1. Концепция баз данных
2. Не реляционные модели данных. Примеры реализации.
3. Реляционная модель данных. Таблицы, записи, поля, связи.
4. Базы данных и СУБД. Определение реляционной СУБД.
5. Основные функции СУБД
6. Транзакции. Свойства транзакции.
7. Журналирование. Зачем нужны журналы транзакций?
8. Восстановление СУБД после сбоев.
9. Архитектура СУБД. Взаимодействие СУБД с клиентом.
10. Преимущества трезвенной архитектуры и область ее применения.
11. Использование индексов и основные сведения о индексах.
12. Пользователи, роли и разграничение прав доступа. Предопределенные роли пользователей СУБД.
13. Удаление записей и целостность базы данных.
14. Триггеры и их использование в СУБД
15. Ключи и атрибуты. Суррогатные ключи как идеальные первичные ключи.
16. BI системы. Путь данных от получения до анализа.
17. Хранимые (встроенные) процедуры в СУБД. Типы хранимых процедур.
18. Обеспечение живучести и отказоустойчивости. Копирование и репликация.
19. Что такое биткойн
20. Какая БД используется в биткойне
21. Где и почему выгодно применение технологии блокчейн
22. Критерии распределенности (по К. Дейту)
23. Методы поддержки распределенных данных
24. Типы табличной фрагментации
25. Репликация БД (общая схема)
26. Репликация с основной копией и её варианты
27. Репликация без основной копии и её варианты
28. Поддержка распределенных БД. Шардинг
29. Протокол двухфазной фиксации
30. Варианты реализации протокола 2ФФ
31. Технология Oracle Streams
32. Стихийные угрозы информационной безопасности
33. Злоумышленные угрозы информационной безопасности
34. Простые действия для обеспечения безопасности IT-систем

1-Концепция баз данных

lec.1 sli.10

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Database - a set of data stored in accordance with the data scheme, which is manipulated in accordance with the rules of data modeling tools.

Концепция баз данных

1-Отчуждение данных от программ

2-Хранение описания данных вместе с самими данными

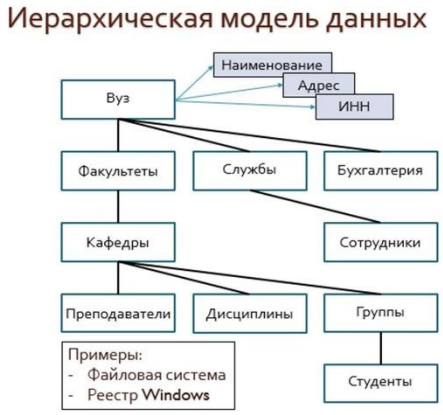
3-Отчуждение данных от носителей

4-Поддержание баз данных в согласованном и актуальном состоянии

5-Защита информации от сбоев аппаратуры

6-Поддержка многопользовательской работы

2-Не реляционные модели данных. Примеры реализации.



У иерархической системы главный признак – один родитель! Т.е. , по сути - все связи односторонние

- Иерархическая модель данных

Примеры:

– Файловая система
– Реестр Windows

Сетевая модель данных
Пример:
– Человеческий мозг
– Криптовалюта

Alaa

NoSQL — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД

Применяется к базам данных, в которых делается попытка решить проблемы масштабируемости и доступности за счёт атомарности и согласованности данных

Характеристики NoSQL баз данных:

1. Не используется SQL
2. Неструктурированные (schemaless)
   – в NoSQL базах в отличие от реляционных структура данных не регламентирована
   – Схема данных отслеживается в коде приложений
3. Представление данных в виде агрегатов
   – Сущности объединены в объекты базы данных как удобно (как попало)
4. Слабые ACID свойства.
5. Распределенные системы, без совместно используемых ресурсов

Теорема CAP (известная также как теорема Брюера) — эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

– согласованность данных (англ. consistency) — во всех вычислительных узлах в один момент времени данные не противоречат друг другу;
– доступность (англ. availability) — любой запрос к распределённой системе завершается корректным откликом;
– устойчивость к разделению (англ. partition tolerance) — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

3-Реляционная модель данных. Таблицы, записи, поля, связи.

Lec.1 Sli.15

Dennis

Реляционная модель - совокупность данных, состоящая из набора двумерных таблиц, в которой все данные доступные пользователю, организованы в виде набора связанных двумерных таблиц, а все операции над данными сводятся к операциям реляционной алгебры.

Реляционная модель является удобной и наиболее привычной формой представления данных, так в настоящее время эта модель является фактическим стандартом, на который ориентируются практически все современные коммерческие СУБД.

На реляционной модели данных строятся реляционные базы данных. Базовой единицей данных в пределах реляционной модели является таблица.

В сравнении с иерархической и сетевой моделью данных, реляционная модель отличается более высоким уровнем абстракции данных.

Alaa

-Впервые термин "реляционная модель данных" появился в статье сотрудника фирмы IBM д-ра Кодда опубликованной 6 июня 1970г. Будучи математиком по образованию Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных может сводится к совокупности двумерных таблиц, которые он назвал отношениями -relation (англ.). Реляционной является БД, в которой все данные доступные пользователю, организованы в виде набора связанных двумерных таблиц, а все операции над данными сводятся к операциям реляционной алгебры.

-Таблица описывает отдельную сущность предметной области (объект или событие). У таблицы есть имя.

-Запись это данные о конкретном экземпляре объекта или события, каждая запись представляет собой набор значений, содержащихся в полях (запись это одна строка таблицы). У записи есть ключ.

-Поле это контейнер для хранения данных об атрибутах

(свойствах) сущностей предметной области, из полей складываются столбцы таблицы. У поля (столбца таблицы) есть имя.

-Связи между таблицами

Связи между таблицами устанавливаются при помощи специальных полей – ключей.

Первичный ключ – поле однозначно идентифицирующее запись в таблице (значение этого поля уникально в столбце)

Внешний ключ – поле, в котором содержится значение первичного ключа другой таблицы, это поле необходимо для создания связи между записями таблиц.

Типы связей

- Один к одному: одной строке таблицы соответствует не более одной строки в другой таблице (группа - староста)
- Один ко многим: одной строке таблицы может соответствовать несколько строк в другой таблице (группа - студент)
- многие ко многим: одна строка первой таблицы связана с несколькими строками второй таблицы, и одна строка второй таблицы связана с несколькими строками первой (предмет преподаватель)

В реляционной базе данных бывают только связи 1:M

4-Базы данных и СУБД. Определение реляционной СУБД.

Lec.1 Sli.11

База данных — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения,о беспечивающиху правлениес озданиеми использованием баз данных.

Реляционная СУБД (или РСУБД) - система управления реляционными БД, СУБД такого типа используют структуры (таблицы) для хранения и работы с данными.,

Популярные РСУБД

- SQLite: очень мощная встраиваемая РСУБД.
- MySQL: самая популярная и часто используемая РСУБД.
- PostgreSQL: самая продвинутая и гибкая РСУБД.

5-Основные функции СУБД

Lec.2 sli.19

1-Управление транзакциями.

- Транзакция — это процесс, который переводит базу данных из одного согласованного состояния, в другое согласованное состояние. Допускается, что в процессе транзакции согласованность может нарушаться, но извне транзакции этого не видно.

2-Управление блокировками и клинчами 3-Управление буферами оперативной памяти.

4-Ведение журнала изменений в БД.

- Все значения данных до выполнения транзакций, приведших к изменениях в БД, также как и сами транзакции, записываются в специальных журналах СУБД.

5-Ведение словаря БД.

- Структура Базы данных – тоже данные!
- Структура Базы данных тоже может храниться в таблицах
- Хранимые процедуры на разных языках
- А еще , например, коды и расшифровки ошибок исполнения SQL
- А еще имена и пароли пользователей
- И, конечно, таблица DUAL

6-Обеспечение целостности и безопасности БД.

7-Поддержка языков БД.

- SQL
- Процедурное расширение (PL SQL)
- Прочие языки программирования

8-Управление данными во внешней памяти.

- Файлы и «сырые диски»
- Устройства, луны и экстенты
- Добавление, изменение и фрагментация
- Размещение и последовательное чтение
- Кеширование, буферизация, тирринг

6-Транзакции. Свойства транзакции.

Lec.2 Sli.20

Транзакция— группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, либо не выполнена вообще

(правильное определение)

Транзакция — это процесс, который переводит базу данных из одного согласованного состояния, в другое согласованное состояние. Допускается, что в процессе транзакции согласованность может нарушаться, но извне транзакции этого не видно

Dennis

Транзакция — логическая единица работы, состоящая из одного или нескольких операторов манипулирования данными (чтения, удаления, вставки, обновления), которую СУБД рассматривает и обрабатывает как неделимое действие, переводящее БД из одного целостного состояния в другое целостное состояние

ACID, или свойства транзакции

1 Atomic - атомарность. Транзакция — это неделимая единица, которая должна быть либо выполнена, либо отменена.

2 Coordination - согласованность. Смысл транзакции состоит в том, чтобы база данных переходила из одного согласованного состояния в другое.

3 Insulativity - изолированность. Каждая транзакция, которая выполняется, не зависит от остальных. Все результаты одного процесса, доступные в промежутках, не должны быть видны другим транзакциям.

4 Duration - надежность. Все результаты, которые были достигнуты в ходе успешной транзакции, наверняка сохраняются в базе данных.

7-Журналирование. Зачем нужны журналы транзакций?

Lec.3 Sli.21

-Журналирование

Все значения данных до выполнения транзакций, приведших к изменениях в БД, также как и сами транзакции, записываются в специальных журналах СУБД.

Зачем нужны журналы транзакций?

-Анализ работы СУБД и действий пользователей

-Репликация данных

-Восстановление данных после сбоев

8-Восстановление СУБД после сбоев.

Lec.3 Sli.26

Причины и последствия

-Пропало питание

Утеряно содержимое оперативной памяти

-Оплошность пользователя Утеряны данные на устройстве долговременного хранения

-Глобальная катастрофа

Потерян целиком ЦОД

крах оперативной памяти

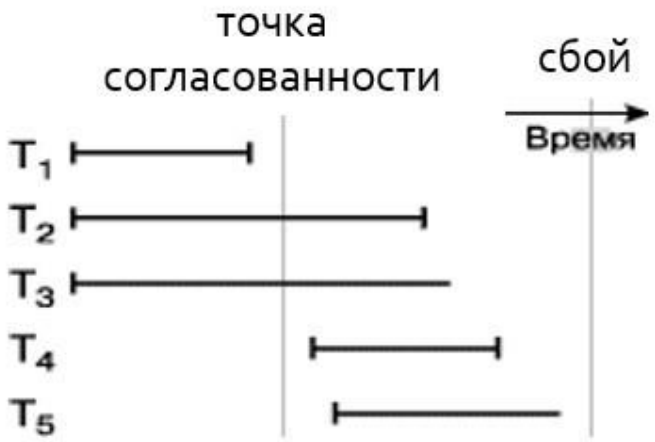
Самовосстановление

-Загрузка ближайшей точки согласованности

-Откат транзакций незафиксированных в долговременной памяти

(журнал REDO)

-Накат транзакций из журнала REDO



WINDOWS failed. Basically, the system recovers itself.

- No action is required for a T1 transaction. It ended until tlpc, and all its results are reflected in the external memory of the database.

-For transaction T2, the rest of the operations (redo) must be reexecuted. Indeed, in the external memory, there are no traces of operations that were performed in the T2 transaction after the tlpc time. Consequently, repeated direct interpretation of T2 operations is correct and will lead to a logically consistent state of the database (since transaction T2 was successfully completed before the moment of a soft failure, the log contains records of all changes made by this transaction). - For transaction T3, the first part of the operation (undo) must be performed in the opposite direction. Indeed, in the external memory of the database there are completely no results of T3 operations that were performed after the time tlpc. On the other hand, in the external memory there are guaranteed results of T3 operations that were performed before the time tlpc. Consequently, the inverse interpretation of T3 operations is correct and will lead to a consistent state of the database (since T3

transaction was not completed by the time of the soft failure, during recovery it is necessary to eliminate all the consequences of its execution).

-For transaction T4, which managed to start after the time tlpc and end before the time of the soft failure, you need to perform a full re-direct interpretation of operations (redo).

-Finally, for the tlpc that started after the tlpc moment and didn't have time to complete by the time of the T5 transaction's mild failure, no action is required. The results of operations of this transaction are completely absent in the external memory of the database.

Восстановление данных после потери (искажения) данных на устройствах долговременной памяти

Ручное восстановление

-Загрузка данных с копии

-При необходимости правка журнала Archived log

-Накат транзакций из журнала Archived log

Восстановление данных после сбоев (ошибки неизвестного происхождения) Здесь правил нет!

9-Архитектура СУБД. Взаимодействие СУБД с клиентом.

Lec.3 Sli.37

Архитектура. В среде СУБД можно выделить след. пять основных компонентов.

1-Аппаратное обеспечение. Одни СУБД предназначены для работы только с конкр типами ОС или оборудования, другие могут работать с широким кругом аппаратного обеспечения и различными ОС. Для работы СУБД обычно требуется некоторый минимум оперативной и дисковой памяти, но ее может быть недостаточно для достижения приемлемой производительности системы.

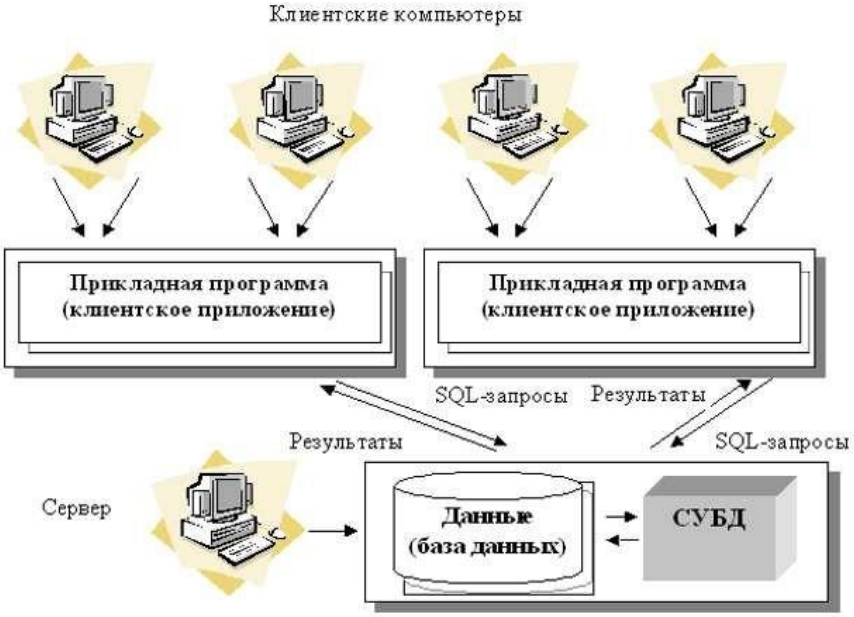
2-Программноео беспечение.Этотк омпонентв ключает операционную систему, программное обеспечение самой СУБД, прикладные программы, включая и сетевое программное обеспечение, если СУБД используется в сети. Обычно приложения создаются на языках третьего поколения, таких как C, COBOL, Fortran, Ada или Pascal, или на языках четвертого поколения, таких как SQL, операторы которых внедряются в программы на языках третьего поколения.

3-Данные – наиболее важный компонент с точки зрения конечных пользователей. База данных содержит как рабочие данные, так и метаданные, т.е. "данные о данных".

4-Процедуры, к которым относят инструкции и правила, которые должны учитываться при проектировании и использовании базы данных: регистрация в СУБД; использование отдельного инструмента СУБД или приложения; запуск и останов СУБД; создание резервных копий СУБД; обработка сбоев аппаратного и программного обеспечения, включая процедуры идентификации вышедшего из строя компонента, исправления отказавшего компонента (например, посредством вызова специалиста по ремонту аппаратного обеспечения), а также восстановления базы данных после устранения неисправности; изменение структуры таблицы, реорганизация базы данных, размещенной на нескольких дисках, способы улучшения производительности и методы архивирования данных на вторичных устройствах хранения.

5-Пользователи: клиенты БД, администратор БД, прикладн. программисты.

Как СУБД общаются с клиентами. Многопользовательская работа



Клиент-серверная архитектура

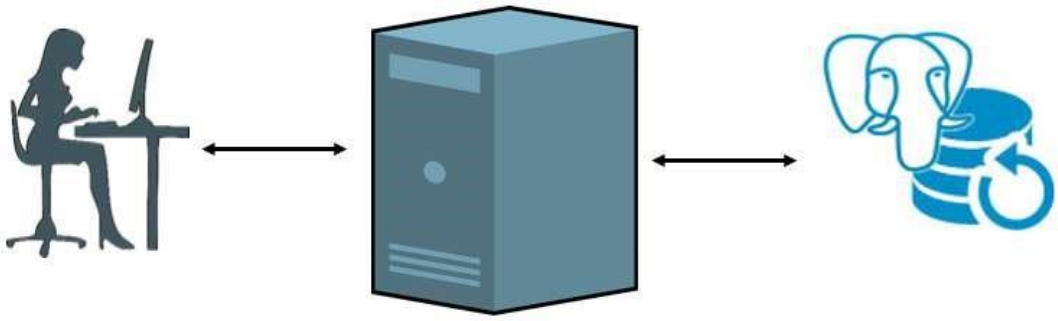
клиент **сервер**

Тонкий клиент - реализуется на базе WEB - приложения

Толстый клиент или Rich-клиент - отдельное приложение

10-Преимущества трехзвенной архитектуры и область ее применения.

Lec.3 sli.39



Это архитектура предполагает три компонента: клиент, сервер приложений (к которому подключено клиентское приложение) и сервер баз данных (с которым работает сервер приложений).

1 Слой клиента

Самый верхний уровень приложения с интерфейсом пользователя. главная функция интерфейса представление задач и результатов, понятных пользователю.

2 Слой логики

Этот слой координирует программу, обрабатывает команды, выполняет логические решения и вычисления, выполняет расчеты. Она также перемещается и обрабатывает данные между двумя окружающими слоями.

3 Слой данных

Здесь хранится информация и извлекается из базы данных и файловой системы. Информация отправляется в логический слой для обработки и в конечном счете возвращается пользователю.

Преимущества трехзвенной архитектуры

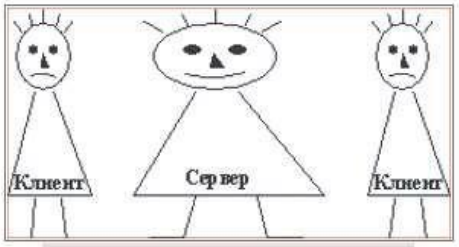
- Простота модификации
- Простота расширения
- Простота интеграции
- Повышение безопасности
- Возможность работы тонкого клиента
- Низкая стоимость внедрения
- Очень простая поддержка
- Независимость от ОС
- Доступность из любой точки мира

Понятие информационной банковской системы (ИБС), - совокупность информационных технологий, используемых в банке в каждый момент времени, автоматизирующих полностью или частично выполняемые предметные технологии, характерные для данного периода развития информатики. *

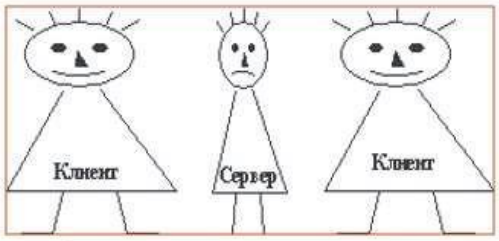
ИБС – совокупность банковских ИТ совместно с обеспечивающими ИТ.*

Взаимодействие в ИБС

- Ориентация на работу большого количества пользователей
- Клиент-серверная архитектура ИБС



Модель «тонкого» клиента

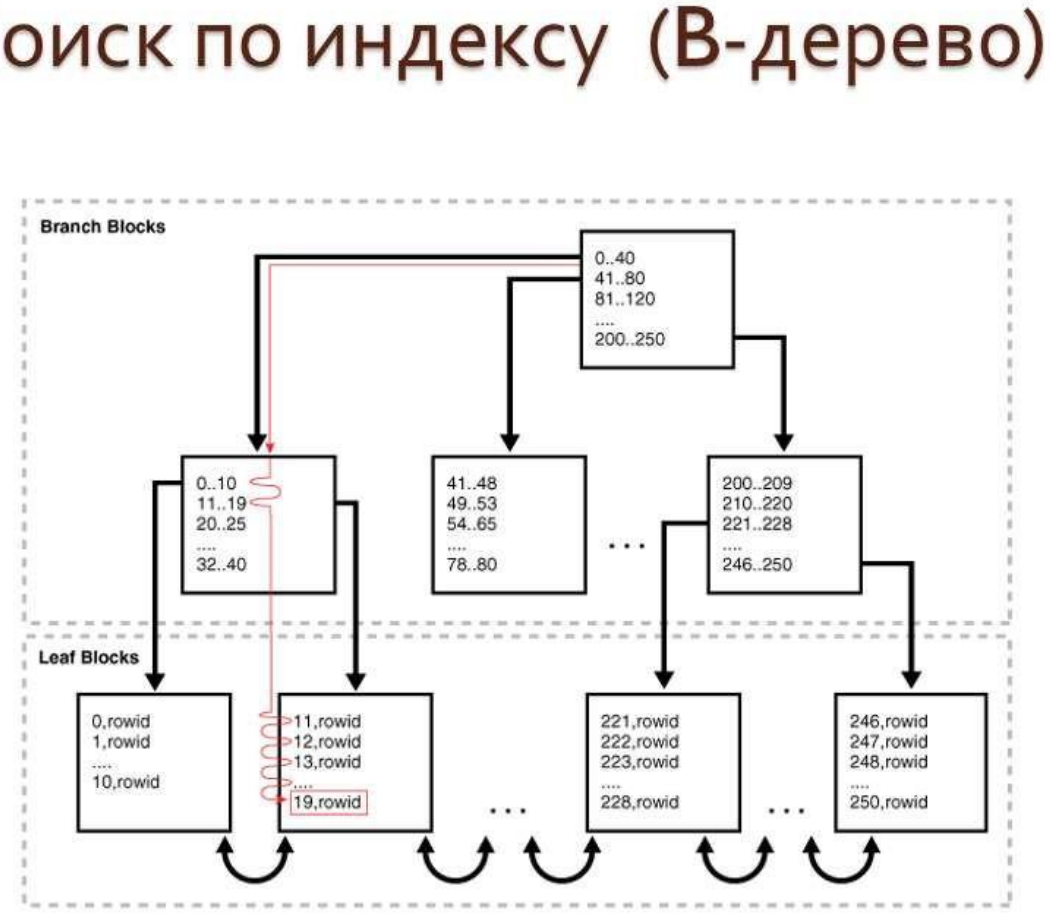


Модель «толстого» клиента

11-Использование индексов и основные сведения о индексах.

Lec.5 Sli.68

Индекс (англ. index) — объект БД, который создается для повышения производительности поиска данных. Индексы создаются для организации (ускорения) поиска и сортировки записей таблицы



-Индексы создаются для ускорения поиска
-Индексы замедляют все операции кроме выборки
-Индексы занимают место
-Для маленьких таблиц индексы не нужны
-Для составного индекса важен порядок полей в индексе
-Для временных таблиц индексы не нужны
-Для таблиц которые часто обновляются используйте м коажкно меньше индексов

12-Пользователи, роли и разграничение прав доступа. Предопределенные роли пользователей СУБД.

Lec.5 Sli.56

Что такое разграничение доступа (What is access control)

Дискреционное управление доступом (discretionary access control) — разграничение доступа между поименованными субъектами и поименованными объектами. Под субъектами здесь понимаются пользователи СУБД, а под объектами – объекты БД, такие как таблицы, представления, индексы и т.д., под доступом понимается возможность использования данных из объектов БД (содержимое таблиц), так и использование самих объектив (индексы, хранимые процедуры и т.д.). Т.е. каждый пользователь имеет конкретные права на работу с конкретными объектами БД, например, доступ только по чтению к таблицам «Страны», «Города» и «Улицы» или доступ к созданию, изменению и удалению всех объектов БД «Адреса».

Роли пользователей СУБД

DBMS user roles

1-CONNECT — конечные пользователи. По умолчанию им разрешено только соединение с базой данных и выполнение запросов к данным.

2-RESOURCE — привилегированные пользователи, обладающие правом создания собственных объектов в базе данных (таблиц, представлений, синонимов, хранимых процедур). Пользователь — владелец объекта обладает полным набором привилегий для управления данным объектом.

RESOURCE — работает с объектами БД, создает, модифицирует и удаляет объекты БД.

3-DBA — категория администраторов базы данных. Включает возможности обеих предыдущих категорий, а также возможность вводить (удалять) в систему (из системы) субъекты защиты или изменять их категорию.

DBA — работает с пользователями, их схемами, табличными пространствами и другими объектами СУБД.

13-Удаление записей и целостность базы данных.

Lec.5 Sli.44,72

-Никогда не удаляйте из базы данных записи!!!

-Почти всегда удаление данных это неправильное действие!

Что же делать с таблицами, которые все растут и растут? НАДО ИХ СЕГМЕНТИРОВАТЬ!

- Секционирование таблиц

Секционирование (англ. partitioning) — разделение хранимых объектов баз данных на отдельные части с раздельными параметрами физического хранения Обеспечения целостности данных

Целостность (согласованность) БД - соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам.

-Сущностная целостность

-Доменная целостность

-Ссылочная целостность

-Пользовательская целостность

DENNIS

Оператор DELETE удаляет данные;

DELETE

[FROM] {имя_таблицы | ONLY (имя_таблицы)} [псевдоним]

[WHERE условие_поиска [{AND | OR} условие_поиска[, ...]]

В различной литературе довольно большое место уделяется процессу удаления данных и возникающим вследствие этого нарушениям ссылочной целостности базы данных. Рассматриваются многочисленные методы и способы приведения БД к согласованному состоянию после удаления данных.

В противовес этим источникам необходимо со всей категоричностью заявить, что из БД ничего удалять нельзя. Любое удаление данных из БД приводит к невозможности получения аналитических отчетов за период, в котором были удалены данные. Если из базы данных кафедры ВУЗА удалить, например, отчисленных студентов, то нельзя будет получить списки групп, статистику посещаемости и успеваемости за прошлые годы. Исходя из невозможности удаления данных, надо предусмотреть методы поддержания изменений состояния сущностей, атрибутов и даже связей между сущностями в процессе жизни и развития ИС (см. раздел «Поддержка историчности»), а также методы ускорения работы БД при постоянном росте ее объемов (см. разделы «Секционирование таблиц» и «Кластеризация таблиц»). В редких случаях в процессе эксплуатации ИС приходится удалять некоторые данные, но это делается вне ИС, администратором БД и последствия этих действий целиком ложатся на его плечи.

Конечно, тезис «из БД ничего удалять нельзя» не относится к процессу разработки и отладки ИС, в эти периоды жизненного цикла удаление данных из БД – обычное дело, только надо помнить о согласованности данных.

из БД ничего удалять нельзя. Любое удаление данных из БД приводит к невозможности получения аналитических отчетов за период, в котором были удалены данные.

Исходя из невозможности удаления данных, надо предусмотреть методы поддержания изменений состояния сущностей, атрибутов и даже связей между сущностями в процессе жизни и развития ИС, а также методы ускорения работы БД при постоянном росте ее объемов.

В редких случаях в процессе эксплуатации ИС приходится удалять некоторые данные, но это делается вне ИС, администратором БД

Целостность (согласованность) БД - соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам.

- Сущностная целостность - Поддержание сущностной целостности состоит в недопущении многократного попадания данных об одной сущности в таблицу БД. Если использовать «естественные» уникальные ключи, такие как номер паспорта или СНИЛС то сущностная целостность поддерживается автоматически. При использовании суррогатных

первичных ключей добиться этого можно используя уникальные «естественные» вторичные ключи, такие как, уже упомянутые, номер паспорта или номер зачетки и т.д.

- Доменная целостность - это достоверность записей в конкретном столбце. Она включает в себя ограничения типа данных, ограничения формата при помощи ограничений CHECK и правил, а также ограничения диапазона возможных значений при помощи ограничений FOREIGN KEY, CHECK, DEFAULT, определений NOT NULL и т.д.
- Ссылочная целостность - гарантирует согласованность значений первичных и внешних ключей во всех таблицах. Обычно поддерживается триггерами, автоматически заполняющими значение ключей или обеспечивающие невозможность их изменения.
- Пользовательская целостность - Определяется бизнесправилами, не входящие ни в одну из категорий целостности, эти правила определяются и отслеживаются не на уровне СУБД, а на уровне ИС. Например, баланс банка должен сходится или число проданных и непроданных билетов на авиарейс должно в сумме равняться вместимости самолета. Пользовательская целостность поддерживается при помощи транзакций и использования триггеров.

14-Триггеры и их использование в СУБД

Lec.5 Sli.64

Триггер - хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнеслогики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера.

Зачем использовать триггеры

-Для автоматической генерации значений полей
-Для логирования при ограничении доступа
-Для сбора статистики
-Для предотвращения dml операций
-Для реализации сложных ограничений целостности данных
-Для организации всевозможных видов аудита
-Для оповещения других модулей о том, что делать в случае изменения информации в БД
-Для реализации бизнес логики
-Для организации каскадных воздействий на таблицы БД

15-Ключи и атрибуты. Суррогатные ключи как идеальные первичные ключи.

Lec.5 Sli.33

Главный принцип определения сущностей и атрибутов

Принцип DRY. (Don’t repeat yourself, DRY (рус. не повторяйся) — это принцип разработки программного обеспечения, нацеленный на снижение повторения информации различного рода, особенно в системах со множеством слоёв абстрагирования. Принцип DRY формулируется так:

«Каждая часть данных должна иметь единственное,
непротиворечивое и авторитетное представление в рамках системы»

Связи между таблицами устанавливаются при помощи специальных полей – ключей.

- Первичный ключ – поле однозначно идентифицирующее запись в таблице (значение этого поля уникально в столбце)
- Внешний ключ – поле, в котором содержится значение первичного ключа другой таблицы, это поле необходимо для создания связи между записями таблиц.

Определение ключей

Первичным ключом может быть только суррогатный ключ. Обычно это счетчики (последовательности), которыми СУБД автоматически заполняет ключевые поля при добавлении записи.

Ключи и индексы

Ключи служат для организации связей между таблицами

Индексы создаются для организации (ускорения) поиска и сортировки записей таблицы

Три основных вопроса про ключи и записи:

1-Может ли внешний ключ принимать неопределённые значения (NULL)?

Конечно может!!!

Например в учебном плане есть дисциплина, но мы еще не знаем, кто будет ее читать!

2-Что должно происходить при ИЗМЕНЕНИИ первичного ключа целевой сущности, на которую ссылается внешний ключ?

База данных развалится!!!

НО, поскольку мы определяем первичные ключи только как суррогатные (т.е. это не атрибут сущности), то меняться они не будут!!!

3-Что должно случиться при попытке УДАЛЕНИЯ целевой сущности, на которую ссылается внешний ключ?

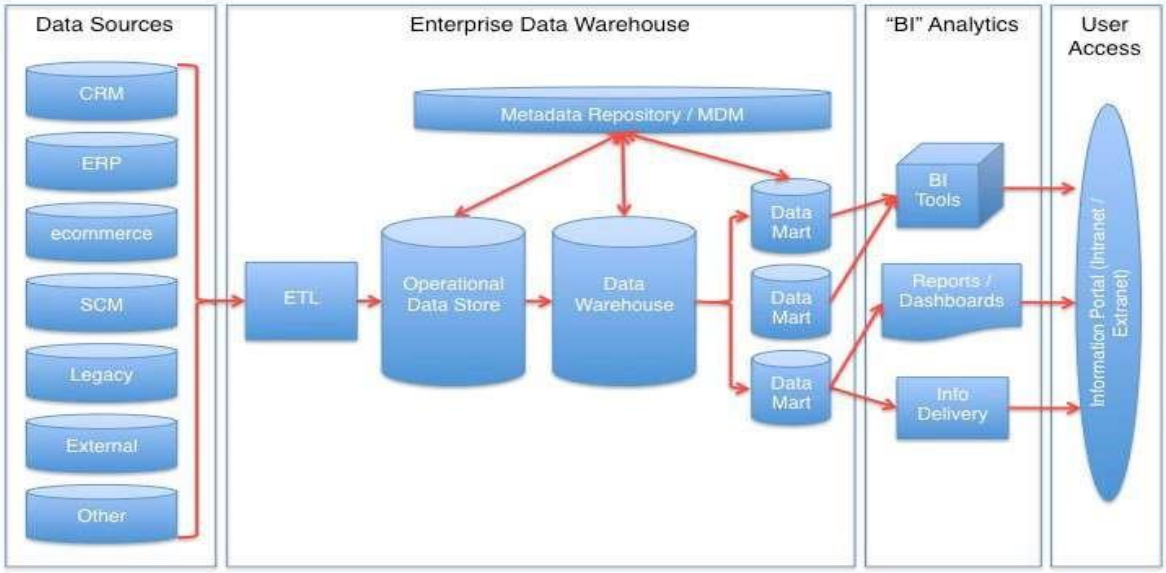
База данных развалится!

Никогда не удаляйте из базы данных записи!!!

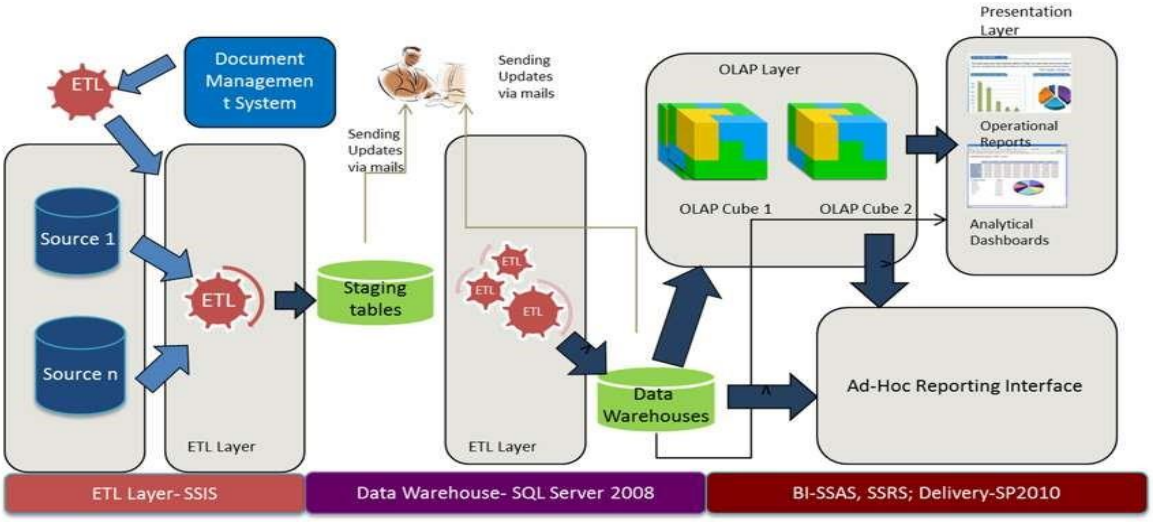
16-BI системы. Путь данных от получения до анализа.

Lec.6 Sli.1

Business intelligence (BI) — это методы и инструменты для перевода необработанной информации в осмысленную, удобную форму. Эти данные используются для бизнес-анализа.



Великий путь данных (BI)



Этапы и части BI:

1-Информационный поиск

2-Трансформация и очистка данных

3-Аналитическая обработка в реальном времени (OLAP),

4-Инструменты предупреждения об отклонениях от ожидаемых показателей

4-Бизнес-аналитика

5-Бизнес-отчётность

17-Хранимые (встроенные) процедуры в СУБД. Типы хранимых процедур.

Lec.5 Sli.62

Типы встроенных процедур

Types of stored procedures

1-Процедуры обработки ошибок

2-Процедуры обеспечивающие целостность и согласованность БД

3-Процедуры реализующие особо сложные запросы

4-Триггеры

Триггер - хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой инициируется СУБД при попытке изменения данных в таблице, с которой он связан. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Все производимые модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера.

Зачем нужны Триггеры

- Для автоматической генерации значений полей
- Для логирования при ограничении доступа
- Для предотвращения dml операций
- Для реализации сложных ограничений целостности данных
- Для организации всевозможных видов аудита и сбора статистики
- Для оповещения других модулей о том, что делать в случае изменения информации в БД
- Для реализации бизнес логики
- Для организации каскадных воздействий на таблицы БД

18-Обеспечение живучести и отказоустойчивости. Копирование и репликация.

Lec. Sli.

Репликация – это поддержание двух и более идентичных копий (реплик) данных на разных узлах РБД (Реляционная база данных).

Реплика может включать всю базу данных (полная репликация), одно или несколько взаимосвязанных отношений или фрагмент отношения.

Достоинства репликации:

- повышение доступности и надежности данных; • повышение локализации ссылок на реплицируемые данные.

Недостатки репликации:

- сложность поддержания идентичности реплик;
- увеличение объема памяти для хранения данных.

Поддержание идентичности реплик называется распространение изменений и реализуется службой тиражирования.

Служба тиражирования должна выполнять следующие функции:

- Обеспечение масштабируемости, т.е. эффективной обработки больших и малых объемов данных.
- Преобразование типов и моделей данных (для гетерогенных РБД).
- Репликация объектов БД, например, индексов, триггеров и т.п.
- Инициализация вновь создаваемой реплики.
- Обеспечение возможности "подписаться" на существующие реплики, чтобы получать их в определенной периодичностью.

Репликация с основной копией

Существуют следующие варианты:

- Классический подход заключается в наличии одной основной копии, в которую можно вносить изменения; остальные копии создаются с определением read only.
- Асимметричная репликация: основная копия фрагментирована и распределена по разным узлам РБД, и другие узлы могут являться подписчиками отдельных фрагментов (read only).
- Рабочий поток. При использовании этого подхода право обновления не принадлежит постоянно одной копии, а переходит от одной копии в другой в соответствии с потоком операций. В каждый момент времени обновляться может только одна копия.
- Консолидация данных:

Репликация без основной копии

Симметричная репликация (без основной копии). Все копии реплицируемого набора могут обновляться одновременно и независимо друг от друга, но все изменения одной копии должны попасть во все остальные копии.

Существует два основных механизма распространения изменений при симметричной репликации:

- синхронный: изменения во все копии вносятся в рамках одной транзакции;
- асинхронный: подразумевает отложенный характер внесения изменений в удаленные копии.

19-Что такое биткойн

Lec.8 sli.4

Биткойн—это способ передачи стоимости от одного лица к другому без участия третьей стороны, посредника (банка). Он сам по себе не имеет денежного эквивалента стоимости; он обладает не внутренней стоимостью, а очень высокой полезностью. По сути— это инструмент (метод, протокол) для осуществления транзакций— передачи некоторой стоимости от одного субъекта к другому. Это своего рода финансовый интернет—сеть, которая обеспечивает совершение безналичных платежей и расчетов.

Достоинства

- Анонимность – другой участник транзакции будет знать только ваш Bitcoin-адрес или QR-код. Другие данные не разглашаются.
- Децентрализованный характер системы – все участники сети равны и независимые.
- Безопасность – взлом кошельков, подмена данных, перехват переводов невозможны.
- Глобальность – биткойн позволяет быстро проводить транзакции между людьми с разных стран, часовых поясов.
- Майнинг – существует самостоятельный способ добычи BTC, рассматриваемый многими в качестве заработка.

Недостатки

- При высокой загрузке сети переводы замедляются, иногда до нескольких часов
- Регуляционная деятельность властей разных стран по отношению к криптовалютам неоднозначна
- Нестабильный курс – из-за этого потенциал BTC как платежной опции уменьшается
- Мошенничество – сам Bitcoin очень надежен. Однако неопытные люди могут попадаться на схемы развода: фейковые обменники, мошеннические инвестиционные проекты, вирусы

Криптовалюты

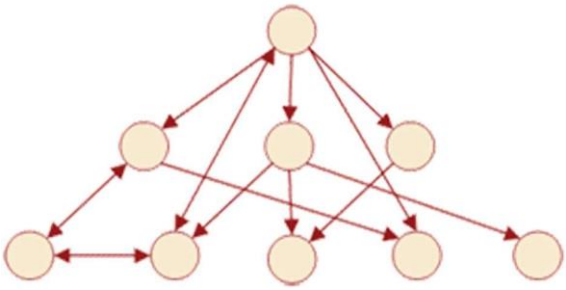
Ethereum - платформа для создания децентрализованных онлайнсервисов на базе блокчейна, работающих на базе умных контрактов.

20-Какая БД используется в биткойне

Lec.8 sli.

Это – сетевая модель БД

Сетевая модель данных — логическая модель данных, являющаяся расширением иерархического подхода. Сетевая модель представляет собой структуру, у которой любой элемент может быть связан с любым другим элементом.Сетевая база данных состоит из наборов записей, которые связаны между собой так, что записи могут содержать явные ссылки на другие наборы записей. Тем самым наборы записей образуют сеть. Связи между записями могут быть произвольными, и эти связи явно присутствуют и хранятся в базе данных.



21-Где и почему выгодно применение технологии блокчейн

Cryptocurrencies

Most cryptocurrencies use blockchain technology to record transactions. For example, the bitcoin network and Ethereum network are both based on blockchain.

Smart contracts

Blockchain-based smart contracts are proposed contracts that could be partially or fully executed or enforced without human interaction. One of the main objectives of a smart contract is automated escrow.Smart contracts based on blockchain technology might reduce moral hazards and optimize the use of contracts in general.

Banks

Major portions of the financial industry are implementing distributed ledgers for use in banking.

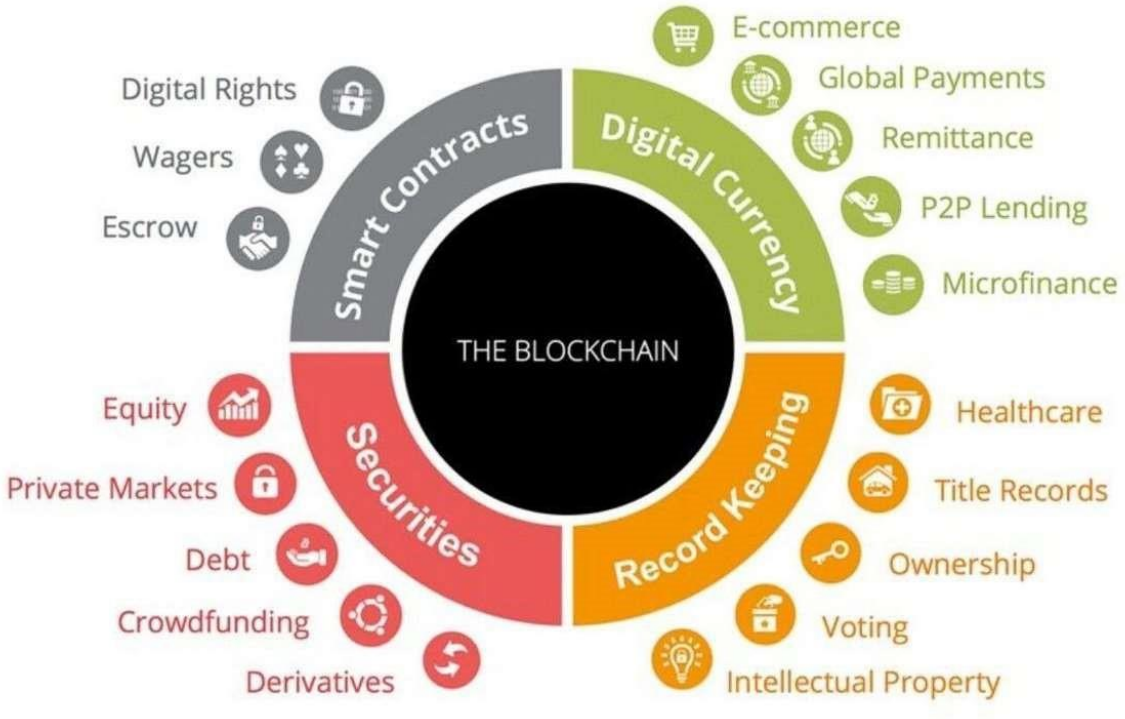
Banks are interested in this technology because it has potential to speed up back office settlement systems.

Blockchain with video games

Some video games are based on blockchain technology.

Other uses

Blockchain technology can be used to create a permanent, public, transparent ledger system for compiling data on sales, tracking digital use and payments to content creators, such as wireless users or musicians.



Smart Contracts

Smart contract code is code that is stored, verified and executed on a blockchain. Smart legal contracts is the use of the smart contract code that can be used as a complement, or substitute, for legal contracts. In its simplest form, this is about "if X happens, do Y," without any human intervention. Use cases include anything from rental cars or apartments, through to complex machine-to-machine transactions involving advanced analytics.

In October 2015, it was announced that Visa was working on a smart contract proof of concept (PoC) for car leases and rentals. Visa worked with DocuSign to create the solution with the aim of

reducing or completely eliminating paperwork that customers need to go through during the car rental process.

The Visa-DocuSign PoC involves registration of each car on the blockchain using a unique digital identity assigned to them. The car's digital identity is then connected to a Visa payment channel and also to the DocuSign platform.

Whenever a customer wants to rent a car, they can pick the car of their choice and finish the formalities online while sitting inside the car itself. Beyond selecting the type of car, they can search, compare and choose insurance plans as well as other in-car payments like tolls, parking fees, etc. It is possible to enable all of these services by authenticating preferences using a DocuSign electronic signature. The information will be updated on the blockchain against the car's unique identity, which can be readily accessed by the company.

https://www.aciworldwide.com/insights/expertview/2017/march/blockchain-for-retailers-producing-real-businessbenefits

---

Dennis

-Наследство и завещания - смарт-контракты для определения действительности воли и распределения инерции

-Туризм - идентификация пассажира, посадка на борт, паспорт, платежная и другая оцифрованная документация

-Недвижимость

-Общественный транспорт

-Музыкальная индустрия

-Медицина - целостность цепочки поставок лекарств, эффективность, конфиденциальность и владение данными о здоровье пациентов

-Медиа - контроль прав собственника, использование смартконтрактов для компенсации художнику

-Маркетинг - обходите посредников, обеспечивая более экономичную рекламу - Юриспруденция

-Право

-Интернет вещей - возможность для приложений IoT вносить транзакционные данные в blockchain

-Страхование

-Рынок труда - проверка анкетных данных: проверка личности, трудовой стаж

-Рынок оружия - отслеживание владения оружием

-Правительство и голосование - minimize government fraud, digitize most processes, integrity of citizen registry data

-Прогнозы

-Энергетика

-Образование - оцифровка, проверка академических полномочий

-Пожертвования - обеспечение аудита пожертвований для предотвращения мошенничества

-Кибербезопасность

-Кредитная история - сделать кредитные отчеты более точными, прозрачными и доступными

-Коммерческий транспорт

-Облачное хранение - снижение транзакционных издержек в децентрализованной сети

-Благотворительность

-Банки и финансы - Оптимизация обработки платежей с высокой эффективностью, быстротой и безопасностью транзакций

-Автомобильная отрасль

22-Критерии распределенности (по К. Дейту)

Lec.7 sli.3

1. Локальная автономность.
   - Локальные данные принадлежат локальным узлам и управляется администраторами локальных БД.
   - Локальные процессы в РБД остаются локальными.
   - Все процессы на локальном узле контролируются только этим узлом.
2. Отсутствие опоры на центральный узел.
   - В системе не должно быть узла, без которого система не может функционировать, т.е. не должно быть центральных служб.
3. Непрерывное функционирование.
   - Удаление или добавление узла не должно требовать остановки системы в целом.
4. Независимость от местоположения.
   - Пользователь должен получать доступ к любым данным в системе, независимо от того, являются эти данные локальными или удалёнными.
5. Независимость от фрагментации.
   - Доступ к данным не должен зависеть от наличия или отсутствия фрагментации и от типа фрагментации.
6. Независимость от репликации.
   - Доступ к данным не должен зависеть от наличия или отсутствия реплик данных.
7. Обработка распределенных запросов.
   - Система должна автоматически определять методы выполнения соединения (объединения) данных.
8. Обработка распределенных транзакций.
   - Протокол обработки распределённой транзакции должен обеспечивать выполнение четырёх основных свойств транзакции:
   - атомарность, согласованность, изолированность и продолжительность.
9. Независимость от типа оборудования.
   - СУРБД должна функционировать на оборудовании с различными вычислительными платформами.
10. Независимость от операционной системы.
    - СУРБД должна функционировать под управлением различных ОС.
11. Независимость от сетевой архитектуры.
    - СУРБД должна быть способной функционировать в сетях с различной архитектурой и типами носителя.
12. Независимость от типа СУБД.
    - СУРБД должна быть способной функционировать поверх различных локальных СУБД, возможно, с различными моделями данных (требование гетерогенности).

23-Методы поддержки распределенных данных

Lec.7 sli.5

1. Фрагментация – разбиение БД или таблицы на несколько частей и хранение этих частей на разных узлах РБД.

Фрагментация – основной способ организации РБД.

Назначение: хранение данных на том узле, где они чаще используются.

Основные проблемы, которые при этом возникают:

- прозрачность написания запросов к данным;
- поддержка распределенных ограничений целостности.

2. Репликация – создание и хранение копий одних и тех же данных на разных узлах РБД.

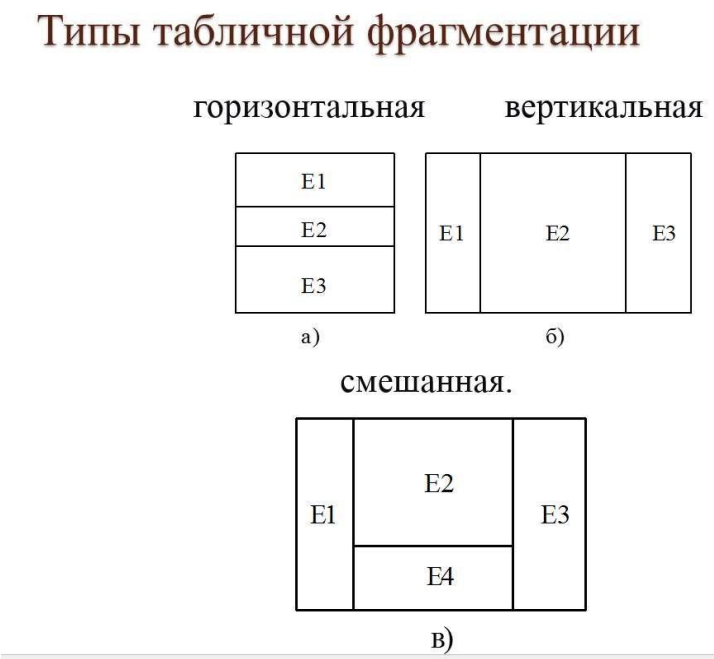
Это поддержание двух и более идентичных копий (реплик) данных на разных узлах РБД.

Реплика может включать всю базу данных (полная репликация), одно или несколько взаимосвязанных отношений или фрагмент отношения

3. Распределенные ограничения целостности – ограничения, для проверки выполнения которых требуется обращение к другому узлу РБД.
4. Распределенные запросы – это запросы на чтение, обращающиеся более чем к одному узлу РБД.
5. Распределенные транзакции – команды на изменение данных, обращающиеся более чем к одному узлу РБД.

24-Типы табличной фрагментации

Lec.7 sli.7



Горизонтальная фрагментация

Разбиение таблицы происходит за счет помещения в отдельные таблицы с одинаковой структурой уникальных групп строк.

Фактически осуществляется хранение строк одной логической таблицы в нескольких идентичных физических таблицах на различных узлах

- Каждый фрагмент хранится на отдельном узле.
- Каждый фрагмент имеет уникальные строки.
- Все уникальные строки имеют одинаковые столбцы

Вертикальная фрагментация

Разбиение таблицы происходит по столбцам. Одни столбцы формируют одну таблицу, другие — другую.

- Каждый фрагмент хранится на отдельном узле.
- Каждый фрагмент имеет уникальные столбцы — за исключением ключевого столбца, который имеется во всех фрагментах.

Смешанная фрагментация

Представляет собой комбинацию вертикальной и горизонтальной фрагментации. Таблица может разделяться на несколько горизонтальных строк, каждая из которых разделяется на множество столбцов.

25-Репликация БД (общая схема)

Lec.7 sli.8

Репликация – это поддержание двух и более идентичных копий (реплик) данных на разных узлах РБД.

Реплика может включать всю базу данных (полная репликация), одно или несколько взаимосвязанных отношений или фрагмент отношения.

Достоинства репликации:

— повышение доступности и надежности данных;

— повышение локализации ссылок на реплицируемые данные.

Недостатки репликации:

— сложность поддержания идентичности реплик;

— увеличение объема памяти для хранения данных.

Поддержание идентичности реплик называется распространение изменений и реализуется службой тиражирования.

26-Репликация с основной копией и её варианты

Lec.7 sli.11

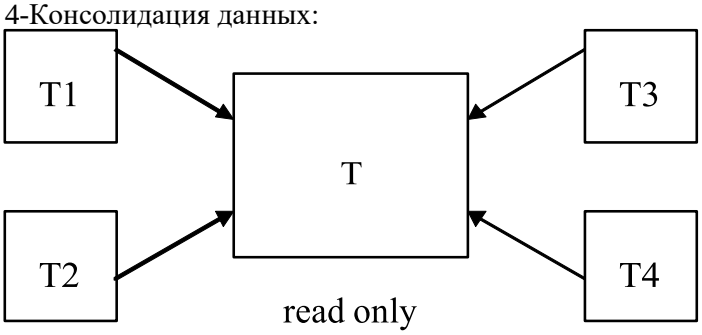
Существуют следующие варианты:

1-Классический подход заключается в наличии одной основной копии, в которую можно вносить изменения; остальные копии создаются с определением read only.

2-Асимметричная репликация: основная копия фрагментирована и распределена по разным узлам РБД, и другие узлы могут являться подписчиками отдельных фрагментов (read only).

3-Рабочий поток. При использовании этого подхода право обновления не принадлежит постоянно одной копии, а переходит от одной копии в другой в соответствии с потоком операций. В каждый момент времени обновляться может только одна копия.

4-Консолидация данных:



27-Репликация без основной копии и её варианты

Lec.7 sli.12

Симметричная репликация (без основной копии). Все копии реплицируемого набора могут обновляться одновременно и независимо друг от друга, но все изменения одной копии должны попасть во все остальные копии.

Существует два основных механизма распространения изменений при симметричной репликации:

- синхронный: изменения во все копии вносятся в рамках одной транзакции;
- асинхронный: подразумевает отложенный характер внесения изменений в удаленные копии.

Достоинство синхронного распространения изменений – полная согласованность копий и отсутствие конфликтов обновления. Недостатки:

- трудоемкость и большая длительность модификации данных,
- низкая надежность работы системы.

28-Поддержка распределенных БД. Шардинг

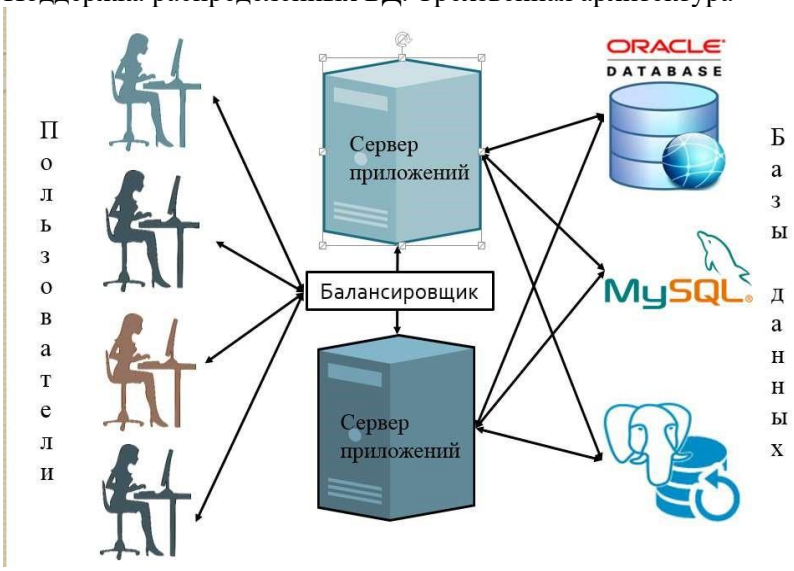
Lec.7 sli.16

Шардинг (иногда шардирование) — это другая техника масштабирования работы с данными. Суть его в разделении (партиционирование) базы данных на отдельные части так, чтобы каждую из них можно было вынести на отдельный сервер.

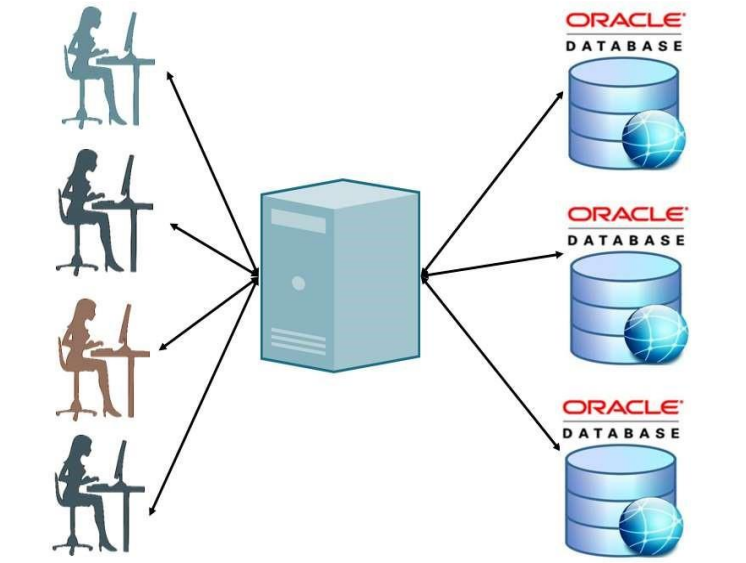
Шардинг – это одна из стратегий масштабирования каких-либо приложений. Шардинг стал одной из топовых тем для обсуждения в сообществе Etherium.

Развивающиеся приложения часто сталкиваются с проблемой масштабирования Тогда базу данных и делят на части, отправляя эти части на разные шарды.

Поддержка распределенных БД. Трехзвенная архитектура



Поддержка распределенных БД. Шардинг



Шардинг. Поддержка в ORACLE

Секционирование (англ. partitioning) — разделение хранимых объектов баз данных на отдельные части с раздельными параметрами физического хранения

29-Протокол двухфазной фиксации

Lec.7 sli.27

Распределенные транзакции обращаются к двум и более узлам и обновляют на них данные.

Основная проблема распределенных транзакций – соблюдение логической целостности данных. Транзакция на всех узлах должна завершиться одинаково: или фиксацией, или откатом.

Выполнение распределенных транзакций осуществляется с помощью специального алгоритма, который называется двухфазная фиксация.

Протокол двухфазной фиксации

а) диаграмма состояний координатора

б) диаграмма состояний участника

Действия координатора транзакции

Координатор выполняет протокол 2ФФ по следующему алгоритму:

I. Фаза 1 (голосование).
   - Занести запись begin_commit в системный журнал и обеспечить ее перенос из буфера в ОП на ВЗУ.
   - Отправить всем участникам команду PREPARE.
   - Ожидать ответов всех участников в пределах установленного таймаута.

II. Фаза 2 (принятие решения).

При поступлении сообщения ABORT:

- занести в системный журнал запись abort и обеспечить ее перенос из буфера в ОП на ВЗУ;
- отправить всем участникам сообщение GLOBAL_ABORT и ждать ответов участников (тайм-аут).

- Если участник не отвечает в течение установленного тайм-аута, координатор считает, что данный участник откатит свою часть транзакции и запускает протокол ликвидации.
- Если все участники прислали COMMIT, поместить в системный журнал запись commit и обеспечить ее перенос из буфера в ОП на ВЗУ. Отправить всем участникам сообщение GLOBAL_COMMIT и ждать ответов всех участников.
- После поступления подтверждений о фиксации от всех участников: поместить в системный журнал запись end_transaction и обеспечить ее перенос из буфера в ОП на ВЗУ.
- Если некоторые узлы не прислали подтверждения фиксации, координатор заново направляет им сообщения о принятом решении и поступает по этой схеме до получения всех требуемых подтверждений.

Действия участника транзакции

Участник выполняет протокол 2ФФ по следующему алгоритму:

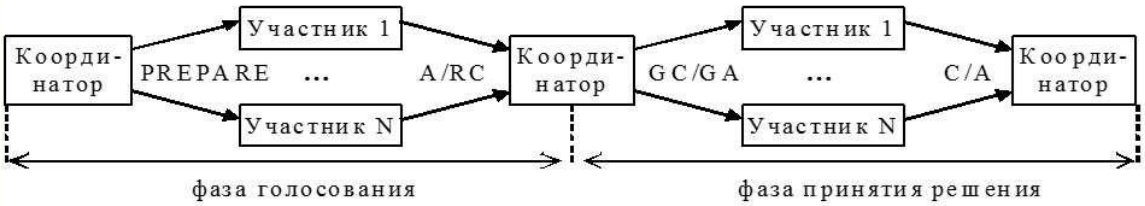
- При получении команды PREPARE, если он готов зафиксировать свою часть транзакции, он помещает запись ready_commit в файл журнала транзакций и отправляет координатору сообщение READY_COMMIT. Если он не может зафиксировать свою часть транзакции, он помещает запись abort в файл журнала транзакций, отправляет координатору сообщение ABORT и откатывает свою часть транзакции (не дожидаясь общего сигнала GLOBAL_ABORT).
- Если участник отправил координатору сообщение READY_COMMIT, то он ожидает ответа координатора в пределах установленного тайм-аута.
- При получении GLOBAL_ABORT участник помещает запись abort в файл журнала транзакций, откатывает свою часть транзакции и отправляет координатору подтверждение отката.
- При получении GLOBAL_COMMIT участник помещает запись commit в файл журнала транзакций, фиксирует свою часть транзакции и отправляет координатору подтверждение фиксации.
- Если в течение установленного тайм-аута участник не получает сообщения от координатора, он откатывает свою часть транзакции.

30-Варианты реализации протокола 2ФФ

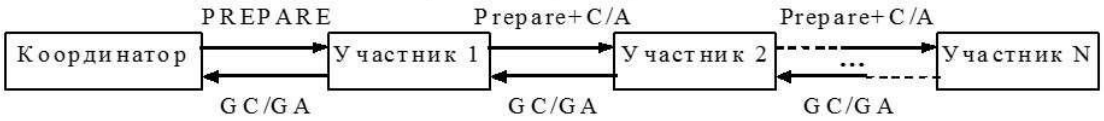
Lec.7 sli.33

Реализация протокола 2ФФ

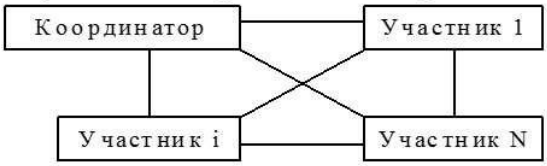
а) традиционная реализация протокола 2ФФ



б) линейная реализация протокола 2ФФ



в) распределенная реализация протокола 2ФФ



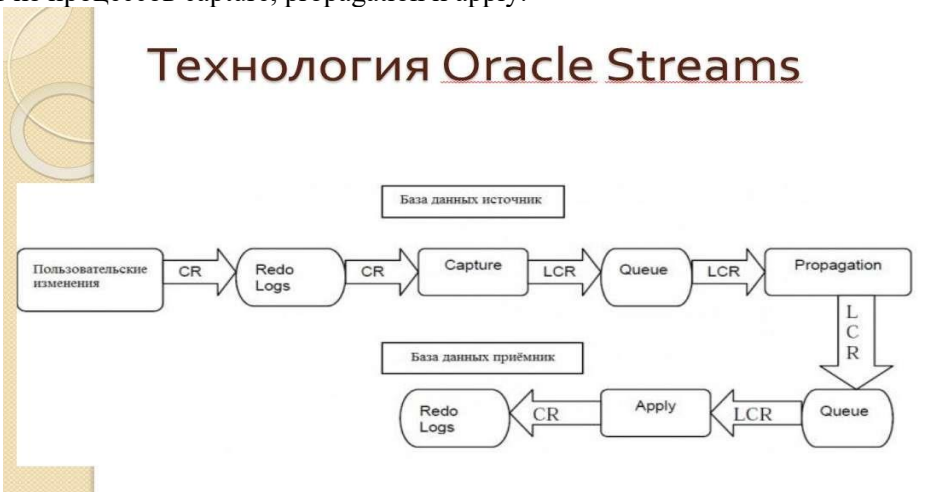
31-Технология Oracle Streams

Lec.7 sli.38

Oracle Streams – универсальный гибкий механизм обмена информацией между серверами в много серверной архитектуре (MTS). Позволяет одновременно реализовать репликацию, обмен сообщениями, загрузку хранилищ данных, работу с событиями, поддержку резервной БД.

Данные следуют по определенным пользователем маршрутам и доставляются к месту назначения. В результате получается механизм, который обеспечивает большую функциональность и гибкость, чем традиционные решения для хранения и распространения данных, а также совместного их использования с другими базами данных и приложениями.

Oracle Streams – отдельная информационная инфраструктура, которая состоит из процессов capture, propagation и apply.



Термины Oracle Streams

LCR, CR

В контексте Oracle Streams информационное представление любого изменения, сделанного в базе данных, называется LCR (logical change record). CR (change record) –запись изменения, используется для того, чтобы обозначить конкретное изменение в базе.

Capture, Propagation and Apply –три основных процесса Oracle Streams.

Основные задачи процесса capture:

- считывание изменений, содержащихся в журналах транзакций;
- преобразование CR в LCR;
- постановка LCR в очередь.

Так же как и capture, процесс propagation выполняет 3 основных задачи:

- просмотр LCR;
- передача LCR из одной очереди в другую, причем очереди могут находится как на одной базе данных, так и на разных;
- удаление LCR.

Apply процесс:

- извлекает принятые LCR из очереди;
- производит изменения с базой данных в соответствии с LCR;
- удаляет LCR из очереди.

32-Стихийные угрозы информационной безопасности

Lec.9 Sli.6

Стихийные угрозы информационной безопасности:

Не предполагается наличие злоумышленника

- Отказ оборудования
- Стихийное бедствие
- Техногенная катастрофа
- Неумышленные действия персонала

33-Злоумышленные угрозы информационной безопасности

Lec.9 sli.7

Злоумышленные угрозы информационной безопасности (Предполагается наличие злоумышленника):

1-Несанкционированное считывание информации

- считывание паролей и отождествление их с конкретными пользователями;
- получение конфиденциальной информации;
- идентификация информации, запрашиваемой пользователями;
- подмена паролей с целью доступа к информации;
- контроль активности абонентов сети для получения косвенной информации о взаимодействии пользователей и характере информации, которой обмениваются абоненты сети.

2-Неразрешенная модификация информации и ее уничтожение

- разрушение данных и кодов исполняемых программ путем внесения тонких, трудно обнаруживаемых изменений в информационные массивы;
- внедрение программных закладок в другие программы и подпрограммы (вирусный механизм воздействий);
- искажение или уничтожение собственной информации сервера, и тем самым нарушение работы сети;
- модификация пакетов сообщений.

3-Изменение функционирования вычислительной системы (сети)

- уменьшение скорости работы вычислительной системы (сети); частичное или полное блокирование работы системы (сети);
- имитация физических (аппаратных) сбоев работы вычислительных средств и периферийных устройств;
- переадресация сообщений;
- обход программно-аппаратных средств криптографического преобразования информации;
- обеспечение доступа в систему с непредусмотренных периферийных устройств.

34-Простые действия для обеспечения безопасности IT-систем

Lec.9 sli.12

-Настройка поддерживаемых систем
-Обновление поддерживаемых систем
-Обучение персонала (пользователей)
-Распределение ролей и ограничение доступа к данным
-Создание программ проверки целостности
-Создание всеохватывающей системы аудита