Московский Государственный Технический Университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Компьютерные системы и сети»

Фомин М.М.

**Методическое пособие по лабораторной работе №4**

**Доработка приложения в среде ORACLE/APEX**

Москва 2017

**Цель работы –** Научиться создавать и дорабатывать простые приложения в СУБД Oracle11g.

## Введение

В основе комплекса технологических программ, на которых проводится лабораторная работа, лежит Oracle Database Express Edition (Oracle Database XE) - это бесплатная, урезанная по объему хранимой информации версия СУБД Oracle Database. Для администрирования и создания базы данных в Oracle Database XE предлагается использовать SQL Developer. Для разработки схемы базы данных предлагается использовать Oracle SQL Data Modeler. В комплекс также включен Oracle Application Express (APEX) - графическая среда разработки для создания web-приложений, основанных на базе данных поддерживаемой Oracle Database XE. В настоящей лабораторной работе требуется доработать простейшую информационную систему, которую вы создали в третьей лабораторной работе.

# Теоретическая часть

# Внешнее соединение запросов

**UNION**

Оператор *UNION* объединяет результаты двух запросов в одно множество.

*UNION* относится к операторам *работы с множествами*. Другие операторы работы с множествами включают *INTERSECT* и *EXCEPT.*  Все эти операторы предназначены для одновременной работы с результирующими множествами нескольких запросов, отсюда и название – операторы работы с множествами.

**Синтаксис SQL2003**

На число запросов, которые можно объединять с помощью *UNION*, никаких ограничений не накладывается. Общий синтаксис оператора следующий:

<*оператор\_select1*>

UNION [ALL | DISTINCT]

<*оператор\_select2*>

UNION [ALL | DISTINCT]

...

**Ключевые слова**

*UNION*

Объединяет результирующие наборы операторов в один набор. Повторяющиеся строки по умолчанию исключаются.

*ALL | DISTINCT*

Позволяет исключить повторяющиеся строки из конечного результата (*DISTINCT*) либо оставить повторяющиеся строки (*ALL*). Столбцы с пустыми значениями считаются одинаковыми. Если не указано ни *ALL*, ни *DISTINCT*, то по умолчанию дубликаты удаляются.

**Общие правила**

При использовании *UNION* следует помнить только об одном существенном правиле: порядок, число и типы данных во всех объединяемых запросах должны совпадать. Типы данных не должны быть идентичными, но они должны быть совместимыми. Например, совместимыми являются типы данных *CHAR* и *VARCHAR*. Как правило, в результирующем множестве будет использован самый длинный тип данных. Например, если объединяются три запроса, в которых столбцы имеют

**СУБД Уровень поддержки**

MySQL Не поддерживается

Oracle Поддерживается с ограничениями

PostgreSQL Поддерживается с ограничениями

SQL Server Поддерживается с ограничениями

**Советы и хитрости**

Хотя согласно стандарту ANSI оператор *INTERSECT* имеет наибольший приоритет среди всех операторов работы с множествами, большинство платформ выполняют все операторы работы с множествами с равным приоритетом. Вы можете явно определять порядок выполнения операторов с помощью скобок. В противном случае с большой вероятностью операторы будут выполняться в порядке следования, сверху вниз.

На некоторых платформах использование *DISTINCT* может приводить к большим затратам на выполнение запроса, так как при этом требуется дополнительный проход по данным для удаления дубликатов. Поэтому если вы уверены, что в данных дубликатов нет (либо их не требуется удалять), то используйте *ALL* для более быстрого выполнения запроса.

В соответствии со стандартом ANSI можно использовать только одну фразу *ORDER BY* для всего оператора. Используйте ее в конце последнего оператора *SELECT*. Для исключения неоднозначности в столбцах и таблицах вы можете снабдить все столбцы соответствующими одинаковыми псевдонимами. Однако для именования столбцов конечного результата используются только псевдонимы из первого запроса в *SELECT…UNION*. Например:

SELECT au\_lname AS "lastname", au\_fname AS "firstname"

FROM authors

UNION

SELECT emp\_lname AS "lastname", emp\_fname AS "firstname"

FROM employees

ORDER BY lastname, firstname

Хотя наборы столбцов запросов могут иметь совместимые типы данных, в некоторых платформах могут быть различия в обработке длин столбцов. Например, если столбец **au\_lname** из первого запроса имеет значительно большую длину, чем столбец **emp\_name** из второго запроса, то правила выбора длины столбца для конечного результата могут быть различными в разных платформах. Хотя обычно просто выбирается тип с наибольшей длиной (наименее ограниченный). Каждая СУБД может иметь свои правила именования столбцов в случае, когда столбцы в запросах имеют разные названия. Обычно используются названия столбцов из первого запроса.

**Oracle**

Oracle поддерживает базовую функциональность операторов *UNION* и *UNION*

*ALL* в соответствии со следующим синтаксисом:

<*оператор\_select1*>

UNION [ALL]

<*оператор\_select2*>

UNION [ALL]

...

Вместо *UNION DISTINCT* следует использовать функционально эквивалентный *UNION*. Нельзя использовать оператор *UNION* в следующих случаях:

**•** Если запросы содержат столбцы типов *LONG*, *BLOB*, *CLOB*, *BFILE* и *VARRAY*.

**•** Если запросы содержат фразы *FOR UPDATE* и выражения *TABLE*.

Если первый запрос содержит выражения в списке столбцов, то для этих выражений при помощи *AS* должны быть заданы псевдонимы. Также только последний запрос может содержать фразу *ORDER BY*. Например, мы можем получить все идентификаторы магазинов, без дубликатов, с помощью следующего запроса:

SELECT stor\_id FROM stores

UNION

SELECT stor\_id FROM sales

# Практическая часть

# Задание на лабораторную работу.

Необходимо доработать приложение на основании макета информационной системы, разработанной в лабораторной №3 и приведенному ниже бизнес процессу:

1. Задание начального и конечного пунктов маршрута.
2. Задание даты вылета в «прямом» направлении.
3. Задание (если необходимо) даты вылета обратного перелета.
4. Выбор из возможных вариантов прямого перелета *как беспересадочного, так и с пересадками*.
5. Выбор из возможных вариантов обратного перелета *как беспересадочного, так и с пересадками*.
6. Утверждение маршрута перелета.
7. Ввод данных клиента.
8. Сохранение выбранного маршрута в заказах клиента (изменение статуса заказа).
9. Формирование маршрутной квитанции для распечатки клиентом.
10. Просмотр клиентом своих маршрутов.

Интерфейс приложения должен быть лаконичен (главное сейчас не красота, а функциональность) и практически повторять интерфейс разработанной в лабораторной №3.

От четы должны обеспечивать ту же функциональность, что и в лабораторной работе №3, но с учетом изменений в бизнес процессе приложения.

# Доработка приложения

Подумайте, что необходимо для поиска возможных перелетов с пересадкой между конечными пунктами. Реализуйте эту возможность в своей информационной системе. Для одновременного отображения в списке выбора прямых (беспересадочных) маршрутов и маршрутов с пересадками, необходимо объединить запросы с помощью предложения UNION

UNION

SELECT из третьей лабораторной работы

UNION

SELECT для поиска маршрутов с одной пересадкой

UNION

SELECT для поиска маршрутов с двумя пересадками

Можно ограничится двумя пересадками, так как в жизни обычно вполне хватает этого количества.

Подумайте, что должно отображаться в строке выбора маршрута, чтобы клиенту было удобно делать выбор.

Подумайте, что должно быть на этот раз во втором поле запроса. Как в дальнейшем обрабатывать содержимое этого поля?

Подумайте, как должны записываться результаты выбора клиента в таблицы базы данных. Каким образом должны выбираться и отображаться все рейсы прямого и обратного маршрутов в отчете «Маршрутная квитанция».

# Дополнительные вопросы.

Подумайте, как сделать запрос с неограниченным числом пересадок

Каком образом должен быть организован интерфейс для полной реализации бизнес процесса приложения?

Как организовать редактирование заказа и маршрутов перелетов?

# Рекомендуемая литература

1. Дейт К. Введение в системы баз данных, 8-е издание: Пер. с англ. – М.:Издательский дом ”Вильямс”. 2005.-1328 с.
2. Грабер М. Введение в SQL: Пер с англ. – М.:Изд-во’ЛОРИ’,1996. – 380с.
3. Кайт Томас. Oracle для профессионалов: архитектура, методика программирования и основные особенности.: Пер с англ. – М.:Издательский дом ”Вильямс”. 2008. – 848 с.
4. Кляйн К., Кляйн Д., Хант Б. SQL. Справочник, 3\_е издание. – Пер. с англ. – СПб: Символ\_Плюс, 2010. –656 с., ил.