МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных предприятия «Лидское пиво»» с реализацией технологии шифрования и маскирования в БД»

Выполнил студент Викторович И.С.

(Ф.И.О.)

Руководитель работы Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[**Введение** 5](#_Toc153930165)

[**1 Постановка задачи** 6](#_Toc153930166)

[2 Проектирование базы данных 7](#_Toc153930167)

[2.1 Схема базы данных 7](#_Toc153930168)

[2.2 Таблицы базы данных 8](#_Toc153930169)

[2.3 Вывод по разделу 10](#_Toc153930170)

[3 Разработка объектов базы данных 11](#_Toc153930171)

[**3.1 Табличное пространство** 11](#_Toc153930172)

[3.1 Таблицы 11](#_Toc153930173)

[3.2 Процедуры 11](#_Toc153930174)

[3.3 Функции 12](#_Toc153930175)

[3.4 Вывод по разделу 12](#_Toc153930176)

[**4 Описание процедур экспорта и импорта** 13](#_Toc153930177)

[4.1 Описание процедуры экспорта 13](#_Toc153930178)

[4.2 Описание процедуры импорта 13](#_Toc153930179)

[4.3 Вывод по разделу 14](#_Toc153930180)

[**5 Тестирование производительности** 15](#_Toc153930181)

[5.1 Тестирование запросов заполнения таблиц 15](#_Toc153930182)

[5.2 Проверка технологии шифрования и маскирования 16](#_Toc153930183)

[5.3 Вывод по разделу 17](#_Toc153930184)

[**6 Описание и применение технологии в БД** 18](#_Toc153930185)

[6.1 Описание взаимодействия с технологией 18](#_Toc153930186)

[6.2 Вывод по разделу 21](#_Toc153930187)

[7 Руководство пользователя 22](#_Toc153930188)

[Вход в систему: Для доступа к базе данных используйте ваше уникальное имя пользователя и пароль. Проверьте, что ваши учетные данные корректны. 22](#_Toc153930189)

[**Заключение** 23](#_Toc153930190)

[**Список используемых источников** 24](#_Toc153930191)

[Приложение A Создание таблиц 25](#_Toc153930192)

[Приложение Б Создание функций 27](#_Toc153930193)

[Приложение С Процедуры 28](#_Toc153930194)

# **Введение**

Современные предприятия, в том числе и компания «Лидское пиво», сталкиваются с необходимостью эффективного управления своими ресурсами, оптимизации бизнес-процессов и обеспечения высокого уровня обслуживания клиентов. В этом контексте разработка базы данных, специализированной для учета товаров, заказов и складских операций, играет ключевую роль.

Целью данного проекта является создание надежной и функциональной базы данных, которая позволит предприятию «Лидское пиво» решать следующие задачи:

1. Учет товаров и складских операций: База данных будет хранить информацию о наличии товаров на складе, движении товаров (поступление, списание, перемещение) и текущем состоянии запасов. Это позволит компании точно контролировать свои ресурсы и избегать дублирования данных.
2. Регистрация заказов и обработка клиентских запросов: Система будет отслеживать заказы клиентов, обрабатывать их быстро и эффективно, а также предоставлять актуальную информацию о статусе заказа. Это поможет удовлетворить потребности клиентов, минимизировать время доставки и повысить уровень обслуживания.
3. Анализ данных и принятие обоснованных решений: База данных будет предоставлять аналитические данные о реализации продукции, популярности определенных товаров, спросе на них и других ключевых показателях. Это поможет руководству компании принимать обоснованные решения о закупках, управлении запасами и стратегии развития.
4. Безопасность данных: Важным аспектом разработки будет обеспечение безопасности данных. Применение технологий шифрования и маскирования поможет защитить конфиденциальные сведения о клиентах, поставщиках и других участниках бизнес-процессов.

В итоге, использование разработанной базы данных способствует повышению конкурентоспособности компании, удовлетворению потребностей клиентов и эффективному управлению ресурсами.

# **1 Постановка задачи**

В рамках данного проекта требуется спроектировать инфраструктуру базы данных предприятия «Лидское пиво». Для этого необходимо провести анализ требований и определить следующие элементы и их содержимое в нашей базе данных: таблицы и связи между ними, ограничения целостности, профили безопасности, пользователи, триггеры, хранимые процедуры, функции и индексы. Затем необходимо разработать эти объекты в базе данных, используя СУБД Oracle и написание SQL-скриптов.

Для заполнения таблиц данными предусмотрен импорт из JSON-файлов. Кроме того, будет использована технология Oracle шифрования и маскирования для хеширования паролей пользователей.

Проект был разработан в Oracle с использованием sqldeveloper, где реализован функционал и возможности, предоставляемые базой данных через процедуры и функции. Важными функциями приложения будут:

* управление заказами (добавление, удаление, изменение);
* определение ролей (администратор, пользователь);
* управление продуктами (добавление, удаление, изменение);
* анализ продукции (количество проданных товаров по периодам, популярные товары, общее количество продукции);

Реализация этих операций будет осуществляться с помощью хранимых процедур и функций.

Важным этапом проекта будет тестирование производительности базы данных на таблице, содержащей не менее 100 000 строк. В случае необходимости будут внесены изменения в структуру базы данных для оптимизации производительности.

Применение технологии шифрования и маскирования для защиты пользовательских данных в базе данных позволит проанализировать преимущества, которые предоставляет данная технология.

2 Проектирование базы данных

2.1 Схема базы данных

Для курсового проекта используется PDB с настроенными параметрами. Определены таблицы, необходимые для хранения информации, используемой реализации требуемого функционала.

В базе данных предприятия пивоваренного завода применяются таблицы для хранения информации о доступной продукции: сорта пива, кваса и других напитков, пивоваренных заводах, складах предприятия, закупках завода, пользовательских данных, а также пользовательских заказов. Диаграмма таблиц и их отношения представлены на рисунке 2.1.

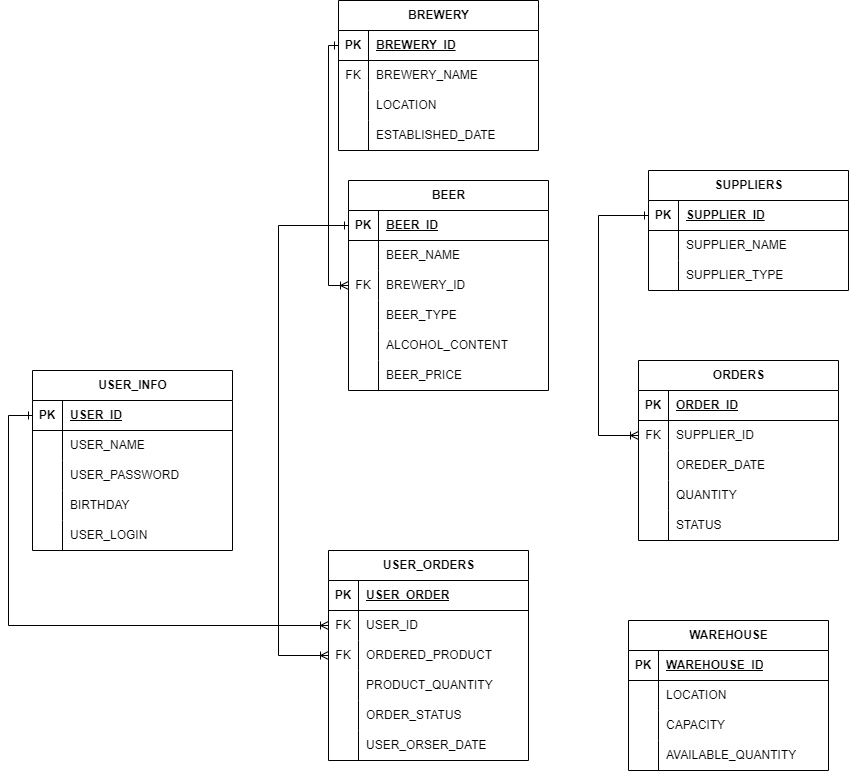


Рисунок 2.1 – Схема базы данных

Для эффективного использования системы важно определить роли пользователей и их сценарии использования. Сценарии описывают, как каждый пользователь будет взаимодействовать с системой в соответствии с их ролью. Это помогает определить доступные функции, доступные данные и организацию навигации в системе. Для визуализации взаимодействия между пользователями и системой используются диаграммы UML.

Роли пользователей включают Admin (администратор) и User (пользователь). Это означает, что администратор будет иметь расширенные возможности и полный доступ ко всем функциям системы, в то время как пользователь будет иметь ограниченный доступ только к определенным функциям, процедурам и данным, соответствующим его роли.

Взаимодействие с системой наглядно демонстрирует Use-case диаграмма – рисунок 2.2.

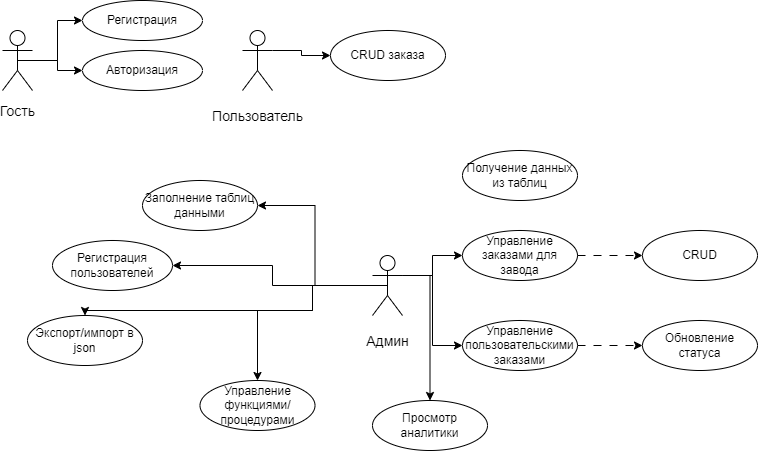


Рисунок 2.2 – Use Case диаграмма

В проекте реализован функционал для трёх групп пользователей – Администраторы, зарегистрированные пользователи и гости. Все действия данных пользователей будут выполняться через хранимые процедуры и функции.

2.2 Таблицы базы данных

Далее приведено описание таблиц базы данных: названия, названия столбцов, типы данных столбцов, описание содержания столбцов. Описание представлено в таблицах 2.1 – 2.10.

Таблица 2.1 – Содержание таблицы BREWERY (Пивоварни)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Brewery\_id | NUMBER | Уникальный идентификатор пивоварни |
| Brewery\_name | VARCHAR2(100) | Название пивоварни |
| Location | VARCHAR2(100) | Местонахождение пивоварни |
| Established\_Date | DATE | Дата основания |

Эта таблица хранит в себе данные пивоварнях, относящихся к придприятию.

Таблица 2.2 – Содержание таблицы BEER (Продукция)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Beer\_id | NUMBER | Уникальный идентификатор пива |
| Beer\_name | VARCHAR2(100) | Название пива/напитка |
| Brewery\_id | NUMBER | Уникальный идентификатор пивоварни |

Продолжение таблицы 2.2

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Beer\_type | VARCHAR2(50) | Тип пива/напитка |
| Alcohol\_content | NUMBER | Процент алкоголя в продукте |
| Beer\_price | NUMBER | Стоимость пива |

В этой таблице содержатся данные продукции, производимой на заводе «Лидское пиво».

Таблица 2.3 – Содержание таблицы Suppliers (Поставщики)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Supplier\_id | NUMBER | Уникальный идентификатор поставщика |
| Supplier\_name | VARCHAR2(100) | Название поставщика |
| Supplier\_type | VARCHAR2(50) | Тип поставщика(по сырью) |

Эта таблица хранит в себе данные о поставщиках.

Таблица 2.4 – Содержание таблицы Orders (Заказы предприятия)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Order\_id | NUMBER | Уникальный идентификатор заказа |
| Supplier\_id | NUMBER | Идентификатор поставщика |
| Order\_date | DATE | Дата заказа |
| Quantity | NUMBER | Количество единиц в заказе |
| Status | VARCHAR2(20) | Статус выполнения заказа |

В данной таблице содержатся данные закупках предприятия.

Таблица 2.5 – Содержание таблицы Warehouse (склады)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Warehouse\_ID | NUMBER | Уникальный идентификатор склада |
| Location | VARCHAR2(100) | Месторасположение склада |
| Capacity | NUMBER | Общая вместимость склада |
| Avaiable\_capacity | NUMBER | Доступная вместимость склада |

В данной таблице содержится информация о складах завода.

Таблица 2.6 – Содержание таблицы User\_info (Пользовательская информация)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| User\_id | NUMBER | Уникальный идентификатор пользователя |
| User\_name | VARCHAR2(40) | Имя пользователя |
| User\_password | VARCHAR2(200) | Хешированный пароль пользователя |
| Birthday | DATE | Дата рождения пользователя |
| User\_login | VARCHAR2(20) | Логин пользователя |

В данной таблице содержится регистрационная информация о пользователе.

Таблица 2.7 – Содержание таблицы User\_orders (Пользовательские заказы)

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| Users\_order\_id | NUMBER | Уникальный идентификатор заказа |

Продолжение таблицы 2.7

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| User\_id | NUMBER | Идентификатор пользователя |
| Ordered\_product | NUMBER | Наименования продукта в заказе |
| Product\_quantity | NUMBER | Количество продуктов в заказе |
| Order\_status | VARCHAR2(20) | Статус заказа |
| Order\_date | DATE | Дата создания заказа |

В данной таблице содержится информация о заказах пользователей.

2.3 Вывод по разделу

В данном разделе создана подключаемая база данных, спроектирована структура таблиц базы данных, реализованы таблицы, столбцы и типы данных для них, определены сценария взаимодействия пользователей базы данных завода с предоставляемыми данными, определены связи между таблицами, а также логика взаимодействия с ними.

3 Разработка объектов базы данных

**3.1 Табличное пространство**

Для каждой таблицы и пользователя создаётся табличное пространство. Листинг показывает пример создания такого табличного пространства для таблицы Beer – листинг 3.1

CREATE TABLESPACE beer\_space

DATAFILE 'beer\_tspace.dbf' SIZE 100M  
AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED

EXTENT MANAGEMENT LOCAL;

Листинг 3.1 – Создание табличного пространства

После создания пользователей необходимо сделать это табличное пространство для них табличным пространством по умолчанию и выделить квоту.

3.1 Таблицы

Можно отметить, что в каждой таблице есть поле, представляющее ID, которое является первичным ключом и заполняется с помощью последовательности с шагом 1 либо присвоением уникального идентификатора при заполнении таблицы данными. Числовые типы данных представлены типами NUMBER. Используется тип DATE для хранения даты, например, даты оформления заказа, даты рождения. Символьные данные представлены типами varchar2(n), где n – максимальная длина последовательности символов.

После создания таблицы заполняются некоторым количеством тестовых данных для проверки работоспособности и разработки объектов следующих пунктов.

В данном проекте базы данных содержится семь таблиц, каждая из которых описана в разделе 2. Соответствующий SQL-код для создания этих таблиц приведен в приложении A.

3.2 Процедуры

В данном проекте широко представлено применение процедур. С помощью процедур реализованы такие операции, как добавление, удаление и обновление данных, регистрация пользователей, исполнение технологии шифрования и маскирования в базе данных, аналитика продукции.

Администратор имеет привилегии dba, следовательно, имеет право выполнить практически любую процедуру. Для пользователей администратором специально выдаются права на исполнение конкретных процедур, определённых их функционалом.

В приложении C представлен листинг SQL-кода для создания процедур, использующихся в курсовом проекте.

3.3 Функции

В данном курсовом проекте функции применены при аутентификации пользователей. Это обусловлено тем, что эта функция многократно вызывается в различных процедурах для выполнения определённых действий.

В приложении B представлен листинг SQL-кода для создания функций, осуществляющих различные операции.

* 1. Вывод по разделу

В данном разделе было описано и выполнено создание объектов для базы данных предприятия, таких как табличные пространства, роли и пользователи, таблицы, хранимые процедуры, функции и триггеры. Определено взаимодействие представленных объектов базы данных, добавлены обработчики исключений, выданы привилегии на вызов и использование пользователями процедур, функций и других объектов.

# **4 Описание процедур экспорта и импорта**

4.1 Описание процедуры экспорта

Для хранения данных используется JSON-формат. Процедура экспорта в JSON-формат для продукции завода – листинг 4.1.

|  |
| --- |
| CREATE OR REPLACE DIRECTORY UTL\_DIR AS '/home/oracle/JSON'; GRANT READ, WRITE ON DIRECTORY UTL\_DIR TO public; CREATE OR REPLACE DIRECTORY UTL\_DIR AS 'C:\WINDOWS.X64\_193000\_db\_home\database';  CREATE OR REPLACE PROCEDURE ExportToJSON IS  v\_file UTL\_FILE.FILE\_TYPE;  v\_row BEER%ROWTYPE; BEGIN  v\_file := UTL\_FILE.FOPEN('UTL\_DIR', 'FACTORY\_PRODUCTS.json', 'W'); UTL\_FILE.PUT\_LINE(v\_file, '[');  FOR v\_row IN (SELECT JSON\_OBJECT(  'beer\_id' is CAST(BEER\_ID as NUMBER),  'beer\_name' is CAST(BEER\_NAME as VARCHAR(100)),  'brewery\_id' is CAST(BREWERY\_ID as NUMBER),  'beer\_type' is CAST(BEER\_TYPE as VARCHAR(50)),  'alcohol\_content' is CAST(ALCOHOL\_CONTENT as NUMBER),  'beer\_price' is CAST(BEER\_PRICE as NUMBER) ) AS JSON\_DATA FROM BEER) LOOP  UTL\_FILE.PUT\_LINE(v\_file, v\_row.JSON\_DATA || ','); END LOOP;   UTL\_FILE.PUT\_LINE(v\_file, ']');  UTL\_FILE.FCLOSE(v\_file); EXCEPTION WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM); RAISE; END; |

Листинг 4.1 – Процедура для экспорта данных из таблицы Beer

Процедура выполняет экспорт данных из таблицы Beer в формат JSON. Результат экспорта данных сохраняется в файл 'FACTORY\_PRODUCTS.json'. В данной процедуре применяется UTL FILE для работы с файлами, создания JSON-объектов для каждой строки таблицы, которые записываются в виде JSON-массива.

4.2 Описание процедуры импорта

Реализована процедура для импорта данных из формата JSON. Данная процедура представлена в листинге 4.2.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ImportFromJSON IS BEGIN FOR json\_rec IN (  SELECT beer\_id, beer\_name, brewery\_id, beer\_type, alcohol\_content, beer\_price FROM JSON\_TABLE(BFILENAME('UTL\_DIR', 'FACTORY\_PRODUCTS.json'), '$[\*]' COLUMNS (  beer\_id number PATH '[$.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2F%24.beer)\_id',  beer\_name varchar2(100) PATH '[$.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2F%24.beer)\_name',  brewery\_id number PATH '$.brewery\_id',  beer\_type varchar2(50) PATH '[$.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2F%24.beer)\_type',  alcohol\_content number PATH '$.alcohol\_content',  beer\_price number PATH '[$.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2F%24.beer)\_price' )) ) LOOP BEGIN  INSERT INTO JSONBEER (beer\_id, beer\_name, brewery\_id, beer\_type, alcohol\_content, beer\_price )  VALUES ([json\_rec.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fjson_rec.beer)\_id, [json\_rec.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fjson_rec.beer)\_name, json\_rec.brewery\_id, [json\_rec.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fjson_rec.beer)\_type, json\_rec.alcohol\_content, [json\_rec.beer](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fjson_rec.beer)\_price);  DBMS\_OUTPUT.PUT\_LINE('Product was added successfully'); EXCEPTION WHEN DUP\_VAL\_ON\_INDEX THEN ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Product with the id already exists.'); WHEN OTHERS THEN ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error inserting user: ' || SQLERRM); RAISE; END; END LOOP; END; |

Листинг 4.2 – Процедура для импорта данных в таблицу JSONBEER

Процедура ImportFromJSON выполняет импорт данных из JSON-формата в специально созданную таблицу JSONBEER для демонстрации корректной работы процедуры. Реализован механизм обработки исключений. В результате импорта мы получим заполненную, ранее экспортированными, данными таблицу JSONBEER.

4.3 Вывод по разделу

В этом разделе были рассмотрены процедуры, выполняющие экспорт/импорт данных в/из формата JSON. Данный механизм позволяет расширить функциональные возможности базы данных, позволяет оптимизировать ввод данных.

# **5 Тестирование производительности**

5.1 Тестирование запросов заполнения таблиц

В базе данных реализованы различные процедуры, функции, а также таблицы заполнены необходимым количеством данных – листинг 5.1. Теперь необходимо протестировать, что всё работает корректно.

|  |
| --- |
| DECLARE  v\_supplier\_id NUMBER;  v\_order\_date DATE;  v\_quantity NUMBER;  v\_status VARCHAR2(20); BEGIN FOR i IN 1..100000 LOOP  v\_supplier\_id := TRUNC(DBMS\_RANDOM.VALUE(1, 20));   v\_order\_date := SYSDATE - TRUNC(DBMS\_RANDOM.VALUE(1, 365)); v\_quantity := TRUNC(DBMS\_RANDOM.VALUE(1, 1000));   v\_status := CASE WHEN i < 30000 THEN 'is complited'  WHEN i < 60000 THEN 'is compliting'  ELSE 'rejected' END; AddOrder(v\_supplier\_id, TO\_CHAR(v\_order\_date, 'DD.MM.YYYY'), v\_quantity, v\_status); END LOOP; END; |

Листинг 5.1 – Заполнение таблицы Orders

При успешном выполнении скрипта, представленного на листинге 5.1, таблица Orders заполняется 100000 строк. Убедится в этом можно, выполнив запрос SELECT count(\*) FROM ORDERS; Результат выполнения запроса представлен на рисунке 5.1

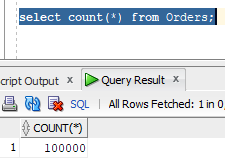


Рисунок 5.1 – Результат выполнения запроса

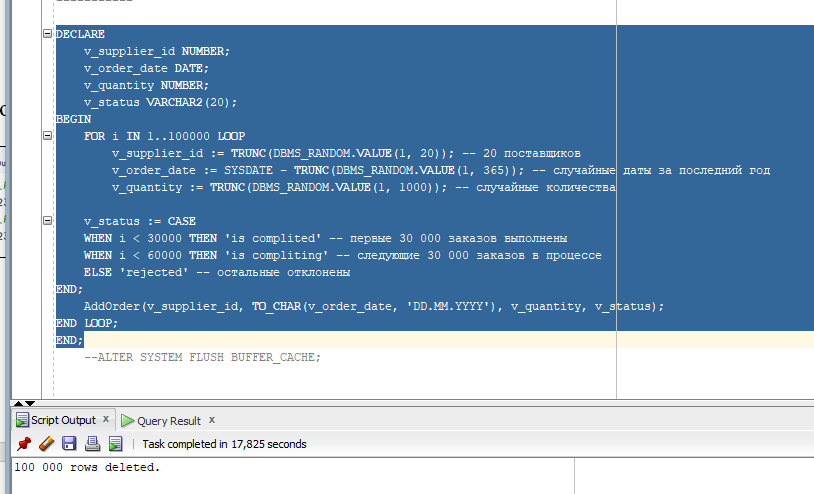


Рисунок 5.2 – Время поиска пользователя

Из рисунка 5.2 можно заметить, что время выполнения скрипта составило 17,825 секунд. На время выполнения запроса могут влиять многие факторы, такие как количество добавляемых данных или загруженность системы.

5.2 Проверка технологии шифрования и маскирования

Одной из задач курсового проекта была реализация технологии шифрования и маскирования. Следует выполнить тестирование процедур, реализующих эту технологию, для проверки корректной работы.

Листинги процедур, реализующих технологию находятся в приложении Г.

В данном курсовом проекте шифрование используется для скрытия паролей для пользователей базы данных методом хеширования паролей.

Для тестирования выполним нового пользователя, а затем выполним select-запрос к таблице User\_info, где, согласно коду процедуры, вместо истинного пароля должен находятся его хеш.

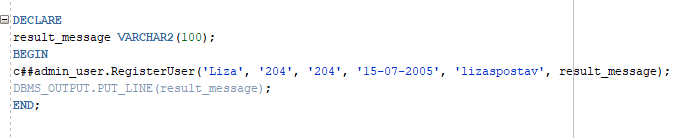


Рисунок 5.3 – Вызов процедуры регистрации

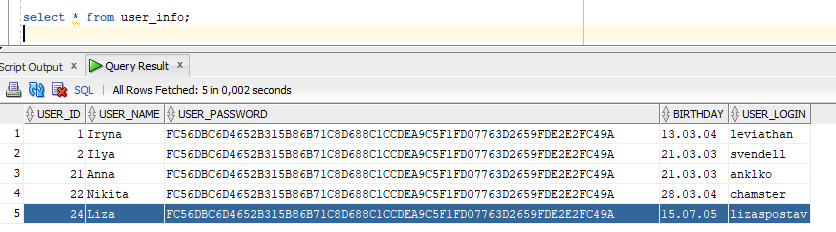


Рисунок 5.4 – Запрос к таблице User\_info

Из результирующего набора мы видим, что шифрование данных выполняется корректно, истинные пароли скрываются, вместо него – хеш.

5.3 Вывод по разделу

Выполнено тестирование заполнение таблицы Orders 100000 строками данных, проверено время выполнения, рассмотрены причины полученных результатов. Протестированы процедуры, отвечающие за реализацию технологий шифрования и маскирования.

# **6 Описание и применение технологии в БД**

6.1 Описание взаимодействия с технологией

Технология маскирования данных в базе данных - это метод обеспечения безопасности информации путем скрытия или искажения конфиденциальных данных от пользователей, которым нет необходимости видеть их полностью.

В курсовом проекте данная технология реализована в виде представления, выполняющего select-запрос. В перечислении столбцов таблицы после ключевого слова select некоторые данные, требующие маскирования, реализовывались через case, где, согласно реализации, для администратора данные остаются в изначальном виде, а для остальных пользователей происходит замена данных конкретного столбца на условные символы.

Убедится в корректной работе технологии маскирования можно выполнив select-запрос, вызывающий представление Masked\_User\_info.

|  |
| --- |
| CREATE OR REPLACE VIEW Masked\_User\_info AS SELECT user\_id, CASE WHEN SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') = 'C#[#ADMIN\_USER](https://vk.com/im?sel=295252893&st=%23ADMIN_USER)' THEN user\_name ELSE 'xxxxxxx' END AS user\_name, CASE WHEN SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA') = 'C#[#ADMIN\_USER](https://vk.com/im?sel=295252893&st=%23ADMIN_USER)' THEN user\_password ELSE '\*\*\*\*\*\*\*\*' END AS user\_password, birthday, user\_login FROM User\_info; |

Листинг 6.1 – Представление с реализацией технологии маскирования

|  |
| --- |
|  |

Рисунок 6.1 – Вызов представления администратором

|  |
| --- |
|  |

Рисунок 6.2 – Вызов представления пользователем

Один из методов шифрования данных в Oracle – хеширование. В Oracle используют функции хеширования, такие как *DBMS\_CRYPTO.HASH*.

В данном курсовом проекте шифрование используется при регистрации пользователей для хеширования сохранённых паролей.

В процедуре, реализующей регистрацию пользователей, применяются функция хеширования, описанная выше. В таблицу записывается хеш. В нашей работе функция *DBMS\_CRYPTO.HASH* принимает в качестве входных данных байтовое представление пароля и использует алгоритм SHA-256 для генерации хеша.

Данная операция является необратимой, то есть нельзя преобразовать хеш к паролю. Процесс проверки корректности следующий:

1. пользователь вводит свой пароль для аутентификации;
2. введённый пароль хешируется с помощью того же алгоритма, что и при сохранении в базе данных;
3. сгенерированный хеш ввода сравнивается с хешем пароля, сохранённым в базе данных;
4. если хеши совпадают, то совпадают и пароли.

6.2 Вывод по разделу

Технологии шифрования и маскирования данных - это важные методы обеспечения безопасности и конфиденциальности информации в базах данных.

Шифрование данных позволяет преобразовывать информацию в зашифрованный формат, который невозможно прочитать без ключа шифрования. Это способ защиты информации в базе данных от несанкционированного доступа к конфиденциальным данным, даже если злоумышленники получили физический доступ к базе данных.

Маскирование данных, с другой стороны, позволяет скрыть чувствительные данные для предотвращения их отображения или раскрытия пользователям или приложениям, которые не имеют соответствующих прав доступа. Это полезный метод для обеспечения конфиденциальности данных в различных сценариях, например, при отображении частичной информации для пользователей, которые не имеют привилегий для полного доступа к данным.

Оба подхода эффективны в защите информации в базах данных, помогая предотвратить утечку конфиденциальной информации и обеспечивая соблюдение требований безопасности и законодательства о защите данных. Однако важно правильно выбирать и реализовывать эти методы, учитывая специфику и требования вашего проекта или организации.

7 Руководство пользователя

Вход в систему: Для доступа к базе данных используйте ваше уникальное имя пользователя и пароль. Проверьте, что ваши учетные данные корректны.

Функции базы данных:

1. Управление продукцией:
   * Добавление новых продуктов: Позволяет добавлять новые виды пива в систему, указывая их характеристики и цены.
   * Изменение данных о продукции: Редактирование информации о продукции (цена, объемы производства и т.д.).
   * Управление складом: Отслеживание количества продукции на складе.
2. Заказы:
   * Размещение заказов: Возможность размещения заказов на поставку продукции.
   * Отслеживание статуса заказов: Мониторинг текущего состояния размещенных заказов.
3. Аналитика:
   * Отчеты и анализ данных: Генерация отчетов о продажах, объемах производства и других аналитических данных.
   * Статистика производства: Просмотр статистики производства для оптимизации работы завода.
4. Управление пользователями:
   * Добавление и удаление пользователей: Управление учетными записями пользователей с различными уровнями доступа.

# **Заключение**

В ходе выполнения курсового проекта была спроектирована и реализована инфраструктура базы данных для предприятия «Лидское пиво», которая содержит информацию о пользователях, продуктах и заказах, с использованием СУБД Oracle 12c. Были определены и разработаны необходимые объекты базы данных, такие как табличные пространства, таблицы, ограничения целостности, пользователи, триггеры, хранимые процедуры. Также был проведен импорт данных из JSON файлов для заполнения таблиц.

Внимание было уделено производительности базы данных. Проведено тестирование на таблице, содержащей более 100 000 строк, и проведён анализ необходимых изменений в структуре базы данных для обеспечения оптимальной производительности.

В результате успешной реализации курсового проекта была создана база данных предприятия «Лидское пиво» с широким функционалом, позволяющим эффективно управлять продуктами, заказами и пользователями. Проект демонстрирует преимущества использования СУБД Oracle 12c для создания мощной и гибкой инфраструктуры базы данных.

# **Список используемых источников**

1. Нистюк О. А. Курс лекций по базам данных, [Электронный ресурс]. – Режим доступа: https://diskstation.belstu.by:5001/. – Дата доступа: 01.10.2023.
2. Работа с файлами в Oracle, [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/cd/F49540\_01/DOC/server.815/a68001/utl\_file.htm. – Дата доступа: 01.10.2023.
3. Конфиденциальность и функции защиты данных, [Электронный ресурс]. – Режим доступа: <https://docs.oracle.com/cloud/latest/related-docs/OMCEZ/ru/EncryptionAtRest.htm>. – Дата доступа: 01.11.2023

Приложение A Создание таблиц

|  |
| --- |
| CREATE TABLE Brewery (      Brewery\_ID NUMBER PRIMARY KEY,      Brewery\_Name VARCHAR2(100),      Location VARCHAR2(100),      Established\_Date DATE  )  TABLESPACE brewery\_space;  CREATE TABLE Beer (      Beer\_ID NUMBER PRIMARY KEY,      Beer\_Name VARCHAR2(100),      Brewery\_ID NUMBER REFERENCES Brewery(Brewery\_ID),      Beer\_Type VARCHAR2(50),      Alcohol\_Content NUMBER,      Beer\_price NUMBER  )  TABLESPACE beer\_space;  CREATE TABLE Suppliers (      Supplier\_ID NUMBER PRIMARY KEY,      Supplier\_Name VARCHAR2(100),      Supplier\_Type VARCHAR2(50)  )  TABLESPACE supplier\_space;  CREATE TABLE Orders (      Order\_ID NUMBER PRIMARY KEY,      Supplier\_ID NUMBER REFERENCES Suppliers(Supplier\_ID),      Order\_date DATE,      Quantity NUMBER,      Status VARCHAR2(20),      CONSTRAINT check\_status check (Status in ('is complited', 'is compliting', 'rejected'))  ) TABLESPACE orders\_space;  CREATE TABLE Warehouse (      Warehouse\_ID NUMBER PRIMARY KEY,      Location VARCHAR2(100),      Capacity NUMBER,      Available\_quantity NUMBER  ) TABLESPACE warehouse\_space;  create table User\_info (      user\_id NUMBER PRIMARY KEY,      user\_name varchar2(40),      user\_password varchar(200),      birthday DATE,      user\_login varchar(20)  ) TABLESPACE user\_space;  create table User\_orders (      users\_order\_id NUMBER PRIMARY KEY,      user\_id NUMBER REFERENCES User\_info(user\_id),      ordered\_product NUMBER REFERENCES Beer(Beer\_ID),      product\_quantity NUMBER  ) TABLESPACE user\_orders;  ALTER TABLE User\_orders  ADD order\_status VARCHAR2(20) DEFAULT 'active' check (order\_status in ('completed', 'active', 'rejected'));  alter table User\_orders  add user\_order\_date DATE; |

Приложение Б Создание функций

|  |
| --- |
| CREATE OR REPLACE FUNCTION AuthenticateUser(      p\_user\_login IN VARCHAR2,      p\_user\_password IN VARCHAR2  ) RETURN BOOLEAN  IS  hashed\_password RAW(256);  stored\_password RAW(256);  BEGIN  SELECT user\_password INTO stored\_password FROM User\_info WHERE user\_login = p\_user\_login;  hashed\_password := DBMS\_CRYPTO.HASH(UTL\_RAW.CAST\_TO\_RAW(p\_user\_password), DBMS\_CRYPTO.HASH\_SH256);  DBMS\_OUTPUT.PUT\_LINE('Comparing stored password: ' || stored\_password || ' with hashed password: ' || hashed\_password);  IF hashed\_password = stored\_password THEN  DBMS\_OUTPUT.PUT\_LINE('User authenticated successfully for login: ' || p\_user\_login);  RETURN TRUE;  ELSE  DBMS\_OUTPUT.PUT\_LINE('Authentication failed for login: ' || p\_user\_login);  RETURN FALSE;  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('User not found for login: ' || p\_user\_login);  RETURN FALSE;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred during authentication for login: ' || p\_user\_login);  RETURN FALSE;  END; |

Приложение С Процедуры

|  |
| --- |
| CREATE OR REPLACE PROCEDURE AddOrder (      Supplier\_ID\_in NUMBER,      Order\_date\_in DATE,      Quantity\_in NUMBER,      Status\_in VARCHAR2  )  IS      v\_order\_id NUMBER;  BEGIN      SELECT order\_seq.NEXTVAL INTO v\_order\_id FROM dual;      INSERT INTO Orders (Order\_ID, Supplier\_ID, Order\_date, Quantity, Status)      VALUES (v\_order\_id, Supplier\_ID\_in, Order\_date\_in, Quantity\_in, Status\_in);  COMMIT;  EXCEPTION  WHEN OTHERS THEN      DBMS\_OUTPUT.PUT\_LINE('Ошибка при вставке данных: ' || SQLERRM);  ROLLBACK;  END AddOrder;    CREATE OR REPLACE PROCEDURE DeleteOrder (      v\_order\_id IN NUMBER  )  IS      v\_deleted\_count NUMBER;  BEGIN      DELETE FROM Orders WHERE Order\_ID = v\_order\_id RETURNING COUNT(\*) INTO v\_deleted\_count;  IF v\_deleted\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'No order found with this ID');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Order was deleted successfully');  COMMIT;  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLCODE || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE UpdateOrder (  u\_order\_id IN NUMBER,  new\_supplier\_id IN NUMBER DEFAULT NULL,  new\_order\_date IN DATE DEFAULT NULL,  new\_quantity IN NUMBER DEFAULT NULL,  new\_status IN VARCHAR2 DEFAULT NULL  )  IS  BEGIN  DECLARE  order\_count NUMBER;  BEGIN  SELECT COUNT(\*) INTO order\_count  FROM Orders  WHERE Order\_ID = u\_order\_id;  IF order\_count = 0 THEN  raise\_application\_error(-20001, 'Order does not exist');  END IF;  IF new\_supplier\_id IS NOT NULL THEN  UPDATE Orders  SET Supplier\_ID = new\_supplier\_id  WHERE Order\_ID = u\_order\_id;  DBMS\_OUTPUT.PUT\_LINE('Supplier was updated. New supplier is: ' || new\_supplier\_id);  END IF;  IF new\_order\_date IS NOT NULL THEN  UPDATE Orders  SET Order\_date = new\_order\_date  WHERE Order\_ID = u\_order\_id;  DBMS\_OUTPUT.PUT\_LINE('Date was updated. New date is: ' || new\_order\_date);  END IF;  IF new\_quantity IS NOT NULL THEN  UPDATE Orders  SET Quantity = new\_quantity  WHERE Order\_ID = u\_order\_id;  DBMS\_OUTPUT.PUT\_LINE('Quantity was updated. New quantity is: ' || new\_quantity);  END IF;  IF new\_status IS NOT NULL THEN  UPDATE Orders  SET Status = new\_status  WHERE Order\_ID = u\_order\_id;  DBMS\_OUTPUT.PUT\_LINE('Status was updated. New status is: ' || new\_status);  END IF;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Order updated successfully');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLCODE || SQLERRM);  END;  END;  CREATE OR REPLACE PROCEDURE AddProduct (      p\_user\_login IN VARCHAR2,      p\_user\_password IN VARCHAR2,      p\_beer\_name IN VARCHAR2,      p\_product\_quantity IN NUMBER,      p\_order\_date IN DATE  )  IS      v\_user\_id NUMBER;      v\_beer\_id NUMBER;      v\_order\_id NUMBER := user\_orders\_seq.NEXTVAL;  BEGIN  IF AuthenticateUser(p\_user\_login, p\_user\_password) THEN  SELECT user\_id INTO v\_user\_id FROM User\_info WHERE user\_login = p\_user\_login;  SELECT beer\_id INTO v\_beer\_id  FROM (  SELECT beer\_id  FROM Beer  WHERE Beer\_Name = p\_beer\_name  ORDER BY DBMS\_RANDOM.VALUE  )  WHERE ROWNUM = 1;  INSERT INTO User\_orders (users\_order\_id, user\_id, ordered\_product, product\_quantity, order\_status, user\_order\_date)  VALUES (v\_order\_id, v\_user\_id, v\_beer\_id, p\_product\_quantity, 'active', p\_order\_date);  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Order added successfully!' || 'Your order id is ' || v\_order\_id);  ELSE  DBMS\_OUTPUT.PUT\_LINE('Authentication failed.');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('User or product not found.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred.');  END;  CREATE OR REPLACE PROCEDURE DeleteProduct (      p\_user\_login IN VARCHAR2,      p\_user\_password IN VARCHAR2,      p\_order\_id IN NUMBER  )  IS  v\_user\_id NUMBER;  v\_order\_status VARCHAR2(20);  BEGIN  IF AuthenticateUser(p\_user\_login, p\_user\_password) THEN  SELECT user\_id INTO v\_user\_id FROM User\_info WHERE user\_login = p\_user\_login;  SELECT order\_status INTO v\_order\_status  FROM User\_orders  WHERE users\_order\_id = p\_order\_id  AND (user\_id = v\_user\_id) -- 1 is the admin user\_id  FOR UPDATE; -- Lock the row for update  IF v\_order\_status = 'active' THEN  DELETE FROM User\_orders WHERE users\_order\_id = p\_order\_id;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Order deleted successfully!');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Order cannot be deleted. Status is not active.');  END IF;  ELSE  DBMS\_OUTPUT.PUT\_LINE('Authentication failed.');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Order or user not found.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred.');  END;  CREATE OR REPLACE PROCEDURE UpdateProduct (      p\_user\_login IN VARCHAR2,      p\_user\_password IN VARCHAR2,      p\_order\_id IN NUMBER,      p\_product\_quantity IN NUMBER,      p\_ordered\_product\_name IN VARCHAR2  )  IS      v\_order\_owner\_id NUMBER;      v\_user\_id NUMBER;      v\_ordered\_product\_id NUMBER;  BEGIN  IF AuthenticateUser(p\_user\_login, p\_user\_password) THEN  SELECT user\_id INTO v\_user\_id FROM User\_info WHERE user\_login = p\_user\_login;  SELECT user\_id INTO v\_order\_owner\_id  FROM User\_orders  WHERE users\_order\_id = p\_order\_id  AND user\_id = v\_user\_id;  IF v\_order\_owner\_id IS NOT NULL THEN  SELECT Beer\_ID INTO v\_ordered\_product\_id  FROM Beer  WHERE Beer\_Name = p\_ordered\_product\_name  AND ROWNUM = 1;  UPDATE User\_orders  SET product\_quantity = p\_product\_quantity,  ordered\_product = v\_ordered\_product\_id  WHERE users\_order\_id = p\_order\_id  AND order\_status = 'active';  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('No rows updated. The order might be inactive or not owned by the user.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Order updated successfully!');  END IF;  ELSE  DBMS\_OUTPUT.PUT\_LINE('The order does not exist or is not owned by the user.');  END IF;  ELSE  DBMS\_OUTPUT.PUT\_LINE('Authentication failed.');  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('No order found for the user.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE ProductAnalysis (      start\_date IN DATE,      end\_date IN DATE  )  IS      v\_total\_quantity NUMBER;  BEGIN      IF end\_date < start\_date THEN          DBMS\_OUTPUT.PUT\_LINE('Ошибка: Некорректные даты');      RETURN;  END IF;  SELECT SUM(product\_quantity) INTO v\_total\_quantity  FROM User\_orders  WHERE user\_order\_date BETWEEN start\_date AND end\_date;  DBMS\_OUTPUT.PUT\_LINE('Общее количество продукции за период с ' || start\_date || ' по ' || end\_date || ': ' || v\_total\_quantity || ' штук');  DBMS\_OUTPUT.PUT\_LINE('Анализ продукции за период с ' || start\_date || ' по ' || end\_date || ':');  FOR product IN (  SELECT ordered\_product, SUM(product\_quantity) AS total\_quantity  FROM User\_orders  WHERE user\_order\_date BETWEEN start\_date AND end\_date  GROUP BY ordered\_product  )  LOOP  DBMS\_OUTPUT.PUT\_LINE('Продукт ' || product.ordered\_product || ': ' || product.total\_quantity || ' штук');  END LOOP;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Данные за указанный период отсутствуют');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);  END;  CREATE OR REPLACE PROCEDURE RegisterUser(      p\_user\_name IN VARCHAR2,      p\_user\_password IN VARCHAR2,      p\_repeat\_password IN VARCHAR2,      p\_birthday IN DATE,      p\_user\_login IN VARCHAR2,      p\_result OUT VARCHAR2  )  IS  login\_exists NUMBER;  hashed\_password RAW(256); --SHA-256  BEGIN  IF p\_user\_password != p\_repeat\_password THEN  p\_result := 'Passwords do not match.';  ELSE  SELECT COUNT(\*) INTO login\_exists FROM User\_info WHERE user\_login = p\_user\_login;  IF login\_exists > 0 THEN  p\_result := 'User login already exists.';  ELSE  hashed\_password := DBMS\_CRYPTO.HASH(UTL\_RAW.CAST\_TO\_RAW(p\_user\_password), DBMS\_CRYPTO.HASH\_SH256);  INSERT INTO User\_info (user\_id, user\_name, user\_password, birthday, user\_login)  VALUES (user\_info\_seq.NEXTVAL, p\_user\_name, hashed\_password, TO\_DATE(p\_birthday, 'DD-MM-YYYY'), p\_user\_login);  IF SQL%ROWCOUNT > 0 THEN  p\_result := 'User registered successfully!';  ELSE  p\_result := 'User registration failed.';  END IF;  END IF;  END IF;  EXCEPTION  WHEN OTHERS THEN  p\_result := 'An error occurred: ' || SQLERRM;  END; |