Содержание

1 Экзамен

1.1 Базы данных и системы управления базами данных. Определения, основные функции и классификация

База данных (БД) — самодокументированное собрание интегрированных записей.

1) БД является самодокументированной, т.е. она содержит описание собственной структуры. Это описание называется словарем данных, каталогом данных или метаданными.
2) БД является собранием интегрированных записей, т.е. она содержит:
   - файлы данных;
   - метаданные (данные о данных);
   - индексы, которые представляют связи между данными, а также служат для повышения производительности приложений базы данных;
   - метаданные приложения.
3) БД является информационной моделью пользовательской модели предметной области.

OLAP — online analtyc proccesing (операциооные данные).

OLTP — online transaction proccessing (перманентые данные).

Таблица 1

OLAP	OLTP
чтение	вставка, удаление, обновление
минимальное время отклика	минимальное время вставки, удаление, обновление

Транзакции — либо все действия, либо никакие действия.

Любая бд хранит:

1) метаданные (данные о данных);
2) файлы данных,
3) Индексы (indexes), которые представляют связи между данными, а также служат для повышения производительности приложений базы данных.
4) Может содержать метаданные приложений (application metadata).

Основные храктеристики, требования

1) **Неизбыточность данных** — каждое данное присутствует в БД в единственном экземпляре.
2) **Совместное использование данных** многими пользователями.

3) **Эффективность доступа** к БД - высокое быстродействие, т. е. малое время отклика на запрос.
4) **Целостность данных** — соответствие имеющейся в БД информации её внутренней логике, структуре и всем явно заданным правилам.
5) **Безопасность данных** — защита данных от преднамеренного или непреднамеренного искажения или разрушения данных.
6) **Восстановление данных** после программных и аппаратных сбоев.
7) **Независимость данных** от прикладных программ.

Система баз данных (СБД) — совокупность одной или нескольких баз данных и комплекса информационных, программных и технических средств, обеспечивающих накопление, обновление, корректировку и многоаспектное использование данных в интересах пользователей.

Система управления базами данных (СУБД) — приложение, обеспечивающее создание, хранение, обновление и поиск информации в базах данных.

Основные функции СУБД

1) Управление данными во внешней памяти.
2) Управление буферами оперативной памяти.
3) Управление транзакциями.
4) Журнализация.
5) Поддержка языка или языкового пакета (-ов).

Классификация СУБД:

1) По модели данных:
   - Дореляционные (Инвертированные списки, иерархические и сетевые)
     - Инвертированные списки (файлы). БД на основе инвертированных списков представляет собой совокупность файлов, содержащих записи (таблиц). Для записей в файле определен некоторый порядок, диктуемый физической организацией данных. Для каждого файла может быть определено произвольное число других упорядочений на основании значений некоторых полей записей (инвертированных списков). Обычно для этого используются индексы. В такой модели данных отсутствуют ограничения целостности как таковые. Все ограничения на возможные экземпляры БД задаются теми программами, которые работают с БД. Одно из немногих ограничений, которое все-таки может присутствовать - это ограничение, задаваемое уникальным индексом.
     - Иерархичекие

  - Сетевые (могут быть представлены в виде графа; логика выборки зависит от физической организации данных)
- Реляционные
  - Структурный (данные — набор отношений)
  - Целостностный (отношения (таблицы) отвечают определенным условиям целостности)
  - Манипуляционный (манипулирования отношениями осуществляется средствами реляционной алгебры и/или реляционного исчисления)
- Постреляционные

2) По архитектуре организации хранения данных:
   - Локальные (все части локальной СУБД размещаются на одном компьютере)
   - Распределенные (части СУБД могут размещаться на 2-х и более компьютерах)
3) По способу доступа к БД:
   - Файл-серверные (при работе с базой, данные перегоняются приложению, которое с ней работает, вне зависимости от того, сколько их нужно. Все операции — на стороне клиента. Файловый сервер периодически обновляется тем же клиентом)
   - Клиент-серверные (вся работа на сервере, по сети передаются результаты запросов, гораздо меньше информации. Обеспечивается безопасность данных, потому что все происходит на стороне сервера. Проще исключить одновременное изменение и тп)
   - Встраиваемые — библиотека, которая позволяет унифицированным образом хранить большие объемы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объемами данных.
   - Сервисно-ориентированные (БД является хранилищем сообщений, промежуточных состояний, метаинформации об очередях сообщений и сервисах)
   - Прочие (пространственная, временная и пространственно-временная)

Архитектура хранения данных

1) Локальные.
2) Распределенные.
3) По способу обращения к данным.
   - Файл серверные.
   - Клиент серверные (PostGress, MSSQL, Oracle, MySQL, Mongo).
   - Встраиваемые (SQLlite).

- Сервисно-ориентированные (KafcaBD).
- Прочее - time series.

1.2 Семантическое моделирование данных

Любая развитая семантическая модель данных, как и реляционная модель, включает структурную, манипуляционную и целостную части. Главным назначением семантических моделей является обеспечение возможности выражения семантики данных. На практике семантическое моделирование используется на первой стадии проектирования базы данных. При этом в терминах семантической модели производится концептуальная схема базы данных, которая затем:

1) Либо вручную преобразуется к реляционной схеме;
2) Либо реализуется автоматизированная компиляция концептуальной схемы в реляционную;
3) Либо происходит работа с базой данных в семантической модели, т.е. под управлением СУБД, основанных на семантических моделях данных.

Наиболее известным представителем класса семантических моделей предметной области является модель «сущность-связь» или ER-модель, , предложенная Питером Ченом в 1976 году.

Модель сущность-связь — модель данных, позволяющая описывать концептуальные схемы предметной области. Предметная область — часть реального мира, рассматриваемая в пределах данного контекста. Под контекстом здесь может пониматься, например, область исследования или область, которая является объектом некоторой деятельности. ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных.

Основными понятиями ER-модели являются сущность, связь и атрибут(свойство).

Сущность - это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. При этом имя сущности - имя типа, а не некоторого конкретного экземпляра этого типа. Каждый экземпляр сущности должен быть отличим от любого другого экземпляра этой сущности.

Связь - это ассоциация, устанавливаемая между сущностями. Эта ассоциация может существовать между разными сущностями или между сущностью и ей же самой (рекурсивная связь). Сущности, включенные в связь, называются её участниками, а количество участников - степенью связи. Связи в ER-модели могут иметь тип «один к одному», «один ко многим», «многие ко многим».

Свойством/атрибут сущности (и связи) является любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности (или связи).

Ключи: Первичный ключ — набор атрибутов однозначно идентифицирующий кортеж значений, и Внешний ключ

1.3 Реляционная модель данных: структурная, целостная, манипуляционная части. Реляционная алгебра. Исчисление кортежей

Реляционная модель данных согласно трактовке Кристофера Дейта состоит из трех частей: структурной, целостной и манипуляционной.

Структурная часть описывает из каких объектов состоит реляционная модель. Основной структурой данных в реляционной модели является нормализованные n-мерные отношения и основными понятиями структурной части реляционной модели является:

- *Тип данных* — множество значений и операций над ними. (понятие такое же как и в языках программирования);
- *Домен* можно считать уточнением типа данных и рассматривать как подмножество значений некоторого типа данных, имеющий определенный смысл. Характеризуется следующими свойствами:
  - имеет уникальное имя в пределах базы данных;
  - определен на некотором типе данных или на другом домене;
  - может иметь логическое условие, позволяющее описать подмножество данных, доступных для данного домена;
  - несет определенную смысловую нагрузку

  Домен отражает семантику, определенной предметной области и может быть не сколько доменов совпадающих как подмножество, но с различным смыслом. Основное значения домена ограничивается сравнением.
- *Атрибут отношения* — это пара вида <имя_атрибута, имя_домена>, при этом имена атрибутов должны быть уникальны в пределах отношения, но могут совпадать с именем домена.
- *Схема отношения* — это именованное множество упорядоченных пар <имя_атрибута, имя_домена>.
- *Схема БД* — это множество именованных схем отношений.
- *Кортеж* — это множество упорядоченных пар <имя_атрибута, значение_атрибута>, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения.
- *Отношение* определенное на множестве из n доменов (не обязательно различных), содержит две части: заголовок (схему отношения) и тело (множество из m кортежей). Значения n и m называются соответственно степенью и кардинальностью отношения.

- Непустое подмножество множества атрибутов схемы отношения будет *потенциальным ключом* тогда и только тогда, когда оно будет обладать свойствами уникальности (в отношении нет двух различных кортежей с одинаковыми значениями потенциального ключа) и неизбыточности (никакое из собственных подмножеств множества потенциального ключа не обладает свойством уникальности).
- В реляционной модели по традиции один из потенциальных ключей должен быть выбран в качестве *первичного ключа*, а все остальные потенциальные ключи будут называться *альтернативными*.
- *Реляционная база данных* — это набор отношений, имена которых совпадают с именами схем отношений в схеме базы данных.

Целостностная часть описывает ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных. Это целостность сущностей и целостность внешних ключей.

Манипуляционная часть описывает два эквивалентных способа манипулирования реляционными данными - реляционную алгебру и реляционное исчисление.

Реляционная алгебра является основным компонентом реляционной модели, опубликованной Коддом, и состоит из восьми операторов, составляющих две группы по четыре оператора:

- Традиционные операции над множествами: объединение (UNION), пересечение (INTERSECT), разность (MINUS) и декартово произведение (TIMES). Все операции модифицированы, с учетом того, что их операндами являются отношения, а не произвольные множества.
- Специальные реляционные операции: ограничение (WHERE) , проекция (PROJECT), соединение (JOIN) и деление (DIVIDE BY).

Результат выполнения любой операции реляционной алгебры над отношениями также является отношением. Эта особенность называется свойством реляционной замкнутости. Утверждается, что поскольку реляционная алгебра является замкнутой, то в реляционных выражениях можно использовать вложенные выражения сколь угодно сложной структуры.

Исчисление существует в двух формах: исчисление кортежей и исчисление доменов. Основное различие между ними состоит в том, что переменные исчисления кортежей являются переменными кортежей (они изменяются на отношении, а их значения являются кортежами), в то время как переменные исчисления доменов являются переменными доменов (они изменяются на доменах, а их значения являются скалярами).

Выражение исчисления кортежей содержит заключенный в скобки список целевых элементов и выражение WHERE, содержащее формулу WFF ("правильно построенную форму-

лу"). Такая формула WFF составляется из кванторов (EXISTS и FORALL), свободных и связанных переменных, литералов, операторов сравнения, логических (булевых) операторов и скобок. Каждая свободная переменная, которая встречается в формуле WFF, должна быть также перечислена в списке целевых элементов.

1.4 Теория проектирования реляционных баз данных: функциональные зависимости, нормальные формы

Пусть

1.5 Теория проектирования хранилищ данных. Основные принципы построения. ETL и ELT процессы

Определение

1.6 Транзакции. Определение, свойства и уровни изоляции транзакций. Неблагоприятные эффекты, вызванные параллельным выполнением транзакци , и способы их устранения. Управление транзакциями и способы обработки ошибок

Свойства для n = 2

1.7 Блокировки. Определение, свойства, иерархии, гранулярность и взаимоблокировки, алгоритмы обнаружения взаимоблокировок

Определение

Блокировка — это механизм, с помощью которого компонент Database Engine синхронизирует одновременный доступ нескольких пользователей к одному фрагменту данных.

Свойства

1) гранулярностью (или размером блокировки)
2) режимом (или типом блокировки)
3) продолжительностью

Гранулярность

1.8 Журнализация. Операции журнала транзакций и его логическая и физическая архитектуры. Модели восстановления. Метаданные

Нет вопроса

1.9 Безопасность и Аудит. Ключевые понятия и участники системы безопасности. Модели управления доступом

Этого вопроса нет

1.10 MPP системы. Распределенное и колоночное хранение. Распределенные вычисления, модель MapReduce. Обеспечение отказоустойчивости.

Пусть

1.11 In-Memory базы данных. Преимущества и недостатки. Примеры использования

Учитывая равенство $P\{Y < y\} = F_Y(y)$, приходим к формуле **??**.

1.12 Инструкции языка описания данных, инструкции языка обработки данных, инструкции безопасности, инструкции управления транзакциями

Когда X_1, X_2 являются *независимыми случайными величинами, то есть их двумерная плотность распределения*

1.13 Объекты базы данных: функции, процедуры, триггеры и курсоры

1.14 Оптимизация запроса: индексы, партиционирование, сегментирование

1.14.1 Индексы

(материал на основе SQL Server)

Индекс – это объект базы данных, обеспечивающий дополнительные способы быстрого поиска и извлечения данных. Индекс может создаваться на одном или нескольких столбцах. Это означает, что индексы бывают простыми и составными. Если в таблице нет индекса, то поиск нужных строк выполняется простым сканированием по всей таблице. При наличии индекса время поиска нужных строк можно существенно уменьшить.

К недостаткам индексов следует отнести:

- дополнительное место на диске и в оперативной памяти,
- замедляются операции вставки, обновления и удаления записей.

В SQL Server (и, наверное, многих других СУБД) индексы хранятся в виде сбалансированных деревьев. Представление индекса в виде сбалансированного дерева означает, что стоимость поиска любой строки остается относительно постоянной, независимо от того, где находится эта строка.

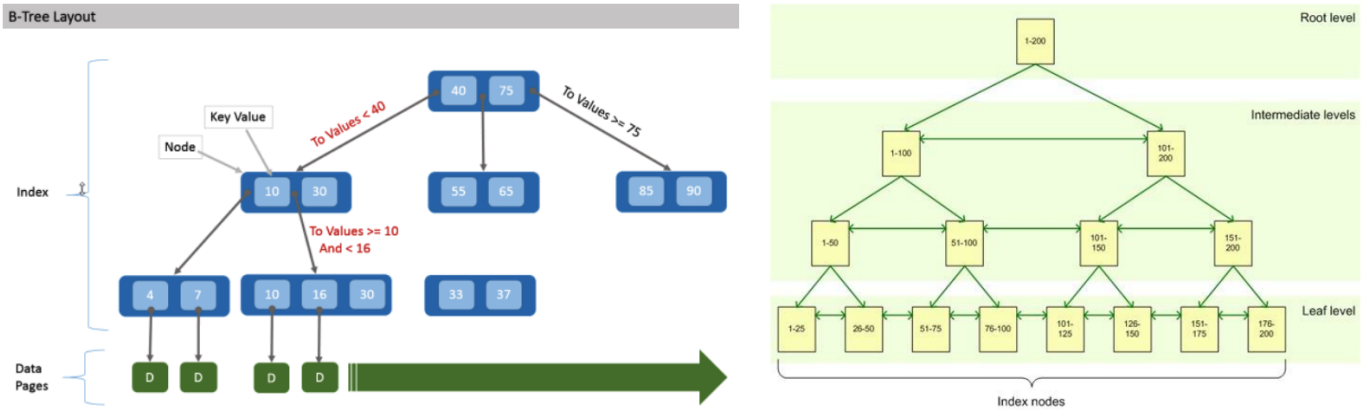

Рисунок 1 – B-Tree

Существует два типа индексов:

- Кластерные индексы
- Некластерные индексы, которые включают:
  - индексы на основе кучи;
  - индексы на основе кластерных таблиц.

Кластерные индексы

В кластерном индексе таблица представляет собой часть индекса, или индекс представляет собой часть таблицы в зависимости от вашей точки зрения. Листовой узел кластерного индекса – это страница таблицы с данными. Поскольку сами данные таблицы являются частью индекса, для таблицы может быть создан только один кластерный индекс. Кластерный индекс является уникальным индексом по определению.

Некластерные индексы на основе кучи

В листьях некластерного индекса на основе кучи хранятся указатели на строки данных. Указатель строится на основе идентификатора файла (ID), номера страницы и номера строки на странице. Весь указатель целиком называется идентификатором строки (RID).

Некластерные индексы, основанные на кластерных таблицах

В листьях некластерного индекса, основанного на кластерных таблицах, хранятся указатели на корневые узлы кластерных индексов. Поиск в таком индексе состоит из двух этапов:

- поиск в некластерном индексе;
- поиск в кластерном индексе.

Создание индекса

В SQL Server индексы создаются при помощи команды **CREATE INDEX**.

Помимо этого, индексы создаются при добавлении некоторых ограничений. Такой тип индексов часто называют «связанными индексами». Связанные индексы создаются при добавлении одного из следующих двух типов ограничений:

- ограничения первичного ключа (PRIMARY KEY);
- ограничения уникальности (UNIQUE).

1.14.2 Партиционирование

(материал на основе PostgreSQL)

Партиционирование – это метод разделения больших (исходя из количества записей, а не столбцов) таблиц на много маленьких.

Для произведения партиционирования нужно решить, каким будет ключ партиционирования – другими словами, по какому алгоритму будут выбираться партиции. Есть пара наиболее очевидных:

1) партиционирование по дате – например, выбирать партиции, основываясь на годе, в котором пользователь был создан.

   Достоинства:

   – легко понять;

   – количество строк в данной таблице будет достаточно стабильным.

   Недостатки:

   – требует поддержки – время от времени нам придётся добавлять новые партиции;

   – поиск по имени пользователя или id потребует сканирования всех партиций;

2) партиционирование по диапазону идентификаторов – например, первый миллион пользователей, второй миллион пользователей, и так далее

   Достоинства:

   – легко понять;

   – количество строк в данной таблице будет абсолютно стабильным.

   Недостатки:

   – требует поддержки – время от времени нам придётся добавлять новые партиции;

   – поиск по имени пользователя потребует сканирования всех партиций;

3) партиционирование по чему-нибудь другому – например, по первой букве имени пользователя.

   Достоинства:

   – легко понять;

   – никакой поддержки — есть строго определенный набор партиций и нам никогда не придется добавлять новые;

   Недостатки:

   – количество строк в партициях будет стабильно расти;

   – в некоторых партициях будет существенно больше строк, чем в других (больше людей с никами, начинающимися на «t*», чем на «y*»);

   – поиск по id потребует сканирования всех партиций;

4) есть еще пара других, не так часто используемых вариантов, вроде «партиционирования по хэшу от имени пользователя».

Пример

```
1    CREATE TABLE measurement (
2    city_id int not null,
3    logdate date not null,
4    peaktemp int,
```

```
5     unitsales int
6     ) PARTITION BY RANGE (logdate);
```

1.14.3 Сегментирование

(материал на основе Vertica)

Для распределения данных по кластерам используется их сегментация(Segmentation), а точнее сегментация проекций(Projection) в которых они находятся. Задача разработчика — подобрать такой список полей и/или такую функцию(например, хэш-функцию), благодаря которым данные равномерно распределятся по нодам кластера. HP Vertica рекомендует использовать встроенные функции HASH и MODULARHASH для этих целей.

Прим. Для справки: *K-Safety* — критерий отказоустойчивости БД, определяющий без какого кол-ва узлов БД сможет функционировать корректно. Он может быть равен 0, 1 или 2.

Для обеспечения K-Safety > 0 создаются сообщные проекции(Buddy Projection). Проекции называются сообщными, если они имеют в себе одинаковые наборы полей и одинаковое выражение сегментации, но хранятся на разных нодах. Сообщные проекции позволяют создать те самые реплики, которые позволяют работать БД в выбранном режиме K-Safety.

Пример

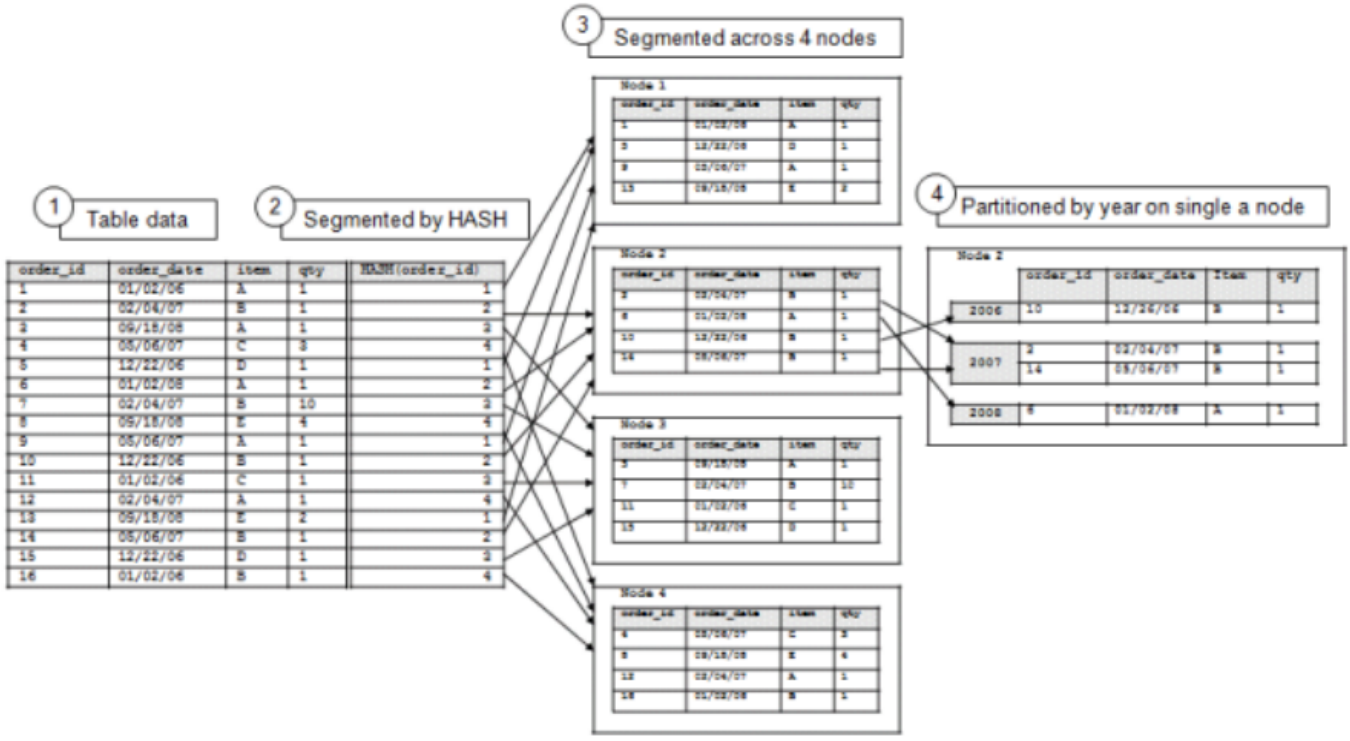

Рисунок 2 – Сегментирование и партиционирование вместе

1.15 План запроса. Этапы выполнения запроса

Пусть X — случайная величина.