Университет машиностроения

Кафедра «Автоматика и процессы управления»

Дисциплина

# Информационные технологии

1 семестр

Тема 13

# Базы данных и СУБД

# Применение SQL

# Запрос на выборку

# SELECT *smth*

# Запрос на выборку
## SELECT *smth.*

```
SELECT expr1,..., exprN
FROM source1,..., sourceN
[ [ INNER ] JOIN othersource [ON join_predicate1 ] ]
[ WHERE predicate2 ]
[ GROUP BY group_expr1,..., group_exprN ]
[ HAVING group_predicate ]
[ ORDER BY field1,..., fieldN [ ASC | DESC ] ]
[ LIMIT [ offset, ] rowcount ]
```

# Запрос на выборку
## SELECT *smth.*

# Оператор WHERE

# WHERE

- Условный оператор в **SQL**
- Используется для отбора записей – указывает оператору языка управления данными (**DML**) записи, на которые он действует
- Не обязательный в **SQL** (**DML**) выражениях
- В выражениях с **SELECT**, **DELETE**, **UPDATE** предваряет критерии отбора данных
- Критерии отбора должны быть записаны в форме предикатов булевского вида (***TRUE***, ***FALSE*** или ***NULL***)
- Действует на исходный набор записей (до группировки)

# Применение **WHERE**

**SQL-DML-**выражение

**FROM** *table_name*

**WHERE** *predicate*

Все записи, для которых значением предиката является **истина** – будут задействованы (возвращены)

Записи, для которых значением предиката является **ложь** или **неопределённость** (**NULL**) – будут исключены из обработки (выборки)

**DELETE**

**FROM** *mytable*

**WHERE** *mycol* **IS NULL OR** *mycol = 100*

# Запрос на выборку
## SELECT *smth.*

# Оператор HAVING

# HAVING

- Условный оператор (параметр) в **SQL**
- Указывает условия на результат агрегатных функций (**MAX**, **SUM**, **AVG**,…)
- Предикаты строятся только из выражений, указанных в разделе **GROUP BY** и значений агрегатных функций, вычисленных для каждой группы, образованной **GROUP BY**
- Необходимо, чтобы в **SELECT** были заданы только столбцы, перечисленные в **GROUP BY** и/или агрегированные значения
- Если параметр **GROUP BY** не указан в **SELECT**, **HAVING** дублирует **WHERE**

```
SELECT  DeptID, SUM(SaleAmount)
FROM  Sales
WHERE SaleDate = '01-Jan-2000'
GROUP BY DeptID
HAVING  SUM(SaleAmount) > 1000
```

*Получение идентификаторов отделов, продажи которых превысили 1000 за 1 января 2000 года*

```
SELECT  d.DeptName, COUNT(*)
FROM  employee e, departament d
WHERE e.DeptID = d.DeptID
GROUP BY d.DeptName
HAVING  COUNT (*) > 1000
```

*Получение списка отделов, в которых работает более чем один сотрудник*

# Запрос на выборку
SELECT *smth*
WHERE *predicate*

# Построение логических предикатов

# Построение предиката для WHERE

- Условный предикат – выражение, которое должно возвращать значения **TRUE**, **FALSE** или **NULL**
- Значение **NULL** возвращается если арифметическая, логическая операция или операция сравнения выполняется над операндом со значением **NULL**), за исключением операций проверки на пустое значение (**EXISTS**, **IS NULL**)
- Выражения, используемые для построения предиката могут использовать парные круглые скобки [ **(** , **)** ] любой степени вложенности
- Для выполнения логических операций над

# Построение предиката для WHERE (2)

- Условный предикат может включать подзапросы
- В выражениях могут использоваться функции определённые в SQL (**NOW**, **YEAR** *etc.*) или определённые в базе данных (**CREATE FUNCTION**…)
- Для проверки значения можно использовать: **LIKE** –для сравнения с шаблоном, **IS** – для специальных значений, **IN** – для вхождения в список
- Контроль вхождения значения в заданный диапазон выполняется с использованием **BETWEEN … AND …**
- (в Access) Для построения выражений в роли констант могут использоваться статистические функции по подмножеству для вычисления вспомогательных числовых значений (**DSum** , **DAvg**, **DMax**, **DMin**, **Dcount** *etc.*)

```
SELECT  ProductID, Name, Color
FROM Production
WHERE Name LIKE ("%Frame%")
    AND ProductID <= 12
    AND Color = "Red" ;
```

*Совпадение с шаблоном, сравнение, объединение условий*

```
SELECT  ProductID, Name, Color
FROM Production
WHERE Name IN ("Blade", "Crown Race",
 "Spokes");
```

*Наличие в списке значений*

```
SELECT  ProductID, Name, Color
FROM Production
WHERE ProductID BETWEEN 725 AND 734;
```

*Вхождение в диапазон между двумя значениями*

# Запрос на выборку
## SELECT *smth*

# Оператор LIKE

# LIKE

- Проверяет символьную строку на совпадение с заданным шаблоном
- Используется как часть выражения **WHERE *smth***

match_expression [ **NOT** ] **LIKE** pattern **ESCAPE** esc_char

| Символ-шаблон | Описание | Пример |
|---|---|---|
| % * | Любая строка длиной от нуля и более символов. | Инструкция WHERE Название LIKE '%компьютер%' выполняет поиск и выдает все названия книг, содержащие слово «компьютер». |
| _ ? | Любой одиночный символ. | Инструкция WHERE фамилия_автора LIKE '_етров' выполняет поиск и выдает все имена, состоящие из шести букв и заканчивающиеся сочетанием «етров» (Петров, Ветров и т.п.). |
| [ ] | Любой одиночный символ, содержащийся в диапазоне ([a-f]) или наборе ([abcdef]). | Инструкция WHERE Фамилия_автора LIKE '[Л-С]омов' выполняет поиск и выдает все фамилии авторов, заканчивающиеся на «омов» и начинающиеся на любую букву в промежутке от «Л» до «С», например Ломов, Ромов, Сомов и т.п. |
| [^] | Любой символ, не содержащийся в диапазоне ([^a-f]) или наборе ([^abcdef]). | Инструкция WHERE Фамилия_автора LIKE 'ив[^а]%' выполняет поиск и выдает все фамилии, начинающиеся на «ив», в которых третья буква отличается от «а». |

```
SELECT p.FirstName, p.LastName, ph.PhoneNumber
FROM Person.PersonPhone AS ph
INNER JOIN Person.Person AS p
ON ph.BusinessEntityID = p.BusinessEntityID
WHERE ph.PhoneNumber LIKE '415%'
ORDER BY p.LastName;
```

| FirstName | LastName | Phone |
|---|---|---|
| Ruben | Alonso | 415-555-124 |
| Shelby | Cook | 415-555-0121 |
| Karen | Hu | 415-555-0114 |
| John | Long | 415-555-0147 |
| David | Long | 415-555-0123 |

*Получение списка телефонных номеров с кодом города 415*

# Запрос на выборку
## SELECT *smth*

# Операции с множествами

# Операции с множествами в WHERE

- Множества (списки значений) в предикатах оператора **WHERE** могут использоваться для проверки наличия заданного значения (в списке) или для выполнения однотипной операции сравнения с элементами списка
- Для проверки наличия значения в списке можно использовать оператор **IN (…)**
- Проверка списка на наличие элементов выполняется предикатом **EXISTS (…)**
- Операция сравнения значения со всеми элементами списка составляется с использованием предикатов **ANY (…)**, **SAME (…)** и **ALL (…)**. **ANY** эквивалентно объединению выражений посредством **OR**, а **ALL** – с помощью **AND**
- В некоторых случаях пустой список может выступать эквивалентом «пустого» значения (**NULL**)

```
SELECT  ProductID, Name, Color
FROM Production
WHERE Vendor IN ("Toyota", "Nissan",
 "Mitsubishi", "Mazda");
```

*Совпадение со значением из списка*

```
SELECT  ProductID, Name, Color
FROM Production
WHERE EXISTS (   SELECT ProductID
                 FROM Production
                 WHERE Vendor ="Ford");
```

*Наличие в списке значений*

```
SELECT  ProductID, Name, Color
FROM Production
WHERE ProductID = ANY ( 725 , 730 , 734 );
```

*Равенство любому значению из списка*

# Запрос на выборку
## SELECT *smth*

# Вложенные запросы (подзапросы)

# Использование подзапросов в WHERE

- Условный предикат может включать подзапрос
- В составе предиката подзапрос заключается в круглые скобки [ **(** , **)** ]
- Вложенный запрос (подзапрос) должен возвращать скалярное (одно) логическое значение (или **NULL**) либо выступать операндом в операции, возвращающем подобное значение
- Операции с подзапросами, возвращающими список значений, должны использовать предикаты **ANY** (**SOME**) и **ALL** для группового сопоставления, **EXISTS** для проверки наличия результатов или **IN** для анализа вхождения значения

```
SELECT  ProductID, Name, Color
FROM Production
WHERE EXISTS (   SELECT ProductID
                 FROM Production );
```

*Наличие в списке хотя бы одного результата*

```
SELECT  ProductID, Name, Color
FROM Production
WHERE Cost >= (  SELECT  AVG( Cost )
                 FROM Production );
```

*Сравнение с единственным возвращаемым значением*

```
SELECT  Name, NCost
FROM  ( SELECT Name, Cost * 1.15 AS NCost
        FROM Production )
WHERE ProductID >= 725  );
```

*Использование подзапроса как источника данных*

# Запрос на выборку
## SELECT *smth*

# Оператор JOIN

# JOIN

- Оператор **SQL** реализующий операцию соединения реляционной алгебры для раздела **FROM**
- В схему таблицы-результата входят столбцы обеих таблиц-операндов («сцепление» схем операндов)
- Каждая строка таблицы-результата является «сцеплением» строк таблиц-операндов
- Результирующий набор строк зависит от типа ***операции соединения*** и ***условия соединения***
- При необходимости соединения нескольких таблиц операция соединения применяется несколько раз (последовательно)

**SELECT** *expessions [,... n]*

**FROM** *table1*

**[ INNER | [ LEFT | RIGHT | FULL ] OUTER | CROSS ] JOIN**

*table2*

**[ ON** *condition* **]**

Для перекрёстного соединения (декартова произведения) CROSS JOIN можно использовать (,)

**SELECT** *expessions [,... n]*

**FROM** *table1,…, tableN*

# Виды оператора **JOIN**

- **INNER JOIN** – внутреннее соединение таблиц
- **OUTER JOIN** – внешнее соединение таблиц
- **LEFT OUTER JOIN** – левое внешнее соединение таблиц (несимметричное)
- **RIGHT OUTER JOIN** – правое внешнее соединение таблиц (несимметричное)
- **FULL OUTER JOIN** – полное внешнее соединение таблиц
- **CROSS JOIN** – перекрёстное соединение таблиц

***Исходные данные***

***Person***

| | |
|---|---|
| Андрей | 1 |
| Леонид | 2 |
| Сергей | 1 |
| Григорий | 4 |

***Address Book***

***City***

| | |
|---|---|
| 1 | Москва |
| 2 | Санкт-Петербург |
| 3 | Казань |

# INNER JOIN

Внутреннее соединение таблиц

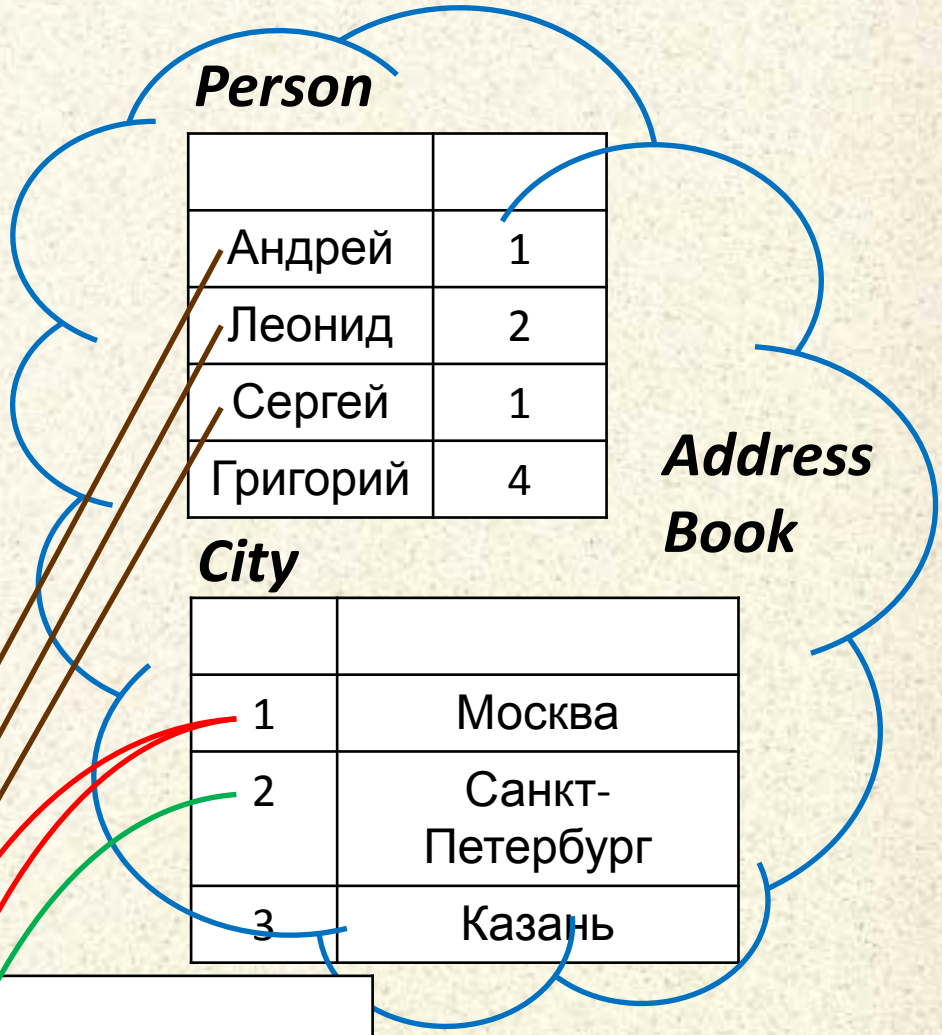
Симметричный оператор

Результат – таблица из соединённых строк таблиц-операндов по предикату

**SELECT** * **FROM** *Person*

**INNER JOIN** *City*

**ON** *Person.CityId* = *City.Id*




***Person-City***

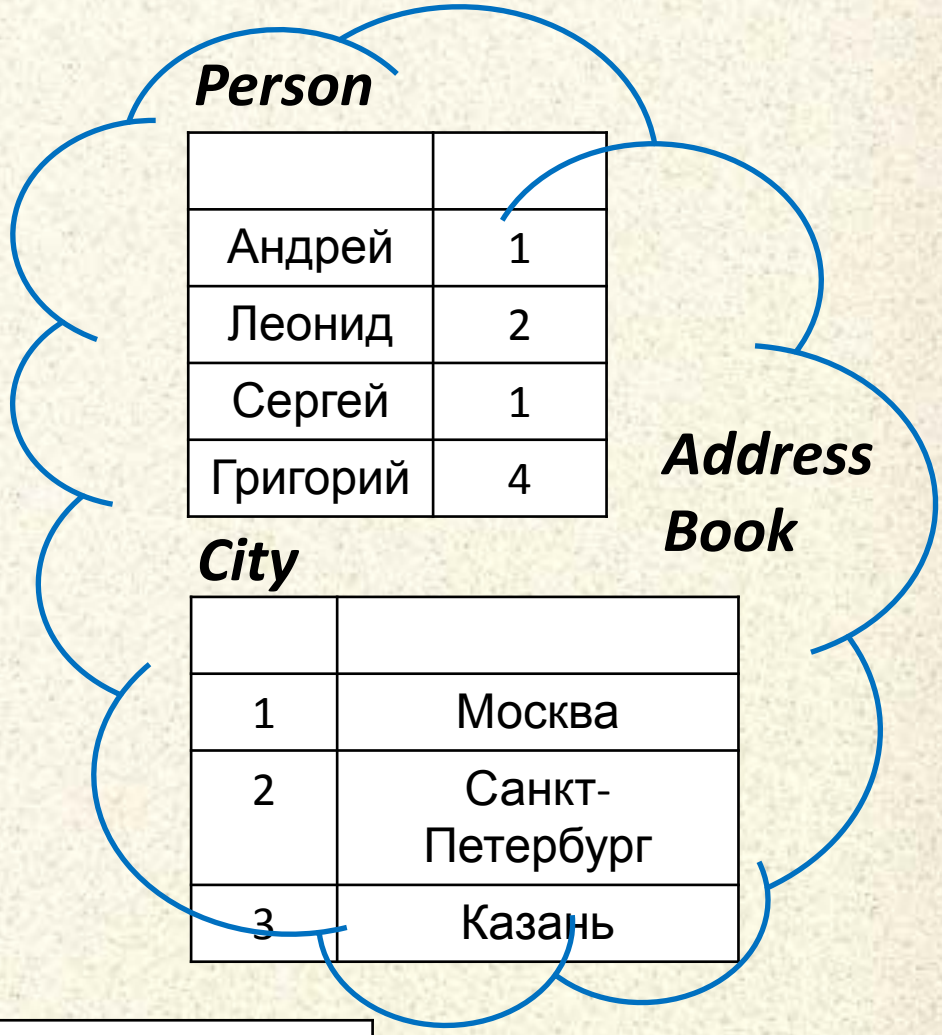
| Person.Name | Person.CityID | | |
|---|---|---|---|
| Андрей | 1 | 1 | Москва |
| Леонид | 2 | 2 | Санкт-Петербург |
| Сергей | 1 | 1 | Москва |

# LEFT OUTER JOIN

Левое внешнее соединение таблиц

Несимметричный оператор

Результат – таблица из соединённых строк таблиц-операндов по предикату дополненных оставшимися строками левой таблицы (дополняются *NULL*)




| | |
|---|---|
| Андрей | 1 |
| Леонид | 2 |
| Сергей | 1 |
| Григорий | 4 |

| | |
|---|---|
| 1 | Москва |
| 2 | Санкт-Петербург |
| 3 | Казань |

```
SELECT * FROM Person
   LEFT OUTER JOIN City
   ON Person.CityId = City.Id
```

| Person.Name | Person.CityID | | |
|---|---|---|---|
| Андрей | 1 | 1 | Москва |
| Леонид | 2 | 2 | Санкт-Петербург |
| Сергей | 1 | 1 | Москва |
| Григорий | 4 | NULL | NULL |

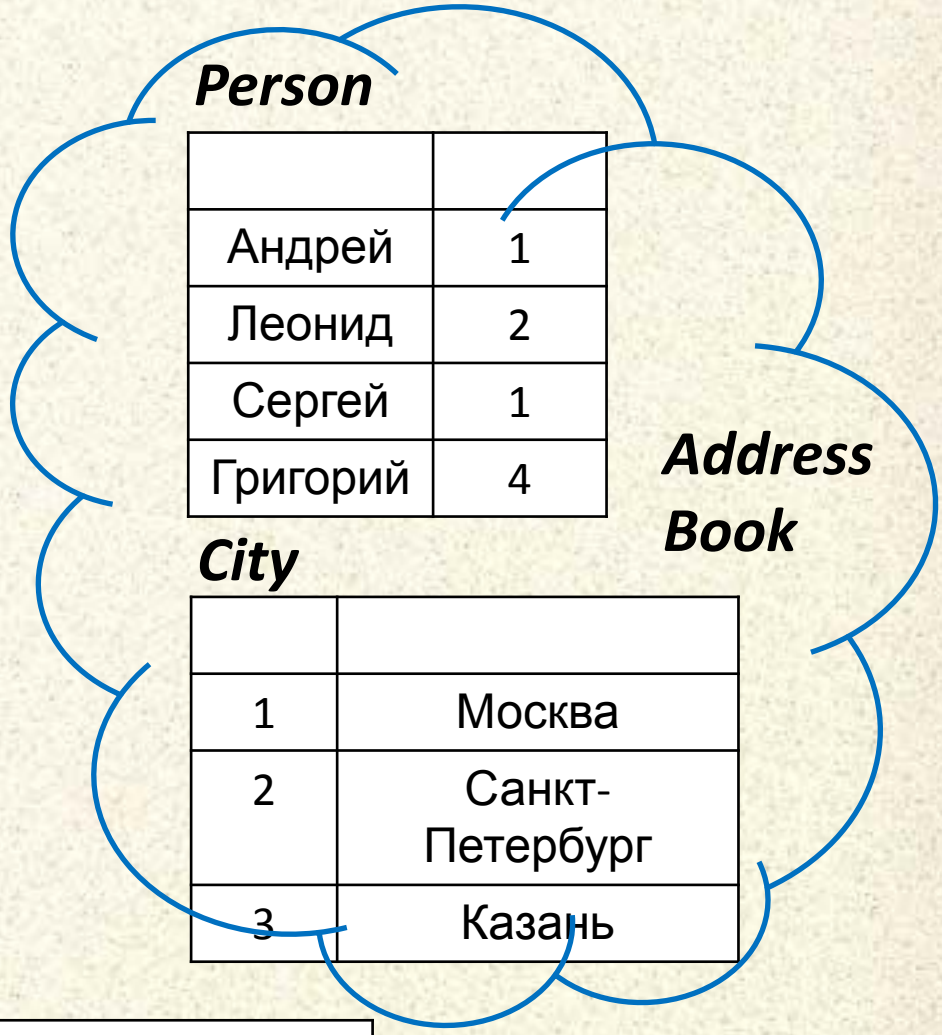
# RIGHT OUTER JOIN

Правое внешнее соединение таблиц

Несимметричный оператор

Результат – таблица из соединённых строк таблиц-операндов по предикату дополненных оставшимися строками правой таблицы (дополняются *NULL*)

```
SELECT * FROM Person
RIGHT OUTER JOIN City
ON Person.CityId = City.Id
```



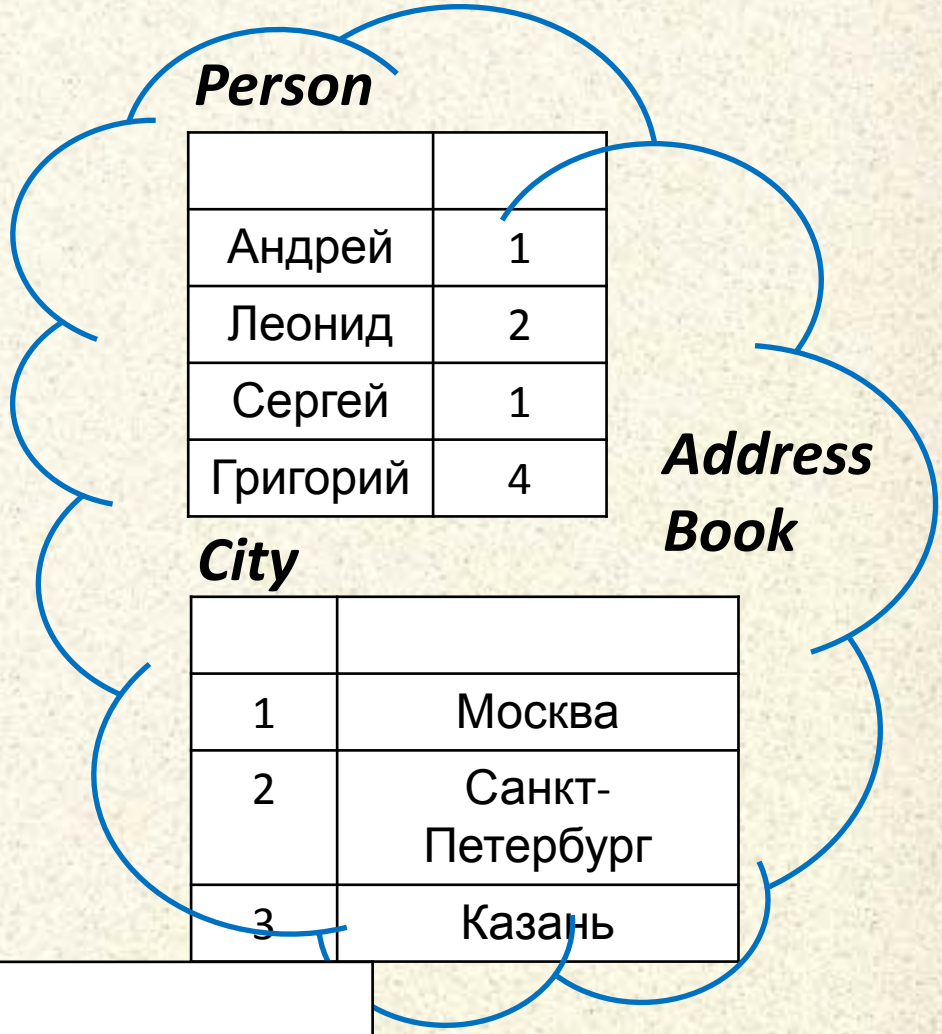

| Person.Name | Person.CityID | | |
|---|---|---|---|
| Андрей | 1 | 1 | Москва |
| Леонид | 2 | 2 | Санкт-Петербург |
| Сергей | 1 | 1 | Москва |
| NULL | NULL | 3 | Казань |

# FULL OUTER JOIN

Полное внешнее соединение таблиц

Симметричный оператор

Результат – таблица из соединённых строк таблиц-операндов по предикату дополненных оставшимися строками обеих таблиц (дополняются *NULL*)




```
SELECT * FROM Person
FULL OUTER JOIN City
  ON Person.CityId = City.Id
```

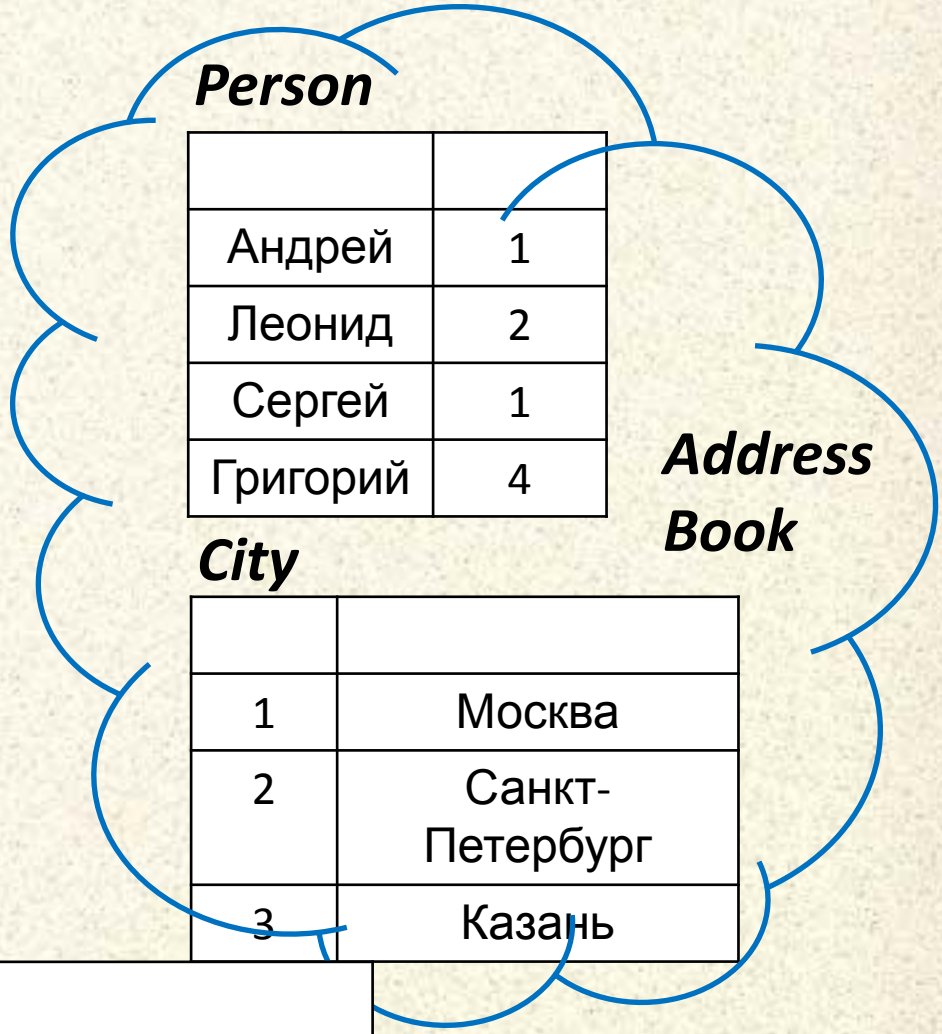
| Person.Name | Person.CityID | | |
|---|---|---|---|
| Андрей | 1 | 1 | Москва |
| Леонид | 2 | 2 | Санкт-Петербург |
| Сергей | 1 | 1 | Москва |
| NULL | NULL | 3 | Казань |
| Григорий | 4 | NULL | NULL |

# CROSS JOIN

Перекрёсное соединение таблиц

Симметричный оператор

Результат – таблица из соединённых строк таблиц-операндов, давая все возможные сочетания строк двух таблиц

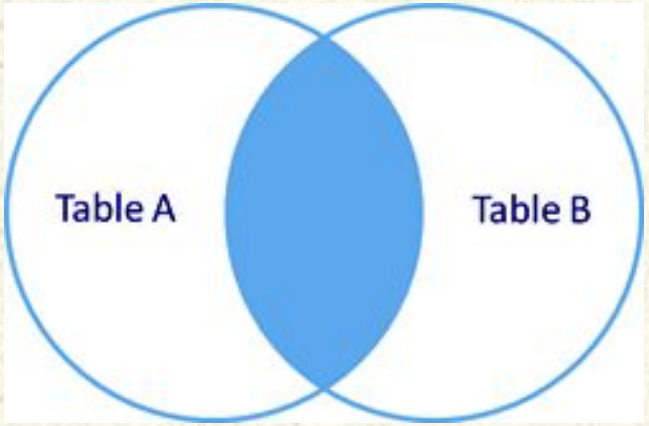



**SELECT * FROM** *Person*
**CROSS JOIN** *City*
**[ WHERE** *predicate* **]**

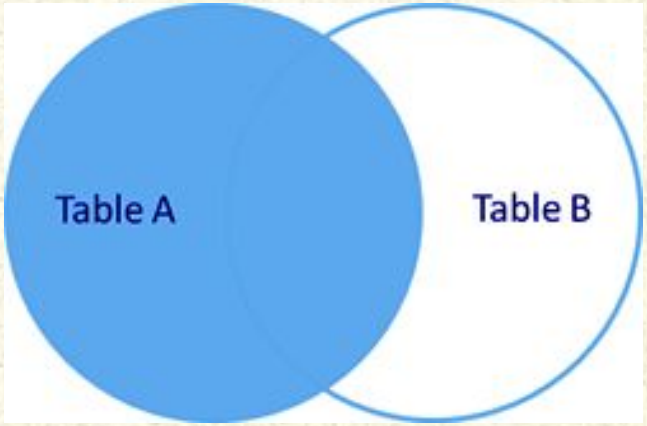
| Person.Name | Person.CityID | | |
|---|---|---|---|
| Андрей | 1 | 1 | Москва |
| Андрей | 1 | 2 | Санкт-Петербург |
| Андрей | 1 | 3 | Казань |
| Леонид | 2 | 1 | Москва |
| … | … | … | … |

# Схемы объединения множеств

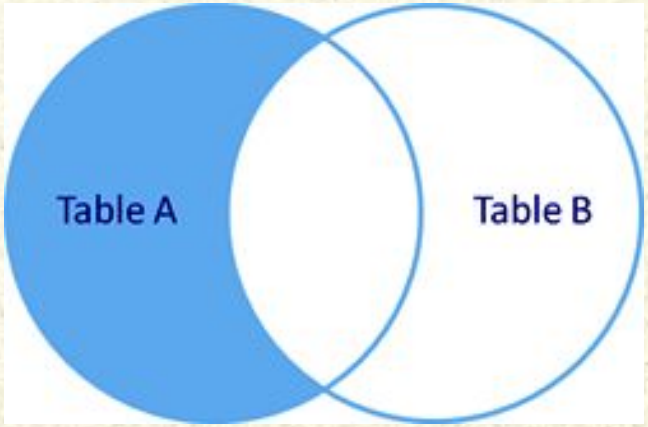
## INNER JOIN




## LEFT OUTER JOIN




## LEFT OUTER JOIN с фильтрацией



# Запрос на выборку
## SELECT *smth*

# Оператор UNION

# UNION

- Оператор объединения результатов двух SQL-запросов в единую таблицу, состоящую их схожих строк
- Оба запроса должны возвращать одинаковое число столбцов с совместимые типы данных

*<запрос1>*
**UNION [ALL]**
*<запрос2>*
**UNION [ALL]**
*<запрос3>*
*…..;*

- В объединение не включаются (скрываются) повторяющиеся строки
- Оператор **ALL** требует включение всех строк в результирующий набор

# UNION

Пример

# Запрос на выборку
## SELECT *smth*

# Групповые запросы

# Запрос на выборку
## SELECT *smth*

# Статистические функции по подмножеству

# Статистические функции по подмножеству

**DAvg** – подсчет среднего арифметического значения столбца или выражения,

**DCount** – подсчет количества записей,

**DFirst** – нахождение первого значения столбца из группы,

**DLast** – нахождение последнего значения столбца из группы,

**DMax** – определение максимального значения столбца или выражения,

**DMin** – определение минимального значения столбца или выражения,

**DSum** – подсчет суммы значений столбца или выражения.

Синтаксис операторов следующий:

**<имя_функции> ("выражение";"источник";"критерий")**

# Статистические функции по подмножеству

```
Select fio, score, groupID
From students
Where score > DAvg ("God",
                    "students",
                    "groupID<>1")
      and groupID <> 1 ;
```

**Важно!** Параметры запроса записываются как **строковые литералы** (в кавычках)

# Недокументированная функциональность СФпП

**Существует возможность использования переменных выражений в качестве параметров статистических функций по подмножеству**

| fio | score | subject | | |
|---|---|---|---|---|
| Иванов И.И. | 4,5 | math | 4 | 5 |
| Петров П.П. | 3,8 | math | 4 | 3 |
| Сидоров С. С. | 4,0 | rus | 5 | 4 |
| Кузнецов К. К. | 4,8 | rus | 5 | 4 |

```
Se
From students
Where score >= DAvg (subject,"students”);
```

| fio | score |
|---|---|
| Иванов И.И. | 4,5 |
| Сидоров С. С. | 4,0 |
| Кузнецов К. К. | 4,8 |

# Запрос на выборку
## SELECT *smth*

# Перекрестные запросы

# Перекрестные запросы

```
Transform <итоговые функции>
Select <заголовки строк и итоги по
  строкам>
From <источник данных>
Group by <заголовки строк>
Pivot <заголовки столбцов>;
```

```
Transform Sum(God) As Sum-God
Select Izd, Sum(god) as SumGod, Count(*) asKolFirm
From Firm
Group by Izd
Pivot Nazf;
<итоговые функции>
```

Университет машиностроения
Кафедра «Автоматика и процессы управления»

Блок дисциплин

# Информатика и информационные технологии

## Далее:

- *Использование Access 2007*
- *Табличные процессоры*
- *Текстовые процессоры*

*Контакты:*

mami.ru/index.php?id=466

timid@mami.ru

inform437@gmail.com