительность. Более того, при расширении системы должна потребоваться лишь минимальная реорганизация существующей базы данных.

Технологии распределенных и параллельных баз данных.

Распределенные и параллельные СУБД предоставляют ту же функциональность, что и централизованные СУБД, если не считать того, что они работают в среде, где данные распределены по узлам компьютерной сети или многопроцессорной системы. Как уже упоминалось, пользователи могут вообще ничего не знать о распределении данных. Таким образом, эти системы обеспечивают пользователям логически интегрированное представление физически распределенной базы данных. Поддержка подобного представления – источник ряда сложных проблем, которые должны решаться системными функциями.

3.2 Хранилища данных

Проблемы разрозненности хранения данных в рамках одного предприятия, необходимость привлечения технических специалистов для извлечения из баз данных нужной для принятия решений информации привели в 1980-е гг. к идее централизованного хранения данных, необходимых для последующего анализа. Возник термин «хранилище данных».

Хранилища данных представляют собой специализированные базы данных, обладающие следующими свойствами:

– *предметная ориентированность.* В хранилище содержатся данные, всесторонне описывающие определенную предметную область;

– *интегрированность.* Данные собираются из множества различных источников, обобщаются и хранятся в едином корпоративном хранилище;

– *обеспечение непротиворечивости данных.* Данные из разных источников могут содержать дублирующие, противоречивые сведения, поэтому перед их загрузкой в хранилище они проходят процедуры проверки, согласования, дополнения, обобщения;

– *неизменяемость.* В отличие от баз данных транзакционных систем, в которых оперативные данные могут редактироваться пользователями, данные в хранилище используются исключительно в режиме чтения и недоступны для корректировки;

– *поддержка хронологии.* Поскольку для целей анализа и прогнозирования развития предметной области необходимо видеть ее показатели в динамике, данные хранятся в привязке в дате и за максимально возможный временной период;

– *оптимизация под выполнение сложных аналитических запросов.* Хранилище проектируется таким образом, чтобы минимизировать время на формирование аналитической отчетности, необходимой для поддержки принятия решений для руководителей и менеджеров.

Если в базах данных транзакционных систем данные поступают в процессе бизнес-деятельности (продажи товаров фиксируются в системе по факту продажи, товары, поступившие на склад, учитываются по факту поступления на склад и т. п.), то для пополнения данных в хранилище требуется их периодическая выгрузка из источников. Процесс размещения информации в хранилищах предусматривает периодический сбор, очистку и интеграцию разрозненных данных с последующим их преобразованием в статичные, постоянные структуры.

В качестве источников данных для информационного хранилища, как правило, используются данные из разрозненных ИС, основанных на различных реляционных СУБД, обслуживающие повседневную деятельность предприятия. Источниками могут быть и данные, получаемые от внешних организаций – информационных агентств, консалтинговых компаний, средств массовой информации, сайтов интернета.

В зависимости от степени детализации и времени хранения в хранилище выделяются текущие детальные данные, архивные данные, агрегированные (суммарные, обобщенные) данные, метаданные (репозиторий).

В отличие от баз данных транзакционных систем, где агрегированные данные не хранятся, а каждый раз вычисляются заново, хранилище содержит и детальные, и агрегированные данные. Это обусловлено необходимостью обеспечения быстрого выполнения запросов пользователей: в хранилище содержится такое большое количество данных, что вычисление суммарных показателей «на лету» занимало бы значительное количество времени.

В хранилище содержится информация из различных источников, которая может иметь различную периодичность обновления, различную структуру, степень достоверности, владельцев данных – сведения об этих характеристиках информации называются метаданными и хранятся в репозитории хранилища. В репозитории могут также храниться бизнес-термины, правила и алгоритмы вычисления показателей, которые определены для рассматриваемого бизнеса. Физически репозиторий представляет собой отдельную базу данных или набор таблиц в рамках базы данных хранилища.

Хранилище может быть реализовано в виде виртуального хранилища данных, витрин данных и глобального хранилища данных.

Под виртуальным хранилищем данных понимают специальные средства доступа к данным транзакционных систем, обеспечивающие работу с этими данными как с хранилищем данных. Этими средствами доступа могут быть как «представления» в базе данных, так и отдельные программные продукты.

Достоинствами виртуального хранилища являются простота и малая стоимость реализации, единая платформа с источником информации, отсутствие необходимости перегрузки данных из источников информации в хранилище данных. К недостаткам такого подхода относятся проблемы производительности, трансформации данных, интеграции данных с другими источниками, отсутствие поддержки хронологии, проверки корректности данных, зависимость от доступности и структуры основной базы данных.

Реализация хранилища данных на основе витрин данных предполагает функционирование двух уровней: уровня источников данных и уровня витрин данных, которые строятся на основе принципов проектирования хранилищ данных и содержат данные о конкретной узкой предметной области. В рамках одного предприятия витрин данных может быть несколько: витрина данных но поставщикам, витрина данных по производимым товарам, витрина данных но доходам и расходам для бухгалтерии и др. Единое центральное хранилище данных при этом не создается. Достоинствами витрин данных являются простота и малая стоимость реализации по сравнению с созданием централизованного хранилища данных, высокая производительность за счет физического разделения регистрирующих и аналитических систем, выделения загрузки и трансформации данных в отдельный процесс, оптимизированный под анализ структуры хранения данных. Витрины данных также позволяют поддерживать хронологию данных, описывать структуру данных в виде метаданных. К недостатку витрин данных можно отнести то, что они не дают единого источника информации обо всем предприятии. Впоследствии интегрировать витрины в единое централизованное хранилище может оказаться проблематичным из-за различающихся форматов и структур хранения данных. Кроме того, разные витрины могут использовать частично повторяющиеся данные, которые нужно извлекать из источника для каждой витрины отдельно, что требует дополнительных затрат на обслуживание.

Глобальное хранилище данных предполагает реализацию трехуровневой архитектуры системы. На первом уровне располагаются источники данных – внутренние транзакционные системы, внешние источники (данные информационных агентств, консалтинговых компаний и т. п.). Второй уровень содержит центральное хранилище, в которое загружается информация из источников данных. При различном регламенте поступления данных из источников в качестве промежуточного звена может использоваться оперативный склад данных, в котором данные подготавливаются, преобразуются, проверяются для их последующей загрузки в центральное хранилище. Описания загруженных данных помещаются в репозиторий. Третий уровень представляет собой набор предметно-ориентированных витрин данных, источником информации для которых является центральное хранилище данных. Именно с витринами данных и работает большинство конечных пользователей.

В основе построения хранилища данных лежит принцип многомерного представления данных, при котором в структуре экономической информации выделяются измерения и факты. Под *измерениями* понимаются категориальные (дискретные) атрибуты, наименования и свойства объектов, участвующих в

бизнес-процессе, например, наименования клиентов, названия товаров, регионов, магазинов. *Факты* – это количественные значения показателей, описывающих бизнес-процесс.

В соответствии с принципом многомерного представления данных в базе данных хранилища выделяются таблицы фактов, таблицы измерений и консольные таблицы. В *таблицах фактов* содержатся количественные значения экономических показателей со ссылками на значения измерений, к которым они относятся. В *таблицах измерений* (справочников) хранятся все возможные значения измерений. *Консольные таблицы* могут использоваться для хранения более сложных измерений с вложенностью и иерархией.

В зависимости от сложности предметной области таблицы базы данных хранилища могут быть связаны по схеме «звезда», «снежинка» или «созвездие».

3.3 Многомерные базы данных. Разработка (извлечение) данных

Многомерные базы данных – разновидность реляционной модели, которая использует многомерные структуры для организации данных и выражает отношения между данными. Поддерживают неограниченное число значений в поле и находят применение там, где необходима эффективная и простая работа с большими массивами символьной информации.

Если целью является именно анализ данных, а не выполнение транзакций, используется многомерная модель данных. Технология многомерных баз данных – ключевой фактор интерактивного анализа больших массивов данных с целью поддержки принятия решения. Подобные базы данных трактуют данные как многомерные кубы, что очень удобно именно для их анализа.

Многомерные модели рассматривают данные либо как факты с соответствующими численными параметрами, либо как текстовые измерения, которые характеризуют эти факты.

Многомерные модели данных имеют три важные области применения, связанные с проблематикой анализа данных:

1) хранилища данных интегрируют для анализа информации из нескольких источников;

2) системы оперативной аналитической обработки (online analytical processing – OLAP) позволяют оперативно получить ответы на запросы, охватывающие большие объемы данных в поисках общих тенденций;

3) приложения добычи данных служат для выявления знаний за счет полуавтоматического поиска ранее неизвестных шаблонов и связей в базах данных.

Основные свойства, присущие многомерным БД.

Агрегируемость данных означает рассмотрение информации на различных уровнях ее обобщения. В информационных системах степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь, управляющий, руководитель.

Историчность данных предполагает обеспечение высокого уровня статичности собственно данных и их взаимосвязей, а также обязательность

привязки данных ко времени.

Прогнозируемость данных подразумевает задание функций прогнозирования и применение их к различным временным интервалам.

В основе многомерных хранилищ данных находится многомерная модель данных, которая опирается на концепцию **гиперкубов**. Такие гиперкубы представляют собой упорядоченные многомерные массивы.

Многомерное представление данных разделяется на две группы: **факты** – численные параметры и **измерения** – текстовые параметры для предоставления максимально возможного контекста для фактов.

В многомерной базе данных **параметры** представляют свойства факта, который пользователь хочет изучить.

Основной таблицей хранилища данных является **таблица фактов**. Обычно такие таблицы содержат сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться.

Таблицы измерений содержат практически неизменяемые данные.

Измерение – упорядоченный набор значений, принимаемых конкретным параметром и соответствующих одной из граней гиперкуба. Измерения качественно описывают исследуемый процесс, они могут быть числовыми, но в любом случае это данные дискретные, т. е. их значения принадлежат ограниченному набору.

Измерения – ключевая концепция многомерных баз данных. Многомерное моделирование предусматривает использование измерений для предоставления максимально возможного контекста для фактов.

Измерения используются для выбора и агрегирования данных на требуемом уровне детализации. Измерения организуются в иерархию, состоящую из нескольких уровней, каждый из которых представляет уровень детализации, требуемый для соответствующего анализа.

В отличие от линейных пространств, с которыми имеет дело алгебра матриц, многомерные модели, как правило, не предусматривают функций упорядочивания или расстояния для значений измерения. Единственное «упорядочивание» состоит в том, что значения более высокого уровня содержат значения более низких уровней. Однако для некоторых измерений, таких как время, упорядоченность значений размерности может использоваться для вычисления совокупной информации, такой как общий объем продаж за определенный период. Большинство моделей требуют определения иерархии измерений для формирования сбалансированных деревьев – иерархии должны иметь одинаковую высоту по всем ветвям, а каждое значение не корневого уровня – только одного родителя.

Факты (меры) – это данные, которые количественно описывают процесс, они непрерывны, т. е. могут принимать бесконечное количество значений.

Факты представляют субъект – некий шаблон или событие, которые необходимо проанализировать. В большинстве многомерных моделей данных факты однозначно определяются комбинацией значений измерений; факт существует только тогда, когда ячейка для конкретной комбинации значений не пуста.

Каждый факт обладает некоторой гранулярностью, определенной уровнями, из которых создается их комбинация значений измерений.

Хранилища данных, как правило, содержат следующие три типа фактов:

1) события (event), по крайней мере, на уровне самой большой гранулярности, как правило, моделируют события реального мира, при этом каждый факт представляет определенный экземпляр изучаемого явления. Примерами могут служить продажи, щелчки мышью на веб-странице или движение товаров на складе;

2) мгновенные снимки (snapshot) моделируют состояние объекта в данный момент времени, такие как уровни наличия товаров в магазине или на складе и число пользователей веб-сайта. Один и тот же экземпляр явления реального мира, например, конкретная банка бобов, может возникать в нескольких фактах;

3) совокупные мгновенные снимки (cumulative snapshot) содержат информацию о деятельности организации за определенный отрезок времени. Например, совокупный объем продаж за предыдущий период, включая текущий месяц, можно легко сравнить с показателями за соответствующие месяцы прошлого года.

Хранилище данных часто содержит все три типа фактов. Одни и те же исходные данные, например, движение товаров на складе, могут содержаться в трех различных типах кубов: поток товаров на складе, список товаров и поток за год к текущей дате.

Параметры – свойства факта.

Параметры состоят из двух компонентов:

1) численная характеристика факта, например, цена или доход от продаж;

2) формула, обычно простая агрегативная функция, скажем, сумма, которая может объединять несколько значений параметров в одно.

Свойство и формула выбираются таким образом, чтобы представлять осмысленную величину для всех комбинаций уровней агрегирования.

Классы параметров.

1 Аддитивные параметры могут содержательным образом комбинироваться в любом измерении. Например, имеет смысл суммировать общий объем продаж для продукта, местоположения и времени, поскольку это не вызывает наложения среди явлений реального мира, которые генерируют каждое из этих значений.

2 Полуаддитивные параметры, которые не могут комбинироваться в одном или нескольких измерениях. Например, суммирование запасов по разным товарам и складам имеет смысл, но суммирование запасов товаров в разное время бессмысленно, поскольку одно и то же физическое явление может учитываться несколько раз.

3 Неаддитивные параметры не комбинируются в любом измерении обычно потому, что выбранная формула не позволяет объединить средние значения низкого уровня в среднем значении более высокого уровня.

Аддитивные и неаддитивные параметры могут описывать факты любого рода, в то время как полуаддитивные параметры, как правило, используются с мгновенными снимками или совокупными мгновенными снимками.

Гиперкуб рассматривают как систему координат, в котором осями являются измерения (теоретически, куб может иметь любое число измерений). По осям будут откладываться значения измерений. В такой системе каждому набору значений измерений будет соответствовать ячейка, в которой можно разместить меры (числовые показатели / факты), связанные с данным набором.

Двумерное представление куба можно получить, «разрезав» его поперек одной или нескольких осей (измерений): мы фиксируем значения всех измерений, кроме двух, и получаем обычную двумерную таблицу. В горизонтальной оси таблицы (заголовки столбцов) представлено одно измерение, в вертикальной (заголовки строк) – другое, а в ячейках таблицы – значения мер. При этом набор мер фактически рассматривается как одно из измерений – мы либо выбираем для показа одну меру (и тогда можем разместить в заголовках строк и столбцов два измерения), либо показываем несколько мер (и тогда одну из осей таблицы займут названия мер, а другую – значения единственного «неразрезанного» измерения).

Двумерное представление куба возможно и тогда, когда «неразрезанными» остаются и более двух измерений. При этом на осях среза (строках и столбцах) будут размещены два или более измерений «разрезаемого» куба.

Многомерная база данных естественным образом предназначена для определенных типов запросов.

Моделирование многомерных кубов на реляционной модели данных может происходить по двум схемам: «звезда» или «снежинка».

Перечень вопросов для написания аудиторной контрольной работы

Вопросы к разделу 1

1 Основные понятия, лежащие в основе концепции баз данных. Информация и данные. Предметная область. Определение системы. Свойства системы. Определение информационной системы. Классификация информационных систем.

2 Понятие базы данных и системы баз данных. Определение базы данных и системы баз данных. Роль баз данных и их сферы применения. Схема взаимодействия пользователей с базой данных. Понятия интегрированности и разделяемости данных. Свойства независимости от данных и целостности данных.

3 Жизненный цикл базы данных. Определение жизненного цикла базы данных и его этапы. Концептуальное проектирование. Логическое проектирование. Физическое проектирование.

4 Модели данных. Сущность моделирования. Классификация моделей. Определение модели данных. Реализация модели данных. Логические модели данных: иерархическая, сетевая, реляционная, объектная, объектно-реляционная. Их достоинства и недостатки. Физические модели данных.

5 Операции реляционной алгебры. Определение и свойства сущности. Определение возможного ключа. Первичный ключ и альтернативные ключи. Синтаксис и семантика выражений реляционной алгебры. Теоретико-множественные реляционные операции объединения, пересечения, разности и декартова произведения. Специальные реляционные операции селекции, проекции, соединения, деления. Примитивные и непримитивные реляционные операции.

6 Типы связей между сущностями. Связь типа «один-к-одному». Связь типа «один-ко-многим». Внешний ключ и его свойства. Связь типа «многие-ко-многим». Преобразование связи «многие ко многим» в две связи «один-ко-многим» и связующую таблицу.

7 Нормализация данных. Понятие нормализации данных. Приведение сущности к первой нормальной форме. Приведение сущности ко второй нормальной форме. Виды аномалий в данных, устраняемые после приведения ко второй нормальной форме. Приведение сущности к третьей нормальной форме. Виды аномалий в данных, устраняемые после приведения к третьей нормальной форме. Нормальная форма Кодда – Бойса. Приведение сущности к четвертой нормальной форме. Приведение сущности к пятой нормальной форме.

8 Целостность данных. Понятие целостности данных. Целостность таблиц. Целостность внешних ключей (ссылочная целостность). Основные и дополнительные правила ссылочной целостности. Целостность типов данных.

Вопросы к разделу 2

1 Проектирование реляционных БД. Диаграммы «сущность – связь»: модель уровня сущностей, модель данных, основанная на ключах, полная атрибутивная модель. CASE-средства для моделирования структур данных. Операции прямого и обратного проектирования. Синхронизация моделей данных.

2 Физическая организация БД. СУБД. Физическая модель данных. Критерии выбора физической организации данных. Понятие, определение и назначение СУБД. Механизмы доступа к данным из прикладных программ.

3 Назначение и функции СУБД. Управление словарем данных. Управление хранением, преобразованием и представлением данных. Обеспечение безопасности данных. Обеспечение целостности данных. Механизм транзакций. Управление многопользовательским доступом к данным. Механизм блокировок. Наличие возможностей экспорта и импорта данных. Наличие языков доступа к данным и интерфейсов прикладного программирования. Наличие интерфейсов взаимодействия с базой данных.

4 Администрирование БД. Безопасность данных. Повышение производительности: индексирование, оптимизация запросов. Резервное копирование и восстановление данных. Полная архивная копия базы данных и журнал транзакций. Средства защиты данных от несанкционированного доступа: аутентификация, пользовательские роли, механизм привилегий.

5 Язык структурированных запросов SQL. Назначение и общая характеристика языка SQL. Команды определения данных и команды манипулирования данными в языке SQL. Предикаты. Логические связки NOT, AND, OR. Операторы IN, BETWEEN, LIKE. Использование метасимволов «%» и «_»

с оператором LIKE. Команда выборки данных из таблиц. Сортировка строк, использование агрегатных функций и вычисляемых полей. Группировка строк и подсчет итоговых данных. Внутреннее и внешнее соединение таблиц. Использование подзапросов. Операторы EXISTS, ANY (SOME) и ALL в командах с подзапросом. Создание индексов, представлений, хранимых процедур и функций. Курсоры и триггеры.

Вопросы к разделу 3

1 Распределенные базы данных. Тиражирование данных. Понятие распределенной базы данных. Удаленные запросы, распределенные транзакции. Проблемы параллельного выполнения транзакций. Двенадцать правил Дейта для распределенных баз данных. Назначение механизма репликации (тиражирования) данных. Системы «клиент/сервер» как частный случай распределенных систем. Классификация клиент/серверных систем.

2 Хранилища данных. Особенности хранилищ данных по сравнению с операционными базами данных. Подготовка данных, предназначенных для хранения в хранилищах данных. Магазины (витрины) данных.

3 Многомерные базы данных. Разработка (извлечение) данных. Понятие многомерной базы данных. Размещение информации и схема адресации. Назначение технологий разработки (извлечения) данных.

Список литературы

1 **Голицына, О. Л**. Базы данных : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва : ФОРУМ; ИНФРА-М, 2020. – 400 с.

2 **Агальцов, В. П.** Базы данных : учебник : в 2 кн. Кн. 1: Локальные базы данных / В. П. Агальцов. – Москва : ФОРУМ : ИНФРА-М, 2020. – 352 с.

3 **Агальцов, В. П.** Базы данных: в 2 кн. Кн. 2: Распределенные и удаленные базы данных : учебник / В. П. Агальцов. – Москва : ФОРУМ; ИНФРА-М, 2021. – 271 с.

4 **Дейт, К.** Введение в системы баз данных / К. Дейт. – 8-е изд. – Москва; Санкт-Петербург; Киев: Вильямс, 2017. – 1328 с.

5 **Коваленко, В. В.** Проектирование информационных систем: учебное пособие / В. В. Коваленко. – Москва: ФОРУМ; ИНФРА-М, 2018. – 320 с.

6 **Коннолли, Т.** Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. – 3-е изд., испр. и доп. – Москва; Санкт-Петербург; Киев: Вильямс, 2017. – 1120 с.

7 **Мартишин, С. А.** Базы данных. Работа с распределенными базами данных и файловыми системами на примере MongoDB и HDFS с использованием Node.js, Express.js, Apache Spark и Scala : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ИНФРА-М, 2021. – 235 с.

8 **Мартишин, С. А.** Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем: учебное пособие /

С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ; ИНФРА-М, 2021. – 368 с.

9 **Полищук, Ю. В.** Базы данных и их безопасность : учебное пособие / Ю. В. Полищук, А. С. Боровский. – Москва : ИНФРА-М, 2022. – 210 с.

10 **Шустова, Л. И.** Базы данных: учебник / Л. И. Шустова, О. В. Тараканов. – Москва : ИНФРА-М, 2021. – 304 с.