Лекція 3-2020_2021

- На попередній лекції
- Моделі данних
  - ОО
  - ОР
  - NoSQL
  - NewSQL
- Архітектура централізованих БД з мережевим доступом

На прошлой лекции

Основные модели представления данных:

иерархическая, сетевая, реляционная, постреляцион-ная, многомерная, объектно-реляционная, объектно-ориентированная, NoSQL, NewSQL.

- Иерархическая модель. БД представляется в виде древовидной структуры, состоящей из объектов различных уровней. Основное правило: запись пото-мок должна иметь в точности одного предка.
- Сетевая модель данных − расширение иерархического подхода. Основное правило: любой объект может быть одновременно и главным, и подчиненным, и может участвовать в образовании любого числа взаимосвязей с другими объектами.

На прошлой лекции

• Реляционная модель. Данные представляются в виде двумерных таблиц (отношений). Основа – теория множеств и логика предикатов. Для того, чтобы таблица была отношением:

− Все строки таблицы должны быть уникальны и должны иметь одну и ту же структуру;

− Имена столбцов таблицы уникальные, данные в одном столбце должны быть однотипными.

− Значения атрибутов должны быть атомарными;

− Порядок следования строк и столбцов в таблице несущественен.

• Постреляционная модель данных – расширенная реляционная модель. Значения атрибутов могут быть не атомарные, допускаются многозначные поля, значения которых являются самостоятельной таблицей, встроенной в основную таблицу.

На прошлой лекции

OLTP (Online Transaction Processing) системы − обработка транзакций в реальном времени: реляционные и постреляционные БД, оперативная обработка.

OLAP (Online Analytical Processing) системы – аналитическая обработка данных, многомерное представление данных.

Многомерная модель – разновидность реляционной модели, которая использует многомерные структуры. Многомерные СУБД предназначены для аналитической обработки информации в системах поддержки принятия решений.

Основные свойства: − рассмотрение информации на различных уровнях ее обобщения; − данные жестко зависят от времени; − возможна прогнозируемость данных.

Основные понятия: измерение (грань гиперкуба) и ячейка (значение при фиксированных значениях измерений).

Объектно-ориентированная модель

ООБД оперируют объектами, между данными и функциями их обработки устанавливаются взаимосвязи подобные соответствующим средствам в объектно-ориентированных языках программирования.

Примеры: G-Base, GemStone, Statice, ObjectStore, Versant, O2, ODB-Jupiter, Iris, Orion.

Примеры ООБД

Проект ORION осуществлялся с 1985 по 1989 г. фирмой MCC под руководством Вона Кима.

ORION: ORION-1 – однопользовательская система; ORION-1SX – сервер в локальной сети;

ORION-2 – распределенная ООСУБД.

Реализация на языке Common Lisp в среде ОС UNIX.

Проект O2 выполнялся французской компанией Altair, с сентября 1986 г. пять лет (три года на прототипирование, два года на разработку промышленного образца).

ObjectStore – долговременное хранение в БД объектов, созданных программами на языках C++ и Java

Модель данных, применяемая в O2

В O2 поддерживаются объекты и значения.

Объект – пара (идентификатор, значение).

Объекты инкапсулированы, их значения доступны только через методы – процедуры, привязанные к объектам.

Классы (экземпляры = объекты → данные + поведение). Типы. Каждому классу сопоставляется тип, описывающий структуру экземпляров класса.

Поведенческая сторона класса определяется набором методов.

Модель данных, применяемая в O2

Метод - программный код, привязанный к конкретному классу и применимый к объектам этого класса.

1. Объявляется сигнатура метода (имя, класс, типы или классы аргументов и тип или класс результата). Методы могут быть публичными или приватными.
2. Определяется реализация класса на языке программирования O2.

В O2 используются два объектно-ориентированных расширения языков Бейсик и Си.

Язык CO2 – расширение Си++.

Реляционные БД vs ООБД

Парадигма ООП в технологии разработки баз данных не особо популярна. Причины:

Популярность. Под РБД создано множество продуктов, которые необходимо поддерживать и развивать. В эти продукты уже вложены большие деньги и заказчики готовы еще вкладывать деньги в их развитие. Напротив, с использованием ООБД разработано сравнительно мало серьезных коммерческих продуктов, существует мало мощных ООСУБД.

Реляционные БД vs ООБД

Математический аппарат. Для РБД – реляционная алгебра и реляционное исчисление, для ООБД нет такого аппарата, хотя работы ведутся с 80-х. годов.

Для хранения данных ООБД разработан стандарт ODMG

– объектная модель;

– язык описания объектов для определения схемы БД;

– язык объектных запросов SQL - подобный декларативный язык для извлечения объектов из БД;

– связывание с ОО-языками (C++, Smalltalk, Java). Каждый ОО-язык имеет свое связывание, поэтому разработчик остается в одной языковой среде, ему нет необходимости разделять средства программирования и доступа к данным.

Реляционные БД vs ООБД

Язык запросов и его стандартизация. Попытки совместить средства манипулирования данными реляционной модели и способы описания внешнего мира объектно-ориентированной модели получили развитие в языке SQL-3.

Объектно-реляционные базы данных

ОРСУБД — реляционная СУБД (РСУБД), поддерживающая некоторые технологии, реализующие объектно-ориентированный подход. Объекты, классы и наследование реализованы в структуре баз данных и языке запросов (очень упрощенное определение).

Возникла из-за необходимости определять пользовательские типы данных.

Примеры: Oracle, DB2, PostgreSQL.

Объектно-реляционные базы данных

Два направления разработки:

Майкл Стоунбрейкер (от реляционной модели за счет добавления средств, свойственных объектным системам). PostgreSQL. Компании, производивших SQL-ориентированные СУБД смогли наращивать функциональные возможности своих систем без потребности в их коренной перестройке.

Под руководством Вон Ким было разработано семейство ООСУБД Orion. Подход с позиций объектно-ориентированного мира. UniSQL – ООСУБД со специальными средствами для обеспечения реляционных свойств. Подход Вона Кима не оказал влияние на ведущие коммерческие СУБД.

Объект класс в СУБД Oracle

CREATE TYPE – описание класса:

```
CREATE TYPE TYPE1 AS OBJECT (
  ID_TYPE1 NUMBER(19),
  PARAM1 VARCHAR2(250),
  MEMBER FUNCTION FUNC1 RETURN NUMBER );
```

CREATE TYPE BODY – код методов класса.

Например:

```
CREATE TYPE BODY TYPE1 AS
MEMBER FUNCTION FUNC1 RETURN NUMBER IS
  BEGIN
  RETURN 1;
  END;
END;
```


Объектные расширения СУБД Oracle

Для Windows 64 в 2010 г. Компанией Оракл разработана версия 11g Release 2 – объектно-реляционная БД, устойчивый, обеспеченный документацией релиз, поддерживает SQL 2008.

Исследования показали, что более 90% крупных предприятий используют комбинацию локальных ИТ-решений, выделенных облачных сред и публичных облачных сервисов. Поэтому вся дальнейшая разработка Оракл для облака, для Windows в 2017 г. 12c Release 2, в 2018 − 18c. Новая версия 18c обеспечивает простую автоматизированную миграцию в облако.

СУБД Oracle 11gR2 – объектно-реляционная БД. Данные хранятся в двумерных таблицах. Работа с объектами:

классами, массивами и вложенными таблицами.

Объектный тип определяется в CREATE TYPE.

Для класса: имя, атрибуты, свойства и методы.

Способы хранения объектов в СУБД Oracle

Объекты можно хранить в столбце таблицы. В этом случае вначале создают объектный тип данных (класс), а потом определяют атрибут родительской таблицы этим типом. При вставке новых значений объектного типа осуществляется обращение к конструктору объекта, который автоматически создается СУБД при заведении нового типа.

Другой способ хранения: создание таблицы объектов.

Таблицы объектов в Oracle представляют собой списки объектов, это таблицы из одного столбца объектного типа. На такие объекты можно ссылаться, в таблице можно разместить как объекты класса, так и объекты - наследники класса.

ORM (Object-Relational Mapping) — объектно-реляционное отображение)

ORM − технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Задача − работать в ОО среде с объектами, хранить данные в реляционной БД, иметь механизм конвертирования данных.

Разработаны пакеты, предоставляющие библиотеки классов, которые способны преобразовывать объекты для хранения в реляционных базах данных автоматически.

ORM (Object-Relational Mapping) — объектно-реляционное отображение)

С точки зрения программиста система выглядит как постоянное хранилище объектов.

Плюсы − ORM избавляет программиста от написания большого количества кода, уменьшаются ошибки, повышается скорость разработки. Большинство ORM позволяют вручную жёстко задать код SQL-запросов.

Минусы − медленная и неэффективная работа с данными, большой расход памяти.

ORM (Object-Relational Mapping) — объектно-реляционное отображение)

На практике заказчик хочет получить продукт как можно раньше, затратив минимум денег. Никто не думает об оптимизации в момент написания первых версий продукта − главный фактор скорость разработки. Через некоторое время, когда база наполнится реальными данными, заказчик и разработчик обнаруживают, что время выборок увеличилась почти вдвое, при работе 10-20 пользователей одновременно СУБД пытается покончить жизнь самоубийством и т.д. и т.п. Требуется искать узкие места, выдирать из ORM автоматические запросы, переписывать их руками, перестраивать индексы в таблицах БД и т.д.

NoSQL

В последнее десятилетие человечество столкнулось с очень большими объемами данных, которые быстро меняются и разнообразны по своей структуре. С ними тяжело работать, используя традиционные реляционные СУБД. Термин NoSQL обозначает «не только SQL» (Not Only SQL). NoSQL БД обеспечивают масштабируемость и доступность, жертвуя атомарностью и согласованностью данных.

Характерные черты NoSQL-решений:

- Применение различных типов хранилищ.
- Возможность разработки БД без задания схемы.

1. Линейная масштабируемость (добавление процессоров увеличивает производительность).
2. Инновационность: «не только SQL» открывает много возможностей для хранения и обработки данных.

SQL и NoSQL системы

Несколько ключевых различий между этими типами СУБД.

1. В SQL БД есть четкие схемы таблиц, в NoSQL нет заранее заданных схем документов.
2. В SQL существуют сложные связи между различными таблицами, в NoSQL, как правило, каждый документ является изолированной единицей и хранит в себе все имеющиеся данные.
3. Механизмы поддержки целостности данных есть в SQL и отсутствуют в NoSQL.
4. В SQL есть механизм транзакций, в NoSQL − только в пределах одного документа.
5. В идеальном случае NoSQL работает быстрее.

SQL и NoSQL системы

Несколько ключевых различий между этими типами СУБД.

1. В SQL БД есть четкие схемы таблиц, в NoSQL нет заранее заданных схем документов.
2. В SQL существуют сложные связи между различными таблицами, в NoSQL, как правило, каждый документ является изолированной единицей и хранит в себе все имеющиеся данные.
3. Механизмы поддержки целостности данных есть в SQL и отсутствуют в NoSQL.
4. В SQL есть механизм транзакций, в NoSQL − только в пределах одного документа.
5. В идеальном случае NoSQL работает быстрее.

Модели для NoSQL БД

Модели и функциональные системы для NoSQL БД:

• Хранилище «ключ-значение». Данные хранятся в простой хеш-таблице. Доступ к данным осуществляется по ключам. По ключу можно получить значение или записать значение в базу. (*Redis, MemcacheDB, Riak, Berkeley DB и т.п.*).

• Хранилище колонок. Основная идея − группировка подобных значений в семейство столбцов. Такое семейство хранится в отдельном файле (*Cassandra, Hbase, Google Big Table и т.п.*).

Модели для NoSQL БД

• Документоориентированные СУБД. Это хранилище для документов типа XML, JSON, BSON и др. Документоориентированные хранилища отлично хранят несвязанную информацию больших объемов, даже если она очень разнится от сущности к сущности (*MongoDB, CouchDB, MarkLogic, eXist* и т.п.).

• Графовые СУБД хранят сущности и отношения между ними. Сущности представляются в виде узлов, которые имеют свойства. Отношения представляются в виде ребер, которые тоже могут иметь свойства. (OrientDB, Neo4J, ArangoDB, FlockDB, HyperGraphDB и т.п.).

Преимущества NoSQL по сравнению с РБД

• линейная масштабируемость – добавление новых узлов в кластер увеличивает общую производительность системы;

• гибкость − возможность оперировать полуструктурированными данными;

• возможность работать с разными представлениями информации, без задания схемы данных;

• высокая доступность за счет репликации данных и автоматического разделения данных по разным узлам сети;

• производительность за счет оптимизации для конкретных видов моделей данных;

• широкие функциональные возможности – собственные языки запросов.

Недостатки NoSQL БД

• ограниченные возможности встроенных языков запросов;

• отсутствие поддержки ACID требований. Вместо ACID:

атомарность - либо все операции, либо ни одной; *согласованность* - по окончании транзакции БД структурно согласованна;

изолированность - транзакции не мешают друг другу; *долговечность* - результаты применения транзакции не теряются. Подход BASE: *обычно доступно* - хранилище доступно большую часть времени; *гибкое состояние* - хранилища не обязаны соблюдать очередность записей, и разные реплики не должны сразу согласовываться; *отложенная согласованность* - хранилища достигают согласованности с задержкой по времени.

• сильная привязка приложения к конкретной СУБД из-за специфики внутреннего языка запросов и гибкой модели данных;

• недостаток специалистов по NoSQL-базам по сравнению с реляционными аналогами;

NewSQL БД

NewSQL− класс современных реляционных СУБД, стремящихся совместить в себе преимущества NoSQL и транзакционные требования классических БД. Потребность в данных системах возникла у компаний, работающих с критическими данными (например, финансового сектора), которым требовались масштабируемые решения, в то время как решения NoSQL не могли предоставить транзакций и не отвечали требованиям надёжности данных.

Представители *VoltDb, Google Cloud Spanner*.

Google Cloud Spanner

1. Наличие NoSQL возможностей;
2. Поддержка распределенных транзакций; (New!)
3. Глобальная согласованность операций чтения между географически распределенными ДЦ, т.о. данные, которые возвращают операции чтения из разных ДЦ, всегда согласованны и непротиворечивы.
4. Автоматическая обработка отказов как вычислительных узлов, так и ДЦ;
5. Автоматическая миграция данных как между вычислительными узлами, так и между ДЦ.
6. Обеспечение быстродействия SQL запросов.

Классификация баз данных

По технологии обработки:

Централизованная база – хранится в памяти одной вычислительной системы.

Распределенная база – состоит из нескольких, возможно, пересекающихся или даже дублирующих друг друга частей, которые хранятся на различных компьютерах вычислительной сети.

По способу доступа к данным:

базы данных с локальным доступом,

базы данных с сетевым доступом.

Архитектура централизованных БД с сетевым доступом. Телеобработка.

Основная большая нагрузка возлагается на центральный компьютер, который должен выполнять не только действия прикладных программ и СУБД, но и работу по обслуживанию терминалов (например, форматирование данных, выводимых на экраны терминалов).

Архитектура централизованных БД с сетевым доступом. Телеобработка.

Один компьютер с единственным процессором соединен с несколькими терминалами. Вся обработка выполняется в рамках единственного компьютера, а присоединенные к нему пользовательские терминалы – "неинтеллектуальные" устройства (не самостоятельные).

Прошлое: Система Виртуальных Машин (VMS) на мэйнфрейме. Настоящее, например: На центральном компьютере установлена система Windows Server 2015 и инсталлировано ПО (exspress studio 2013 для изучения C#, Rad Studio XE с Delphi XE). Все остальные компьютеры класса используются как терминальные станции (тонкие клиенты).

Архитектура централизованных БД с сетевым доступом. Файловый сервер.

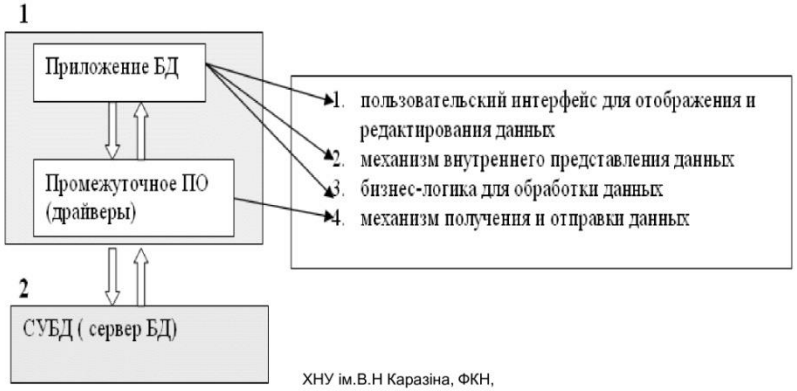
Пользовательские приложения и сама СУБД размещены и функционируют на отдельных рабочих станциях, и обращаются к файловому серверу только для получения доступа к нужным файлам. Файловый сервер – совместно используемый жесткий диск. Например, расшарена папка на центральном компьютере, там хранится *.accdb БД в режиме многопользовательского доступа, на каждом компьютере – среда выполнения Access.

Недостатки: – значительный сетевой трафик,

– трудности при управлении параллельностью доступа к информации, при восстановлении информации, при поддержании целостности.

Архитектура централизованных БД с сетевым доступом. Клиент-сервер.

Двухуровневая (локальные сети): на выделенном компьютере – сервер БД, на рабочих станциях – приложения БД.



Архитектура централизованных БД с сетевым доступом. Клиент-сервер.

Извлеченные данные транспортируются по сети от сервера к клиенту.

Приложения БД могут быть реализованы на любом объектно-ориентированном языке (ООЯ), поддерживающим возможность доступа к данным сервера БД. Это, так называемые «толстые» клиенты БД.

Двухуровневая модель используется в локальных сетях.

Основа работы сервера БД – использование языка запросов (SQL).

Архитектура централизованных БД с сетевым доступом. Клиент-сервер.

Между приложением и БД находится специальное ПО, связывающее программу и источник данных и управляющее процессом обмена данными.

Промежуточное ПО может быть реализовано как:

– окружение приложения, без которого оно вообще не будет работать,

– набор драйверов и динамических библиотек, к которым обращается приложение,

– может быть интегрировано в само приложение.

Например: MySQL сервер, приложение на C# или Java, промежуточное ПО – ODBC драйвер или технология ADO, JDBC.

Архитектура централизованных БД с сетевым доступом. Клиент-сервер.

Web сервер умеет просто отдавать запрошенную страницу. Для возможности обработки данных и динамического сформирования ответа браузеру существует технология CGI (Common Gateway Interface).

CGI программа находится на сервере приложений, она содержит бизнес-логику. CGI программа может быть реализована на языках серверных скриптов Perl, PHP. CGI программа осуществляет доступ к серверу БД.

Архитектура централизованных БД с сетевым доступом. Клиент-сервер.

Сервер приложений – формирование запросов к БД данных. Он может быть Web сервером или спец. программой (например, Oracle Forms Server).