**2 Лабораторная работа №2**

**«Язык SQL. Генераторы. Триггеры»**

**2.1 Цель работы:**

Ознакомится с принципом работы генераторов и триггеров, продемонстрировать работу на примерах.

**2.2 Постановка задачи**

2.2.1 Записать запросы, соединяющие две таблицы с помощью JOIN и без него.

2.2.2 Записать запросы, соединяющие более двух таблиц с помощью JOIN и без него.

2.2.3 Продемонстрировать следующие возможности SQL:

* использование псевдонимов на примере рекурсивного запроса;
* привести пример запроса с подзапросом;
* использование агрегатных функций в подзапросе;
* подзапросы, возвращающие единственное и множественные значения;
* подзапросы, использующие вычисление;
* использование подзапросов в HAVING.

2.2.4 Записать запрос, соединяющий таблицу со своей копией.

2.2.5 Привести пример коррелированного запроса, использующего две разные таблицы.

2.2.6 Продемонстрировать следующие возможности SQL

* работу оператора EXISTS;
* работу оператора ALL;
* работу оператора ANY.

2.2.7 В соответствии с вариантом задания (Вариант – 5) создать генератор и триггер:

Создать генераторы для полей «Номер фирмы», «Номер типа помещения». Автоматическая генерация полей «Номер фирмы», «Номер типа помещения», создать обновляемое представление «ПомещениеТип» (в таблицу «Помещение» вместо номера типа помещения подставить наименование типа помещения).

2.2.8 Изменить значение генератора, в соответствии с хранимыми данными.

2.2.9 Ввести данные в таблицу, используя генератор (не менее 5 строк). Просмотреть полученный результат.

2.2.10 Внести изменения в указанные таблицы, используя триггеры (не менее 5 строк). Просмотреть полученный результат.

**2.3 Ход работы**

2.3.1 Было осуществлено подключение к базе данных из прошлой лабораторной работы с помощью следующего запроса:

CONNECT "D:\5\_semester\Databases\Lab\_1\Firms.fdb" user 'SYSDBA' password 'masterkey';

Был сделан запрос, соединяющий две таблицы без Join: “Вывести из таблицы Помещение атрибут номер фирмы, из таблицы Телефон номер телефона, Из таблиц Помещение и Телефон, Где в таблице Помещение атрибут адреса равен атрибуту адреса в таблице Телефон”. Результат выполнения запроса отображен на рисунке 2.1.

SELECT ROOM.COMPANY\_NUM, TELEPHONE.PHONE\_NUM

FROM ROOM, TELEPHONE

WHERE ROOM.ADRESS = TELEPHONE.ADRESS;

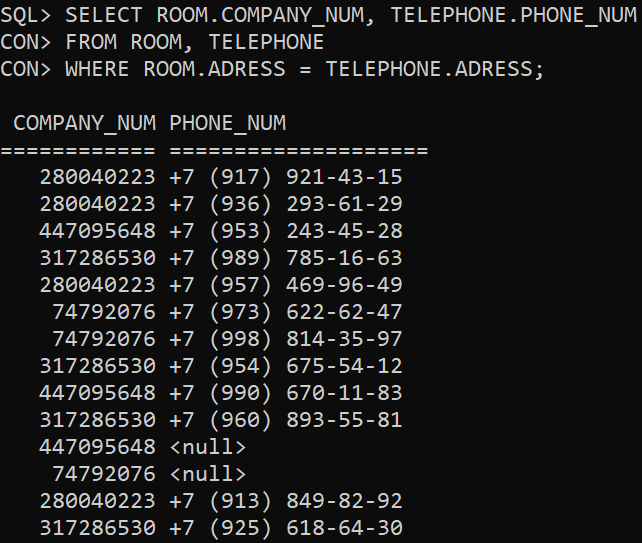


Рисунок 2.1 – Результат запроса соединения двух таблиц без использования Join

Затем такой же запрос был сделан с использованием Join. Результат выполнения запроса отображен на рисунке 2.2. Запрос был следующий: “Вывести из таблицы Помещение атрибут номер фирмы, из таблицы Телефон номер телефона, Из таблиц Помещение соединяя с таблицей Телефон Где Помещение атрибут адреса равен атрибуту адреса в таблице Телефон”.

SELECT ROOM.COMPANY\_NUM, TELEPHONE.PHONE\_NUM

FROM ROOM JOIN TELEPHONE

ON ROOM.ADRESS = TELEPHONE.ADRESS;

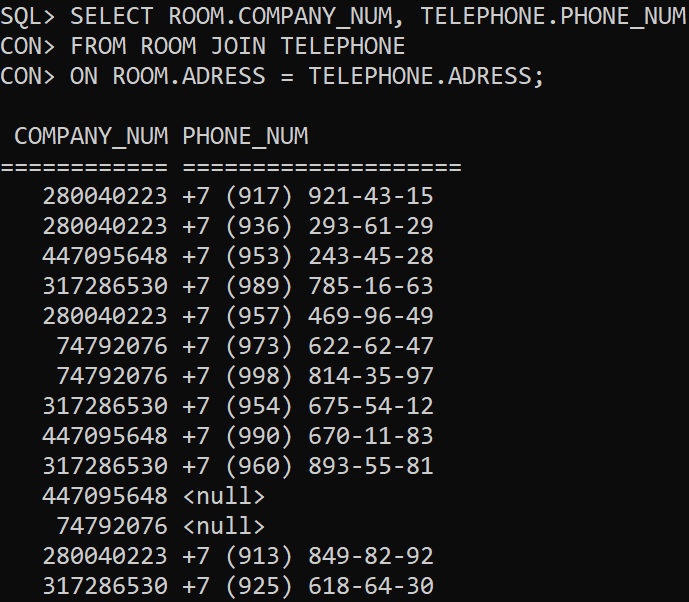


Рисунок 2.2 – Результат запроса соединения двух таблиц с использованием Join

2.3.2 Были соединены три таблицы сперва без использования Join и результат выполнения запроса отображен на рисунке 2.3. Запрос был следующий: “Вывести на экран из таблицы Помещение атрибут номер фирмы, из таблицы Тип помещения атрибут тип помещения, из таблицы Телефон атрибут номер телефона Из таблиц Помещение, Телефон, Тип помещения Где адрес из таблицы Помещение равен адресу из таблицы Телефон И где номер типа помещения в таблице Тип помещения равен номеру типа помещения из таблицы Помещение”.

SELECT ROOM.COMPANY\_NUM, ROOM\_TYPE.ROOM\_TYPE, TELEPHONE.PHONE\_NUM

FROM ROOM, TELEPHONE, ROOM\_TYPE

WHERE ROOM.ADRESS = TELEPHONE.ADRESS AND

ROOM\_TYPE.ROOM\_TYPE\_NUM = ROOM.ROOM\_TYPE\_NUM;

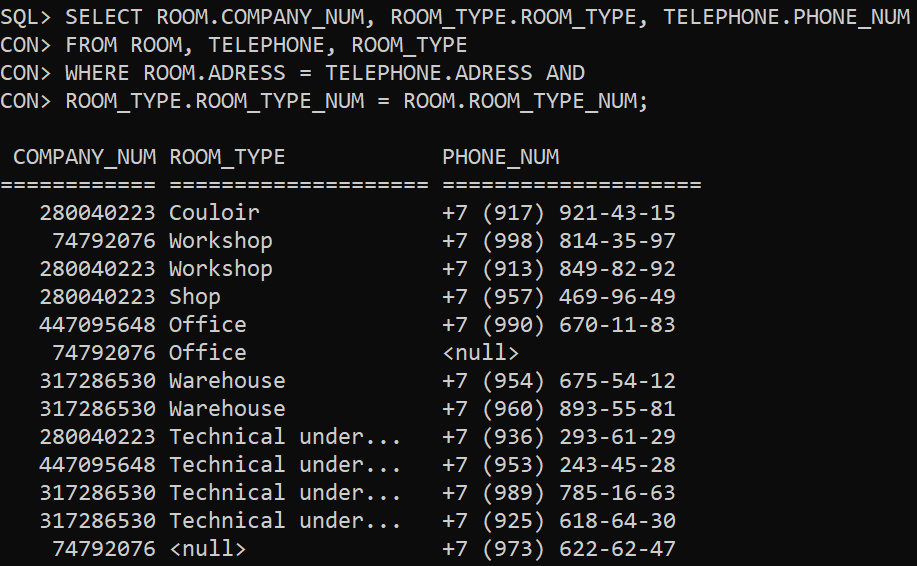


Рисунок 2.3 – Результат запроса соединения трёх таблиц без использования Join

Затем были соединены три таблицы сперва с использования Join и результат выполнения запроса отображен на рисунке 2.4. Запрос был следующий: “Вывести на экран из таблицы Помещение атрибут номер фирмы, из таблицы Тип помещения атрибут тип помещения, из таблицы Телефон атрибут номер телефона Из таблиц (Помещение соединяя с таблицей Телефон, где адрес из таблицы Помещение равен адресу из таблицы Телефон) Соединяя с таблицей Тип помещения, где номер типа помещения в таблице Тип помещения равен номеру типа помещения из таблицы Помещение Сортируя по номеру фирмы из таблицы Помещение”.

SELECT ROOM.COMPANY\_NUM, ROOM\_TYPE.ROOM\_TYPE, TELEPHONE.PHONE\_NUM

FROM (ROOM JOIN TELEPHONE ON ROOM.ADRESS = TELEPHONE.ADRESS)

JOIN ROOM\_TYPE ON ROOM\_TYPE.ROOM\_TYPE\_NUM = ROOM.ROOM\_TYPE\_NUM

ORDER BY ROOM.COMPANY\_NUM;

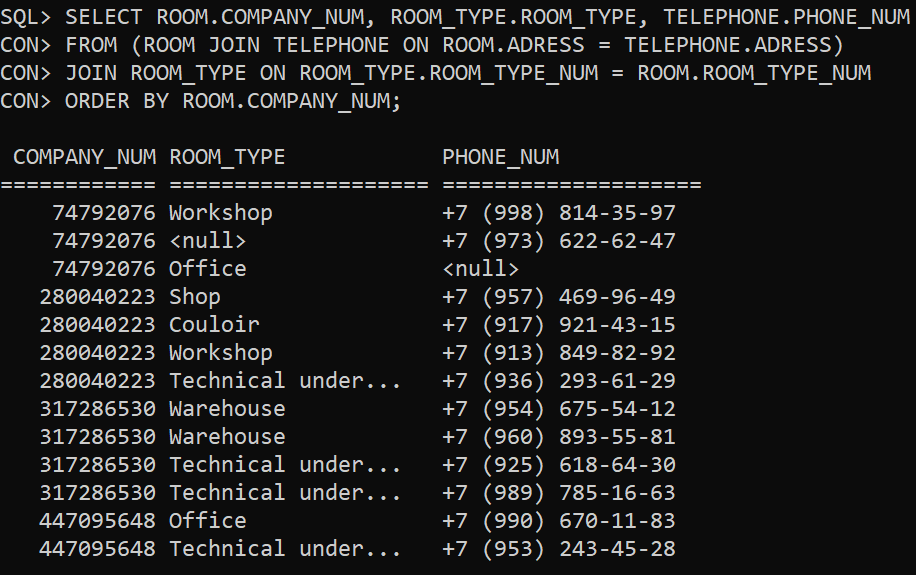


Рисунок 2.4 – Результат запроса соединения трёх таблиц с использованием Join

2.3.3 Был приведен пример запроса с подзапросом, содержащий следующую смысловую нагрузку: “Вывести на экран Размер помещения, Номер типа помещения, Номер фирмы из таблицы Помещение, где Номер фирмы равен (номеру фирмы из таблицы Фирма, где имя компании = «Мучной мастер»)”. Результат выполнения запроса отображен на рисунке 2.5.

SELECT AREA\_SIZE, ROOM\_TYPE\_NUM, COMPANY\_NUM FROM ROOM

WHERE COMPANY\_NUM =

(SELECT COMPANY\_NUM FROM COMPANY

WHERE COMPANY\_NAME = ‘Flour\_master’);

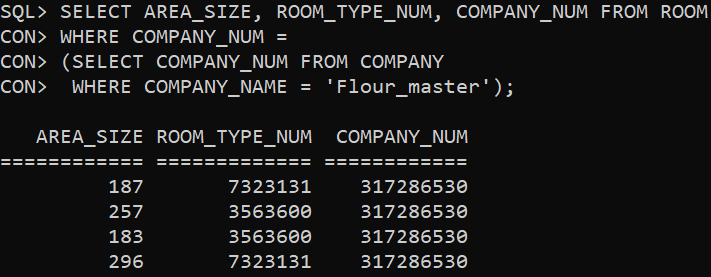


Рисунок 2.5 – Результат выполнения запроса с подзапросом

Для демонстрации использования агрегатных функций в подзапросах был сделан запрос означающий: “Вывести на экран номер фирмы, номер телефона из таблицы Телефон, где номер фирмы меньше среднего значения номера фирмы из таблицы Фирма”. Результат отображения запроса отображен на рисунке 2.6.

SELECT COMPANY\_NUM, PHONE\_NUM FROM TELEPHONE

WHERE COMPANY\_NUM <

(SELECT AVG(COMPANY\_NUM) FROM COMPANY);

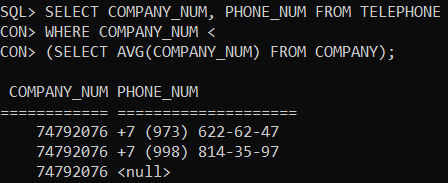


Рисунок 2.6 – Результат выполнения запроса с подзапросом с использованием агрегатной функции

Для демонстрации работы подзапроса возвращающего множественное значение был совершен запрос означающий: “Выбрать все атрибуты из таблицы Помещение, где номер типа помещения в таблице тип помещения не равен null”. Запрос выдал ожидаемый результат и отображен на рисунке 2.7.

SELECT \* FROM ROOM

WHERE ROOM\_TYPE\_NUM IN

(SELECT ROOM\_TYPE\_NUM FROM ROOM\_TYPE

WHERE ROOM\_TYPE <> '<null>');

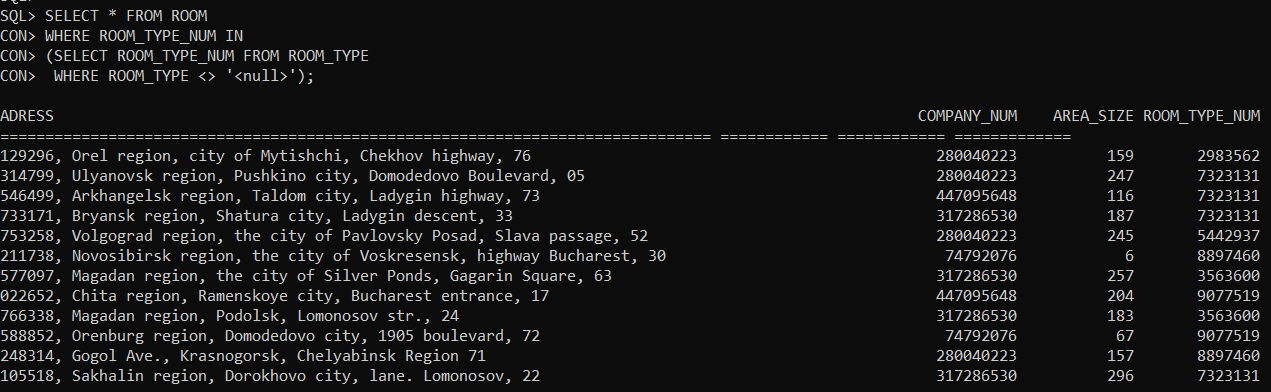


Рисунок 2.7 – Результат подзапроса, использующего множественное значение

Для демонстрации работы подзапросов, использующих вычисление был сделан запрос, означающий следующую информацию: “Вывести на экран номер фирмы, номер типа помещения из таблицы Помещение, где размер помещения должен быть меньше чем, номер типа помещения деленный на 100000 и где тип помещения офис”. Результат выполнения запроса отображен на рисунке 2.8.

SELECT COMPANY\_NUM, ROOM\_TYPE\_NUM FROM ROOM

WHERE AREA\_SIZE <

(SELECT ROOM\_TYPE\_NUM/100000 FROM ROOM\_TYPE

WHERE ROOM\_TYPE = 'Office');

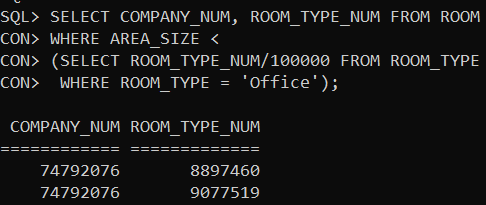


Рисунок 2.8 – Результат выполнения подзапроса с использованием вычисления

Наконец для демонстрации работы Having в подзапросах был сделан запрос означающий: “Вывести на экран номер фирмы, среднее значение размера помещения для этого номера фирмы, это среднее значение будет называться AVG\_AREA\_SIZE, все данные из таблицы Помещение, группируя по номеру фирмы, группировать только те значения где среднее значение размера помещения больше среднего значения помещения в таблице Помещение где номер фирмы не равен 74792076”. Результат выполнения запроса отображен на рисунке 2.9.

SELECT COMPANY\_NUM, AVG(AREA\_SIZE) AS AVG\_AREA\_SIZE FROM ROOM

GROUP BY COMPANY\_NUM

HAVING AVG(AREA\_SIZE) >

(SELECT AVG(AREA\_SIZE) FROM ROOM

WHERE COMPANY\_NUM <> 74792076);

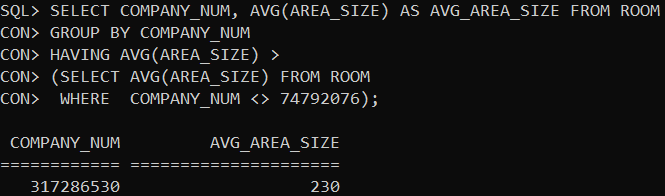


Рисунок 2.9 – Результат выполнения подзапроса с использованием Having

2.3.4 Сделан запрос, выводящий номера фирм, номера типов помещения и площадь помещения из таблицы помещение, где площадь помещения выше средней по таблице. Результат выполнения запроса отображен на рисунке 2.10.

SELECT COMPANY\_NUM, ROOM\_TYPE\_NUM, AREA\_SIZE

FROM ROOM

WHERE AREA\_SIZE >=

(SELECT AVG(AREA\_SIZE)

FROM ROOM);

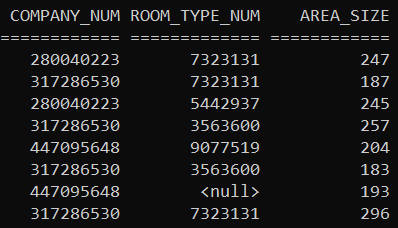


Рисунок 2.10 – Результат запроса, соединяющего таблицу со своей копией

2.3.5 Для демонстрации работы коррелированного запроса с двумя таблицами был сделан запрос, несущий в себе следующий смысл: “Вывести номера фирм, имеющие больше трёх помещений”. Результат выполнения запросы соответствует ожиданиям и отображен на рисунке 2.11.

SELECT COMPANY\_NUM FROM COMPANY main

WHERE 3 <

(SELECT COUNT (AREA\_SIZE) FROM ROOM

WHERE main.COMPANY\_NUM = COMPANY\_NUM);

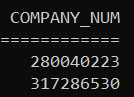


Рисунок 2.11 – Результат выполнения, коррелированного запроса с двумя таблицами

2.3.6 Были созданы запросы с использованием EXISTS, ALL и ANY для демонстрации их работы в языке SQL. Результаты выполнения запросов отображены на рисунках 2.12 – 2.14.

Вывести все значения из таблицы Телефон Где не существует компаний с помещениями площадью меньше 100.

SELECT \* FROM TELEPHONE main

WHERE EXIST

(SELECT COMPANY\_NUM FROM ROOM

WHERE AREA\_SIZE < 100 AND main.COMPANY\_NUM <> COMPANY\_NUM);

Выбрать все записи из таблицы телефон, где номер фирмы равен какому-либо номеру фирмы из таблицы Помещение, где размер помещения меньше 100.

SELECT \* FROM TELEPHONE main

WHERE COMPANY\_NUM = ANY

(SELECT COMPANY\_NUM FROM ROOM

WHERE AREA\_SIZE < 100);

Вывести кортежи из таблицы Помещение, где размер помещения больше, чем любой другой в этой фирме.

SELECT \* FROM ROOM main

WHERE AREA\_SIZE >= ALL (SELECT AREA\_SIZE FROM ROOM WHERE main.COMPANY\_NUM = COMPANY\_NUM);

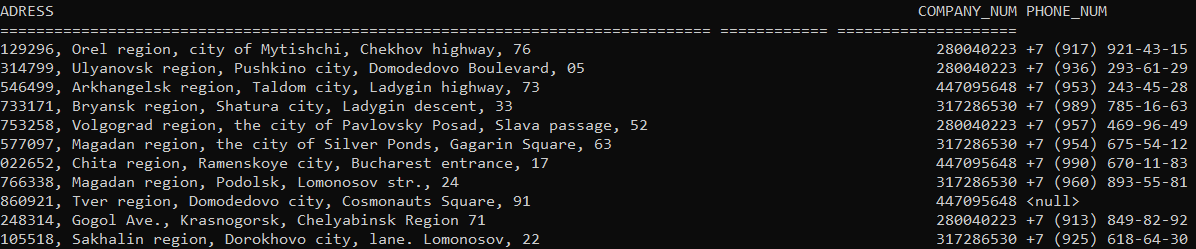


Рисунок 2.12 – Демонстрация работы оператора EXISTS

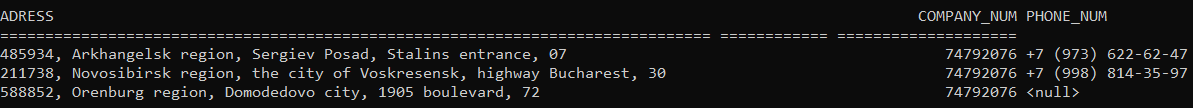


Рисунок 2.13 – Демонстрация работы оператора ANY

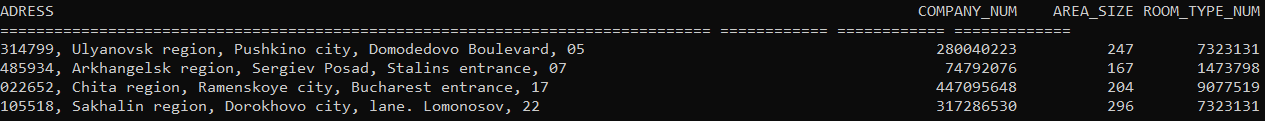


Рисунок 2.14 – Демонстрация работы оператора ALL

2.3.7 Были созданы генераторы для Номера фирмы и Номера типа помещения и установлены для них начальные значения.

CREATE GENERATOR COMPANY\_NUM\_GEN;

SET GENERATOR COMPANY\_NUM\_GEN TO 1;

CREATE GENERATOR ROOM\_TYPE\_NUM\_GEN;

SET GENERATOR ROOM\_TYPE\_NUM\_GEN TO 10;

Создано представление:

CREATE VIEW RoomType AS

SELECT ADRESS, COMPANY\_NUM, AREA\_SIZE, ROOM\_TYPE

FROM ROOM, ROOM\_TYPE

WHERE ROOM.ROOM\_TYPE\_NUM = ROOM\_TYPE.ROOM\_TYPE\_NUM;

Был создан триггер, который делает то, что от него хотят.

SET TERM !! ;

CREATE TRIGGER RoomType\_Insert2 FOR RoomType BEFORE INSERT AS

BEGIN

INSERT INTO ROOM values (NEW.ADRESS, NEW.COMPANY\_NUM, NEW.AREA\_SIZE, GEN\_ID(ROOM\_TYPE\_NUM\_GEN,1));

END !!

SET TERM ; !!

По правде говоря, я не понял, чего от меня хотят, чтобы я сделал. Хотелось бы чтобы постановка задачи формулировалась яснее. А так я пока могу лишь гадать что от меня хотят. Вставлю рисунок, демонстрирующий, что тригер всё же у меня написался без возникновения ошибок (Рисунок 2.15) и пожалуй на сегодня закончим.

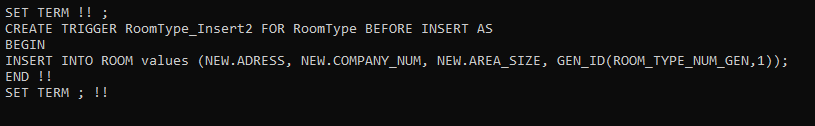


Рисунок 2.15 – Создание тригера

**Выводы**

При выполнении данной лабораторной работы были получены навыки работы с коррелированными и вложенными подзапросами, соединения двух и более таблиц с использованием и без использования Join. Были изучены соотнесения таблицы со своей копией. На практике были изучены и применены операторы EXISTS, ANY, ALL. Были изучены рекурсивные объединения и псевдонимы. Изучены и закрпелены на практике вложенные подзапросы с использованием агрегатных функций, вычислений, множественных значений. Ознакомились с принципом работы генераторов и триггеров. Полученные знания и опыт помогут более комфортно, качественно и быстро работать с базами данных в будущем с помощью языка запросов SQL.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение генераторов?

Генератор – объект базы данных, служащий для генерации последовательностей целых чисел.

2. Как сгенерировать следующее значение генератора?

Для того чтобы получить следующее значение генератора, необходимо применить функцию GEN\_ID (здесь name – имя генератора, step – инкремент генератора):

GEN\_ID( name, step);

3. Как переустановить значение генератора?

Для того чтобы установить генератор в другое значение, необходимо использовать команду SET GENERATOR:

SET GENERATOR name TO value;

4. Как удалить генератор?

DELETE FROM RDB$GENERATORS

WHERE RDB$GENERATORS\_NAME = 'Имя\_генератора';

5. Повышение надежности данных?

ДА, это важно. Триггер в основном используется для поддержания целостности информации в базе данных.

6. Организация многопользовательского режима доступа к данным?

Такое реализовать можно, но мы такое не изучали. А если и изучали, то я не знал.

7. Что такое «триггер»?

Триггер – процедура, автоматически вызываемая при операциях с таблицей.

8. Из каких частей состоит триггер?

Оператор состоит из заголовка триггера и тела триггера.

9. Какая информация содержится в заголовочной части триггера?

- имя триггера, уникальное по БД;

- имя таблицы, с которой ассоциируется триггер;

- указание на момент, когда триггер должен вызываться.

10. Как сделать триггер временно неактивным? Как удалить триггер?

ACTIVE|INACTIVE

Указывает «активность» триггера. Не активные триггеры игнорируются. По умолчанию триггер активен.

Удаление:

DROP TRIGGER <имя триггера>

11. Для чего используются триггеры?

Триггер в основном используется для поддержания целостности информации в базе данных.

12. Назовите элементы языка хранимых процедур и триггеров.

- BEGIN...END

определяет блок операторов (операторные скобки). Скобка после END не нужна.

- variable = expression

оператор присваивания

- /\* comment\_text \*/

многострочный комментарий

- EXCEPTION exception\_name

вызывает исключительную ситуацию с именем exception\_name, если она не обрабатывается оператором WHEN

- EXECUTE PROCEDURE

proc\_name [var [, var ...]]

[RETURNING\_VALUES var [, var ...]]

Вызывает хранимую процедуру с именем proc\_name, с указанными входными и выходными параметрами. Параметры процедуры должны быть локальными переменными.

- FOR select\_statement DO

compound\_statement

повторяет выполнение блока кода следующего за DO для каждой строки, возвращенной оператором select\_statement.

- select\_statement.

select\_statement - обычный запрос на выборку, за исключением того, что он обязательно должен содержать часть INTO, и данная часть должна идти на последнем месте.

- compound\_statement

или одиночный оператор языка, или блок операторов заключенных в операторные скобки.

- IF (condition)

THEN compound\_statement [ELSE

compound\_statement]

условный оператор condition - условие, выражение булевской трехзвенной логики (TRUE, FALSE, UNKNOWN). Обычно два выражения как операнды оператора сравнения.

- NEW.column

контекстная переменная, содержащая новое значение столбца с именем column при выполнении операций INSERT или UPDATE.

- OLD.column

контекстная переменная, содержащая старое значение столбца с именем column при выполнении операций INSERT или UPDATE.

- POST\_EVENT event\_name

отсылает сообщение с именем event\_name.

- WHILE (condition) DO compound\_statement

цикл с предусловием.

- WHEN {error [, error .]|ANY}

DO compound\_statement

Оператор обработки исключительных ситуаций и ошибок. В случае если имеет место одна из ошибок перечисленных в операторе WHEN, вызывается compound\_statement. Оператор WHEN должен быть последним оператором перед END тела триггера. Error: EXCEPTION exception\_name, SQLCODE errcode or GDSCODE number. Ошибкой может быть - возбуждение исключительной ситуации с именем exception\_name, равенство константы SQLCODE значению errcode, равенство константы GDSCODE значению number. ANY – обрабатывает любую ошибку, если она имела место.