Оглавление

1 Лекция первая. Введение в БД. 3
1.1 Базы данных. 3
1.2 Основные требования к БД. 3
1.3 СУБД, журнализация. 4
1.4 Основные компоненты СУБД 4
1.5 Классификация СУБД 5

2 Лабораторная работа 1. 6
2.1 Задание: 6

3 Семинар 1. 7
3.1 SQL 7
3.2 Основные языки 8
3.3 Способы хранения данных 9

4 Реляционная модель (Лекция 2) 11
4.1 ER - модель 11
4.2 Реляционная модель 12

5 Реляционная алгебра (Лекция 3) 14
5.1 Синтаксис реляционной алгебры 15
5.2 Примеры 16

6 Семинар 2 20

7 Группировка (Лекция 4) **25**
7.1 Реляционное сравнение 25
7.2 Исчисление кортежей 26

8 Семинар 3 **27**
8.1 Объекты БД 27

9 Функциональная зависимость (Лекция 5) **30**

10 Семинар 4 **36**
10.1 Курсор 40
10.2 Индексы 42
10.3 Партицирование 43

11 РК1 **44**
11.1 Задание 2 44

12 Семинар 5 **51**
12.1 Грамматика 51

13 Теория проектирования **52**
13.1 Формализация 52

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.

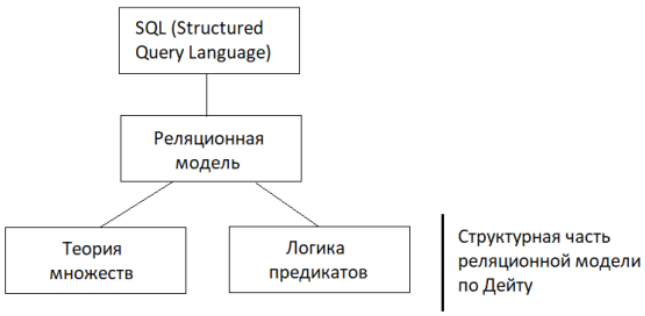

Рис. 3.1: SQL

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

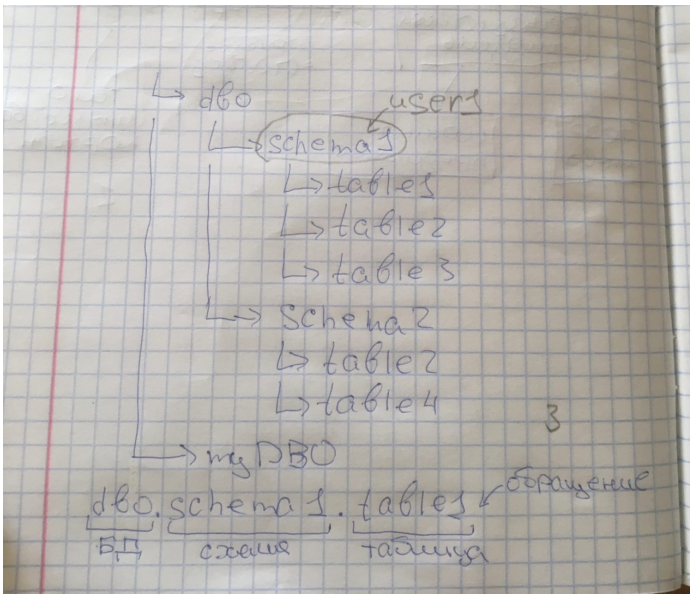

Рис. 3.3: Структура БД.

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.

7

Группировка (Лекция 4)

SP GROUP (Pno, Qny) as PQ

7.1 Реляционное сравнение

<рел. выр> <опер. сравн.> <рел выр.> опер. сравн. >= супермножество > - срьств. супермножество

<расширение> - добаление новых атрибутов по горизонтали.

<расширение> ::= EXTEND <реляц. выр.> ADD <список добавляемых расширений>

представляет собой выражение, после которого следует ключевое слово AS (Проименованное выражение)

<добавл расшир> ::= <выражение> AS <имя атрибута>

Обобщение - горизонтально группирует записи.

<Обобщение> :== SUMMARIZE <реляц выр> PER <рел выр> ADD <список добавляемых обобщений>

<добавл обобщение> :== <тип обобщения> [(скалярн выражение) AS <имя атрибута>]

<тип обобщения> :== COUNT | MIN | MAX | ANG | SUM | ALL | ANY

7.2 Исчисление кортежей

<объявление кортежной переменной> :== RANGE OF <переменная> IS <список областей>

<область> :== <отножение> | <рел выражение>

<рел выр> :== (список целевых элементов) [WHERE wff]

<целевой элемент> ::= переменная | переменная атрибут [AS <псевдоним>]

правильно построенная функция wff ::= условие | NOT wff | условие AND wff | if условие then wff | exists переменная (wff) | FORAll переменная (wff) | (wff)

8

Семинар 3

8.1 Объекты БД

- table (+ temp)
- view
- constraints
  - PK/FK
  - default, not null, cheek
- function

Функции.

1. По поведению:
   (a) детерминированные
   (Пример:
   select add date('day', '2020-10-03'))
   (b) нет (Пример:
   getDate())
2. По возвращаемому значению

(a) Скалярная функция (**Синтаксис**:
create function [схема].имя (<параметры>)
returns <ск.тип>
[*with* <опции>]
[as]
begin
<тело>
return <ск. переменная> end [;])
(**Пример:**
create functions dbo.AvgPrice()
returns smallmoney
with schemabinding
as
begin
return (select avg(Cnt) from P)
end;
) (**Пример:**
create function dbo.PriceDiff(@Pricesmallmoney)
returns smallmoney
as
begin
return @Price_dbo AvgPrice()
end)
select Price.dbo.AfgPrice()
dbo.PriceDiff(Price)
from P

(b) Подставляемая функция.
Синтаксис: create function [схема].имя (<парам>)
returns table
[with <опции>]
[as]
return <sql-запрос>
end [;]
Пример: create function dbo.fullSpy()

return table
as
return (select Sname, Pname
from S join SP on S.Sno = SP.Pno
join P on SP.Pno=P.Pno)
where P.color=@color

(c) Многооператорная функция.
Синтаксис: create function [схема].имя (<парам>)
returns @возвр.перем. table <опред. таблицы>
[with <опции>]
[as] declare
begin
<тело>
return
end[;]
Пример:
create function dbo.FnGetReport(@id as id)
returns @Reports table (eid int, repid int)
as
begin
declare @Empid as int
select @Empid = 1
select id from ...
into @Empid
insert into @Reports(...) values(...)

9

Функциональная зависимость (Лекция 5)

Функциональная зависимость (ФЗ)

R - отношение

x, y - подмножество мн-ва атрибутов.

x->y <=> любое x связано в точности с одним y (Биекция)

{Sno} -> {City}

детерминант - левая часть.

зависимая часть - правая.

Функциональную зависимость строим исходя из имеющихся данных. А не на основе какой-то логике.

Зависимая часть может содержать несколько значений. Детерминант тоже может содержать нес. значений.

Тривиальная функциональная зависимость - когда y явл. подмножеством x.

{Sno, Pno} -> {Sno}

{Sno, Qty} -> {Qty}

Множество всех функциональных зависимостей, которые задаются данным множеством ФЗ является **замыканием множества**.

Правило Амстронга

- Правило рефлексивности: Если B подможество A, то B функционально зависит от A (A->B)

- Правило дополнения: Если B функционально зависит от A (A->B), то AC->BC (Т.е. мы можем добавить абрибут справа и слева)
- Правило транзитивности: A->B, B->C => A->C
- (Выше были основные далее вытекают из ранее приведенных) Самоопределния: A->A
- Декомпозиции: A->BC => A->B и A->C
- Объединения: A->B и A->C => A->BC
- Композиция: A->B и C-> D => AC -> BD
- Общая теорема объединения: A->B и C->D => A(C-B)->BD

Пример (Задач, которые будут на РК)
1. R(A,B,C,D,E,F)
S = { A->BC,
B->E,
CD->EF }
Задача: AD->F ?
Решение:
1. A->BC => A->B и A->C
2. A->C => AD->CD
3, AD->CD И CD->EF => AD->EF
4. AD->EF => AD->F

Супер ключ - Супер ключ R - множество атрибутов R, котор. содержит в виде подмножества хотя бы один (не обязательно собственный) потенциальный ключ. (ключ с множество ключей с доп. атрибутами).

K - подмножество R
K->A для любого A принадлежащего R.

Алгоритм нахождения ключа
1. K = R
2. Для каждого атрибута из K выполняем след. действия

2.1 Вычислим замыкание K-A+
2.2 Если замыкание K-A+ = R, то K = K-A+
Алгоритм вычисления замыкания

1. $J_(new)$ = k
2. repeat
3. $J_(old) = J_(new)$
4. foreach (X->Y in S) do
5. J(X подмножество $J_(new)$) then $J_(new) = J_(new)$ + J
6. until ($J_(old) = J_(new)$)

Пример:
R(A,B,C,D,E,F) S = { A->BC,
E->CF,
B->E,
CD->EF }
Найти: A, B+ ?
Решение: A, B+ = A, B, C, E, F
ЕСЛИ БЫ ТАМ ЕЩЕ БЫЛО D, ТО ЭТО ЯВЛЯЛОСЬ БЫ ПОТЕНЦИАЛЬНЫМ КЛЮЧОМ.
Два множества ФЗ S1 и S2 явл **эквивалентными** тогда и только тогда когда они явл. покрытиями друг друга.
Пример:
Есть F - Набор ФЗ.
F = { A -> C,
AC -> D,
E -> AD,
E -> H }
G = { A -> CD,
E -> AH }
Задача: Доказать что они явлю эквивалентными (или не явл.)

Решение:

1. G - покрывает F ?

{A}+ = {A,C,D} (Строим по F) = A->CD (По G)

{E}+ = {E, A, D, H, C} (ПО F) = {E, A, H, C, D} (Множества совпадают, значит G покрывает F)

1. F - покрывает G ?

(По кому покрываем по тому и строим)

{A}+ = {A,C,D} (По G) = {A,C,D}

{AC}+ = {A,C,D} (По G) = {ACD}

{E}+ = {E, A, D, H, C} (По G) = {E, A, H, C, D} => F эквивалентно G.

Множество ФЗ явл. **неприводимым** тогда и только тогда, когда обладает след. свойствами:

1. Любая ФЗ X->Y, Y - один элемент
2. Ни одну ФЗ нельзя удалить без изменения замыкания
3. Ни один атрибут не может быть удален из детерминента без изменения замыкания

Примеры (Явл. min покрытием?):

1. S:

{Pno} -> {Pname, Color}

{Pno} -> {City}

Нарушает! =>

{Pno} -> {Pname}

{Pno} -> {Color}

2.

S:

Pno -> Pno

Pname -> Pname

Pno -> City

Нарушает! =>

Pno -> City

3. E - множество ФЗ.

min покрытие F (Проверяем все правила)
1. F = E
X -> {A1,...,An} =>
декомпозиция:
X - > A1
X - > An
2. Любая ФЗ X -> A из F
Любой B из X:
F - {X-A} объединение {(X-B) -> A} = F
3. Для каждой ФЗ, где A -> X
Мы проверяем
F - {X -> A} = F
Если так, то мы можем удалить.
Пример (Найти min покрытие):
R (A,B,C,D)
A -> BC
B -> C
A -> B
AB -> C
AC -> D
1. Разбиваем (Удаляем те ФЗ, которые выводимы)
A -> B
A -> C (Это можно удалить)
B -> C
A -> B (Это тоже можно удалить)
AB -> C
AC -> D
2. Разбиваем (Удаляем те ФЗ, которые выводимы)
Объединяем первые две и делаем композицию с третьей и получается выводим третью, значит ее можно удалить
A -> B
B -> C
AB -> C
AC -> D
3.

A -> B
B -> C
AC -> D
4.
A -> B
B -> C
A -> D

10

Семинар 4

Процедуры и функции.

Параметры - имя, тип.

Отличие: параметры у процедуры не берутся в скобки.

@ - локальная переменная.

@@ - глобальная переменная.

Параметры которые передаются по ссылки передаются с OUTPUT

Глубина рекурсии - 32.

create procedure [схема].имя <параметры>

[with <опции>]

as

<тело функции>

end;

Пример: (Факториал)

У postgres'а другие ключевые слова!

Листинг 10.1: Пример

```
create procedure dbo.Factorial @Valin bigint,
@ValOut bigint OUTPUT
as
begin
if @Valin > 20
begin
print N'Error'
```

```
return -99
end
-- Объявляем ' переменную '
declare @WorkValin bigint
-- Создаем ' переменную '
@WorkValout bigint
if @Valint != 1
begin
set @WarkValin = @Valin - 1
print @@NestedLevel
exec dbo.Factorial @WarkValIn, @WorkValout
set @ValOut = @WorkIN(WorkValOut)
end;
else
set @ValOut
end
```

Вызов процедуры:

Листинг 10.2: Пример

```
-- Определяем ' двепеременные  '
declare @FactIn
bigint
declare @Factout
bigint
set dbo.Factorial @Factin @FactOut OUTPUT
print convert(Varchar(20) @Factout)
```

Пишем истинную процедуру, которая ничего не возвращает, а лишь изменяет таблицу.

Имеем таблицу.

Задание: Добавить данные в таблицу. Добавить в конец еще одну строку.

Можно вставить python код.

$$ - тело процедуры в долларах

(В первую переменную (maxid) запишется max(id) + 1) (Во вторую (maxname) 'test' || max(id) + 1)

id	name
1	test1
2	test2
3	test3
...	...
n	testn

Таблица 10.1: Таблица

Листинг 10.3: Пример

```
create function addTest()
returns void
language PipgSQL
as $$
declare
        maxid int
        maxname Varchar(10)
begin
        select max(id) + 1, 'test' || max(id) + 1
        from test
        into maxid, maxname
        insert into test(id, name)
        values (maxis, maxName)
```

Либо вот так:

Листинг 10.4: Пример

```
create function addTest()
returns void
language PipgSQL
as $$
declare
        maxid int
        maxname Varchar(10)
begin
        select insert into test
```

```
        values ('||max(id)+1|', "test '||max(id)+1||'");
        from test
        into maxid, maxname
```

Запускаем:
query execute qSrting
Триггер - Объект. Ответ на событие. Реакция.
Виды:

1. DDl -триггеры (create, drop, alter)
2. DDM - триггер (insert, delete, update)
   (a) instead of (Вместо) (Кол-во 1)
   (b) before/alter Реагируют на какое-то действие и добавляют свое (делают действие до и после) (Вместе с) (Кол-во бесконечено)

Хар-ка	instead of	before/alter
тип	вместо	вместе с
кол-во	1	бесконечно
применение	таблица, представления	таблица

Таблица 10.2: Таблица

for - указываем на какие события мы реагируем.
as - после этого ключевого слова указывает действия, которые хотим сделать.

Листинг 10.5: Пример DDL триггера.

```
create trigger safety
on database
for drop_table, alter_table
as begin
        print N'Error'
        rollback
end;
```

Создадим триггер.

Листинг 10.6: Создадим триггер.

```
create trigger inserSP
on SP
after update
as
begin
        raiserror 'Новая подставка'
end;
```

10.1 Курсор

Курсор - это набор из результата sql запроса и указатель. (Зло). Это объект.

Используется - когда нужно выполнить что-то в цикле (к примеру удалть объекты)

Классификация

1. По области видимости:
   - Локальные
   - Глобальные
2. По типу:
   - Static (Статичный) (Требует доп. ресурса. Также таблица не должна менять)
   - Dynamic (Динамический). Позволяет отлеживать все изменения. (На это нужно огромные ресурсы. Тем самым время)
   - Key set - модификация динамического курсора.
   - Fast forward - всегда идем только вниз (просматриваем вниз)

3. По способу перемещения.

   - Forward_only Вниз
   - Scroll - Гуляем туда сюда.
   - По параллельному доступу.
     (a) read-only
     (b) Optimistic - разрешаем другим читать.
     (c) Scroll lock (Пессимистичная блокировка) - Не дает доступа другим.

Листинг 10.7: Курсор

```
declare 'имя курсора'
cursor
['область видимости']
['тип']
['Способ применения']
['Параллельный доступ']
for <sql запрос—>
```

Листинг 10.8: Курсор. Пример

```
declare myCursor cursor
for select Sname, Pname
from S join SP
on S.Sno = SP.Sno
join P on P.Pno = Sp.Pno

declare @Swork Varchar(10), @Pwork Varchar(10)

open myCursor

-- fetch - Запись ' переменной '
-- next - В ' каком ' порядке ''
fetch next from myCursor
```

```
into @Swork, @Pwork
-- - @@Fetch_status - 0 - больше ' не ' можем ' читать ' (EOF)
while @@Fetch_status == 0
begin fetch next from myCursor
into ...
end;
close myCursor
```

10.2 Индексы

Оптимизация запросов.

В основе лежит b_tree - сбалансированное дерево.

Кластеризованный индекс - Реальные данные в листе.

1. Можно создать 1 кластеризованный индекс
2. primary key / unique

Некластеризованный и ндекс - Если хранятся ссылки на объекты.

1. n штук
2. Создается по запросу

Листинг 10.9: Индекс

```
select *
from test
where id = 29
```

Листинг 10.10: Индекс. Создание

```
create nonclustered index my myId
on SP
include (Sno, Pno)
```

```
select *
from SP
where Qty=5
```

10.3 Партицирование

(partirion by)

Разбиение данных в таблице на подтаблицы.

11

РК1

11.1 Задание 2

Типы задания:

1. **Замыкание.**
Отношение:
R(A,B,C,D,E,F)
Заданы функциональные зависимости:
S = {
A –> BC,
AC –> DE,
D –> F,
E –> AB
}
Найти замыкание {A}+

ФЗ / Этап	A	A, B, C, D, E, F
A –> BC	A, B, C	A, B, C, D, E, F
AC –> DE	A, B, C, D, E	A, B, C, D, E, F
D –> F	A, B, C, D, E, F	A, B, C, D, E, F
E –> AB	A, B, C, D, E, F	A, B, C, D, E, F

Таблица 11.1: Таблица

Решение: Таблица 11.1.
Ответ {A}+ = {A, B, C, D, E, F}
Отношение:
R(A,B,C,D,E,F)
Заданы функциональные зависимости:
R(A,B,C,D,E,F)
S = {
A –> BC,
E –> CF,
B –> E,
CD –> EF
}
Найти: {A, B}+ ?
Решение: Таблица 11.2.

ФЗ / Этап	A, B	A, B, C, E	A, B, C, E, F
A –> BC	A, B, C	A, B, C, E	A, B, C, E, F
E –> CF	A, B, C	A, B, C, E, F	A, B, C, E, F
B –> E	A, B, C, E	A, B, C, E, F	A, B, C, E, F
CD –> EF	A, B, C, E	A, B, C, E, F	A, B, C, E, F

Таблица 11.2: Таблица

Ответ: {A, B}+ = {A, B, C, E, F}

2. **Неприводимое покрытие (min покрытие == ФЗ явл. неприводимым)**

Множество ФЗ явл. **неприводимым (min покрытие)** тогда и только тогда, когда обладает след. свойствами:

Детерминант - левая часть.

Зависимая часть - правая.

1. Для любого ФЗ X->Y, Y - один элемент.

2. Ни одну ФЗ нельзя удалить без изменения замыкания. (Пробуем удалить и смотрим на замыкание, поменялось?)

3. Ни один атрибут не может быть удален из детирменанта без изменения замыкания

Пример: Найти min покрытие:
R (A,B,C,D)
S = { A –> BC, B –> C, A –> B, AB –> C, AC –> D }
Решение: Таблица 11.3 - 11.4.
2.1 Удаляем зависимость A –> C потому что (выводима):
A –> B, B –> C => A –> C
2.2 Удаляем AB –> C Потому что (выводима):
A –> B, B –> C => AB –> BC
AB –> BC => AB –> C, AB –> B (Вывели)
3. Удаляем атрибут из AC –> D и пытаемся вывести AC –> D. (Удалили C):
A –> D, A –> C (Это у нас есть(можно вывести)) => A –> DC

R (A,B,C,D)	1. Раскрываем	2.1 Удаляем зависимости
A –> BC	A –> B	A –> B
B –> C	A –> C	B –> C
A –> B	B –> C	AB –> C
AB –> C	A –> B	AC –> D
AC –> D	AB –> C	
	AC –> D	

Таблица 11.3: Таблица

2.2 Удаляем зависимости	3. Удаляем атрибуты
A –> B	A –> B
B –> C	B –> C
AC –> D	A –> D

Таблица 11.4: Таблица

ПРИМЕР 2

Найдите неприводимое покрытие множества функциональных зависимостей S=AB–>D, B–>C, AE–>B, A–>D, D–>EF, заданных для переменной-отношения R(A, B, C, D, E, F).

23. Найдите неприводимое покрытие множества функциональных зависимостей `S={AB->D, B->C, AE->B, A->D, D->EF}`, заданных для переменной-отношения `R(A, B, C, D, E, F)`.

Используя алгоритм минимального покрытия, найдите минимальное покрытие
$F = \{AB \to D, B \to C, AE \to B, A \to D, D \to EF\}$
Шаг 1. Сделайте правые стороны атомными - $G = \{AB \to D,\ B \to C,\ AE \to B,\ A \to D,\ D \to E,\ D \to F\}$
Шаг 2. Удалите избыточные функциональные зависимости:
Для $AB \to D$ вычисляют $AB+$ исключая из G зависимость $AB \to D$
$AB+$ = ABCDEF
D входит в $AB+$, поэтому удалите $AB \to D$ из G
$G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$
Для $B \to C$ вычисляют $B+$ исключая из G зависимость $B \to C$: $B+ = B$
C не входит в$B+$ => $G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$
Для $AE \to B$ вычисляют $AE+$ исключая из G зависимость $AE \to B$: $AE+ = AEDF$
B не входит в$AE+$ => $G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$ Для A→ D вычисляют $A+$ исключая из G зависимость $A \to D$: $A+ = A$
D не входит в$A+$ => $G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$ Для D→ E вычисляют $D+$ исключая из G зависимость $D \to E$: $D+ = DF$
E не входит в$D+$ => $G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$
Для D→ F вычисляют $D+$ исключая из G зависимость $D \to F$: $D+ = DE\ F$невходит в$D+$ => $G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$
Шаг 3. Удалите все избыточные признаки с левой стороны ФЗ
$G = \{B \to C, AE \to B, A \to D, D \to E, D \to F\}$
Для $AE \to B$
Для A: вычислите $E+$ исключив из G $AE \to B$ и добавив $E \to B$
$E+$ with $\{B \to C,\ E \to B,\ A \to D,\ D \to E,\ D \to F\}$ = EBC => не содержит A, таким образом, A не избыточный атрибут в A
Для E: вычислите $A+$ исключив из G $AE \to B$ и добавив $A \to B$
$A+$ with $\{B \to C,\ A \to B,\ A \to D,\ D \to E,\ D \to F\}$ = $ABCDEF$ => содержит E, таким образом, E избыточный атрибут в AE
Минимальное покрытие = $\{B \to C, A \to B, A \to D, D \to E, D \to F\}$

Рис. 11.1: SQL

3. **Эквивалентность**

Два множества ФЗ S1 и S2 явл эквивалентными тогда и только тогда когда они явл. покрытиями друг друга.

Есть F - Набор ФЗ.

F = {
A –> C,
AC –> D,
E –> AD,
E –> H
}
G = {
A –> CD,

E –> AH

}

Задача: Доказать что они явл. эквивалентными (или не явл.)

Решение:

1. Проверим G покрывает F?

Найдем замыкание для каждой зависимой части (левой) из G:

{A}+ = A, C, D (Строим по F)

{E}+ = {E, A, D, H, C}(Строим по F) = {E, A, D, H, C}(Множества совпадают, значит G покрывает F)

1. Проверим F покрывает G?

Найдем замыкание для каждой зависимой части (левой) из F:

{A}+ = {A, C, D} (Строим по G)

{AC}+ = {A, C, D} (Строим по G)

{E}+ = {E, A, H, C, D} (Строим по G) = {E, A, H, C, D}

4. **Поиск потенциального ключа** (90%) / суперключа / всех ключей.

Потенциальный ключ (их мб несколько) является не избыточным (Нельзя что-то удалить).

Супер ключ (их мб несколько) - это потенциальный ключ с доп. атрибутами. (не всегда явл супер ключом)

Супер ключ - Супер ключ R - множество атрибутов R, котор. содержит в виде подмножества хотя бы один (не обязательно собственный) потенциальный ключ.

1. Найти все возможные ключи:

Схема отношений

R(A, B, C, D, E)

Набор ФЗ:

S = {A –> C, B –> D, C –> E, E –> A}

Чтобы найти все возможные ключи перечисляем наборы атрибутов.

{A, B, C, D, E}+ = {A, B, C, D, E}+ **Супер ключ**

Проверяем, можно ли удалить какой-то атрибут:

1. {A, B, C, D}+ = {A, B, C, D, E} (Т.к. == R, то это **Супер ключ**. Можно обойтись без E. Т.е. E - не явл. ключевым атрибутом.)

{A, B, C}+ = {A, B, C, D, E} **Супер ключ** (Аналогично) (D - лишнее)

{A, B}+ = {A, B, C, D, E} **Супер ключ и потенциальный ключ** (Аналогично) (CS - лишнее)

{A}+ = {A, C, E}+ (Это не полный набор нашей схемы. B - обязательно) - не ключ

{B}+ = {B, D}+ (Аналогично. A - обязательно) - не ключ.

2. {A, B, C, E}+ = {A, B, C, E, D} **Супер ключ**

{A, B, E}+ = {A, B, E, C, D}+ **Супер ключ**

{A, E}+ = {A, E, C}+ - (Это не полный набор нашей схемы. C - обязательно) - не ключ

{B, E}+ = {B, E, D, A, C}+ **Супер ключ и потенциальный ключ**

{B}+ = {B, D}+ - Не ключ

{E}+ = {E, A, C}+ - Не ключ.

И также продолжаем по аналогии...

3. {A, B, D, E}+ = ...

4. {A, C, D, E}+ = ...

5. {B, C, D, E}+ = ...

1. Найти потенциальные ключи:

R(A, B, C, D, E)

S = {A –> B, BC –> E, ED –> A}

Атрибуты, встречающиеся только в левой части: C,D – (входят во все потенциальные ключи).

Атрибуты, встречающиеся только в правой части: - (не входят в потенциальные ключи).

Атрибуты, не вошедшие в первые 2 группы (которые встречаются и там и там): A, B, E.

{C,D,A}+ = {C,D,A,B,E} - **Потенциальный ключ**

Проверка:

{D,A}+ = {D, A, B}

{C,A}+ = {C,A,B,E}

{D,A}+ = {D,A,B,}

{C,D,B}+ = {C,D,A,B,E,A}- **Потенциальный ключ**

Проверка: ...

{C,D,E}+ = {C,D,A,B,E,A} - **Потенциальный ключ**
Проверка: ...
5. **Выводимость зависимости (да / нет)**
Отношение:
R(A,B,C,D,E,F)
Заданы функциональные зависимости:
S = {
A –> BC,
B –> E,
CD –> EF
}
Задача: AD->F Выводимо?
Решение:

1. A –> BC => A –> B, A –> C
2. A –> C => AD –> CD
3. AD –> CD, CD –> EF => AD –> EF
4. AD –> EF => AD –> E, AD –> F

12

Семинар 5

12.1 Грамматика

- Атом - зарезервированные слова (и переменные)
- Категории.

Построение грамматики:
SFW - Select From Where.

13

Теория проектирования

Типы проектирования:

- Логическое
- Физическое

Аномалии:

- Вставки
- Обновления (Когда дублируется информация. В одном месте обновили в другом забыли)
- Удаления (Две слипшиеся сущности. Не можем по отдельности удалить.)

13.1 Формализация

Формализация - убирает аномалии.
Декомпозиция без потерь - это
Пример:
R(a, b, c)
Разбиваем:
$N = 2^i = 2^3 = 8$

1. {a}, {b}, {c}
2. {a}, {b, c}
3. {a, b}, {c}
...

Без потери - значит, что если объединим (сделаем JOIN) ничего не потеряем.

Нормальные формы:

- 1NF - каждая след. норма форма лучше предыдущей. Каждый атрибут явл. атомарным (нужно избавляться от массивов).
- 2NF - Должна быть первая норм форма (1NF) и каждый неключевой атрибут функционально полно неприводимо зависит от потенциального ключа
- 3NF - это 2NF и каждый неключевой атрибут нетривиально зависит от первичного ключа.
- BCNF (Бойса-Кодда) - это 3NF с улучшениями (оптимизация). Каждая нетривиальная и неприводимая слева функциональная зависимость имеет в кач-ве детерминанта перевичный ключ.
- 4NF - это 3NF или BCNF и все нетривиальные многозначные зависимости функционально зависят от потенциальных ключей.
- 5NF - 4NF и нетривиальная зависимость соединения определяется потенциальным ключом.
- 6NF - находится в 5NF и больше ее декомпозицировать нельзя.

Денормализация - обратный процесс нормализации. Объединяем таблицы.