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

**Содержание**

[Введение 3](#_Toc154186003)

1. Практическая работа №1[. 4](#_Toc154186005)

1.1 [Практическое задание №1 4](#_Toc154186006)

1.2 Выполнение [практического задания №1 4](#_Toc154186007)

1.3 [Заключение по выполнению практической работы №1. 5](#_Toc154186008)

2. [Практическая работа №2 6](#_Toc154186009)

2.1 [Практическое задание №2. 7](#_Toc154186010)

2.2 [Выполнение практического задания №2 7](#_Toc154186011)

2.3 [Заключение по выполнению практической работы №2. 7](#_Toc154186012)

3. [Практическая работа №3. 8](#_Toc154186013)

3.1 [Практическое задание №3 8](#_Toc154186014)

3.2 [Выполнение практического задания №3. 8](#_Toc154186015)

3.3 [Заключение по выполнению практической работы №3 9](#_Toc154186016)

4. [Практическая работа №4. 10](#_Toc154186017)

4.1 [Практическое задание №4 12](#_Toc154186018)

4.2 [Выполнение практического задания № 4 13](#_Toc154186019)

4.3 [Заключение по выполнению практической работы №4. 14](#_Toc154186020)

5. Практическая работа №5....................................................................... 14

5.1 Практическое задание №5................................................................... 16

5.2 Выполнение практического задания №5........................................... 17

5.3 Заключение по выполнению практической работы№ 5 .................. 20

**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**

**1.1 Практическое задание №1**

Выбрать предметную область из предложенного списка. Важно, чтобы

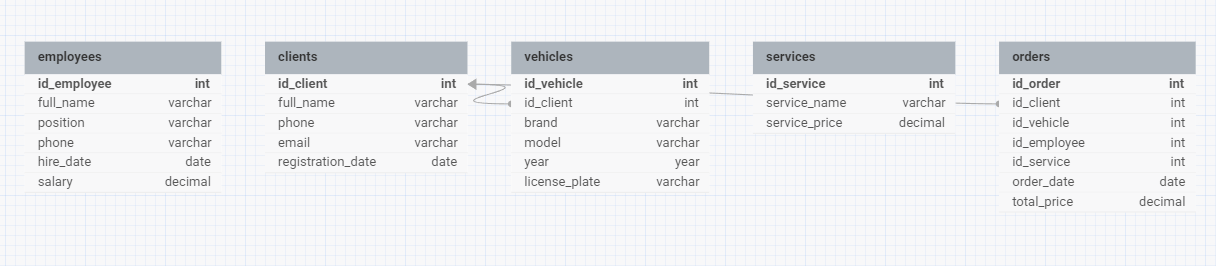
предметные области не повторялись! По этой предметной области необходимо

продумать базу данных, а именно составить список таблиц с полями и их

описать (указать название таблицы, полей, их тип данных). Не менее 5 таблиц.

**1.2 Выполнение практического задания №1**

1. Определился с выбором предметной области, по которой буду, в дальнейшем, создавать базу данных. Предметная область – **Автомастерская**.
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 таблицы на сайте Dbdesigner.



**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 в Dbdesigner.

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.

# **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.

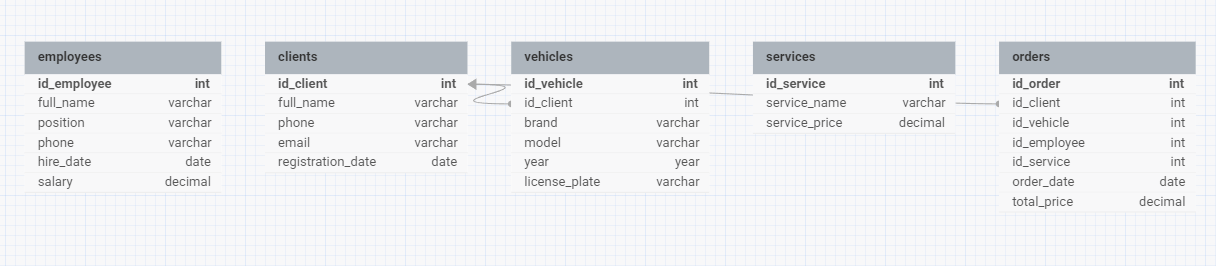
.

**2.1 Практическое задание №2**

1. Привести базу данных к 3 нормальной форме.

# **2.2 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.



**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.

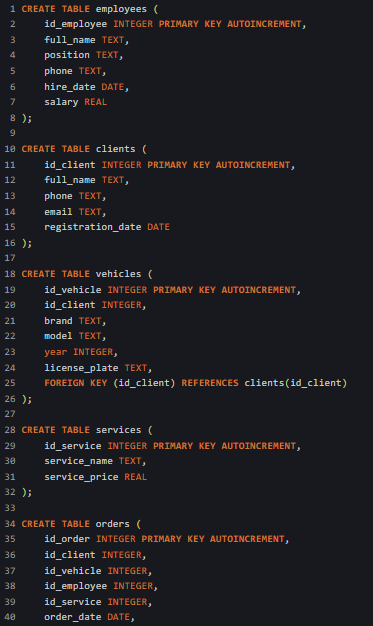
Вывод: в результате выполнения этого задания я приобрёл новые знания и навыки в области проектирования баз данных, узнал о важности нормализации для избежания избыточности данных и обеспечения их целостности. Это позволило мне понять, как эффективно организовать информацию в базах данных, уменьшая дублирования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

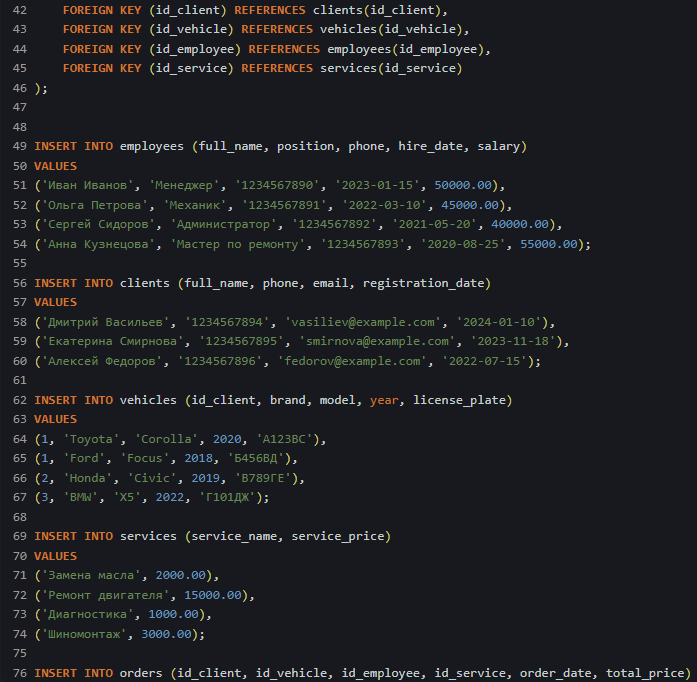
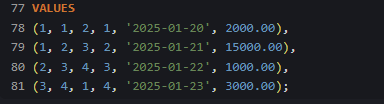
**3. Практическая работа №3**

# **3.1** **Практическое задание №3.**

Создать все таблицы и заполнить их данными (не менее 10 записей в каждой таблице, по возможности)

# **3.2 Выполнение практического задания №3**

1. Создание базы данных
2. Заполнение таблицы данными

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

1. Я научился создавать таблицы

2. Я научился заполнять таблицы данными

# **4. Практическая работа №4**

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

# **4.1 Практическое задание №4.**

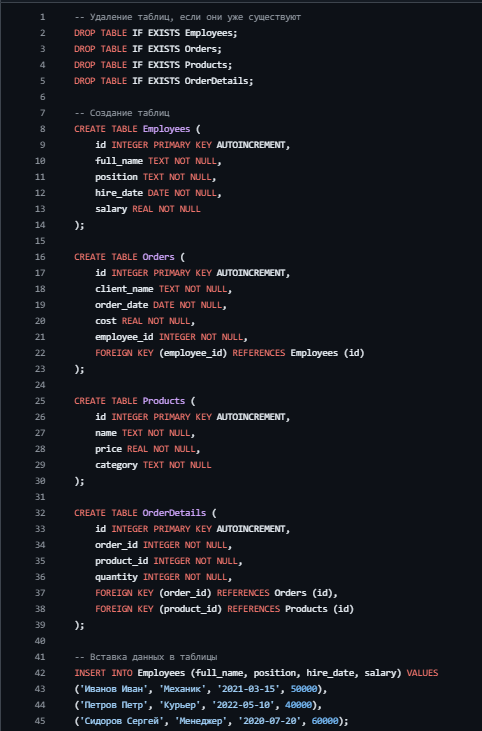
Самостоятельно изучить тему «Соединение таблиц».

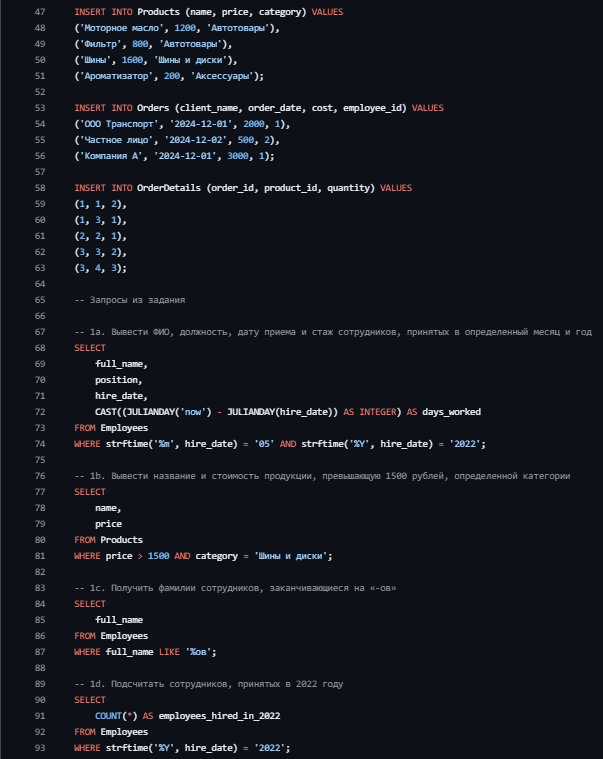
Написать не менее 5 запросов (!= 5), которые будут выводить данные из 2-3 таблиц (по возможности), используя разные виды соединения.

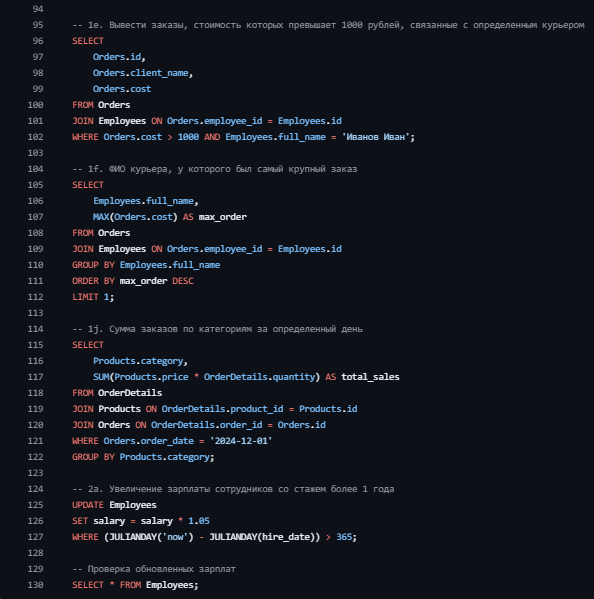
Обновить поле/я таблицы

Удалить записи в таблице

# **4.2 Выполнение практического задания №4**







* 1. **Заключение по выполнению практической работы №4**

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.

# **5. Практическая работа №5**

**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 какую-либо 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);

1. 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.

**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.

# **5.1 Практическое задание №5**

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.

Написать функцию, которая переводит число в число прописью, до млн.

Например, 1 – один, 15 – пятнадцать, 100 – сто, 1001 – тысяча один.

Написать процедуру заполнения таблицы случайными значениями,

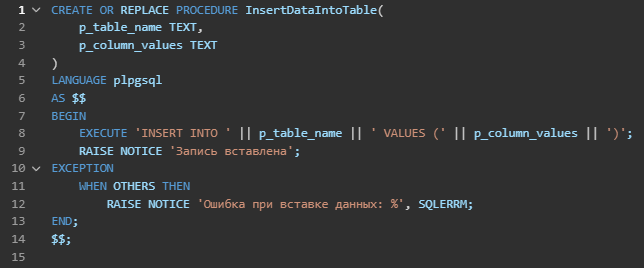
использовать пакет dbms\_random. Там есть вариант случайных символов

– такой вариант решения не подходит. Можно использовать либо

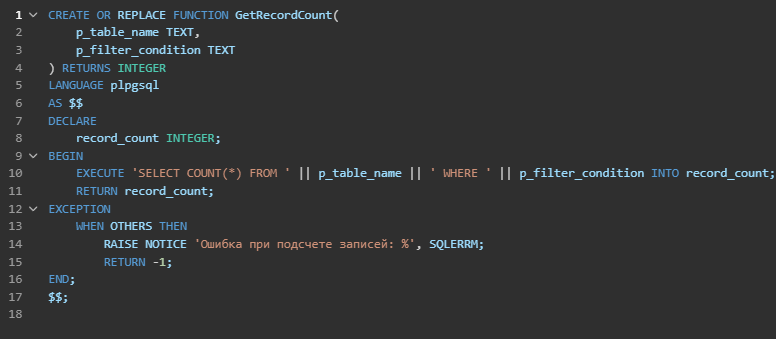
массивы, либо таблицы-«справочники»

# **Выполнение практического задания №5**

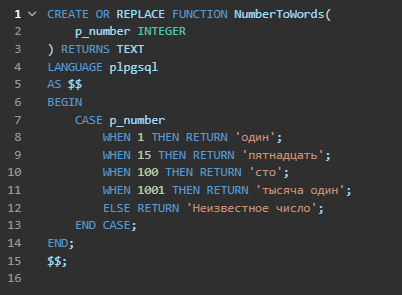
1. Написал процедуру вставки данных в таблицу



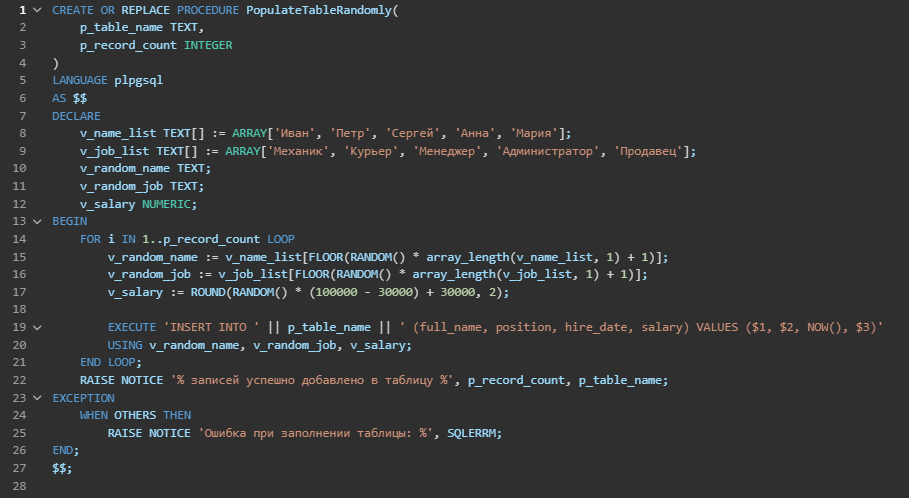
1. Написал функцию, которая возвращает количество записей из таблицы в зависимости от заданного фильтра



1. Написал функцию, которая переводит число в число прописью



1. Написал процедуру заполнения таблицы случайными значениями



**5.3 Заключение по выполнению практической работы №5**

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.