ЛАБОРАТОРНАЯ РАБОТА №6

Тема: Раздел FROM предложения SELECT. Методы конструирования источника данных.

Цель: Пробрести навыки конструирования источника данных предложения SELECT.

В данной лабораторной работе используются база данных HR, созданная в рамках первой лабораторной работы. Также используется таблица TB_ELEKTROSTAL_2018, которая создана и наполнена данными в процессе выполнения лаб. работы №.1.

Дополнительно используется удаленная база данных проекта stackoverflow.com. Структура этой базы данных описана в методических указаниях к выполнению лабораторной работы № 3. Напомним, адрес ресурса http://data.stackexchange.com/stackoverflow/query/new.

Раздел FROM оператора SELECT

В разделе FROM оператора SELECT конструируется источник данных, один или несколько столбцов из полученной временной таблицы используются в качестве элементов списка колонок оператора SELECT. В качестве простых источников данных могут использоваться таблицы, представления (view), внутренние запросы, хранимые процедуры.

В некоторых запросах источник данных не требуется. Например, нужно вывести число 10. В подобных случаях используют специальные таблицы, которые имеются всегда и всегда содержат одну строку. В СУБД ORACLE такой псевдо таблицей является таблица dual, в СУБД Firebird – таблица rdb$database. Диалект языка SQL некоторых марок СУБД позволяет опускать раздел FROM оператора SELECT. К таким СУБД можно отнести MsSQL, mySQL, postgreSQL.

Примеры.

Вывести числа 10

a) в СУБД ORACLE

select 10 from dual

b) в СУБД Firebird

select 10 from rdb$database

c) в СУБД PostgreSQL, MySQL, MsSQL

select 10

В простых запросах в качестве источника данных используется одна таблица. Например (запрос к БД HR):

select first_name, last_name from employees

В более сложных случаях источник данных строится из нескольких таблиц. В стандартах языка SQL ранних версий предусматривалось перечисление таблиц (источников данных) через запятую в разделе FROM. При этом множество строк нового временного источника данных представляет собой полное (декартово) произведение множеств строк двух исходных таблиц. А схема источника данных получалась объединением схем исходных отношений.

Рассмотрим две таблицы (отношения)

Таблица 1.

Отношение «tb_city» (Города)

shifrStr	name
1	Луганск
2	Юбилейный
3	Видный

Таблица 2.

Отношение «tb_street» (Улицы)

shifrStr	shifrCit	tame
1	1	Оборонная

2	1	Ватутина
3	2	Шахтерский
4	-	Артема

Выполнение оператора

select * from tb_city, tb_street

Возвратит следующую таблицу

Таблица 3. Результат выполнении операции соединения

Tb_city.shifrCit	Tb_city.name	Tb_street.shifrStr	Tb_street.shifrcCit	Tb_streer.Name
1	Луганск	1	1	Оборонная
1	Луганск	2	1	Ватутина
1	Луганск	3	2	Шахтерский
1	Луганск	4	-	Артема
2	Юбилейный	1	1	Оборонная
2	Юбилейный	2	1	Ватутина
2	Юбилейный	3	2	Шахтерский
2	Юбилейный	4	-	Артема
3	Видный	1	1	Оборонная
3	Видный	2	1	Ватутина
3	Видный	3	2	Шахтерский
3	Видный	4	-	Артема

Анализ результирующей таблицы показывает, что она содержит 12 строк. Это количество легко вычислить умножив мощность отношения tb_city на мощность отношения tb_street (3*4=12). Другими слова, множество строк результирующего отношение представляет собой декартово произведение множеств строк двух исходных отношений. Схема результирующего отношения получена объединением схем двух исходных отношений. Обратите внимание, что в схеме первого отношения (tb_city) два поля (его арность равна 2). В схеме второго отношения (tb_street) – втором отношении 3 поля. Арность результирующего отношения – 6 (2*3).

Семантический анализ полученного отношения показывает, что данные этого отношения не соответствуют реальному расположению улиц в городах. Получается, что каждый город содержит все улицы и, кроме того, каждая улица расположена в каждом городе. Безусловно, полученные данные противоречат здравому смыслу. Тем не менее, в практике встречаются случаи, когда используются подобные виды соединений. Например, когда в одной из таблиц содержится ровно одна строка. Безусловно, такие таблицы из одной строки встречаются редко. Но, в качестве такое таблицы может использоваться хранимая процедура, которая возвращает одну строку, уникальную для каждой строки другой таблицы.

Наиболее часто встречаются случаи условного соединения таблиц. В таких случаях формулируется условие соединения таблиц. Это условие проверяется для каждой пары строк из двух исходных таблиц. В результирующее отношение включаются только те комбинации строк, для которых выполняется заданное условие. В большинстве случаев условие соединения таблиц выражается в виде условия равенства значений двух полей. Одно поле из одной таблиц, второе поле – из другой таблицы. В рассматриваемом примере наиболее естественным будет условие равенства значения поля shifrcit из таблицы tb_city значению поля shifrcit из таблицы tb_street. В таком случае запрос будет выглядеть следующим образом:

select * from tb_city, tb_street

where tb_city.shifrcit=tb_street.shifrcit

Выполнив этот запрос получим следующий результат.

Таблица 4.

tb_city.shifrCit	tb_city.name	tb_street.shifrStr	tb_street.shifrcCit	tb_streer.Name
1	Луганск	1	1	Оборонная
1	Луганск	2	1	Ватутина
2	Юбилейный	3	2	Шахтерский

Полученный результат имеет очевидный смысл. Для каждого города указаны расположенные в нем улицы. И для каждой улицы указан город, в

котором она расположена. Обратите внимание, что в полученном результате отсутствуют строки, которые не имеют пары. Например, для поселка Видный в таблице tb_street отсутствует запись об улице, расположенной в поселке Видный. И, наоборот, для улицы Артема не указан город, в котором она расположена.

Иногда в ответ требуется включать и строки, не имеющие пары согласно указанному условию, в другом отношении. В условиях первых вариантов стандарта языка SQL описывать подобные условия соединения таблиц довольно трудно. По этой причине в 80-х годах был принят новый стандарт языка SQL. Этот стандарт предусматривает использование ключевого слова JOIN для соединения таблиц. Вся информация о свойствах операции соединения помещается в раздел формирования источника данных запроса (ключевое слово FROM). Условие соединения таблиц указывается после ключевого слова ON.

Рассмотрим разновидности операции JOIN.

CROSS JOIN

Результирующее множество строк этой операций представляет декартово произведение множеств строк двух исходных отношений. Схема результирующего отношения – объединение схем исходных отношений.

Например

select * from tb_city cross join tb_street

вернет тот же результат (см. таблицу 3), что и

select * from tb_city,tb_street

Во первом случае требования к формированию результата заданы явно с акцентом на требуемые условия соединения таблиц. Предыдущие версии языка SQL не предназначались для явного описания требований к соединению таблиц, и результат операции соединения подразумевался неявно.

NATURAL JOIN

При использовании этой разновидности операции *JOIN* таблицы должны иметь совпадающие, по имени, столбцы. В результирующее отношение будут включены строки из обеих таблиц, у которых значения в столбцах с одинаковыми именами совпадают. Легко заметить, что рассматриваемый режим просто автоматизирует составление условия соединения таблиц. Вместо ручного указания имен столбцов, по которым выполняется соединение, используется автоматический поиск среди атрибутов обоих таблиц на предмет совпадающих имен столбцов. Рассмотрим таблицы departments и locations базы данных HR. Можно отметить, что в обеих таблицах присутствует поле **location_id**. Это соответствие найдет операция **NATURAL JOIN** и автоматически построит условие соединения **departments.location_id=locations.location_id**.

Пример

select * from departments natural join locations

В таблицах **tb_street** и **tb_city** присутствуют два поля с одинаковым названием это – **shifrcit** и **name**. Поэтому операция natural natural join построит условие

tb_city.shifrcit=tb_street.shifrcit and tb_city.name=tb_street.name

Этому условию не соответствует ни одна комбинация соединения строк. По этой причине запрос

select * from tb_city natural join tb_street

вернет пустой набор записей.

JOIN USING

Этот вариант операции join предусматривает указание в скобках имени поля (или полей), которое имеется (имеются) в схемах обоих отношений. Именно по равенству значений этих полей в с троках и будет выполняться операций соединения. Другими словами, в отличии от предыдущего случая поля, по которым требуется выполнять соединение таблиц, указываются

непосредственно разработчиком. Исключить операцию сравнения полей name из последнего примера можно следующим образом.

select * from tb_city join tb_street using (shifrcit)

Результат этой операции показан в таблице 4.

Внутренне соединение по условию (INNER JOIN … ON)

В большинстве случаев требуется явно указывать условие, по которому необходимо выполнять соединение таблиц. Обычно такое условие формулируется как равенство значений указанных полей в двух таблицах. Условие соединения таблиц записывается после ключевого слова **ON**. Например

select * from tb_city

join tb_street on tb_city.shifrcit=tb_street.shifrcit

Следует отметить, что в таком варианте операции join в условии соединения можно указывать имена полей, которые не совпадают в разных таблицах. В случае же совпадения имен полей в двух таблицах нужно использовать уточняющие префиксы: либо имена таблиц, либо псевдонимы таблиц. Например, предыдущий запрос можно перезапись следующим образом:

select * from tb_city a

join tb_street b on a.shifrcit=b.shifrcit

В условиях соединения таблиц необязательно использовать знак равенства, т.е. искать совпадающие значения полей. Хотя поиск по равенству – это наиболее частый случай конструирования условия соединения таблиц.

Пусть например требуется найти работника с максимальным окладом из таблицы employees. Напомним, что оклад хранится в поле salary этой таблицы. Можно использовать следующий оператор

select first_name||' '||last_name, salary from employees a

left outer join employees b on a.salary<b.salary

where b.employee_id is null

В данном примере используется внешнее соединение (outer join). Этот вид соединений рассматривается в следующей лабораторной работе. Здесь условие соединения задано в виде условия «меньше». Алгоритм работы следующий. Строка каждого работника соединяется со строками этой таблицы, причем для соединения выбираются строки, соответствующие работникам с большим окладом. Строку работника с максимальным окладом соединять не с чем. По этой причине она имеет значения null в полях парной строки. Именно это и используется для ее отбора в условии **where**.

Порядок выполнения работы

Порядок выполнения работы.

Составить следующие запросы

- Составить запрос к БД stackoverfow. Результат запроса должен содержать две колонки. Первая колона содержание поста. Вторая колонка имя пользователя, создавшего этот пост. Указание: необходимо использовать операцию соединения таблиц в «старом» формате, т е без ключевого слова **join**.
- Составить запрос к БД stackoverfow. Результат запроса должен содержать две колонки. Первая колона содержание поста. Вторая колонка имя пользователя, создавшего этот пост. Указание: необходимо использовать операцию соединения таблиц в «новом» формате, т е с ключевым словом **join**.
- Вывести таблицу из двух колонок (база данных HR). В первой колонке имя и фамилия работника (first_name и last_name), разделенных одним пробелом. Во второй колонке адрес департамента, к котором работает

сотрудник Адрес департамента задать в виде city||''||street_address. Оба поля содержатся в таблице location. (Поскольку у таблиц employees и locations нет общих полей использовать таблицу departments как промежуточную). При составлении запроса использовать вид соединения …JOIN… USING

- Составить запрос как в предыдущем задании, но использовать вид соединения … JOIN… ON ….
- Вывести список сотрудников из таблицы employees (БД HR) с минимальным окладом.
- Вывести список сотрудников из таблицы employees (БД HR) с максимальным уровнем зарплаты. Зарплата определяется как сумма поля salary и выражения salary*commission_pct.

Содержание отчета:

1. Тема, цель лабораторной работы.
2. Примеры выполнения запросов к базе данных.
3. Составленные согласно заданию запросы и скриншоты полученных результатов.
5. Выводы.

Контрольные вопросы:

1. Опишите особенности диалектов языка SQL для различных марок СУБД для случая, когда раздел FROM оператора SELECT не требуется.

2. Как называются мнимые (фиктивные) таблицы в СУБД ORACLE? Firabird? MySQL? postgreSQL? MsSQL?

3. Дайте характеристику способу описания соединения таблиц без ключевого слова JOIN.

4. Охарактеризуйте особенности декартового (полного) соединения таблиц БД.

5. Как определяется схема результирующего отношения в операции соединения двух таблиц.

6. Охарактеризуйте операцию CROSS JOIN.

7. Охарактеризуйте операцию NATURAL JOIN.

8. Охарактеризуйте операцию JOIN … USING (..).

9. Укажите способы уточнения принадлежности полей таблицам для случая, когда имена полей в обоих соединяемых таблицах совпадают.

10. Охарактеризуйте операцию … JOIN … ON ...

Файл:	Лабораторная работа 6
Каталог:	C:\Users\sss\Documents
Шаблон:	C:\Users\sss\AppData\Roaming\Microsoft\Шаблоны\Normal.dotm
Заголовок:	
Содержание:	
Автор:	sss
Ключевые слова:	
Заметки:	
Дата создания:	08.11.2018 22:07:00
Число сохранений:	89
Дата сохранения:	12.11.2018 13:04:00
Сохранил:	sss
Полное время правки:	450 мин.
Дата печати:	12.11.2018 13:04:00
При последней печати	
страниц:	10
слов:	1 956 (прибл.)
знаков:	11 155 (прибл.)