МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ОТЧЁТ**

**к проекту**

**по дисциплине** "Методы сбора, хранения, обработки и анализа данных"

**Исполнитель**

студент(а) 3 курса 1 группы А. Н. Халалеенко

подпись, дата

**Руководитель**

ассистент  А.В. Харланович

должность, учен. степень, ученое звание подпись, дата

Минск 2023

**Введение**

Современные организации, независимо от своего масштаба и направления деятельности, всегда стремятся к повышению эффективности и качества своих бизнес-процессов. Одним из ключевых аспектов достижения этой цели является обеспечение эффективной работы с клиентами и внутренними запросами. В этом контексте сервис-деск (Service Desk) играет важную роль, предоставляя комплексный инструмент для управления запросами, инцидентами, проблемами и задачами.

Данный отчет посвящен разработке и проектированию базы данных для сервис-деска. Сервис-деск представляет собой информационную систему, предназначенную для обработки и управления запросами пользователей, обеспечивая контроль, отслеживание и решение разнообразных задач. Важной частью такой системы является хранение и обработка данных, что делает проектирование и оптимизацию базы данных ключевой задачей.

В данном отчете мы представим детальное описание базы данных для сервис-деска, включая структуру таблиц, описания атрибутов, ключевые индексы, последовательности, функции, процедуры и представления. Мы также рассмотрим логическую схему базы данных, зависимости между таблицами и денормализованные атрибуты, если применимо. Отчет будет включать информацию о безопасности данных и предполагаемые сроки выполнения проекта.

Цель данного проекта - создать эффективную и надежную базу данных, которая поддержит работу сервис-деска и обеспечит высокий уровень обслуживания клиентов и пользователей.

## **Список типичных бизнес-задач:**

* Управление запросами и инцидентами: Регистрация, мониторинг и управление запросами и инцидентами, связанными с информационной технологией или другими аспектами деятельности компании.
* Решение проблем и поддержка пользователей: Предоставление технической поддержки и решение проблем пользователей, связанных с компьютерами, программным обеспечением и другой IT-инфраструктурой.
* Мониторинг сервисных уровней: Отслеживание и управление уровнями обслуживания, включая время реакции на запросы и инциденты.
* Управление базой знаний: Создание и обновление базы данных с решениями для часто возникающих проблем и запросов.
* Аналитика и отчетность: Сбор и анализ данных о производительности сервис-деска для улучшения качества обслуживания.

## **Список пользователей проекта:**

* Администраторы сервис-деска: ответственные за управление и настройку сервис-деска, а также назначение ролей и прав доступа.
* Специалисты технической поддержки: Сотрудники, решающие запросы и инциденты, поступающие от пользователей.
* Клиенты/Пользователи: люди или организации, обращающиеся в сервис-деск с запросами и проблемами.
* Менеджеры проекта: отслеживают производительность сервис-деска и могут требовать отчеты и аналитику.
* Системные администраторы: ответственные за обслуживание и поддержание IT-инфраструктуры, с которой связаны запросы и инциденты.

## **Оценка сроков проекта:**

* Проектирование базы данных: Этот этап может занять от нескольких недель до нескольких месяцев, в зависимости от сложности базы данных и требований к ней.
* Разработка базы данных: Сроки зависят от размера и сложности базы данных, а также от опыта команды разработчиков. Обычно это может занять от нескольких месяцев до года.
* Тестирование и отладка: Зависит от сложности базы данных и объема данных. Оценочно это может занять от нескольких недель до нескольких месяцев.
* Внедрение и обучение: Сроки зависят от того, насколько гладко происходит внедрение и обучение пользователей. Обычно это может занять несколько недель.
* Поддержка и обновление: После внедрения базы данных требуется постоянная поддержка и обновление, что может занимать много месяцев и лет.

## **Требуемые ресурсы:**

* Команда разработчиков: вам потребуются опытные баз данных и разработчики программного обеспечения, включая архитектора баз данных, программистов и администратора баз данных.
* Аналитики и эксперты по бизнес-процессам: эти специалисты помогут определить требования к базе данных и бизнес-правила, которые ей следует соблюдать.
* Тестировщики: необходимы специалисты по тестированию для проверки функциональности и безопасности базы данных.
* Инфраструктура и оборудование: для хранения и обработки данных может потребоваться специализированное оборудование и инфраструктура.
* Бюджет: необходимо выделить бюджет на оплату персонала, оборудование, лицензии ПО и другие расходы.
* Время: команда должна иметь достаточно времени для выполнения проекта согласно установленным срокам.
* Обучение пользователей: для успешного внедрения базы данных необходимо обучить пользователей её использованию.

## **UML-ДИАГРАММА**

Демонстрация диаграммы использования сервиса:

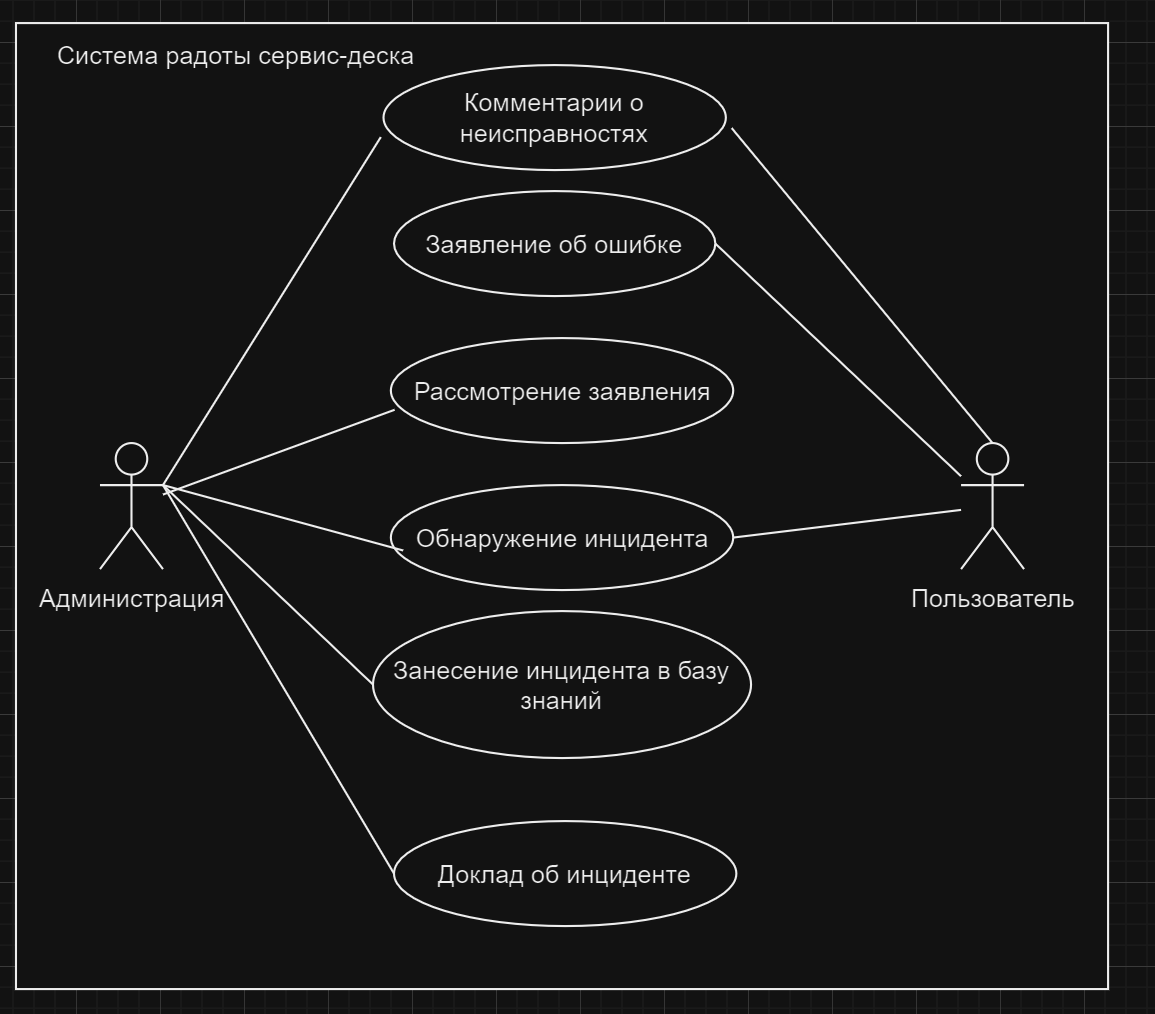
****

Рисунок 5.2 – UML-диаграмма с точки зрения использования

Администраторы данной системы имеют право доступа ко всем возможным операциям, кроме заявления об ошибке. Пользователь данной системы может обнаружить инциденты, произошедшие за время использования. Также, он может сообщить администрации об ошибке, либо прокомментировать что-либо из имеющихся неисправностей; такое возможно, если в ходе использования он обнаружил что-то ещё по этому недочёту.

## **Физическая схема**

На данной схеме отобразим связи объектов базы данных.

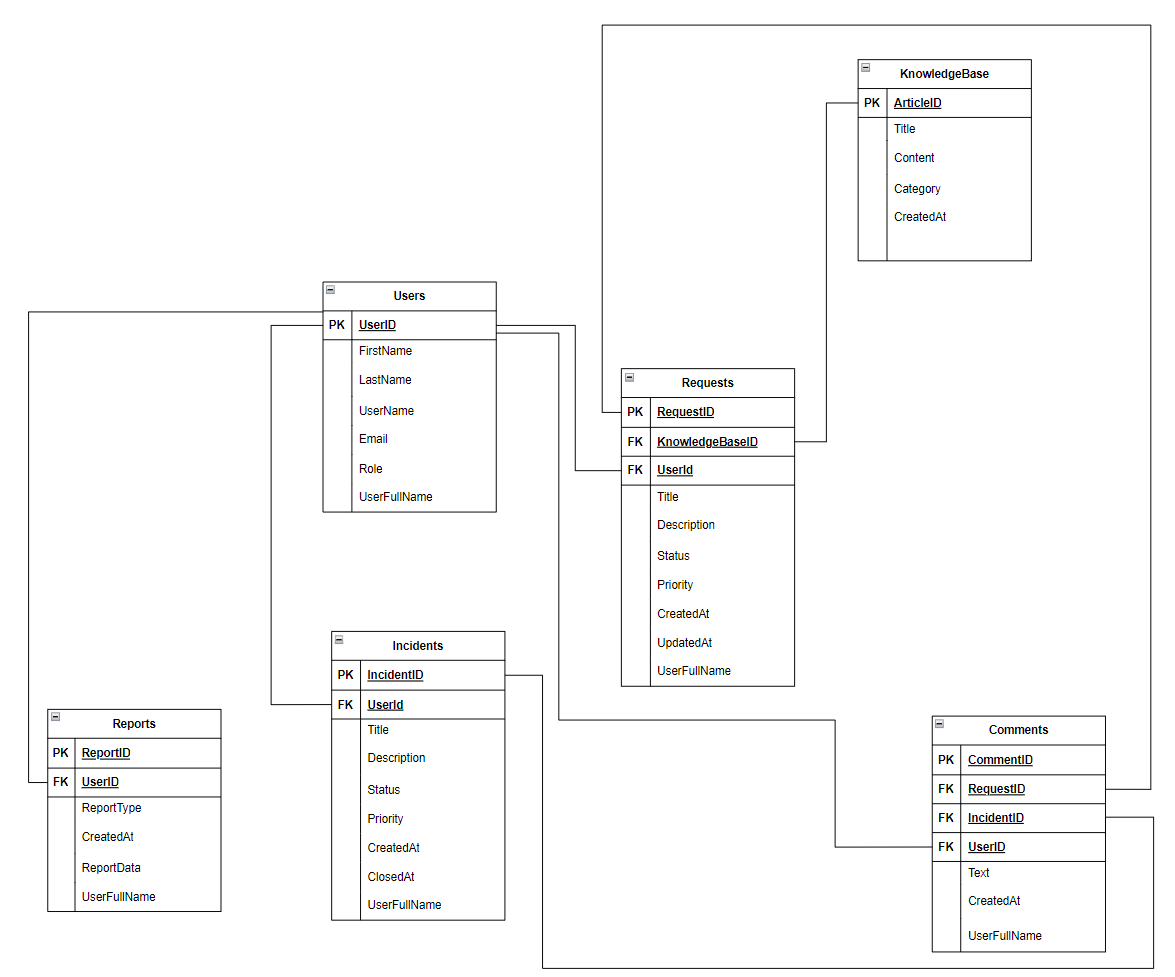
****

Рисунок 6.1 – Физическая схема БД

1. Users (Пользователи):

UserID (Идентификатор пользователя) - NUMBER (Первичный ключ)

UserName (Имя пользователя) - NVARCHAR2

UserFullName (Полное имя пользователя) - NVARCHAR2

Email (Адрес электронной почты) - NVARCHAR2

1. Requests (Запросы):

RequestID (Идентификатор запроса) - NUMBER (Первичный ключ)

Title (Заголовок запроса) - NVARCHAR2

Description (Описание запроса) - NVARCHAR2

Status (Статус запроса) - NVARCHAR2

Priority (Приоритет запроса) - NVARCHAR2

CreatedAt (Дата создания запроса) - TIMESTAMP

UpdatedAt (Дата обновления запроса) - TIMESTAMP

UserID (Идентификатор пользователя) - NUMBER (Внешний ключ, связан с Users.UserID)

UserFullName (Полное имя пользователя, денормализованный атрибут)

1. Incidents (Инциденты):

IncidentID (Идентификатор инцидента) - NUMBER (Первичный ключ)

Title (Заголовок инцидента) - NVARCHAR2

Description (Описание инцидента) - NVARCHAR2

Status (Статус инцидента) - NVARCHAR2

Priority (Приоритет инцидента) - NVARCHAR2

CreatedAt (Дата создания инцидента) - TIMESTAMP

UpdatedAt (Дата обновления инцидента) - TIMESTAMP

UserID (Идентификатор пользователя) - NUMBER (Внешний ключ, связан с Users.UserID)

UserFullName (Полное имя пользователя, денормализованный атрибут)

1. Comments (Комментарии):

CommentID (Идентификатор комментария) - NUMBER (Первичный ключ)

Text (Текст комментария) - NVARCHAR2

CreatedAt (Дата создания комментария) - TIMESTAMP

UserID (Идентификатор пользователя) - NUMBER (Внешний ключ, связан с Users.UserID)

UserFullName (Полное имя пользователя, денормализованный атрибут)

RequestID (Идентификатор запроса) - NUMBER (Внешний ключ, связан с Requests.RequestID)

IncidentID (Идентификатор инцидента) - NUMBER (Внешний ключ, связан с Incidents.IncidentID)

1. Reports (Отчеты):

ReportID (Идентификатор отчета) - NUMBER (Первичный ключ)

Title (Заголовок отчета) - NVARCHAR2

Content (Содержание отчета) - NVARCHAR2

CreatedAt (Дата создания отчета) - TIMESTAMP

1. Knowledge (Знания):

KnowledgeID (Идентификатор знания) - NUMBER (Первичный ключ)

Title (Заголовок знания) - NVARCHAR2

Content (Содержание знания) - NVARCHAR2

CreatedAt (Дата создания знания) – TIMESTAMP

В базе данных были использованы денормализованные атрибуты для повышения производительности и улучшения доступности данных в некоторых таблицах. Денормализация предполагает хранение дублированных данных в таблице для уменьшения необходимости выполнения сложных запросов соединения таблиц. Денормализация данных в этих таблицах позволяет улучшить производительность при отображении и фильтрации данных, так как уменьшает необходимость в соединениях между таблицами. Однако при этом необходимо следить за согласованностью данных и обновлять денормализованные атрибуты при изменении связанных данных в других таблицах (например, при изменении имени пользователя).

## **Логическая схема**

На данной схеме отобразим связи таблиц через первичные и внешние ключи.

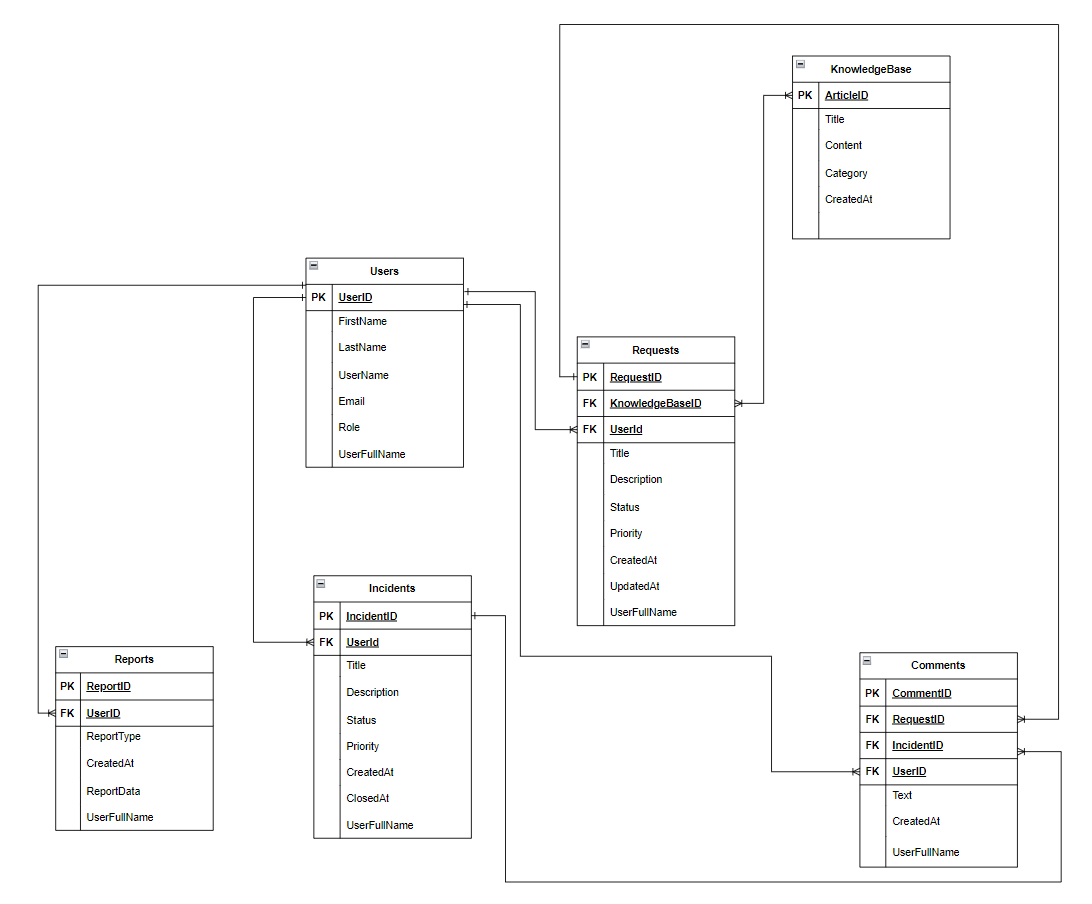


Рисунок 7.1 – Логическая схема БД

На данном рисунке видно следующее:

* Связь между пользователями и запроса2ми (Users - Requests):

Причина: эта связь позволяет отслеживать, какие запросы созданы каждым пользователем. Это важно для учета и управления запросами, а также для обеспечения контроля над доступом пользователей к своим запросам.

* Связь между пользователями и инцидентами (Users - Incidents):

Причина: аналогично запросам, эта связь помогает отслеживать инциденты, созданные каждым пользователем. Это важно для контроля над инцидентами и обеспечения быстрого реагирования на проблемы.

* Связь между пользователями и комментариями (Users - Comments):

Причина: эта связь позволяет знать, какие комментарии оставлены каждым пользователем. Комментарии могут содержать важную информацию о запросах и инцидентах, поэтому отслеживание их авторов полезно.

* Связь между запросами и комментариями (Requests - Comments) и инцидентами и комментариями (Incidents - Comments):

Причина: эти связи помогают связать комментарии с соответствующими запросами или инцидентами. Комментарии предоставляют дополнительный контекст и информацию к запросам и инцидентам.

* Связь между запросами и базой знаний (Requests - Knowledge):

Причина: эта связь позволяет связать запросы с соответствующими статьями из базы знаний. Это полезно для предоставления пользователям релевантной информации и решений на основе существующих знаний.

* Связь между отчетами и пользователями (Reports - Users):

Причина: эта связь позволяет определить, какие отчеты созданы каждым пользователем. Отчеты могут содержать важные данные о работе сервиса и могут быть использованы для анализа и принятия решений.

* Связь между инцидентами и комментариями (Incidents - Comments):

Причина: эта связь аналогична связи между запросами и комментариями и позволяет связать комментарии с инцидентами для уточнения информации о происходящих инцидентах.

## **Безопасность**

Для обеспечения безопасности в системе можно использовать несколько базовых возможностей баз данных:

1. Аутентификация и авторизация:

Аутентификация - проверка подлинности пользователей. Базы данных обычно поддерживают различные методы аутентификации, такие как пароли, сертификаты, интеграция с LDAP и другие.

Авторизация - определение прав доступа пользователей к данным и операциям. Это позволяет контролировать, какие действия могут выполнять пользователи.

1. Роли и привилегии:

Роли - группируют пользователей с общими привилегиями. Создание ролей позволяет управлять доступом к данным более гибко и безопасно.

Привилегии - определяют, какие операции могут выполнять пользователи или роли. Привилегии могут быть назначены таблицам, представлениям, процедурам и другим объектам базы данных.

1. Шифрование данных:

Шифрование данных в базе данных помогает защитить конфиденциальную информацию от несанкционированного доступа. Это может включать в себя шифрование паролей, данных кредитных карт и других конфиденциальных данных.

1. Аудит и журналирование:

Аудит - запись событий и операций, производимых в базе данных. Это позволяет отслеживать действия пользователей и обнаруживать потенциальные угрозы безопасности.

Журналирование - сохранение журналов аудита и другой важной информации о безопасности. Это важно для анализа событий и восстановления данных в случае инцидентов.

1. Ограничения доступа к данным:

Использование представлений и хранимых процедур для ограничения доступа к данным. Это позволяет скрыть сложные запросы и предотвратить прямой доступ к таблицам.