# Лабораторная работа №4. Оператор JOIN

## Краткие теоретические данные.

Join – оператор объединения. JOIN в SQL используется при построении select выражений. Инструкция Join позволяет объединить колонки из нескольких таблиц в одну. Объединение происходит временное и целостность таблиц не нарушается. Существует три типа join-выражений:

inner join;

outer join

---left,

---right

---full

cross join;

**Внутреннееобъединение INNER JOIN**

Внутреннее объединение INNER JOIN (синоним JOIN, слово INNER можно опустить).

Выбираются только совпадающие данные из объединяемых таблиц.

Синтаксис

*SELECT<список выбираемых столбцов>*

*FROM <Table A>*

*INNERJOIN<TableB>ON<условиеобъединения>*

**Пример 4.1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Пусть даны:*  *- таблицы t1, содержащая данные о id студента, фамилию, средний балл, id группы, в которой студент учится;*  *- таблицы t2, содержащая данные о id группы, названии группы, факультете*   |  |  |  |  |  | | --- | --- | --- | --- | --- | | t1 | | | |  | | Id | group\_id | Faculty | Name | Exm | | 1 | 1 | ФПК | Ахрем |  | | 2 | 2 | ФПК | Петров |  | | 3 | 3 | ФПИ | Иванов |  |  |  |  |  | | --- | --- | --- | | **Id** | **Name\_gr** | **Faculty** | | 1 | по1701 | ФПК | | 2 | по1705 | ФПК | | 3 | по1708 | ФПИ | | 4 | по1709 | ФПИ |   Необходимо сформировать запрос, результатом которого будет выборкаc информацией о студентах, факультетах, группах, оценках  *SELECT \* FROM tbl2*  *JOIN tbl1*  *ON tbl1.id\_gr = tbl2.id*  2) необходимо вывести информацию о студентах учащихся в группе по1701  *SELECT \* from tbl1*  *JOIN tbl1*  *ON tbl1.id\_gr = tbl2.id*  *WHERE tbl2.name\_gr like 'по1701'* |

**LEFT JOIN**

Внутренний SELECTможет возвращать не одно а множество значений одного поля. В таких случаях необходимо использовать оператор *IN.*

Выбираются все столбцы левой таблицы, столбцы удовлетворяющие условию объединения из правой таблицы. Если не найдено соответствия в правой таблице, поля итоговой таблицы заполняются NULL.

Синтаксис

SELECT <список выбираемых столбцов>

FROM <Table A>

LEFT JOIN <Table B> ON <условиеобъединения>

**Пример 4.2**

|  |
| --- |
| На основании таблиц примера 4.1 вывести список всех групп с указанием студентов, которые там учатся  *SELECT \* fromtbl2*  *LEFTJOINtbl1*  *ontbl2.id = tbl1.id\_gr* |

**RIGHT JOIN**

Выбираются все столбцы правой таблицы, столбцы удовлетворяющие условию объединения из левой таблицы. Если не найдено соответствия в левой таблице, поля итоговой таблицы заполняются NULL.

*Синтаксис*

*SELECT<список выбираемых столбцов>*

*FROM <Table A>*

*RIFHT JOIN <Table B> ON <условие объединения>*

**Пример 4.3:**

|  |
| --- |
| На основании таблиц t1 и t2 (см. пример 4.1) необходимо вывести список студентов с указанием факультета и номера группы:  *SELECT \* from tbl2*  *RIGHT JOIN tbl2*  *on tbl2.id = tbl1.id\_gr* |

**FULL JOIN**

Возвращает все объединение левого и правого объединения)

*Синтаксис*

*SELECT <список выбираемых столбцов>*

*FROM <Table A>*

*full JOIN <Table B>*

*on tbl2.f1 = tbl1.id (условие объединения)*

**Пример 4.4:**

|  |
| --- |
| На основании таблиц t1 и t2 (см. пример 4.1) необходимо вывести список студентов с указанием факультета и номера группы:  *SELECT\* from tbl1 (что)*  *full JOIN tbl2 (с чем)*  *tbl2.f1 = tbl1.id (условие объединения)* |

**CROSS JOIN**

Возвращает объединение каждой строки одного отношения с каждой строкой второго (Декартово произведение)

*Синтаксис*

*SELECT <список выбираемых столбцов>*

*FROM <Table A>*

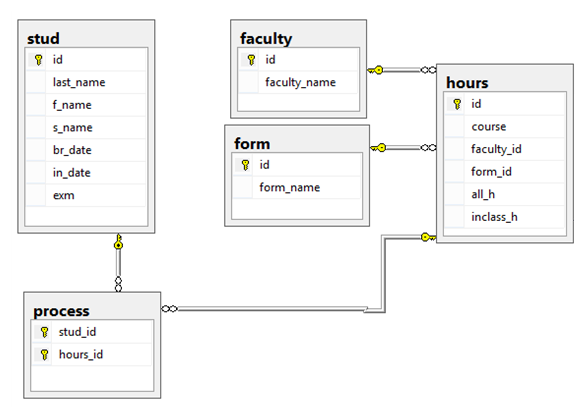
*full JOIN <Table B>*

**Пример 4.5:**

|  |
| --- |
| *SELECT \* fromtbl1*  *CROSSJOINtbl2* |

## Задание к лабораторной работе №4 ЧАСТЬ 1

1. Вывести информацию студентах (факультет, курс) заочной формы обучения младше 37 лет
2. Определить количество студентов на каждом факультете
3. Определить количество студентов, учащихся на каждой форме обучения
4. Определить средний возраст студентов для каждого факультета на конец года (лет)
5. Найти всех студентов-иностранцев (без отчества), вывести сведения о них (дата поступления, факультет, курс, форма обучения)
6. Определить, на какой факультет поступило наибольшее количество человек в 2015 году
7. Определить число студентов поступивших на каждый факультет и форму обучения в 2014 году
8. Вывести список всех факультетов, на которых доступна заочная форма обучения.
9. Вывести **список всех факультетов,** для каждого факультета вывести список форм обучения и номера курсов,
10. Вывести список факультетов, с указанием числа студентов обучающихся на каждой форме обучения
11. Вывести число всех студентов первого и третьего курсов для всех факультетов и форм
12. Для **всех** факультетов и курсов найти число иностранцев, которое на них учится
13. Для каждого курса с учетом факультета найти количество студентов, средний балл которых больше 7,5
14. Вывести **список всех факультетов с учетом формы**с указанием числа студентов старше 45
15. Вывести информацию **для каждого факультета, формы, курса** о количестве студентов младше 27
16. Определить сколько студентов с фамилиями на C на каждом факультете
17. *Вывести список преподавателей и студентов у которых совпадает форма обучения (Преподаватель-Форма обучения-Студент).*
18. *Вывести список преподавателей с фамилией как у студента.*
19. *Вывести список всех студентов у которых фамилии не такие как у преподавателей.*



# Хранимые процедуры.

## Краткие теоретические данные.

Хранимая процедура (ХП) - объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. ХП- это способ инкапсуляции повторяющихся действий

Преимущества:

- разделение логики с между приложениями (разные приложения вызывают разные процедуры к одной и той же БД)

- изоляция пользователей от таблиц базы данных. (дает доступ к ХП а не к самим данным).

- улучшение выполнения как следствие сокращения сетевого трафика. С помощью хранимых процедур множество запросов могут быть объединены.

Недостатки:

- сложно обеспечить целостность логики между СУБД и приложением

- сложность миграции между СУБД

- увеличивается нагрузка на сервер (теперь он выполняет и часть логики)

### 1. Создание ХП

При создании ХП задается:

1. имя ХП и перечень параметров
2. свойства ХП
3. тело ХП

**Пример 6.1**

|  |
| --- |
| *GO* # разделитель,необходим для непрерывного выполнения ХП  *CREATE PROCEDURE sp1*# (1) создаем ХП с именем sp1  *[({@param1} type1= default1] [OUTPUT])]*  *{, …}* #(2)опредляем параметры  *[WITH {RECOMPILE | ENCRYPTION | EXECUTE AS 'user\_name'}] [FOR REPLICATION]* # (3) определяем свойства ХП (см. ниже)  *AS*# (4)начало тела ХП    *SELECT 'Hello World !';* # (4)тело ХП  *GO* # (5)окончания тела ХП  (1) - Имена процедур не чувствительны к регистру, поэтому pr1 = PR1. Нельзя использовать две процедуры с одним именем в одной и той же базе данных; Можно использовать имена в формате "имя-процедуры.имя-базы-данных"; Максимальная длина имени процедуры составляет 64 символа; Избегайте использования имен встроенных функций MSSQL;  (2) - Параметры процедуры это переменные которые могут быть использованы как для передачи информации процедуре, так и для возвращения данных процедурой, параметров у процедуры может и не быть  Пример описания параметров:  @X int = 1, @y char = ‘no\_text’  (3) характеристики хранимой процедуры:  RECOMPILE – компелируется каждый раз при вызове (снижает производительность СУБД)  Ключевое слово ENCRYPTION предписывает серверу выполнить шифрование кода хранимой процедуры, что может обеспечить защиту от использования авторских алгоритмов, реализующих работу хранимой процедуры.  EXECUTE AS 'user\_name‘ – определяет список пользователей с правами которых выполняется процедура  FOR REPLICATION – характеристика, определяющая, что процедура является реплецирующейся (должна копироваться на другие сервера) |

Внутри ХП могут быть объявлены переменные, видимые в пределах ХП

**Пример 6.2: Создать ХП подсчета числа студентов**

|  |
| --- |
| *GO*  *CREATE PROCEDURE count\_stud*  *AS*  *DECLARE @count INT =0;* # объявление переменной count целочисленного типа со значением 0  *select @count =count(id) from stud;*  *print @count;*  *GO* |

### 2 Параметры хранимых процедур

2.1 Передача данных в ХП

Для передачи данных в ХП используются входные параметры. Свойства параметров - количество и тип данных, определяется при создании ХП. При создании ХП определяются ее параметры. Каждый параметр имеет имя, тип данных, тип (входной или выходной), а также может иметь значение по умолчанию

**Пример 6.3. создать ХП для добавления студента. в качестве входных параметров использовать имя, фамилию, отчество (NULL по умолчанию), дату рождения, дату поступления (по умолчанию 01.09.2018)**

|  |
| --- |
| *CREATE PROCEDURE add\_stud*  *@l\_nm nvarchar(20),* # входные параметры, без значения по умолчанию  *@f\_nm nvarchar(20), @br\_d date,*  *@in\_d date ='20180901',* # входные параметры, со значением по умолчанию  *@exm int =8, @s\_nm nvarchar(20) = NULL*  *AS*  *insert into stud*  *(last\_name,f\_name,s\_name,br\_date,in\_date,exm) values*  *(@l\_nm,@f\_nm,@s\_nm,@br\_d,@in\_d,@exm);*  *GO* |

Значение параметров определяется при вызове ХП.

|  |
| --- |
| *EXECUTE add\_stud N'Иванов',N'И', '19820901';* |

где, EXECUTE - оператор вызова ХП

*add\_stud* - имя вызываемой ХП

***N'Иванов'*** - значение присваемое параметру ХП - ***@l\_nm***

***N'И'*** - значение присваемое параметру ХП - ***@f\_nm***

и т.д.

2.2 Вывод данных из ХП

Для вывода данных из ХП используются выходные параметры. Свойства параметров - количество и тип данных, определяется при создании ХП.

**Пример 8.4: создать ХП. подсчитывающую количество студентов**

|  |
| --- |
| *CREATE PROCEDURE count\_stud @count\_st int output*  *AS*  *select @count\_st=count(id) from stud;*  *GO*  *declare @count1 int =0;* #объявляем и инициализируем переменную, в которую будет возвращено значение выходного параметра ХП  *EXECUTE count\_stud @count1 output;* # передаем переменную ХП  *print N'В институте учится ' + CONVERT(varchar(5),@count1) + N'* # Выводим полученное значение |

### 3. Управление потоком внутри ХП

Внутри ХП могут использоваться операторы управления потоком

3.1 Оператор IF

|  |
| --- |
| *IF <условие>*  *BEGIN*  *<инструкции>*  *END*  *ELSE*  *BEGIN*  *<инструкции>*  *END* |

**Пример 6.5: создать ХП, определяющую есть ли студенты в институте**

|  |
| --- |
| *CREATE PROCEDURE find\_stud*  *AS*  *IF EXISTS (select \* from stud)*  *BEGIN*  *SELECT N'В институте учатся студенты';*  *END*  *ELSE*  *BEGIN*  *SELECT N'В институте никто не учится';*  *END*  *GO*  *EXECUTE find\_stud;* |

### 4 Изменение и удаление ХП

ALTER PROCEDURE test1………- изменит процедуру (перепишет)

DROP PROCEDURE test1; - удалит процедуру test1

**Задание к лабораторной работе №4 ЧАСТЬ 2**

1) Создать хранимую процедуру для подсчета количествa студентов на факультете и форме обучения (входные данные название факультета и наименование формы обучения. Результат вывести в процессе выполнения процедуры)

2) Создать хранимую процедуру для подсчета числа предметов на факультетах (наименования), а также числа одинаковых предметов (без входных данных, результат вывести через выходные переменные отдельно для каждого факультета, использовать их при формировании результата с вида: Для ФПК читается \_\_ предметов, для ФПМ читается\_\_\_ предметов, всего \_\_\_\_ предметов (\_\_\_ из которых идентичны)) (print)

3) Создать хранимую процедуру: добавления студента на указанный факультет и форму обучения (входные данные название факультета и наименование формы обучения, дата рождения, дата поступления, ФИО). Зачислять на первый курс. Если указанной формы, факультета или их комбинации не существует, то возвращать сообщение об ошибке.