Оглавление

1

Лекция первая. Введение в БД.

1.1 Базы данных.

БД - это самодокументированная собрание интегрированных записей. Набор таблиц.

Самодокументированная - хранятся метаданные, т.е. данные о данных.

Интегрированные записи - Файлы данных. Целый комплекс. Имеются индексы. Метаданные

1.2 Основные требования к БД.

- Не избыточность - не храним лишнюю информацию.
- Эффективность доступа - малое время отклика на действие пользователя.
- Совместное использование.
- Безопасность. Также внутренняя безопасность - защита от дурака (пример: вместо числа ввел букву).

- Восстановление после сбоя.
- Целостность - если ссылаемся на какой-то объект, то он должен быть. Не ссылаться на несуществеющие объекты.
- Независимость от сторонних приложений. Если программа отправляет ерунду БД должна обработать.

1.3 СУБД, журнализация.

СУБД - (Средства управления БД) приложение, обеспечивающее создание, хранение, обновление и поиск информации в БД.Программа. **СУБД управляет БД**.

Система БД - совокупность БД.

Транзакция - набор действий, которые выполняются одновременно. (Пример: онлайн перевод, одновременно в одном месте деньги ушли, в другом появились.)

Журнализация - информация о действиях, которые происходили в системе. Помогает в откате каких-то действий. **БД** сохраняет запросы в журнале.

СУБД должна поддерживать языки.

1.4 Основные компоненты СУБД

- Ядро - управление памятью. Журнализация.
- Процессор языка БД - оптимизация. Выполнение.
- Подсистема поддержки времени исполнения.
- Сервисные программы - те утилиты, которые мы пишем, доп. возможность. (Вывод звездочек вокруг имени.)

1.5 Классификация СУБД

- По модели данных
  - Дореляционная.
    - Инвертированный список (рис 1)
    - Иерархия. (Дерево)
    - Сетевые (граф)
  - Реляционная.
  - Постреляционная
- По архитектуре.
  - Локальные - на одном устройстве.
  - Распространенные - на многих устройствах.
- По способу доступа к БД
  - Файл-серверный подход - Подключились, взяли всё. Нагружаем клиента, а не сервер. **Минусы: У каждого клиента своя копия.**
  - клиент-серверные - запросы выполняются на сервере, клиент получает только нужное
  - Встраиваемые - маленькие базы, которые не нужны всем.

2

Лабораторная работа 1.

2.1 Задание:

- Выбрать тему.
- Рисуем ER-модель нашей базы.
- Создать БД. Создать таблицу. (>= 3ёх объектов (таблицы связки не считаются объектами)). Создать ключи. (Все это в SQL-скрипт)
- Наполнить (csv) >= 1000 строк.

По итогу 2 фала: 1 SQL-скрипт и 1 модель.

3

Семинар 1.

3.1 SQL

SQL - SQL (Structured Query Language – язык структурированных запросов)
декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных.

SQL - работает в любой БД. В основах лежит реляционная модель.

В основе реляционной модели лежит теория множеств и логика предикатов.

T-SQL - нек-ое дополнение. Надстройка.

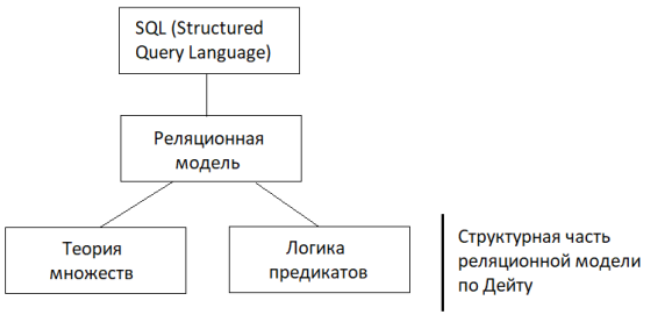

Рис. 3.1: SQL

Заголовок – набор атрибутов (В SQL - столбцы), каждый из которых имеет определенный тип.
Атрибут – совокупность имени и типа данных (Атрибут==столбец).Атрибут – название столбца, его тип + дополнительные настройки
Тело – множество картежей (В SQL – строки).
Заголовок кортежа – заголовок отношения.

Рис. 3.2: Пример таблицы.

3.2 Основные языки

Логику работы с данными можно разделить на три основных языка:

- DLL (*Data Definition Language*) - (Создаем объекты для хранения данных).Служит для описания структуры БД:
  - Создать (Create)
  - Удалить (Drop)
  - Изменить (Alter)
- DML (*Data Manipulation Language*) - Язык для работы с данными
  - Обновить (update)
  - Загружать (insert)
  - Удалять (delete/truncate)

– Читать (select)

- DCL (*Data Control Language*) - Служит для управления доступа к объектам.
  - Выдача прав доступа к объекту (grand)
  - Удаление прав доступа на объект (revoke)

Обращение к таблице. Схема обращения к таблице: [название БД].[название схемы].Название таблицы. Рис. 3.3

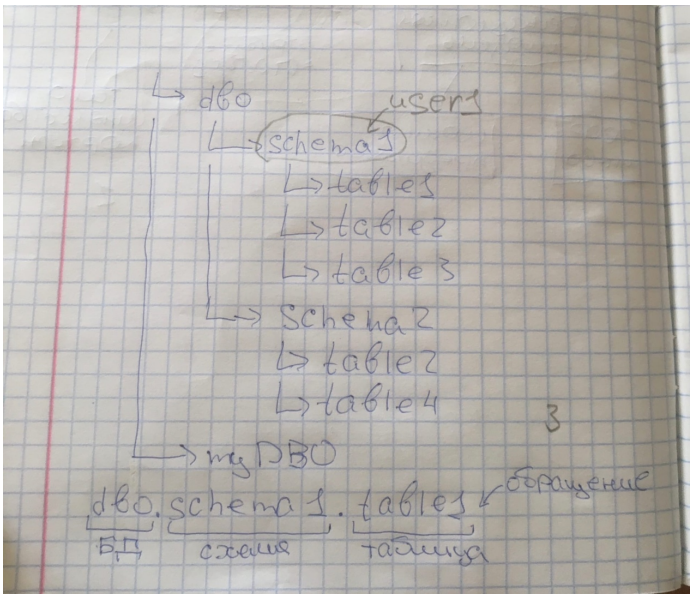

Рис. 3.3: Структура БД.

3.3 Способы хранения данных

- Таблица (table).
- Временные таблица (temp table). По завершению сессии таблица удаляется.
- Представление (View)

- Производные таблицы. (Временная)
- Индексированное представление.

Пример создания таблицы

```
CREATE TABLE dbo.EmployeePhoto
(
Id int IDENTITY(1, 1),
EmployeeId int NOT NULL PRIMARY KEY,
Photo varbinary(max) FILESTREAM NULL,
MyRowGuidColumn uniqueidentifier NOT NULL ROWGUIDCOL
UNIQUE DEFAULT NEWID()
);
```

id - АТРИБУТ типа int счетчик шагаем начиная с единицы с шагом 1.
Employer id - поле, которое используем в кач-ве идентификатора.
Photo - По умолчанию NULL - пустой.
MyRawGuidColumn - Уникальное, DEFAULT - по умолчанию поле задается newID.
nvarchar - Выделяет столько памяти, какова длина строки
varchar - Физически занята вся строка. Занято пробелами.
Salary - numeric(15, 2) - Сколько всего цифр выделено в нашем числа, сколько знаков после запятой.

4

Реляционная модель (Лекция 2)

4.1 ER - модель

- Сущность
- Связь

Объекты обозначаются прямоугольниками. Внутри пишем название.

Виды сущностей:

- Сильные - Обозначаются просто в рамке.
- Слабые - не могут существовать друг без друга. Факультет и предметы. Обозначается вложенным квадратом (рамочка).

Атрибуты отображаются овалами. Внутри пишем название атрибута.

Виды связей:

- Один к одному. Студент-зачетка.
- Один ко многим. Статья-рецензия. Добавляем внешний ключ со стороны многих. Из многих в сторону одного.

- Многие ко многим. Студент-преподаватель. Добавляем связочную таблицу.

4.2 Реляционная модель

Реляционная модель

- Структурная часть - отвечает за то, какие объекты есть.
- Целостная - отвечает за ссылки. DDL.
  - Ссылочная целостность (FK)
  - Целостность сущности (PK) - говорит о том, что есть первичный ключ. Нет повторения. Всегда знаем на что ссылаемся.
- Манипуляционная - за механизм работы с данными. DML.

Домен = (примерно равно) тип данных.
Атрибут (отношения) = (примерно равно) столбец. Упорядоченная пара вида:
имя-атрибута,имя-домена
Схема отношений = (примерно равно) Заголовок. имя-отношение, имя-домена
Кортеж = (примерно равно) Строка. Имя-атрибута, значение-атрибута
Отношение = (примерно равно) таблица.

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)
- неизбыточности (никакое из собственных подмножеств множества потенциального ключа не обладает свойством уникальности

Внешний ключ в отношении R2 – это непустое подмножество множества атрибутов FK этого отношения, такое, что:

- Существует отношение R1 (причем отношения R1 и R2 необязательно различны) с потенциальным ключом CK;
- Каждое значение внешнего ключа FK в текущем значении отношения R2 обязательно совпадает со значением ключа CK некоторого кортежа в текущем значении отношения R1.

5

Реляционная алгебра (Лекция 3)

Реляционная алгебра - замкнутая система.

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

- Традиционные
  - Объединение. (union)
  - Пересечение. (intersect)
  - Вычитание. (minus) (В mysql иначе называется)
  - Декартово произведение - все со всеми. (times)
- Специальные
  - Проекция. (PROJECT, []) помогает выбирать не все из нашего отношения. Можно набрать только те атрибуты, которые будем использовать далее.
  - Фильтрация. (WHERE)
  - Соединения. (JOIN)

- Деление. (DIVIDE BY)

Деление. (DIVIDE BY)
R1 {A, B}
R2 {B}
R1 DIVIDE BY R2 = R1[A] minus(R2 YIMER R1[A]) minus R1)[A]

5.1 Синтаксис реляционной алгебры

Любое реляционное выражение - это унарное выражение или бинарное выражение

- Унарное - выражение с одним элементом.
  - Переименование := терм RENAME имя_атрибута AS новое_имя_атрибута
  - Ограничение := терм WHERE логическое_выражение
  - Проекция := терм | терм[список атрибутов]
- Бинарное - с двумя элементами
  - Бинарное выражение := проекция бинарная_операция (реляционное_выражение)
  - Бинарный операция := UNION | INTERSECT | MINUS | TIMES | JOIN | DIVIDEBY

Терм - либо отношение, либо другое реляционное выражение. Реляционное выражение всегда берется в круглые скобки.

Имеются таблицы:

- S - поставщик S(Sno: integer, Sname: string, Status: integer, City: string)
- P - поставщик P(Pno: integer, Pname: string, Color: string, Weight: real, City: string)
- SP - Таблица связка SP(Sno: integer, Pno: integer, Qty: integer)

id	Имя детали	цвет	вес	Город
1	Гвоздь	К	10.3	Москва
2	Винт	З	15.8	Рязань
3	Гвоздь	С	3.4	Смоленск

Таблица 5.1: Детали

id	Имя поставщика	статус	город
1	ООО Ромашка	5	Рязань
2	ООО Рубин	3	Красногорск

Таблица 5.2: Поставщики

5.2 Примеры

Реляционные алгебра. Выражения.

1. Получить имена поставщиков, которые поставляют деталь под номером 2.

Листинг 5.1: Пример 1

```
(( S JOIN SP ) HWEPE Pno = 2 ) [ Sname ]
```

Листинг 5.2: Пример 1

```
select Sname
from S
join SP on S.Sno = SP.Sno
where SD.Pno = 2
```

Пример 1.2 быстрее

Листинг 5.3: Пример 1.2

```
((SP where Pno=2) join S) [Sname]
```

2. Получить имена поставщиков, которые поставляют по крайней мере одну красную деталь.

Листинг 5.4: Пример 2

```
((( PH WERE Color = 'Красный' ) JOIN SP)
[ Sno ] JOIN S ) [ Sname ]
```

Листинг 5.5: Пример 2

```
select Sname
from S
join SP on S.Sno = SP.Sno
join P on P.Pno = SP.Pno
where color='K'
```

3. Получить имена поставщиков, которые поставляют все детали.

Листинг 5.6: Пример 3

```
(( SP [ Sno, Pno] DIVIDE BY P
[ Pno ] JOIN S ) [ Sname ]
```

Листинг 5.7: Пример 3

```
with group SP(Sno, cnt) as (
        select Sno, count(distinct Pno)
        from SP
        group by Sno
)
select Sname
from group SP тут( необязательно  as) gSP
join S on gSP.Sno=S.Sno
where cnt=(select count(distinct Pno)
from P)
```

4. Получить номера поставщиков, поставляющих по крайней мере все те детали, которые поставляет поставщик под номером 2.

Листинг 5.8: Пример 4

```
SP [Sno, Pno] DIVIDE BY
(SP HWEPE Sno = 2)[ Pno ]
```

Листинг 5.9: Пример 4

```
with group SP(Sno, cnt) as (
        select Sno, count(distinct Pno)
        from SP
        where Sno in ()
```

```
        group by Sno in (select count(distinct Pno)
        from SP
        where Sno=2)
)
select Sname
from group SP тут( необязательно  as) gSP
join S on gSP.Sno=S.Sno
where cnt=(select count(distinct Pno)
from SP
where Sno=2)
```

5. Получить все пары номеров поставщиков, размещенных в одном городе

Листинг 5.10: Пример 5

```
(((  S MRENAE Sno AS FirstSno  )
[  FirstSno ,  City  ] JOIN
(S MRENAE Sno AS SecondSno  )
[  SecondSno  ,  City  ])H
WEPE FirstSno < SecondSno  )
[  FirstSno ,  SecondSno  ]
```

Листинг 5.11: Пример 5

```
select firstS.Sno, SecondS.Sno
from S firstS inner join S secondS
on firstS.Sno = secondS.Sno
where firstS.Sno < SecondS.SnoЭта
(  фильтрацияизбавитотдублей     )
```

6. Получить имена поставщиков, которые не поставляют деталь под номером 2.

Листинг 5.12: Пример 6

```
((S[  Sno  ] MINUS (SP HWEPE Pno = 2  )
  [  Sno  ]  ) JOIN S  ) [Sname]
```

Листинг 5.13: Пример 5

```
select Snp from S
minus
select distinct Sno
from SR
where Pno=2
```

6

Семинар 2

Таблица P.

id	Pname	Color	Weight	City
1	Гвоздь	К	10.3	Москва
2	Винт	З	15.8	Рязань
3	Гвоздь	С	3.4	Смоленск
4	шуруп	К	11	Рязань
5	шайба	с	17.8	Смоленск

Таблица 6.1: Таблица деталей P.

Таблица поставщика - S

Sno	Sname	Status	City
1	ООО Ромашка	5	Москва
2	ООО Рубин	3	Рязань
2	ООО Зеленоглазое такси	4	Смоленск

Таблица 6.2: Таблица поставщика - S

Агрегатная функция - sum, max, min, count,

Листинг 6.1: Пример

```
select Color, count(*)
from p
```

Sno	Pno	Cnt
1	1	100
2	1	150
3	1	180
1	2	180
3	2	180
4	3	180
5	3	180

Таблица 6.3: Таблица SP

```
group by Color
```

having используется для фильтрации групп.

Листинг 6.2: Пример

```
select Color, count(*)
from p
group by Color
having count(*)>1
```

order by - сортировка. Есть прямой порядок () и обратный (desc). По умолчанию по возрастанию.

Отсортируем таблицу деталей по весу.

Листинг 6.3: Пример

```
select Color, count(*)
from p
order by Pname, Weight desc;
```

Порядок записи инструкций. Цифрами показан порядок выполнения.

Листинг 6.4: Порядок записи инструкций

```
select (5)
from (1)
where (2)
group by (3)
```

```
having (4)
order by (6)
```

Нерабочий пример (потому что имя задаем на этапе позже) На этапе where использовать псевдонимы, которые мы создаем в select нельзя.

(2) - выполнится вторым действием,но у нас еще нет псевдонима. (Пример 6.5)

Листинг 6.5: Пример

```
select Pname as myName
from P (1)
from where myName ='Гвоздь' (2)
order by myName
```

Листинг 6.6: Пример внутренних запросов

```
select Sname
from S
where Sno in
(select distinct Sno
from SP
where Pno=2)
```

Запрос - найти цвет с max кол-вом деталей. Действия

- Сначала группируем по цвету. grop by Color
- Найти max.
- Вернуться к табл. и найти

with - показывает, что след. запрос будет выполняться до select. Это только для запроса.

Листинг 6.7: with. найти цвет с max кол-вом деталей.Обобщенное табличное выражение

```
with group Color(Color, cnt) as
select color count(*)
```

```
from P
group by Color(
(select max(cnt)
from (select Color,  count(*) as cnt
from P
grop by Color))
)
select Color
from group Color
where cnt in (
select max(cnt)
from group Color
)
```

Теперь переходим на *JOIN* - соединения
Виды:

- Внутренний. inner join (Это пересечение на кругах Эллера.);
- Внешние. outer join. (3 вида)
  - left join (На кругах это весь круг A)
  - right join (На кругах это весь круг B)
  - full join(полное) (На кругах это оба круга (и A и B))

Листинг 6.8: Внутреннее соединение

```
select A.id, B.id, A.name, B.fio
from A join B
on A.id = B.id
```

Виды join, которыми мы сможем воспользоваться

- Nested loops join. (Сложность n*n). Можно нашей СУБД указать, что нужно использовать её. Минусы: избыточность.
- Hash join (Сложность n+n). Можно сравнивать только на равенство. Минусы: доп расходы на таблицу.

- Merge join (Изначально таблицы должны отсортированы по ключу.) Минусы: нужно сортировать

Операция над множествами. Тело - это множество кортежей.

Ниже представленный запрос даст таблицу с двумя столбцами (id, name). Атрибуты называются по верхней схеме.

Листинг 6.9: union

```
select id, name, from A
union [all]
select id, FIO from B;
```

Листинг 6.10: minus

```
select id, name, from A
minus
select id, FIO from B;
```

join - добавляет столбцы, union - дописывает в конец.