Департамент образования Вологодской области бюджетное профессиональное образовательное учреждение

Вологодской области «Череповецкий химико-технологический колледж»

Специальность 09.02.07

Информационные системы и программирование

##### Отчёт по практическим работам по дисциплине Oracle PL/SQL

Проект разработал: Кудряшов Дмитрий Александрович, студент группы 82/2022 (подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководители практики от предприятия: Цветков К.Н., Яфизова А.В.

(подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Череповец, 2024

[1. ПРАКТИЧЕСКАЯ РАБОТА №1 4](#_Toc188607294)

[1.1 Практическое задание №1 5](#_Toc188607295)

[1.2 Выполнение практического задания №1 6](#_Toc188607296)

[1.3 Заключение по выполнению практической работы №1 8](#_Toc188607297)

[2. ПРАКТИЧЕСКАЯ РАБОТА №2 9](#_Toc188607298)

[2.1 Практическое задание №2 11](#_Toc188607299)

[2.2 Выполнение практического задания №2 12](#_Toc188607300)

[2.3 Заключение по выполнению практической работы №2 13](#_Toc188607301)

[3. ПРАКТИЧЕСКАЯ РАБОТА №3 14](#_Toc188607302)

[3.1 Практическое задание №3 18](#_Toc188607303)

[3.2 выполнение практического задания №3 19](#_Toc188607304)

3.3 Заключение по выполнению практической работы №3 .................. 26

1. Практическая работа №4. ........................................................................... 27
   1. Практическое задание №4 ................................................................... 29
   2. Выполнение практического задания № 4 .......................................... 30
   3. Заключение по выполнению практической работы №4. ................. 31
2. Практическая работа №5 ............................................................................ 32
   1. Практическое задание №5. .................................................................. 33
   2. Выполнение практического задания №5. .......................................... 34
   3. Заключение по выполнению практической работы№ 5 .................. 35

Cписок литературы ......................................................................................... 36

**ВВЕДЕНИЕ**

В отчете по практическим работам по дисциплине Oracle PL/SQL предполагается представить описание выполненных заданий, результаты заданий в виде рисунков и выводы по каждой работе по изучению языка программирования PL/SQL в контексте работы с базой данных Oracle. Отчет будет включать в себя описание выполненных задач, обзор использованных конструкций и инструментов языка PL/SQL. Целью данной дисциплины является изучение и практическое применение данного языка программирования.

# 1. ПРАКТИЧЕСКАЯ РАБОТА №1

**PL/SQL** (Procedural Language/Structured Query Language) – язык программирования, встроенный в СУБД Oracle. Этот язык используется для создания хранимых процедур, триггеров, функций и пакетов, которые позволяют управлять данными и осуществлять бизнес-логику непосредственно на сервере базы данных.

**База данных** – это организованная коллекция данных, хранящихся электронно на компьютере или сервере. Она представляет собой структурированный набор данных, который можно легко обрабатывать, обновлять, извлекать и управлять.

**Предметная область** – это конкретная область знаний, которая изучается или рассматривается в определенном контексте.

**Тип данных** – это концепция в программировании, которая определяет, какие виды данных могут храниться и обрабатываться в компьютерной программе.

Язык SQL предназначен для работы с реляционными базами данных — множеством таблиц, которые связаны между собой.

**Первичный ключ** – это уникальный идентификатор, который используется для однозначной идентификации каждой записи в базе данных.

### 1.1 Практическое задание №1

Изучение предметной области. Продумать базу данных, а именно составить список таблиц с полями и описать их (указать название таблицы, полей, их тип данных). Написать программу вывода таблиц в Pl/Sql.

### 1.2 Выполнение практического задания №1

1. Определился с выбором предметной области, по которой буду, в дальнейшем, создавать базу данных. Предметная область – **<название предметной области>**.
2. Тщательно изучил данную предметную область, отобрал нужные данные для создания базы данных.
3. Определил ключевые атрибуты, которые будут являться столбцами в таблицах, типы данных для каждого столбца.

4. Создал модель базы данных в DBdesigner и заполнил столбцы с данными. (Рисунок 1)

5. После создания модели в DBdesigner создал эти таблицы в PL/SQL. (Рисунок 2)

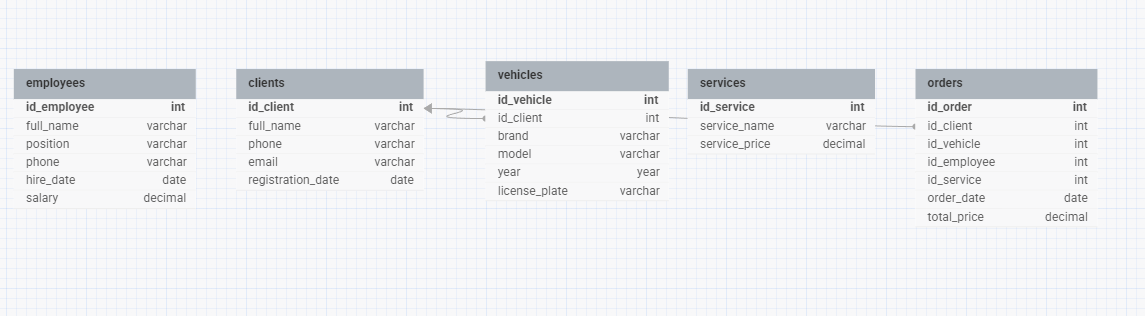


Рис. 1. Модель в DBdesigner

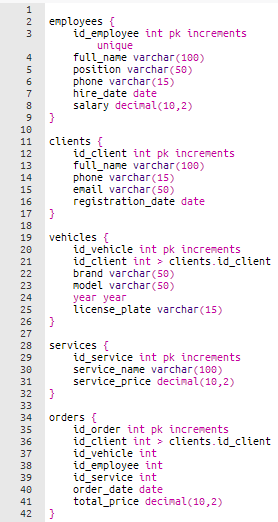


Рис. 2. Создание таблиц в PL/SQL

### 1.3 Заключение по выполнению практической работы №1

1. Я изучил предметную область и определил структуры базы данных, учитывая сущности и связи между ними.

2.Определил атрибуты, являющиеся столбцами в таблицах.

3.Создал таблицы и заполнил столбцы данными.

4.Воспользовался инструментами Pl/Sql, перенёс таблицы в программу.

5.Проверил программу на работоспособность.

Вывод: Исходя из задания, я пришёл к выводу, что составление списка таблиц с описанием полей и их типов данных позволяет создавать чёткую структуру для хранения информации. Это основа для дальнейшей работы. Написание программы вывода таблиц в PL/SQL предоставляет инструмент для получения доступа к данным базе, что важно для работы с хранимой информацией.

# 2. ПРАКТИЧЕСКАЯ РАБОТА №2

**Нормальная форма базы данных** – это набор правил и критериев, которым должна отвечать база данных.

**Цель нормализации:** исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении кортежей (строк таблицы).

**Процесс нормализации** – это последовательный процесс приведения базы данных к эталонному виду, т.е. переход от одной нормальной формы к следующей.

В реальном мире нормализация до третьей нормальной формы (3NF) является обычной, стандартной практикой, так как 3NF устраняет достаточное количество аномалий, при этом производительность базы данных, а также удобство ее использования не снижается, что нельзя сказать о всех последующих формах.

**1НФ, 2НФ и 3НФ** – это первые три типа нормализации базы данных. Они обозначают первую нормальную форму, вторую нормальную форму и третью нормальную форму соответственно.

Все типы нормализации базы данных являются кумулятивными – это означает, что каждый из них строится поверх тех, что находятся под ним. Таким образом, все концепции из 1НФ также переносятся в 2НФ, и так далее.

**Первая нормальная форма** должна соответствовать следующим критериям:

* одна ячейка не должна содержать более одного значения (атомарность);
* для идентификации должен быть первичный ключ;
* нет дублирующихся строк или столбцов;
* каждый столбец должен содержать только одно значение для каждой строки в таблице.

**Вторая нормальная форма** соответствует следующим критериям:

* таблица уже находится в 1НФ;

-не имеет частичной зависимости, то есть все неключевые атрибуты полностью зависят от первичного ключа.

**Третья нормальная форма** соответствует если:

* таблица находится в формате 2НФ, это устраняет повторяющиеся группы и избыточность, но не устраняет транзитивную частичную зависимость.

Это означает, что не простой атрибут зависит от другого не простого атрибута. Это то, что исключает третья нормальная форма.

### 2.1 Практическое задание №2

1. Привести базу данных к 3 нормальной форме.
2. Создать все таблицы и заполнить их данными (не менее 10 записей в каждой таблице).

### 2.2 Выполнение практического задания №2

1. Самостоятельно изучил тему «Нормализация БД». Начал постепенно приводить БД к 3 нормальной форме.
2. Привёл таблицы к 1 НФ.
3. Привёл таблицы ко 2 НФ.
4. Привёл таблицы к 3 НФ.
5. По возможности сделал в каждой таблице не менее 10 записей.
6. После создания таблиц в excel начал переносить их в PL/SQL. (Рисунок 4)

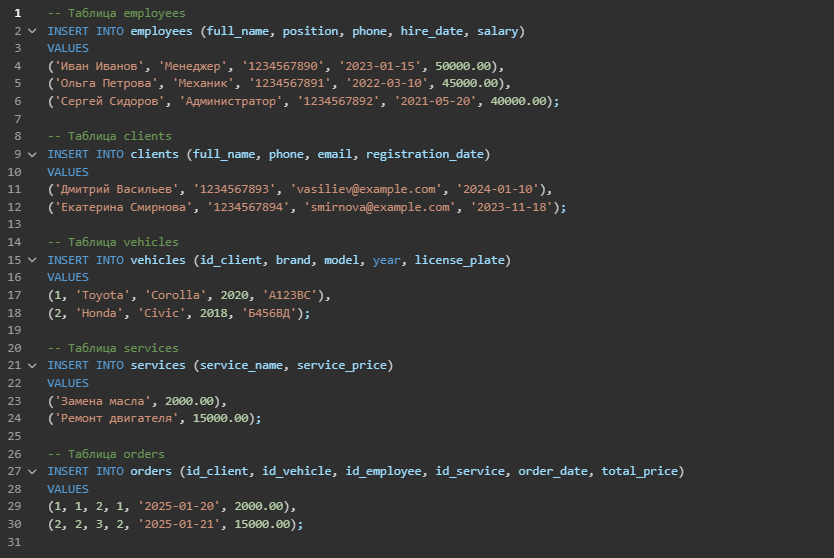


Рис. 4. Таблица в PL/SQL в 3 НФ

### 2.3 Заключение по выполнению практической работы №2

1. Я самостоятельно изучил информацию по нормализации таблиц в базе данных
2. Проанализировал имеющиеся таблицы на наличие повторяющихся данных и связей между ними и произвёл нормализацию, устраняя избыточность данных.
3. С помощью приведённых таблиц, написал рабочий код в программе PLSQL Developer.

Вывод: в результате выполнения этого задания я приобрёл новые знания и навыки в области проектирования баз данных, узнал о важности нормализации для избежания избыточности данных и обеспечения их целостности. Это позволило мне понять, как эффективно организовать информацию в базах данных, уменьшая дублирования и упрощая их обслуживание.

# 3. ПРАКТИЧЕСКАЯ РАБОТА №3

**Join** — оператор, который используют, чтобы объединять строки из двух или более таблиц на основе связующего столбца между ними. Такой столбец еще называют ключом.

Общий синтаксис оператора join:

JOIN <Название таблицы для присоединения> ON <Условие присоединения на основе связующих столбцов>. Соединять можно и больше двух таблиц: к запросу добавьте еще один оператор join. **Внутреннее соединение INNER JOIN**

Оператор языка SQL JOIN предназначен для соединения двух или более таблиц базы данных по совпадающему условию. Этот оператор существует только в реляционных базах данных. Именно благодаря JOIN реляционные базы данных обладают такой мощной функциональностью, которая позволяет вести не только хранение данных, но и их, хотя бы простейший, анализ с помощью запросов.

Если использовать оператор INNER JOIN, в результат запроса попадут только те записи, для которых выполняется условие объединения. Еще одно условие — записи должны быть в обеих таблицах. В финальный результат из примера выше не попали записи с CustomerID=23 и OrderID=304102: для них нет соответствия в таблицах.

Запрос с оператором INNER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON. **Внешние соединения OUTER JOIN**

Если использовать внешнее соединение, то в результат запроса попадут не только записи с совпадениями в обеих таблицах, но и записи одной из таблиц целиком. Этим внешнее соединение отличается от внутреннего.

Указание таблицы, из которой нужно выбрать все записи без фильтрации, называется направлением соединения.

##### LEFT OUTER JOIN / LEFT JOIN

В финальный результат такого соединения попадут все записи из левой, первой таблицы. Даже если не будет ни одного совпадения с правой. И записи из второй таблицы, для которых выполняется условие объединения.

LEFT JOIN – левая\_таблица LEFT JOIN правая\_таблица ON условия\_соединения.

Возвращаются все строки левой\_таблицы (ключевое слово LEFT). Данными правой\_таблицы дополняются только те строки левой\_таблицы, для которых выполняются условия\_соединения. Для недостающих данных вместо строк правой\_таблицы вставляются NULL-значения.

##### RIGHT OUTER JOIN / RIGHT JOIN

В финальный результат этого соединения попадут все записи из правой, второй таблицы. Даже если не будет ни одного совпадения с левой. И записи из первой таблицы, для которых выполняется условие объединения. Запрос с оператором RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из второй по порядку (правой) таблицы, даже если они не соответствуют условию. У записей правой таблицы, которые не соответствуют условию, значение столбца из левой таблицы будет NULL (неопределённым).

RIGHT JOIN – левая\_таблица RIGHT JOIN правая\_таблица ON условия\_соединения.

Возвращаются все строки правой\_таблицы (ключевое слово RIGHT). Данными левой\_таблицы дополняются только те строки правой\_таблицы, для которых выполняются условия\_соединения. Для недостающих данных вместо строк левой\_таблицы вставляются NULL-значения.

##### FULL OUTER JOIN / FULL JOIN

В финальный результат такого соединения попадут все записи из обеих таблиц. Независимо от того, выполняется условие объединения или нет.

Возвращаются все строки левой\_таблицы и правой\_таблицы. Если для строк левой\_таблицы и правой\_таблицы выполняются условия\_соединения, то они объединяются в одну строку. Для строк, для которых не выполняются условия\_соединения, NULL-значения вставляются на место левой\_таблицы, либо на место правой\_таблицы, в зависимости от того данных какой таблицы в строке не имеется.

##### Перекрестное соединение CROSS JOIN

Этот оператор отличается от предыдущих операторов соединения: ему не нужно задавать условие объединения (ON table1.column\_name = table2.column\_name). Записи в таблице с результатами — это результат объединения каждой записи из левой таблицы с записями из правой. Такое действие называют декартовым произведением.

Объединение каждой строки левой\_таблицы со всеми строками правой\_таблицы. Этот вид соединения иногда называют декартовым произведением.

##### Изменение одного значения

Изменение значения всех полей в таблице необходимо крайне редко. Чаще всего необходимо поменять значение какой-то конкретной записи. Для этого в завершении строки с командой UPDATE будет добавлена директива WHERE, в которой указывается условие, определяющее с какой именно строкой нужно выполнить операцию обновления.

##### Внесение изменений в несколько строк с условием отбора

Если вспомнить все многообразие условий в запросе, можно представить себе насколько разнообразными могут быть выборки. Поэтому и запросы на обновления могут выполняться либо с одной строкой, либо с группой строк, либо со всеми строками таблицы. Все зависит от стоящей перед вами задачи, а также с какими строками таблицы нужно выполнять операции обновления. **Обновление значений в нескольких полях строки**

При необходимости обновлять сразу несколько полей, все поля с их значениями указываются после директивы SET через запятую.

##### Использование оператора DELETE с условием

Cпособ удаления данных из ячейки SQL - использование оператора DELETE с условием. Оператор DELETE позволяет удалять строки из таблицы

## 3.1 Практическое задание №3

1. Самостоятельно изучил тему «Соединение таблиц». Написал не менее 5 запросов (!= 5), которые будут выводить данные из 2-3 таблиц.
2. Обновил поле таблицы, исходя из данных.
3. Удалил записи в таблице, данные в которых не актуальны.

## 3.2 выполнение практического задания №3

1. Первым для соединения я выбрал таблицы автомастерской, владельцев автомастерской. Использовал классический вариант соединения JOIN…ON. (Рисунок 13)



Рис. 13. Соединение таблиц автомастерской, владельцев

2. Вторым для соединения я выбрал таблицы услуги и мастера. Соединял также с помощью варианта JOIN…ON. (Рисунок 14)

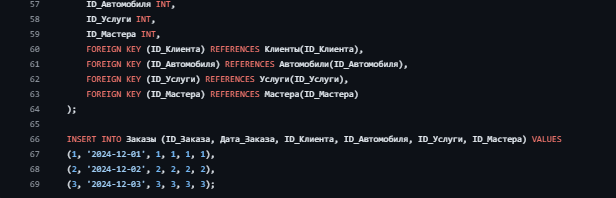


Рис. 14. Соединение таблиц услуги и мастера

###### 3.3 Заключение по выполнению практической работы №3

1. В процессе выполнения задания по изучению темы «Соединение таблиц» и написанию запросов для получения данных из нескольких таблиц, я повысил свой уровень знаний в области SQL и работе с различными типами соединений (INNER JOIN, LEFT JOIN, RIGHT JOIN и т. д.).

2.Выполнение задания по обновлению полей в таблице на основе имеющихся данных позволило мне получить практические навыки в обновлении информации в базе данных с помощью оператора UPDATE в SQL.

3. Задание по удалению записей в таблице на основе условий оказалось полезным для понимания процесса управления данными в базе данных и использования оператора DELETE.

Вывод: в результате выполнения этого задания я повысил свои знания в области с соединениями. Эта работа позволила мне лучше понять, как объединить данные из различных таблиц на основе определенных условий.

##### 4. ПРАКТИЧЕСКАЯ РАБОТА №4

**Встроенные исключительные ситуации** – это исключительные ситуации, которые имеют определенные имена в PL/SQL. Они определены в стандартном пакете в PL/SQL и не могут быть определены программистом.

**Хранимая процедура** – это блок инструкций Oracle SQL, которые компилируются один раз и затем могут использоваться несколько раз. Объединение инструкций в блок и последующая их отправка между приложением и базой данных сокращает сетевой трафик. Oracle создает расширения процедурного языка для SQL (PLSQL) для добавления программных функций к SQL-запросам или операторам.

В Oracle существует 3 типа хранимых процедур:

1. Автономные подпрограммы – процедуры, созданные вне пакета.
2. Подпрограммы - Процедуры, определенные в пакете
3. Локальные подпрограммы – процедуры, вложенные в другие подпрограммы.

**CREATE OR REPLACE PROCEDURE** – это ключевое слово, используемое для создания процедуры хранения. Ключевые слова Replace изменят параметр или другую информацию без удаления существующей процедуры.

IS: содержит все объявленные инструкции. Если вы хотите создать какуюлибо переменную для хранимой процедуры, то ее можно разместить здесь.

BEGIN – это основная игровая площадка, на которой написана вся программа для выполнения хранимой процедуры в соответствии с требованием. Это начало основного содержимого.

EXCEPTION: срабатывало, когда программа выдавала какую-либо непредвиденную ошибку.

END: объявляет завершение хранимой процедуры. Когда программа считывает end, она понимает, что теперь все операции завершены.

INSERT INTO используется:

1. Для вставки новых строк в таблицу базы данных.
2. Синтаксис инструкции SQL.
3. Для вставки новой строки в таблицу базы данных.
4. Вставить в SQL сразу несколько строк.
5. Для вставки новых строк в базу данных INSERT INTO table\_name VALUES.
6. Вставка несколько строк в таблицу table\_name.

Функция SQL COUNT используется для подсчета количества записей в таблице. Она имеет следующий синтаксис: SELECT COUNT(\*) FROM имя\_таблицы

Функция COUNT принимает один из нескольких параметров:

1. означает то, что функция COUNT возвращает все записи в таблице; column\_name - функция COUNT возвращает количество записей конкретного столбца (только NOT NULL);
2. DISTINCT column\_name - функция COUNT возвращает количество только разных записей конкретного столбца (только NOT NULL).

SELECT COUNT, возвращающий количество записей в таблице users.

SELECT COUNT, возвращающий количество записей в таблице users. Конструкторы и методы для типа данных count. v\_count и cnt являются синонимами, каждый из которых создает новый v\_count объект, созданный из подкласса vctrs\_vctr.

###### 4.1 Практическое задание №4

Написать процедуру вставки данных в таблицу. При успешном завершении выдавать сообщение - «запись вставлена», в случае ошибки - вывод сообщения. Использовать exception.

Написать функцию, которая будет возвращать кол-во записей из таблицы с определенным фильтром. Таблицу выбираем самостоятельно. У функции должен быть как минимум 1 входной параметр - фильтр.

###### 4.2 Выполнение практического задания №4

1. Изучил материал для написания процедуры вставки в таблицу и функцию exception.
2. После поиска материалов, определился с командами для написания кода и таблицу, в которую буду вставлять данные.
3. С помощью CREATE OR REPLACE PROCEDURE выбрал параметры для вставки в таблицу.
4. С помощью is создал переменную для хранимой процедуры.
5. Для вставки новых строк в таблицу использовал конструкцию INSERT INTO …. VALUES ….
6. Далее вывел сообщение «Запись вставлена».
7. Написал EXCEPTION для выведение непредвиденных ошибок процедуры.
8. Запустил процедуру, все сработало. (рис.20)

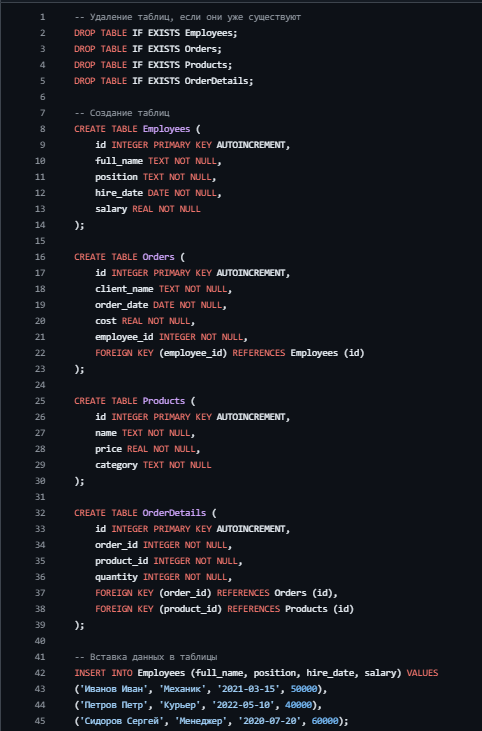
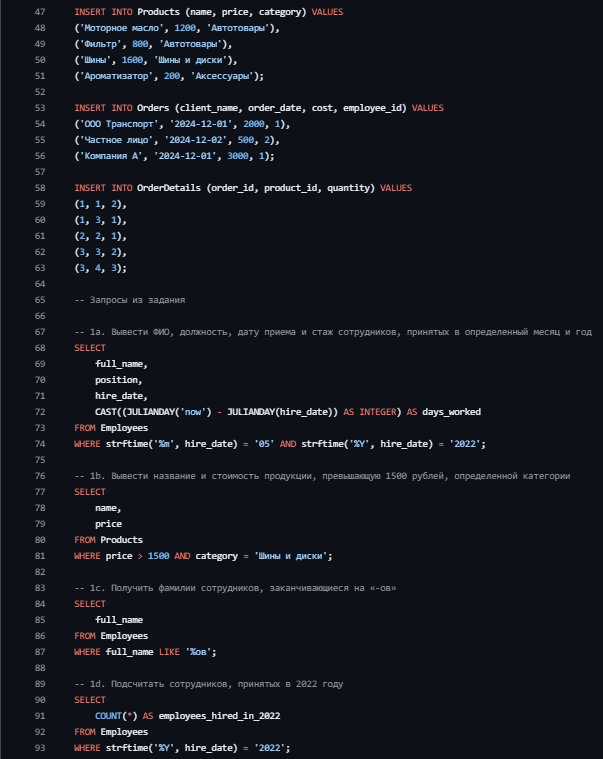
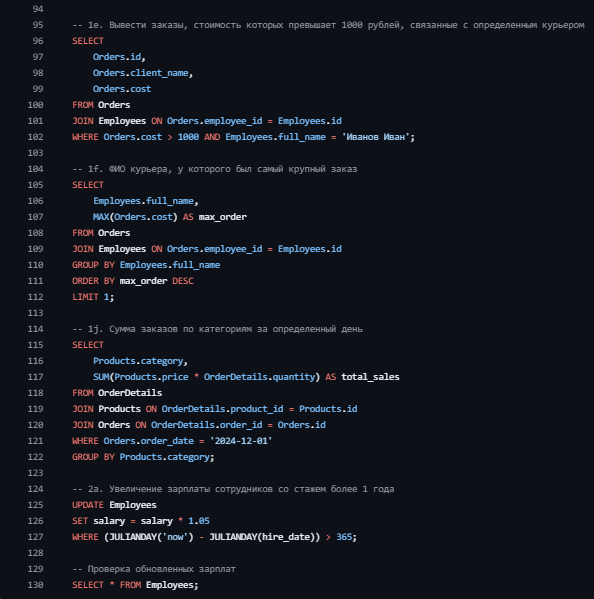


Рисунок 20 - Процедура вставки данных в таблицу

Рисунок 21 - Подсчёт кол-ва записей с определенным фильтром

###### 4.3 Заключение по выполнению практической работы №4

По результатам выполнения заданий, написанная мной процедура вставки данных в таблицу позволяет эффективно добавлять информацию, а использование механизма исключений помогает обрабатывать ошибки при выполнении операций вставки, что сделает работу с базой данных более безопасной и надежной.

Функция, которую я написал, возвращает количество записей из таблицы в зависимости от заданного фильтра. Эта функция может быть полезной при выполнении запросов, которые требуют указанного подмножества данных.

Вывод: с помощью написания процедуры баз вставки данных в таблицу, с использованием механизма исключений, я освоил методы обработки ошибок и выполнения соответствующих действий. Создание функции, возвращающей количество записей из таблицы с учётом определенного фильтра, помогло мне понять, как использовать входные параметры функции для более универсальной обработки данных.

##### 5. ПРАКТИЧЕСКАЯ РАБОТА №5

**DBMS\_RANDOM** может быть инициализирован явно, но не требует инициализации перед вызовом генератора случайных чисел. Он автоматически инициализируется датой, идентификатором пользователя и идентификатором процесса, если явная инициализация не выполняется.

Инструкция Oracle SELECT используется для извлечения записей из одной или нескольких таблиц в базе данных Oracle.

SQL оператор ORDER BY используется для сортировки записей в наборе результатов запроса SELECT.

**ROWNUM** – это виртуальный столбец (не настоящий столбец), Может использоваться в запросах. ROWNUM будет преобразован в числа 1, 2, 3, 4, ..., N, где N - количество строк в наборе, используемом с ROWNUM. Значение ROWNUM не всегда будет присвоено строке (записи). (Это распространенное заблуждение). Строке (записи) в таблице нет соответствующего номера, вы не можете попросить взять пятую строку из таблицы - такого нет.

INSERT INTO используется:

1. Для вставки новых строк в таблицу базы данных.
2. Синтаксис инструкции SQL.
3. для вставки новой строки в таблицу базы данных.
4. Вставить в SQL сразу несколько строк.
5. Для вставки новых строк в базу данных INSERT INTO table\_name VALUES.
6. Вставка несколько строк в таблицу table\_name.

* 1. **Практическое задание №5**

Написать процедуру заполнения таблицы случайными значениями, использовать пакет dbms\_random.

* 1. **Выполнение практического задания №5**
     1. Самостоятельно изучил материал для выполнение практического задания.
     2. Выбрал таблицу для заполнения случайными значениями.
     3. Первой строкой запустил объявление процедуры с именем

"fill\_art\_randomly" и объявил пользовательский тип таблицы, для этого использовал код CREATE OR REPLACE PROCEDURE….. TYPE…. IS table OF varchar2(100).

* + 1. Далее объявил и инициализировала экземпляр типа "full\_name\_array" с именем "full\_name" с примерами полных имен. С помощью full\_name full\_name\_array := full\_name\_array.
    2. Далее объявил переменную с именами и и установленными типами.
    3. Сгенерировал случайную дату рождения путем вычитания случайного числа дней, с помощью : birthday\_date := (to\_date('2000/09/28', 'yyyy/mm/dd') - round(dbms\_random.value(1, 400)));
    4. Выбрал случайный идентификатор произведения из таблицы "art". Этот подзапрос упорядочивает идентификаторы произведений случайным образом, а затем выбирает первый из них, используя ROWNUM.
    5. Сгенерировал случайный индекс между 1 и количеством элементов в массиве командой: "full\_name".random\_index := DBMS\_RANDOM.VALUE(1, full\_name.COUNT);
    6. Далее так же сгенерировал случайное ФИО.
    7. С помощью команды INSERT INTO…. VALUES вставил случайно выбранное полное имя, дату рождения и идентификатор произведения в свою таблицу.
    8. Сохранил изменения в таблице. (Рисунок 22)
    9. Проверил код на работоспособность.

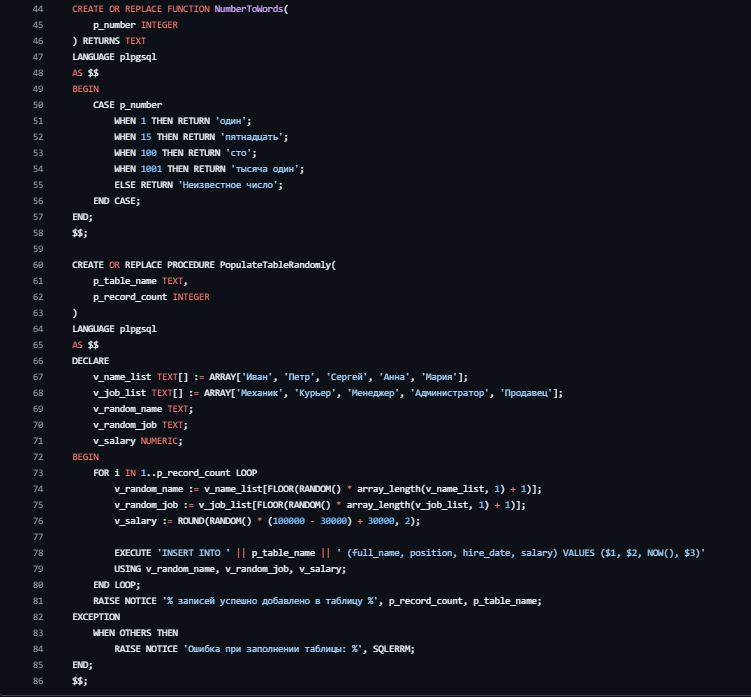
 

Рисунок 22 - Процедура практической работы №5

###### 5.3 Заключение по выполнению практической работе № 5

1. Использование пакета dbms\_random предоставляет доступ к различным функциям для генерации случайных значений, что позволяет создать реалистичную выборку данных для таблицы.
2. Оптимальное использование массивов или справочников может упростить процесс и улучшить производительность при заполнении таблицы.

Вывод: при выполнении этого задания, я научился использовать пакет dbms\_random в PL/SQL для генерации случайных данных, и это поможет мне лучше понимать и использовать возможности генерации случайных данных.

##### Список источников

1. CREATE OR REPLACE PROCEDURE:

https://docs.oracle.com/en/database/oracle/oracle-database/21/lnpls/plsql-

subprograms.html#GUID-E03F512B-1A30-43B4-8DCA-64DA68AE7CE4

1. round sql: https://function-x.ru/sql\_round.html
2. dbms\_random: https://oracle-base.com/articles/misc/dbms\_random
3. Информация о базе данных: https://habr.com/ru/articles/564390/#что-такоеsql
4. Создание таблиц: https://habr.com/ru/articles/564390/#что-такое-sql
5. Материал по НФ баз данных: https://info-comp.ru/database-normalization
6. Соединение таблиц: https://sky.pro/media/join-v-sql/
7. Обновление данных: https://space-base.ru/library/sql/zaprosyi-sql-dlyaobnovleniya-dannyix-update
8. Exeption: https://oracleplsql.ru/named-system-exceptions.html