

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	Партиционирование	43

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 несуществующие 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. (≥ 3 объектов (таблицы связи 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. Настройка.

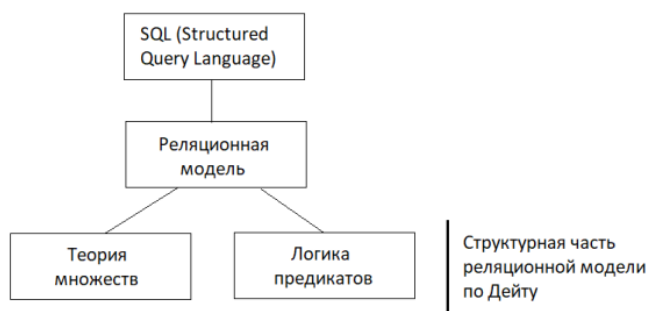


Рис. 3.1: SQL

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.

Заголовок отношения				
Атрибут				
1	2	3	4	5
aaa	aba	abb	bba	bab
bbb	bcb	bcc	ccb	cbc
ccc	cdc	cdd	ddc	dcd
ddd	ded	dee	eed	ede

Кортеж

Тело

отношения

Рис. 3.2: Пример таблицы.

3.2 Основные языки

Логику работы с данными можно разделить на три основных языка:

- DDL (*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 (grant)
 - Удаление прав доступа на объект (revoke)

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

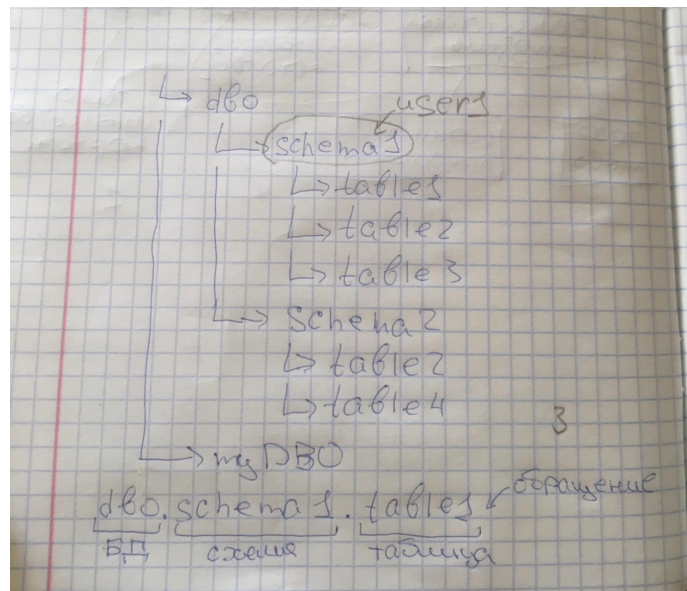


Рис. 3.3: Структура БД.

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 - АТТРИБУТ типа int счетчик шагаем начиная с единицы с шагом 1.

EmployeeId - поле, которое используем в кач-ве идентификатора.

Photo - По умолчанию NULL - пустой.

MyRowGuidColumn - 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 числе, 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 R_2 – это непустое подмножество множества атрибутов FK этого отношения, такое, что:

- Существует отношение R_1 (причем отношения R_1 и R_2 обязательно различны) с потенциальным ключом $СК$;
- Каждое значение внешнего ключа FK в текущем значении отношения R_2 обязательно совпадает со значением ключа $СК$ некоторого кортежа в текущем значении отношения R_1 .

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 ) WHERE 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 WHERE 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 поставяет 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 Р.

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 Р.

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. `group 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
group 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 А)
 - right join (На кругах это весь круг В)
 - full join(полное) (На кругах это оба круга (и А и В))

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 $n \cdot n$). 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> - добавление 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 | AVG | SUM | ALL | ANY

7.2 Исчисление кортежей

$\langle \text{объявление кортежной переменной} \rangle ::= \text{RANGE OF } \langle \text{переменная} \rangle \text{ IS } \langle \text{список областей} \rangle$

$\langle \text{область} \rangle ::= \langle \text{отножение} \rangle \mid \langle \text{рел выражение} \rangle$

$\langle \text{рел выр} \rangle ::= (\text{список целевых элементов}) [\text{WHERE wff}]$

$\langle \text{целевой элемент} \rangle ::= \text{переменная} \mid \text{переменная атрибут} [\text{AS } \langle \text{псевдоним} \rangle]$

правильно построенная функция $\text{wff} ::= \text{условие} \mid \text{NOT wff} \mid \text{условие AND wff} \mid \text{if условие then wff} \mid \text{exists переменная (wff)} \mid \text{FORALL переменная (wff)} \mid (\text{wff})$

8

Семинар 3

8.1 Объекты БД

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

```

(с) 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 int)
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.

$x \rightarrow y \iff$ любое x связано в точности с одним y (Биекция)

$Sno \rightarrow City$

детерминант - правая часть. зависимая часть - левая.

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

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

Тривиальная функциональная зависимость - когда y явл. подмножеством x .

$Sno, Pno \rightarrow Sno$ $Sno, Qty \rightarrow Qty$

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

Правило Амстронга

- Правило рефлексивности: Если B подмножество A , то B функционально зависит от A ($A \rightarrow B$)

- Правило дополнения: Если В функционально зависит от А ($A \rightarrow B$), то $AC \rightarrow BC$ (Т.е. мы можем добавить атрибут справа и слева)
- Правило транзитивности: $A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$
- (Выше были основные далее вытекают из ранее приведенных)
Самоопределения: $A \rightarrow A$
- Декомпозиции: $A \rightarrow BC \Rightarrow A \rightarrow B$ и $A \rightarrow C$
- Объединения: $A \rightarrow B$ и $A \rightarrow C \Rightarrow A \rightarrow BC$
- Композиция: $A \rightarrow B$ и $C \rightarrow D \Rightarrow AC \rightarrow BD$
- Общая теорема объединения: $A \rightarrow B$ и $C \rightarrow D \Rightarrow A(C-B) \rightarrow BD$

Пример (Задач, которые будут на РК)

1. $R(A, B, C, D, E, F)$

$S = \{ A \rightarrow BC,$

$B \rightarrow E,$

$CD \rightarrow EF \}$

Задача: $AD \rightarrow F$?

Решение:

1. $A \rightarrow BC \Rightarrow A \rightarrow B$ и $A \rightarrow C$

2. $A \rightarrow C \Rightarrow AD \rightarrow CD$

3, $AD \rightarrow CD$ И $CD \rightarrow EF \Rightarrow AD \rightarrow EF$

4. $AD \rightarrow EF \Rightarrow AD \rightarrow F$

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

K - подмножество R

$K \rightarrow A$ 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 ($X \rightarrow Y$ in S) do
5. $J(X \text{ подмножество } J_{(new)})$ then $J_{(new)} = J_{(new)} + J$
6. until ($J_{(old)} = J_{(new)}$)

Пример:

$R(A,B,C,D,E,F)$ $S = \{ A \rightarrow BC,$
 $E \rightarrow CF,$
 $B \rightarrow E,$
 $CD \rightarrow EF \}$

Найти: A, B^+ ?

Решение: $A, B^+ = A, B, C, E, F$

ЕСЛИ БЫ ТАМ ЕЩЕ БЫЛО D , ТО ЭТО ЯВЛЯЛОСЬ БЫ ПО-
ТЕНЦИАЛЬНЫМ КЛЮЧОМ.

Два множества ФЗ S_1 и S_2 явл **эквивалентными** тогда и только
тогда когда они явл. покрытиями друг друга.

Пример:

Есть F - Набор ФЗ.

$F = \{ A \rightarrow C,$

$AC \rightarrow D,$

$E \rightarrow AD,$

$E \rightarrow H \}$

$G = \{ A \rightarrow CD,$

$E \rightarrow AH \}$

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

Решение:

1. G - покрывает F ?

$\{A\}^+ = \{A, C, D\}$ (Строим по F) = $A \rightarrow CD$ (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) = $\{E, A, H, C, D\} \Rightarrow F$ эквивалентно G.

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

1. Любая ФЗ $X \rightarrow Y$, 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:

$\{Pno\} \rightarrow \{Pname, Color\}$

$\{Pno\} \rightarrow \{City\}$

Нарушает! \Rightarrow

$\{Pno\} \rightarrow \{Pname\}$

$\{Pno\} \rightarrow \{Color\}$

2.

S:

$Pno \rightarrow Pno$

$Pname \rightarrow Pname$

$Pno \rightarrow City$

Нарушает! \Rightarrow

$Pno \rightarrow City$

3. E - множество ФЗ.

min покрытие F (Проверяем все правила)

1. $F = E$

$X \rightarrow \{A_1, \dots, A_n\} \Rightarrow$

декомпозиция:

$X \rightarrow A_1$

$X \rightarrow A_n$

2. Любая ФЗ $X \rightarrow A$ из F

Любой B из X:

$F - \{X-A\}$ объединение $\{(X-B) \rightarrow A\} = F$

3. Для каждой ФЗ, где $A \rightarrow X$

Мы проверяем

$F - \{X \rightarrow A\} = F$

Если так, то мы можем удалить.

Пример (Найти min покрытие):

R (A,B,C,D)

$A \rightarrow BC$

$B \rightarrow C$

$A \rightarrow C$

$AB \rightarrow C$

$AC \rightarrow D$

1. Разбиваем (Удаляем те ФЗ, которые выводимы)

$A \rightarrow B$

$A \rightarrow C$ (Это можно удалить)

$B \rightarrow C$

$A \rightarrow B$ (Это тоже можно удалить)

$AB \rightarrow C$

$AC \rightarrow D$

2. Разбиваем (Удаляем те ФЗ, которые выводимы)

Объединяем первые две и делаем композицию с третьей и получается выводим третью, значит ее можно удалить

$A \rightarrow B$

$B \rightarrow C$

$AB \rightarrow C$

$AC \rightarrow D$

3.

$A \rightarrow B$
 $B \rightarrow C$
 $AC \rightarrow D$
4.
 $A \rightarrow B$
 $B \rightarrow C$
 $A \rightarrow D$

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 ссылке передаются с OUTPUT

Глубина рекурсии - 32.

create procedure [схема].имя <параметры>

[with <опции>]

as

<тело функции>

end;

Пример: (Факториал)

У postgres'a 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 ' две переменные '
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 (maxid, 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 бесконечно)

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
    raiseerror '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 удалить объекты)

Классификация

1. По области видимости:

- Локальные
- Глобальные

2. По типу:

- Static (Статичный) (Требует доп. ресурса. Также таблица не должна менять)
- Dynamic (Динамический). Позволяет отслеживать 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.

Класторизованный индекс - Реальные данные в листе.

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

Некласторизованный индекс - 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

(partition by)

Разбиение данных в таблице на подтаблицы.