**Еще какая-то инфа**

**1) Файловые системы и базы данных: понятие, назначение, отличие**

Файл – это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные.

Для файл-ого хранения характерна централизованная система упр-ия.

Система управления файлами берет на себя распределение внешней памяти, отображение имен файлов в соответствующие адреса внешней памяти и обеспечение доступа к данным.

· При первом подходе пользователи представляют файл как последовательность записей.

· Второй подход с операционной системой UNIX, состоит в том, что любой файл представляется как последовательность байтов

Таким образом, файловые системы обычно обеспечивают хранение слабо структурированной информации, оставляя дальнейшую структуризацию прикладным программам.

**2) Типы файловых систем**

Современные файловые системы поддерживают многоуровневое именование файлов за счет поддержания во внешней памяти дополнительных файлов со специальной структурой - каталогов.

Хранилища организованы в виде деревьев, как правило размещаются на отдельных дисках – изолированная файловая система.

Представление всей совокупности каталогов и файлов как единого дерева. Система, выполняя поиск файла по имени, запрашивает установку необходимых дисков - централизованная.

**3) Характеристики информационных систем**

· Ориентированы на хранение, выборку и модификацию данных;

· Структура данных информационных систем сложна, и задача обеспечения к ним оперативного доступа требует более развитых средств их структуризации при хранении;

· Структуры данных в разных информационных системах различны;

· Использование одних и тех же данных различными прикладными программами;

· Частое изменение состава и модернизация отдельных прикладных программ при практическом исп-ии.

**4) Ограничения для файловых систем**

· зависимость от программ и данных. Физическая структура и способ хранения записей файлов данных жестко зафиксированы в коде программ приложений. Очевидно, что когда структура данных изменяется,

необходимо модифицировать каждую из программ, обращающихся к файлу, хранящему эти данные.

· несовместимость файлов. Поскольку структура файлов определяется кодом приложений, она также зависит от языка программирования этого приложения. Программная несовместимость файлов, созданных на разных языках программирования, затрудняет процесс их совместной обработки;

· разделение и изоляция данных. Когда данные изолированы в отдельных файлах, доступ к ним затруднен;

· дублирование данных. Такой процесс нежелателен по двум причинам: дублирование данных сопровождается неэкономным расходованием ресурсов компьютера и дублирование данных может привести к нарушению их целостности;

· фиксированные запросы и быстрое увеличение количества приложений. Во многих организациях типы создаваемых запросов и отчетов имели фиксированную форму. В других организациях наблюдалось быстрое увеличение количества файлов и приложений. Программное обеспечение было неспособно адекватно отвечать запросам пользователей, эффективность его падала, а недостаточность документирования дополнительно усложняла сопровождение программ.

**5) БД. Определение.**

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

**6) СУБД. Основные задачи**

Система управления базами данных (СУБД) - совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами.

Система управления базами данных (СУБД) - это комплекс программных и языковых средств, необходимых для создания баз данных, их поддержания в актуальном состоянии и организации в них поиска необходимой информации.

Основные задачи, решаемые с использованием СУБД, следующие:

▪ организация и поддержание логической структуры данных (схемы базы данных);

▪ организация и поддержание физической структуры данных во внешней памяти;

▪ организация доступа к данным и их обработка в оперативной и внешней памяти.

**7) Типы языков, используемых в СУБД**

Группа DBTG также предложила стандартизировать три различных языка:

▪ язык определения данных (Data Definition Language – DDL) для схемы, который позволяет АБД описывать ее;

▪ язык определения данных (также DDL) для подсхемы, который позволяет определять в приложениях те части базы данных, доступ к которым будет необходим;

▪ язык манипулирования данными (Data Manipulation Language – DML), предназначенный для управления данными.

**8) Виды реляционных СУБД**

**9) Преимущества и недостатки СУБД**

К преимуществам СУБД относятся:

▪ контроль за избыточностью данных. Избыточность информации в базах данных не исключается, а лишь контролируется степень ее избыточности. В одних случаях ключевые элементы данных необходимо дублировать для

моделирования связей, в других случаях некоторые данные потребуется дублировать для повышения производительности системы;

▪ непротиворечивость данных. Устранение избыточности данных или контроль над ними позволяет сократить риск возникновения противоречивых состояний. Если элемент данных хранится в базе только в одном экземпляре, то для изменения его значения потребуется выполнить только одну операцию обновления, причем новое значение станет доступным сразу всем пользователям базы данных. Если же элемент данных с ведома системы хранится в базе данных в нескольких экземплярах, то такая система должна следить за тем, чтобы копии не противоречили друг другу.

▪ совместное использование данных. База данных принадлежит всей организации в целом и может совместно использоваться всеми зарегистрированными пользователями. При такой организации работы большое количество пользователей может работать с большим объемом данных. Более того, можно создавать новые приложения на основе уже существующей в базе данных информации и добавлять в нее только те данные, которые еще не хранятся в ней, а не определять заново требования ко всем данным, необходимым новому приложению. Новые приложения также могут использовать такие предоставляемые типичными СУБД функциональные возможности, как определение структур данных и управление доступом к данным, организация параллельной обработки и обеспечение средств копирования/восстановления;

▪ поддержка целостности данных. Целостность данных означает корректность и непротиворечивость хранимых в ней данных. Целостность обычно описывается с помощью ограничений, т. е. правил поддержки непротиворечивости, которые не должны нарушаться в базе данных;

▪ повышенная безопасность. Безопасность базы данных заключается в защите базы данных от несанкционированного доступа со стороны пользователей. Без привлечения соответствующих мер безопасности интегрированные данные становятся более уязвимыми, чем данные в файловой системе. Однако интеграция позволяет определить требуемую систему безопасности базы данных, а СУБД – привести ее в действие. Система обеспечения безопасности может быть выражена в форме учетных имен и паролей для идентификации пользователей, которые зарегистрированы в базе данных;

▪ применение стандартов. Интеграция позволяет АБД определять и применять необходимые стандарты. Например, стандарты отдела и организации, государственные и международные стандарты могут регламентировать формат данных при обмене ими между системами, соглашения об именах, форму представления документации, процедуры обновления и правила доступа;

▪ повышение доступности данных. Данные по различным отделам в результате интеграции становятся непосредственно доступными конечным

пользователям. Потенциально это повышает функциональность системы, что приводит к более качественному обслуживанию конечных пользователей или клиентов организации. Во многих СУБД предусмотрены языки запросов, инструменты для создания отчетов, которые позволяют пользователям задавать непредусмотренные заранее вопросы и почти немедленно получать требуемую информацию на своих терминалах, не прибегая к помощи программиста;

▪ улучшение показателей производительности. На базовом уровне СУБД обеспечивает все низкоуровневые процедуры работы с файлами, которые обычно выполняют приложения. Наличие этих процедур позволяет программисту сконцентрироваться на разработке более специальных, необходимых пользователям функций, не заботясь о подробностях их воплощения на более низком уровне;

▪ упрощение сопровождения системы за счет независимости от данных. В файловых системах описания данных и логика доступа к данным встроены в каждое приложение, что делает программы зависимыми от данных. В СУБД реализован другой подход: описания данных отделены от приложений, поэтому приложения защищены от изменений в описаниях данных (принцип независимости от данных). Наличие независимости программ от данных значительно упрощает обслуживание и сопровождение работающих с базой данных;

▪ улучшенное управление параллельность. В некоторых файловых системах при одновременном доступе к одному и тому же файлу двух пользователей может возникнуть конфликт двух запросов, результатом которого будет потеря информации или утрата ее целостности. В свою очередь, во многих СУБД предусмотрена возможность параллельного доступа к базе данных и гарантируется отсутствие подобных проблем;

▪ развитые службы резервного копирования и восстановления. Ответственность за обеспечение защиты данных от сбоев аппаратного и программного обеспечения в файловых системах возлагается на пользователя. При этом в случае сбоя может быть восстановлена резервная копия, но результаты работы, выполненной после резервного копирования, будут утрачены, и данную работу потребуется выполнить заново. В современных СУБД предусмотрены средства сокращения объема потерь информации от возникновения различных сбоев.

Однако СУБД имеют и ряд недостатков:

▪ сложность. Обеспечение функциональности, которой должна обладать каждая хорошая СУБД, сопровождается значительным усложнением программного обеспечения СУБД. Чтобы воспользоваться всеми преимуществами СУБД, проектировщики и разработчики баз данных, администраторы данных и администраторы баз данных, а также конечные пользователи должны хорошо понимать функциональные возможности СУБД;

▪ размер. Сложность и широта функциональных возможностей приводит к тому, что СУБД становится чрезвычайно сложным программным продуктом, который может занимать много места на диске и требовать большого объема оперативной памяти для эффективной работы;

▪ стоимость СУБД. В зависимости от имеющейся вычислительной среды и требуемых функциональных возможностей, стоимость СУБД может варьироваться в очень широких пределах. Кроме того, следует учитывать ежегодные расходы на сопровождение системы, которые составляют некоторый процент от ее общей стоимости;

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

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

▪ производительность. Обычно файловая система создается для некоторых специализированных приложений, например, для оформления счетов, и потому ее производительность может быть весьма высока. Однако СУБД предназначены для решения более общих задач и обслуживания сразу нескольких приложений. В результате многие приложения в новой среде будут работать не так быстро, как прежде;

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

**10) Трехуровневая архитектура ANSI/SPARK**

Модель ANSI/ представляет собой основу для понимания функциональных особенностей СУБД.

Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления, что обусловлено рядом причин:

▪ каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое представление о них, и изменять свое представление о данных, не оказывая при этом влияние на других пользователей;

▪ пользователи не должны знать особенности физического хранения данных в базе;

▪ администратор должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияние на пользовательские представления;

▪ администратор должен иметь возможность изменять концептуальную или глобальную структуру базы данных без влияния на всех пользователей.

Внешний уровень – это представление базы данных с точки зрения пользователей. Этот уровень описывает ту часть базы данных, которая относится к каждому пользователю.

Концептуальный уровень – это обобщающее представление базы данных. Этот уровень описывает то, какие данные хранятся в базе данных, а также связи, существующие между ними.

Внутренний уровень – это физическое представление базы данных в компьютере. Этот уровень описывает, как информация хранится в базе данных.

Ниже внутреннего уровня находится физический уровень (physical level), который контролируется операционной системой, но под руководством СУБД.

ОТТЕЛЬНО ЭТОЙ ТЕМЕ ПОСВЕЩЕННА ПРЕЗЕНТАЦИЯ 2, 3 (1-17 слайд)

**11) Схема базы данных и уровни ее представления**



Внешний уровень- это представление базы данных с точки зрения пользователей. Этот уровень описывает ту част базы данных ,которая относится к каждому пользователю.

Концептуальный уровень–это обобщающе епредставление базы данных. Этот уровень описывает то, какие данные хранятся в базе данных, а так же связи, существующие между ними. 9логическая структура всей базы)

Внутренний уровень–это физическое представление базы данных в компьютере. Этот уровень описывает, как информация хранится в базе данных.

**12) Преимущества и недостатки трехуровневой архитектуры**

Достоинства

масштабируемость

· конфигурируемость — изолированность уровней друг от друга позволяет (при правильном развертывании архитектуры) быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней

· высокая безопасность

· высокая надёжность

· низкие требования к скорости канала (сети) между терминалами и сервером приложений

· низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости. Терминалом может выступать не только компьютер, но и, например, мобильный телефон.

Недостатки

· более высокая сложность создания приложений;

· сложнее в разворачивании и администрировании;

· высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования;

· высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

**13) Концептуальные модели и их применение, преимущества и недостатки.**

Модель данных–это интегрированный набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные вне которой организации. По существу, модель данных — это совокупность трех составляющих : типов структур данных, операций над данными, ограничений целостности

Для архитектуры ANSI/SPARC можно идентифицировать следующие три связанные модели данных:

§внешнюю модель данных,отображающую представления каждого существующего в организации типа пользователей, которую иногда называю тпредметной областью(UniverseofDiscourse–UoD);

§концептуальную модель данных,отображающую логическое(или обобщенное) представление о данных,н езависимое от типа выбранной СУБД;

§внутреннюю модель данных, отображающую концептуальную схему определенным образом, понятным выбранной целевой СУБД

В настоящее время ER-модель стала одним из основны хметодов концептуального проектирования баз данных..

Достоинства и недостатки????

**14) Типы объектных моделей**

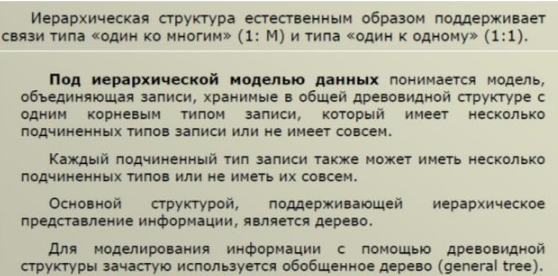
· модельтипа«сущность–связь»илиER-модель(Entity-Relationshipmodel);

· семантическая модель;

· функциональная модель;

· объектно-ориентированная модель.

**15)Иерархическая модель**



Важной особенностью иерархической структуры является то,что в ней дочерние экземпляры не могут существовать без наличия родительских экземпляров.

**16) Сетевая модель**

Под сетевой моделью данных понимается модель, состоящая из записей, элементов данных и связей типа «один ко многим» (1:M), установленных между записями. При этом связи типа «многие ко многим» (M:N) и рекурсивные связи поддерживаются с помощью декомпозиции.

Сетевая база данных состоит из набора записей, соответствующих каждому экземпляру объекта предметной области, и набора связей между ними.

В качестве базовой физической структуры данных выступает сеть, в которой записи связаны друг с другом в один набор с помощью указателей. Записи могут содержать встроенные в них указатели. Под логической структурой данных понимается набор, в котором один тип записи-родителя может быть связан со многими типами записей-потомков. Если записи-потомки закреплены за записью-родителем, то удаление записи-родителя приводит к каскадному удалению записей-потомков.

В качестве недостатков сетевой модели можно отметить следующие:

▪ если концептуальная схема должна быть изменена в соответствии с изменениями структуры хранения записей, то это может вызвать необходимость изменения прикладных программ. Следовательно, реализация макета базы данных может существенно усложниться;

▪ новые или измененные приложения могут не обладать достаточной производительностью, т. к. база данных структурирована для существовавших ранее приложений. Может оказаться невозможной эффективная поддержка всех требуемых приложений;

▪ при проектировании базы данных потребуется квалифицированная помощь, особенно при определении необходимых структур данных и при разработке связанных с ними прикладных программ.

**17) 12 правил Кодда**

1. Данные в виде таблиц

2. Данные доступны логически

3. Значения как неизвестные (а не ноль)

4. БД должна включать метаданные (В системных таблицах хранятся метаданные, есть еще пользовательский вид таблиц)

5. СУБД использует единый язык описания процедур

6. Обеспечивается альтернативный вид отображения данных (Пользователь должен иметь возможность строить виртуальные таблицы - представления (View).

7. Должны поддерживать операции реляционной алгебры

8. Независимы от физической организации данных (Приложения, оперирующие с данными реляционных БД, не должны зависеть от физического хранения данных (от способа хранения, формата хранения и др.)

9. Независимы от логической организации данных (Приложения, оперирующие с данными реляционных БД, не должны зависеть от организации связей между таблицами (логической организации). При изменении связей между таблицами не должны меняться ни сами таблицы, ни запросы к ним).

10. СУБД отвечает за целостность данных (Под целостностью данных в общем случае понимается готовность БД к работе.)

11. Целостность данных не может быть нарушена при любых манипуляциях

12. Должны поддерживаться распределенные операции. База данных может быть распределённой, может находиться на нескольких компьютерах, и это не должно оказывать влияния на приложения. Перенос базы данных на другой компьютер не должен оказывать влияния на приложения.

**18) Реляционная и постреляционная модели, достоинства и недостатки**

Реляционная модель данных – позволяет представлять информацию о предметной области с помощью взаимосвязанных таблиц.

В реляционных базах данных вся информация сведена в таблицы, строки и столбцы, которые называются записями и полями соответственно. Эти таблицы получили название реляций. Записи в таблицах не повторяются. Их уникальность обеспечивается первичным ключом, содержащим набор полей, однозначно определяющих запись.

Достоинства

· Эта модель данных отображает

информацию в наиболее простой для

пользователя форме

· Основана на развитом математи-ческом

аппарате, который позволяет достаточно

лаконично описать основные операции над

данными.

· Позволяет создавать языки

манипулирования данными не процедурного

типа.

· Манипулирование данными на уровне

выходной БД и возможность изменения.

Недостатки

· Самый медленный доступ к данным.

· Трудоемкость разработки

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля — поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу. Помимо обеспечения вложенности полей постреляционная модель поддерживает ассоциированные многозначные поля (множественные группы). При этом в строке первое значение одного столбца ассоциации соответствует первым значениям всех других столбцов ассоциации. На длину полей и количество полей в записях таблицы не накладывается требование постоянства. Это означает, что структура данных и таблиц имеют большую гибкость.

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостатком постреляционной модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

ВКРАТЦЕ Особенности постреляционной модели: снимается ограничение, однозначность полей, нет ограничений на длину поля и их количества.

**19) Многомерная модель**

Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия решения.

Реляционные СУБД предназначались для информационных систем оперативной обработки информации. В системах аналитической обработки они показали себя недостаточно гибкими. Более эффективными здесь оказываются многомерные СУБД. Многомерные СУБД являются специализированными СУБД, предназначенными для интерактивной аналитической обработки информации.

Рассмотрим основные понятия многомерных моделей данных, к числу которых относятся измерение и ячейка.

Измерение (Dimension) – это множество однотипных данных, образующих одну из граней гиперкуба. В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка (Cell) или показатель – это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой.

В зависимости от того, как формируются значения некоторой ячейки, обычно она может быть:

▪ переменной (значения изменяются и могут быть загружены из внешнего источника данных или сформированы программно)

▪ формулой (значения, подобно формульным ячейкам электронных таблиц, вычисляются по заранее заданным формулам).

В существующих МСУБД используются два основных варианта (схемы) организации данных:

▪ В поликубической схеме предполагается, что в БД может быть определено несколько гиперкубов с различной размерностью и с различными измерениями в качестве граней.

▪ В случае гиперкубической схемы предполагается, что все показатели определяются одним и тем же набором измерений. Это означает, что при наличии нескольких гиперкубов БД все они имеют одинаковую размерность и совпадающие измерения.

Основным достоинством многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем.

Недостатком многомерной модели данных является ее громоздкость для простейших задач обычной оперативной обработки информации.

**20) Объектно-ориентированная модель**

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом (например, строковым – string) или типом конструируемым пользователем (определяется как class).

Каждый объект-экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект-экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в БД образуют связную иерархию объектов.

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними состоит в методах манипулирования данными. Поиск в объектно-ориентированной БД состоит в выяснении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в БД.

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

**21) Сущность, связи, домен и т.д. в реляционной СУБД**

Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные.

Сущность есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении.

Атрибуты представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Связь является ассоциативным отношением между сущностями.

Домен представляет собой множество всех возможных значений определенного атрибута отношения.

Тип данных задает множество значений данного типа, набор операций операций, применимых применимых к значениям значениям типа и способ внешнего внешнего представления значений типа

**22) Типы ключей и цели их использования в СУБД**

Ключ или потенциальный ключ – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности.

Первичным ключом или ключом отношения (ключевым атрибутом) называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей.

Внешний ключ – это столбец, значения которого однозначно характеризуют сущности, подставленные строками некоторого другого отношения, т. е. задают значения их первичного ключа.

Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом.

Ключи обычно используют для достижения следующих целей:

1. Исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются);

2. Упорядочения кортежей. Возможно упорядочение по возрастанию или убыванию значений всех ключевых атрибутов;

3. Ускорения работы с кортежами отношения;

4. Организации связывания таблиц.

**23) Индексирование таблиц, схемы индексации**

Определение ключа для таблицы означает автоматическую сортировку записей, контроль отсутствия повторений значений в ключевых полях записей и повышение скорости выполнения операций поиска в таблице. Для реализации этих функций в СУБД применяют индексирование. Термин «индекс» тесно связан с понятием «ключ», хотя между ними есть и некоторое отличие. Под индексом понимают средство ускорения операции поиска записей в таблице, а, следовательно, и других операций, использующих поиск: ▪ извлечение, ▪ модификация, ▪ сортировка и т. д. Таблицу, для которой используется индекс, называют индексированной.

Индекс выполняет роль оглавления таблицы, просмотр которого предшествует обращению к записям таблицы.

Проиллюстрируем организацию индексирования таблиц двумя схемами: одноуровневой и двухуровневой.

При одноуровневой схеме в индексном файле хранятся короткие записи, имеющие два поля:

▪ поле содержимого старшего ключа (хеш-кода ключа) адресуемого блока;

▪ поле адреса начала этого блока.

Если в индексном файле хранятся хеш-коды ключевых полей индексированной таблицы, то алгоритм поиска нужной записи (с указанным ключом) в таблице включает в себя следующие три этапа.

1. Образование свертки значения ключевого поля искомой записи.

2. Поиск в индексном файле записи о блоке, значение первого поля которого больше полученной свертки (это гарантирует нахождение искомой свертки в этом блоке).

3. Последовательный просмотр записей блока до совпадения сверток искомой записи и записи блока файла.

На практике для создания индекса для некоторой таблицы БД пользователь указывает поле таблицы, которое требует индексации. Ключевые поля таблицы во многих СУБД как правило индексируются автоматически. Индексные файлы, создаваемые по ключевым полям таблицы, часто называются файлами первичных индексов.

Индексы, создаваемые пользователем для не ключевых полей, иногда называют вторичными (пользовательскими) индексами.

Индексные файлы, создаваемые для поддержания вторичных индексов таблицы, обычно называются файлами вторичных индексов.

**24) Связывание таблиц (типы связей, ограничения, контроль целостности связей)**

Связывание таблиц при выполнении таких операций как поиск, просмотр, редактирование, выборка и подготовка отчетов обычно обеспечивает возможность обращения к произвольным полям связанных записей.

Между таблицами могут устанавливаться бинарные (между двумя таблицами), тернарные (между тремя таблицами) и, в общем случае, n-арные связи. При связывании двух таблиц выделяют основную (главную) и дополнительную (подчиненную) таблицы. Логическое связывание таблиц производится с помощью ключа связи.

В зависимости от того, как определены поля связи основной и дополнительной таблиц (как соотносятся ключевые поля с полями связи), между двумя таблицами в общем случае могут устанавливаться следующие четыре основные вида связи:

▪ один к одному (1:1)

Связь вида 1:1 образуется в случае, когда все поля связи основной и дополнительной таблиц являются ключевыми. Поскольку значения в ключевых полях обеих таблиц не

повторяются, обеспечивается взаимно-однозначное соответствие записей из этих таблиц. Сами таблицы, по сути, здесь становятся равноправными.

▪ один к многим (1:М)

Связь 1:М имеет место в случае, когда одной записи главной таблицы соответствует несколько записей подчиненной таблицы.

▪ многие к одному (М:1)

Связь М:1 имеет место в случае, когда одной или нескольким записям основной таблицы ставится в соответствие одна запись дополнительной таблицы.

▪ многие к многим (М:М или M:N)

Самый общий вид связи М:М возникает в случаях, когда нескольким записям основной таблицы соответствует несколько записей дополнительной таблицы.

Контроль целостности связей

Контроль целостности связей обычно означает анализ содержимого двух таблиц на соблюдение следующих правил:

▪ каждой записи основной таблицы соответствует нуль или более записей дополнительной таблицы;

▪ в дополнительной таблице нет записей, которые не имеют родительских записей в основной таблице;

▪ каждая запись дополнительной таблицы имеет только одну родительскую запись основной таблицы.

Опишем действие контроля целостности при манипулировании данными в таблицах.

Рассмотрим три основные операции над данными двух таблиц:

-ввод новых записей

При вводе новых записей возникает вопрос определения последовательности ввода записей в таблицы такой, чтобы не допустить нарушение целостности. Исходя из приведенных правил, логичной является схема, при которой данные сначала вводятся в

основную таблицу, а потом – в дополнительную. Очередность ввода может быть установлена на уровне целых таблиц или отдельных записей (случай одновременного ввода в несколько открытых таблиц).

-модификацию записей

При редактировании полей связи дополнительной таблицы очевидным требованием является то, чтобы новое значение поля связи совпадало с соответствующим значением какой-либо записи основной таблицы, Т.е. дополнительная запись может сменить

родителя, но остаться без него не должна. Редактирование поля связи основной таблицы разумно подчинить одному из следующих правил:

▪ редактировать записи, у которых нет подчиненных записей. Если есть подчиненные записи, то блокировать модификацию полей связи.

▪ изменения в полях связи основной записи мгновенно передавать во все поля связи всех записей дополнительной таблицы (каскадное обновление).

-удаление записей

Удаление записей основной таблицы логично подчинить одному из следующих правил:

▪ удалять можно запись, которая не имеет подчиненных записей;

▪ запретить (блокировать) удаление записи при наличии подчиненных записей, либо удалять ее вместе со всеми подчиненными записями (каскадное удаление).

**25) Функциональные зависимости (идея, ограничения применения)**

Функциональная зависимость – это связь между атрибутами.

Например, если нам известен номер счета клиента, то мы можем определить состояние его счета. В таком случае мы можем сказать, что атрибут Состояние\_Счета\_Клиента функционально зависит от атрибута Номер\_Счета\_Клиента.

Функциональные зависимости обозначаются следующим образом:

Номер\_Студента > Специальность

Данное выражение читается так:

▪ атрибут Номер\_Студента функционально определяет атрибут Специальность;

▪ атрибут Номер\_Студента определяет атрибут Специальность;

▪ атрибут Специальность зависит от атрибута Номер\_Студента.

Аномалией удаления – называется такая ситуация, когда, удаляя факты, относящиеся к одной сущности, мы непроизвольно удаляем факты, относящиеся к другой сущности. Выполнив одну операцию удаления, мы теряем информацию о двух сущностях.

Аномалия вставки – это ограничение, которое делает невозможным запись в таблицу некоторого факта об одной сущности, не указав дополнительно некоторый факт о другой сущности.

Разбиение одного отношения на два имеет, однако, один недостаток. Предположим, что студент хочет записаться в несуществующую секцию. Должна ли система каким-либо образом препятствовать добавлению строк в таблицу СТУДЕНТ-СЕКЦИЯ, если название соответствующей секции отсутствует в таблице СЕКЦИЯ- ПЛАТА? Ответ на этот вопрос содержится в требованиях пользователей. Если такое действие должно быть запрещено, данное ограничение необходимо внести в схему базы данных (выполнение этого ограничения будет возложено на СУБД). Если нет, то соблюдение ограничения должно обеспечиваться прикладными программами.

При проектировании базы данных мы можем записать это ограничение любым из следующих способов:

▪ множество значений атрибута Секция в таблице СТУДЕНТ-СЕКЦИЯ является подмножеством множества значений атрибута Секция в таблице СЕКЦИЯ-ПЛАТА;

▪ СТУДЕНТ-СЕКЦИЯ [Секция] является подмножеством СЕКЦИЯ-ПЛАТА [Секция];

▪ СТУДЕНТ-СЕКЦИЯ [Секция] СЕКЦИЯ-ПЛАТА [Секция]. Таким образом, прежде чем ввести название какой-то секции в отношение СТУДЕНТ-СЕКЦИЯ, мы должны проверить, что такое название уже существует в отношении СЕКЦИЯ-ПЛАТА.

Ограничения, подобные этому, называются ограничениями ссылочной целостности, или ограничениями целостности по внешнему ключу.

1. **Обеспечение безопасности СУБД (инструкции Grant и Revoke)**

В перечне базовых инструкций языка SQL представлены инструкции GRANT и REVOKE, предоставляющие или отменяющие привилегии пользователям. Структура инструкции GRANT и REVOKE выглядит следующим образом:

GRANT <список привилегий>

ON <Имя Объекта>

ТО <Имя Пользователей>

[WITH GRANT OPTION];

REVOKE <список привилегий >

ON < Имя Объекта >

FROM < Имя Пользователей >;

В большинстве случаев право использования команд GRAND и REVOKE по конкретному объекту автоматически имеют пользователи, создавшие данный объект, т.е. их владельцы. В других подходах этим правом наделяются доверенные субъекты, т.е. администраторы.

1. **Ролевое разграничение доступа к СУБД**

Основная идея использования ролей заключается в том, чтобы упаковать и управлять дискретными наборами привилегий, которые могут назначаться и отменяться как одно целое.

Роль создается в базе данных и доступна только в этой базе данных. CREATE ROLE < имя роли>;

Роль представляет своего рода контейнер для набора привилегий.

GRANT <привилегии> ТО < имя роли>;

После назначения роли привилегий она становится доступна как привилегия. Может быть создано множество ролей.

GRANT < имя роли> ТО <имя пользователя>

[WITH ADMIN OPTION];

Роль никогда не может быть предоставлена как часть пакета ALL, хотя роли могут быть назначены привилегии ALL.

Процесс реализации ролей состоит из четырех шагов:

1. Создание роли с использованием оператора CREATE ROLE.

2. Назначение привилегий этой роли посредством GRANT привилегия ТО роль.

3. Назначение роли пользователям посредством GRANT роль ТО пользователь.

4. Задание роли вместе с именем пользователя при соединении с базой данных.

CREATE ROLE Users;

COMMIT;

GRANT ALL ON Department TO Users;

GRANT Users TO John;

[WITH ADMIN OPTION];

1. **Функциональные возможности СУБД**

Система управления базами данных (СУБД) - это комплекс программных и языковых средств, необходимых для создания баз данных, их поддержания в актуальном состоянии и организации в них поиска необходимой информации.

**Функциональные возможности СУБД:**

▪ позволяет определять базу данных с помощью языка определения данных;

▪ позволяет вставлять, обновлять, удалять и извлекать информацию из базы данных с помощью языка управления данными;

▪ предоставляет контролируемый доступ к базе данных;

Обладание указанными выше функциональными возможностями превращает СУБД в мощный инструмент обработки информации.

Возможности СУБД

СУБД предоставляет контролируемый доступ к базе данных с помощью следующих средств:

▪ система обеспечения безопасности, предотвращающая несанкционированный доступ к базе данных со стороны пользователей;

▪ система поддержки целостности данных, обеспечивающая непротиворечивое состояние хранимых данных;

▪ система управления параллельной работой приложений, контролирующая процессы их совместного доступа к базе данных;

▪ система восстановления, позволяющая восстановить базу данных до предыдущего непротиворечивого состояния, нарушенного в результате сбоя аппаратного или программного обеспечения; ▪ доступный пользователям каталог, содержащий описание хранимой в базе данных информации.

1. **Представления (идея, принцип работы, преимущества и недостатки)**

СУБД усложняет работу, предоставляя пользователям гораздо большее количество данных, чем им действительно требуется. Для решения этой проблемы в СУБД имеется механизм создания представления (view), который позволяет любому пользователю иметь собственный взгляд на базу данных.

Представления обладают следующими *достоинствами*:

▪ обеспечивают дополнительный уровень безопасности;

▪ предоставляют механизм настройки внешнего интерфейса базы данных;

▪ позволяют сохранять внешний интерфейс базы данных непротиворечивым и неизменным даже при внесении изменений в ее структуру;

Однако использование *представлений* в среде SQL не лишено *недостатков*.

**Ограниченные возможности обновления**

В некоторых случаях *представления* не позволяют вносить изменения в содержащиеся в них данные.

**Структурные ограничения**

Структура *представления* устанавливается в момент его создания. Если определяющий запрос представлен в форме SELECT \* FROM\_, то символ \* ссылается на все столбцы, существующие в исходной таблице на момент создания *представления*. Если впоследствии в исходную таблицу базы данных добавятся новые столбцы, то они не появятся в данном *представлении* до тех пор, пока это *представление* не будет удалено и вновь создано.

**Снижение производительности**

Использование *представлений* связано с определенным снижением производительности. В одних случаях влияние этого фактора совершенно незначительно, тогда как в других оно может послужить источником существенных проблем. Например, *представление*, определенное с помощью сложного многотабличного запроса, может потребовать значительных затрат времени на обработку, поскольку при его разрешении потребуется выполнять соединение таблиц всякий раз, когда понадобится доступ к данному *представлению*. Выполнение разрешения *представлений* связано с использованием дополнительных вычислительных ресурсов.

1. **Классификация СУБД**

В качестве основных классификационных признаков СУБД можно использовать следующие:

▪ вид программы;

▪ характер использования;

▪ модель данных.

Классификация СУБД по виду программ

К СУБД относятся следующие основные виды программ:

▪ Полнофункциональные СУБД (ПФСУБД). ПФСУБД имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД. Для создания запросов и отчетов используется язык QBE.

▪ Серверы БД. Предназначены для организации центров обработки данных в сетях ЭВМ. Серверы БД реализуют функции управления базами данных, запрашиваемые клиентскими программами обычно с помощью операторов SQL.

▪ Клиенты БД. В роли клиентских программ для серверов БД в общем случае могут использоваться различные программы: ПФСУБД, электронные таблицы, текстовые процессоры, программы электронной почты и т. д.

▪ Средства разработки программ работы с БД. Используются для создания пользовательских приложений по работе с БД, серверов БД и их отдельных компонентов, клиентских программ.

▪ Дополнительные средства. Используются для управления данными и организации обслуживания БД (например, мониторы транзакций).

Классификация СУБД по характеру использования

▪ Персональные СУБД обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД или разработанные с их помощью приложения зачастую могут выступать в роли клиентской части многопользовательской СУБД. К персональным СУБД, например, относятся Visual FoxPro, Paradox, Clipper, dBase, Access и др.

▪ Многопользовательские СУБД включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (с разными типами ЭВМ и операционными системами). К многопользовательским СУБД относятся, например, СУБД Oracle и Informix.

Классификация СУБД по используемой модели данных

Модели данных:

▪ Иерархические.

▪ Сетевые.

▪ Реляционные.

▪ Постреляционные.

▪ Многомерные.

▪ Объектно-ориентированные.

Некоторые СУБД могут одновременно поддерживать несколько моделей данных.

1. **Организационно-управленческое обеспечение СУБД**

Совокупность видов обеспечения составляет комплекс средств автоматизации некоторой деятельности.

Различают следующие виды обеспечения СУБД:

▪ информационное;

▪ математическое;

▪ лингвистическое;

▪ программное;

▪ аппаратное;

▪ организационное;

▪ правовое.

Прояснения:

Информационное обеспечение СУБД - это совокупность информации, необходимой для нормального функционирования АИС, представленная в заданной форме. Таким образом, информационное обеспечение включает в себя как собственно рабочие данные, так и метаданные, хранящиеся в системном каталоге: ▪ имена, типы и размеры элементов данных; ▪ имена связей; ▪ ограничения целостности данных; ▪ имена зарегистрированных пользователей, которым предоставлены определенные права доступа к данным; ▪ используемые индексы и структуры хранения.

Математическое обеспечение СУБД – это математические методы и алгоритмы расчетов и принятия решений, которые используются при организации функционирования АИС.

Лингвистическое обеспечение СУБД - это совокупность языков для реализации и эксплуатации АИС. Приложения БД создаются на языках третьего поколения (C, Pascal) или на языках четвертого поколения (SQL, операторы которого внедряются в программы на языках третьего поколения). Однако СУБД может иметь свои собственные инструменты четвертого поколения, предназначенные для быстрой разработки приложений с использованием встроенных непроцедурных языков запросов, генераторов отчетов, форм, графических изображений и полномасштабных приложений.

Программное обеспечение СУБД - это совокупность соответствующих системных и прикладных программ, необходимых для работы АИС. Оно включает в себя программное обеспечение самой СУБД и приложений БД, операционную систему, сетевое программное обеспечение (если СУБД используется в сети).

Организационное обеспечение СУБД - это совокупность инструкций, распоряжений, приказов и т.п., регламентирующих права, обязанности и ответственность персонала и пользователей БД в процессе проектирования и эксплуатации АИС.

Аппаратное обеспечение СУБД – это совокупность средств вычислительной техники, средств связи и специального оборудования, необходимого для эксплуатации АС. Используемое аппаратное обеспечение зависит от требований организации и используемой СУБД. Одни СУБД предназначены для работы только с конкретными типами операционных

систем или оборудования, другие работают с широким кругом аппаратного обеспечения и различными операционными системами.

Правовое обеспечение СУБД включает в себя совокупность правовых норм: ▪ определяющих юридическую силу информации на носителях данных и документов, используемых при функционировании АИС и создаваемых системой;

▪ регламентирующих правоотношения между людьми, входящими в состав персонала АИС (права, обязанности и ответственность), а также между персоналом АИС и персоналом систем, взаимодействующих с АИС.

**37) Основные этапы проектирования СУБД**

1. Осознание о необходимости разработки БД

2. Определение бизнес-процесса

3. Определение входных и выходных данных

4. Отрисовка базы в какой-либо нотации

5. Определение связи

6. Выбор программного средства для реализации

7. Регистрируем БД

8. Разработка таблиц, приложений и т.д. (тела) БД

9. Определение правил доступа к базе

10. Определение правил (принципов) хранения базы

11. Настройка правил резервного копирования

12. Архивирование базы

**38) Администрирование информационных систем**

В общем плане термин «администрирование» определяет комплекс процессов при создании, эксплуатации и использовании АИС, связанных с обеспечением надежности и эффективности функционирования АИС, безопасности данных и организацией коллективной работы пользователей различных категорий.

Этот комплекс процессов можно разделить по решаемым задачам на следующие группы:

▪ обеспечение и поддержание настройки структурного, интерфейсного и технологического компонентов АИС на структуру и процессы предметной области системы;

▪ обеспечение надежности и сохранности данных;

▪ организация и обеспечение коллективной работы пользователей с общими данными

Обеспечение надежности и сохранности данных является одной из главных обязанностей администратора АИС и включает, в свою очередь, решение ряда следующих технологических и профилактических задач:

▪ планирование, конфигурирование и поддержание системы использования устройств внешней памяти, на которых размещаются файлы данных;

▪ архивирование и резервирование данных;

▪ восстановление данных после сбоев и повреждений;

▪ проверка и поддержание целостности данных.

**39) Клиент-серверная архитектура СУБД**

В структуре СУБД выделяют три компонента:

▪ компонент представления, реализующий функции ввода и отображения данных;

▪ прикладной компонент, включающий набор запросов, событий, правил, процедур и других вычислительных функций;

▪ компонент доступа к данным, реализующий функции хранения, извлечения, физического обновления и изменения данных.

**40) Преимущества и недостатки различных архитектур хранения данных** (удаленный доступ, файловый сервер и т.д.).

*Файловый сервер(FS)*

Описание: (Один из компьютеров сети определяется файловым сервером (общим хранилищем данных), а все основные компоненты СУБД размещаются на клиентских установках.

При обращении к данным ядро СУБД обращается с запросами на ввод-вывод данных к файловой системе. В оперативную память клиентской установки на время сеанса работы полностью или частично копируется файл базы данных)

Достоинства: простота и отсутствие высоких требований к производительности сервера.

Недостатки: высокий сетевой трафик, отсутствие специальных механизмов СУБД по обеспечению безопасности данных.

*Удаленный доступ к данным (RDA)*

Описание: эта модель основана на учете специфики хранения и физической обработки данных во внешней памяти для реляционных СУБД. В данной модели компонент доступа к данным реализуется в виде самостоятельной программной части СУБД, называемой SQLсервером, и размещается на сервере. SQL-сервер выполняет низкоуровневые операции по организации, размещению, хранению и манипулированию данными. На сервере размещаются также файлы БД и системный каталог БД.

На клиентских установках размещаются программы, реализующие интерфейсные и прикладные функции СУБД. Прикладной компонент клиента формирует необходимые SQLинструкции и направляет их SQL-серверу, который принимает, интерпретирует, выполняет, проверяет эти инструкции, обеспечивает выполнение ограничений целостности и безопасности данных и направляет клиентам результаты обработки SQLинструкций (наборы данных).

Достоинства: в результате реализации такого подхода резко уменьшается загрузка сети. RDA-модель позволяет также унифицировать интерфейс взаимодействия прикладных компонентов СУБД с общими данными. Такое взаимодействие стандартизовано в рамках языка SQL специальным протоколом ODBC, играющим важную роль в обеспечении независимости от типа СУБД на клиентских установках. Это позволяет интегрировать уже существующие локальные БД в создаваемые распределенные информационные системы независимо от типов СУБД клиентов и сервера.

Недостатки: высокие требования к клиентским вычислительным установкам, так как на них выполняются прикладные программы обработки данных. Значительный трафик сети, поскольку с сервера направляются клиентам наборы данных, которые могут иметь существенный объем.

*Сервер базы данных(DBS)*

Описание: на клиентских установках в DBS-модели размещается только интерфейсный компонент, а все остальные компоненты СУБД размещаются на сервере. От клиентов на сервер направляются только вызовы необходимых процедур, запросов и других функций по обработке данных.

Все операции по обработке данных выполняются на сервере, а пользователю направляются лишь результаты обработки (а не наборы данных, как в RDA-модели).

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

*Сервер приложений (AS)*

Чтобы разнести требования к вычислительным ресурсам сервера в отношении быстродействия и памяти по разным вычислительным установкам, используется модель сервера приложений. Суть данной модели состоит в переносе прикладного компонента СУБД на специализированный в отношении повышенных по быстродействию ресурсов дополнительный сервер системы – сервер приложений.

На клиентских установках располагается интерфейсная часть СУБД, откуда вызовы функций обработки данных направляются на сервер приложений. За выполнением низкоуровневых операций по доступу к данным сервер приложений обращается к SQL-серверу, направляя ему вызовы SQL-процедур и получая от него наборы данных. Таким образом, сервер приложений управляет формированием транзакций, которые выполняет SQL-сервер.