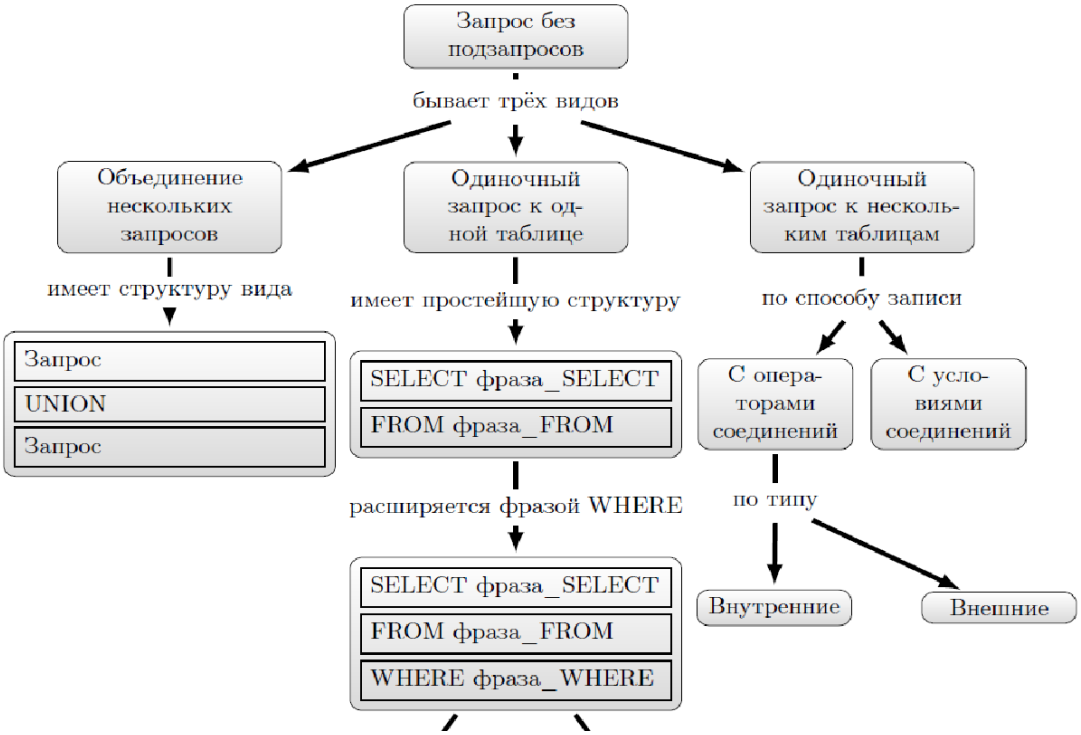
Соединения в SQL Oracle

Виды соединений



JOIN — оператор языка SQL, который является реализацией операции соединения реляционной алгебры. Входит в предложение FROM операторов SELECT, UPDATE и DELETE.

Операция соединения, как и другие бинарные операции, предназначена для обеспечения выборки данных из двух таблиц и включения этих данных в один результирующий набор. Отличительными особенностями операции соединения являются следующие:

- в схему таблицы-результата входят столбцы обеих исходных таблиц (таблиц-операндов), то есть схема результата является «сцеплением» схем операндов;

- каждая строка таблицы-результата является «сцеплением» строки из одной таблицы-операнда со строкой второй таблицы-операнда.

Определение того, какие именно исходные строки войдут в результат и в каких сочетаниях, зависит от типа операции соединения и от явно заданного *условия соединения*. Условие соединения, то есть условие сопоставления строк исходных таблиц друг с другом, представляет собой логическое выражение (предикат).

При необходимости соединения не двух, а нескольких таблиц, операция соединения применяется несколько раз (последовательно).

SQL-операция JOIN является реализацией операции соединения реляционной алгебры только в некотором приближении, поскольку в реляционной модели данных соединение выполняется над отношениями, которые являются множествами, а в SQL — над таблицами, которые являются мультимножествами. Результаты операций тоже, в общем случае, различны: в реляционной алгебре результат соединения даёт отношение (множество), а в SQL — таблицу (мультимножество).

Описание оператора

```
FROM
  Table1
  {INNER | {LEFT | RIGHT | FULL} OUTER | CROSS } JOIN
  Table2
    {ON <condition> | USING (field_name [,... n])}
```

В большинстве СУБД при указании слов LEFT, RIGHT, FULL слово OUTER можно опустить. Слово INNER также в большинстве СУБД можно опустить.

В общем случае СУБД при выполнении соединения проверяет условие (предикат) *condition*. Если названия столбцов, по которым происходит

соединение таблиц, совпадают, то вместо ON можно использовать USING. Для CROSS JOIN условие не указывается.

Для перекрёстного соединения (декартова произведения) CROSS JOIN в некоторых реализациях SQL используется оператор «запятая» (,), что актуально для Oracle:

```
FROM
  Table1,
  Table2
```

Виды оператора JOIN

Для дальнейших пояснений будут использоваться следующие таблицы:

City (Города)

Id	Name
1	Москва
2	Санкт-Петербург
3	Казань

Person (Люди)

Name	CityId
Андрей	1
Леонид	2
Сергей	1
Григорий	4

INNER JOIN

Оператор *внутреннего соединения* INNER JOIN соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является коммутативным.

Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

Тело результата логически формируется следующим образом. Каждая строка одной таблицы сопоставляется с каждой строкой второй таблицы, после чего для полученной «соединённой» строки проверяется условие соединения (вычисляется предикат соединения). Если условие истинно, в таблицу-результат добавляется соответствующая «соединённая» строка.

Описанный алгоритм действий является строго логическим, то есть он лишь объясняет результат, который должен получиться при выполнении операции, но не предписывает, чтобы конкретная СУБД выполняла соединение именно указанным образом. Существует несколько способов реализации операции соединения, например, соединение вложенными циклами (англ. *inner loops join*), соединение хешированием (англ. *hash join*), соединение слиянием (англ. *merge join*). Единственное требование состоит в том, чтобы любая реализация логически давала такой же результат, как при применении описанного алгоритма.

В рамках работы в SQLDeveloper указывать слово INNER необязательно.

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

ИЛИ

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

Результат:

Person.	Person.	City	City.Name

Name	CityId	.Id	
Андрей	1	1	Москва
Леонид	2	2	Санкт-Петербург
Сергей	1	1	Москва

OUTER JOIN

Соединение двух таблиц, в результат которого обязательно входят все строки либо одной, либо обеих таблиц.

LEFT OUTER JOIN

Оператор *левого внешнего соединения* LEFT OUTER JOIN соединяет две таблицы. Порядок таблиц для оператора важен, поскольку оператор не является коммутативным.

Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

Тело результата логически формируется следующим образом. Пусть выполняется соединение левой и правой таблиц по предикату (условию) *p*.

1. В результат включается внутреннее соединение (INNER JOIN) левой и правой таблиц по предикату *p*.
2. Затем в результат добавляются те строки левой таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких строк столбцы, соответствующие правой таблице, заполняются значениями NULL.

В рамках работы в SQLDeveloper указывать слово OUTER необязательно.

```
SELECT *
FROM
  Person -- Левая таблица
  LEFT OUTER JOIN
```

```
  City   -- Правая таблица
    ON Person.CityId = City.Id

ИЛИ
SELECT *
FROM Person -- Левая таблица
LEFT JOIN City ON Person.CityId = City.Id -- Правая таблица
```

Результат:

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Леонид	2	2	Санкт-Петербург
Сергей	1	1	Москва
Григорий	4	NULL	NULL

RIGHT OUTER JOIN

Оператор *правого внешнего соединения* RIGHT OUTER JOIN соединяет две таблицы. Порядок таблиц для оператора важен, поскольку оператор не является коммутативным.

Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

Тело результата логически формируется следующим образом. Пусть выполняется соединение левой и правой таблиц по предикату (условию) *p*.

1. В результат включается внутреннее соединение (INNER JOIN) левой и правой таблиц по предикату *p*.
2. Затем в результат добавляются те строки правой таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких строк столбцы, соответствующие левой таблице, заполняются значениями NULL.

В рамках работы в SQLDeveloper указывать слово OUTER необязательно.

```
SELECT *
FROM
  Person -- Левая таблица
  RIGHT OUTER JOIN
  City   -- Правая таблица
    ON Person.CityId = City.Id

ИЛИ

SELECT *
FROM Person -- Левая таблица
RIGHT JOIN City ON Person.CityId = City.Id -- Правая таблица
```

Результат:

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Сергей	1	1	Москва
Леонид	2	2	Санкт-Петербург
NULL	NULL	3	Казань

FULL OUTER JOIN

Оператор *полного внешнего соединения* FULL OUTER JOIN соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является коммутативным.

Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

Тело результата логически формируется следующим образом. Пусть выполняется соединение первой и второй таблиц по предикату (условию) *p*. Слова «первой» и «второй» здесь не обозначают порядок в записи выражения (который неважен), а используются лишь для различения таблиц.

1. В результат включается внутреннее соединение (INNER JOIN) первой и второй таблиц по предикату *p*.

2. В результат добавляются те строки первой таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких строк столбцы, соответствующие второй таблице, заполняются значениями NULL.

3. В результат добавляются те строки второй таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких строк столбцы, соответствующие первой таблице, заполняются значениями NULL.

В рамках работы в SQLDeveloper указывать слово OUTER необязательно.

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

или

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

Результат:

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Сергей	1	1	Москва
Леонид	2	2	Санкт-Петербург
NULL	NULL	3	Казань
Григорий	4	NULL	NULL

CROSS JOIN

Оператор *перекрёстного соединения*, или *декартова произведения* CROSS JOIN соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является коммутативным.

Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

Тело результата логически формируется следующим образом. Каждая строка одной таблицы соединяется с каждой строкой второй таблицы, давая тем самым в результате все возможные сочетания строк двух таблиц.

```
SELECT *
FROM
  Person
  CROSS JOIN
  City
```

или

```
SELECT *
FROM Person, City
```

Результат:

Person.Name	Person.CityId	City.Id	City.Name
Андрей	1	1	Москва
Андрей	1	2	Санкт-Петербург
Андрей	1	3	Казань
Леонид	2	1	Москва
Леонид	2	2	Санкт-Петербург
Леонид	2	3	Казань
Сергей	1	1	Москва
Сергей	1	2	Санкт-Петербург
Сергей	1	3	Казань
Григорий	4	1	Москва

Григорий	4	2	Санкт-Петербург
Григорий	4	3	Казань

Если в предложении *WHERE* добавить условие соединения (предикат *p*), то есть ограничения на сочетания кортежей, то результат эквивалентен операции INNER JOIN с таким же условием:

```
SELECT *
FROM
  Person,
  City
WHERE
  Person.CityId = City.Id
```

Таким образом, выражения t1, t2 WHERE p и t1 INNER JOIN t2 ON p синтаксически являются альтернативными формами записи одной и той же логической операции внутреннего соединения по предикату *p*. Синтаксис CROSS JOIN + WHERE для операции соединения называют устаревшим, его не рекомендует стандарт SQL ANSI.

Запросы с группированием (фразы Group BY и HAVING).

Описание

SQL оператор **GROUP BY** можно использовать в SELECT для сбора данных по нескольким записям и группировки результатов одного или нескольких столбцов.

Оператор **HAVING** используется в сочетании с оператором GROUP BY, чтобы ограничить группы возвращаемых строк только теми, для которых выполняется некоторое условие.

Синтаксис

```
SELECT expression1, expression2, … expression_n,
aggregate_function (aggregate_expression)
FROM tables
[WHERE conditions]
GROUP BY expression1, expression2, … expression_n

[HAVING conditions]
```

Параметры или аргументы

expression1, expression2, … expression_n — выражения, которые не входят в aggregate_function и должны быть включены в предложение GROUP BY в конце оператора SQL.

aggregate_function — это функция, такая как SUM, COUNT, функции MIN, MAX или AVG.

aggregate_expression – это столбец или выражение, которое используется в aggregate_function.

Tables – таблицы, из которых вы хотите выгрузить данные. После оператора FROM должна быть указана хотя бы одна таблица.

WHERE conditions. Необязательный. Условия, которые должны быть выполнены для записей, которые будут выбраны.

HAVING conditions — это дополнительные условия, применяемые только к агрегированным результатам для ограничения групп возвращаемых строк. В результирующий набор будут включены только те группы, состояние которых соответствует TRUE.

Пример использования функции MIN и MAX

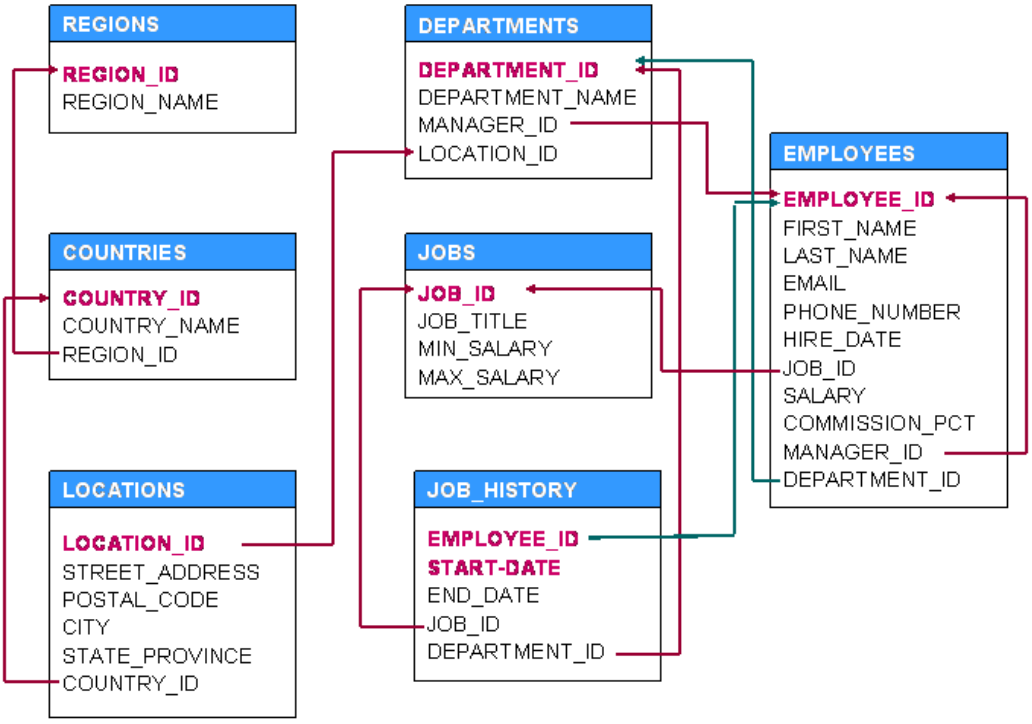
Определить номера департаментов с минимальной зарплатой больше или равной 2500 и максимальной зарплатой меньшей 10000.

```
SELECT emp.department_id, min(emp.salary), max(emp.salary)
FROM employees emp
GROUP BY emp.department_id
HAVING min(emp.salary)>=2500 AND max(emp.salary)<10000
ORDER BY min(emp.salary)
```

Задание для аудиторной и самостоятельной работы:

Создать и наполнить базу данных по схеме таблиц HR. Условия размерности берётся из лабораторной работы № 4, ограничение целостности генерируются самостоятельно на каждое поле в каждой таблице.

Схема таблиц HR



После создания и наполнения реализовать следующие запросы:

1. Получить данные по департаментам: для каждого департамента определить минимальную и максимальную зарплату, определить дату устройства на работу первого и последнего сотрудника, а также общее количество сотрудников в департаменте. Сортировать по количеству сотрудников (по убыванию);

2. Найти сотрудника с минимальной оплатой по каждому из отделов (их названия);

3. Для каждого года вывести количество сотрудников, нанятых в этом году. Сортировать по количеству сотрудников (по убыванию);

4. Найти управляющих, у которых количество подчиненных больше 5 и сумма всех зарплат подчиненных больше 50000;

5. Получить максимальную и минимальную зарплату по должности среди всех департаментов (необходимо найти должности, которые есть в каждом или почти каждом департаменте и выполнить требования по этим департаментам и этой должности).