МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

УКРАЇНИ «КПІ імені Ігоря Сікорського»

факультет «Інформатики та обчислювальної техніки»

кафедра «Автоматики та управління в технічних системах»

КУРСОВА РОБОТА

по курсу «Бази даних-2»

на тему: «База даних системи обліку робочого часу за електронними картками»

Студента III курсу ІТ-51 групи

спеціальності «Програмна інженерія»

Сопова Олексія Олександровича

Керівник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Київ – 2017 рік

**ЗМІСТ**

[ВСТУП 6](#_Toc501764762)

[1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 7](#_Toc501764763)

[1.1 Аналіз інформаційних потреб та визначення предметної області дослідження 7](#_Toc501764764)

[1.2 Архітектура та засоби реалізації бази даних формування обліку робочого часу за електронними картками 13](#_Toc501764765)

[1.3 Опис Microsoft SQL Server 15](#_Toc501764766)

[2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ 17](#_Toc501764767)

[2.1 Аналіз інформаційних та датологічних процесів 17](#_Toc501764768)

[2.2 Проектування структури бази даних обліку робочого часу за електронними картками 18](#_Toc501764769)

[3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ 23](#_Toc501764770)

[3.1 Проектування інтерфейсу обробки даних 23](#_Toc501764771)

[3.2 Фізична реалізація бази даних 24](#_Toc501764772)

[3.3 Виконання основних запитів 26](#_Toc501764773)

[3.4 Підтримка міграцій бази даних 33](#_Toc501764774)

[4 АДМІНІСТРУВАННЯ БАЗИ ДАНИХ 35](#_Toc501764775)

[4.1 Порядок налаштування сервера 35](#_Toc501764776)

[4.2 Налаштування прав доступу 37](#_Toc501764777)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 39](#_Toc501764778)

[ДОДАТОК А. Діаграми бази даних 42](#_Toc501764779)

# ВСТУП

У наш час проблема контролю, обліку робочого часу є дуже поширеною серед кампаній, чисельність персоналу яких є великою та стрімко росте. Здавна, було придумано багато варіантів для вирішення цієї проблеми – встановлення охорони, ведення щоденників робочого часу тощо. Проте, вищенаведені методи мають багато недоліків, де серед основних є те, що ці дані можуть бути занадто неточні, або, навіть недостовірні.

В останній час був придуманий інший метод – контроль робочого часу за електронними картками. Кожному співробітнику видається картка, за допомогою якої він може отримати доступ до робочого приміщення. Кожний раз, коли співробітник намагається доступитися до робочого місця, або, навпаки, вийти з нього – необхідно прикласти картку до спеціального зчитувача, який в свою чергу надає доступ, або ні.

Метою даної курсової роботи є розробка бази даних для обліку робочого часу за електронними картками. База даних має надавати дані для повного аналізу робочого часу персоналу, контролю доступу карток до приміщень, ідентифікації персони на основі даних картки.

1. ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ
   1. Аналіз інформаційних потреб та визначення предметної області дослідження

Основною ціллю розроблюваної системи є облік робочого часу за електронними картками. Для висунення функціональних та не функціональних вимог слід чітко сформувати предметну область, від якої можна буде відштовхуватись і при розробці бази даних та програмного засобу. Опишемо предметну область.

Кожному співробітнику при працевлаштуванні надається картка з унікальним ідентифікатором. Кожна картка може належати одному і тільки одному співробітнику. Один співробітник може мати більше однієї картки. При вході в офісне (або інше місце роботи) встановлено спеціальний апарат, який зможе зчитати дані з картки співробітника.

Кожний апарат має унікальний ідентифікатор та «попускає» тільки за заздалегідь визначеними картками – тобто є інформація за якими картками співробітники може отримати доступ до приміщення.

Кожна спроба доступитися/вийти з робочого приміщення зберігається.

Співробітник працює за спеціальним розкладом, який може бути прийнятий до багатьох співробітників і складається з одного або декількох розкладів за днями. У денному розкладі вказано години зміни співробітника та норма відпрацювання у конкретний день.

Також, має бути можливість вказати те, що співробітник у конкретний день не має бути в робочому приміщенні (наприклад, відпустка, хвороба тощо).

Покажемо потік даних у системі за допомогою діаграми IDEF0.

На рисунку 1.1 зображено контекстну IDEF0 діаграму систему обліку робочого часу за електронними картками.

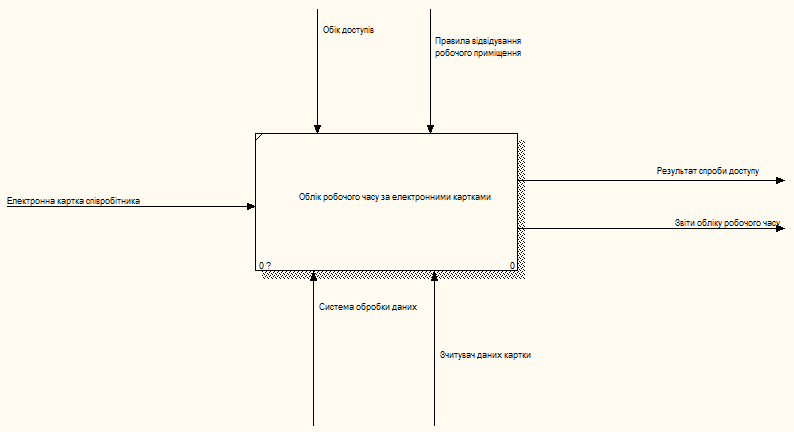


Рисунок 1.1 – Контекстна діаграма систему обліку робочого часу за електронними картками

Опишемо контекстну діаграму більш детально. Основним вхідним параметром обліку робочого процесу співробітників є електронна картка співробітника – саме за допомогою її даних проводиться облік робочого процесу.

Безпосередньо облік робочого процесу відбувається за допомогою «Зчитувача даних картки» який може надавати/блокувати доступ, зчитувати дані картки та сама система обробки даних.

Система обробки даних реалізовує перетворення «сирих» даних: код картки та час доступу до репрезентативних: статистики відвідування, різноманітної аналітики тощо.

Облік, надання доступу, створення аналітики неможливе без надання інформації як саме це робити. Тому, на вхід «Управління» подано «Облік доступів» та «Правила відвідування робочого приміщення».

В управлінні «Облік доступів» збережена інформація про список карток співробітників та до яких перепусток у них є доступ. В управлінні «Правила відвідування робочого приміщення» вказано у котрий час співробітник має знаходитись у офісі, та норму його робочого дня.

Декомпозуємо контекстну діаграму та опишемо її детальніше.

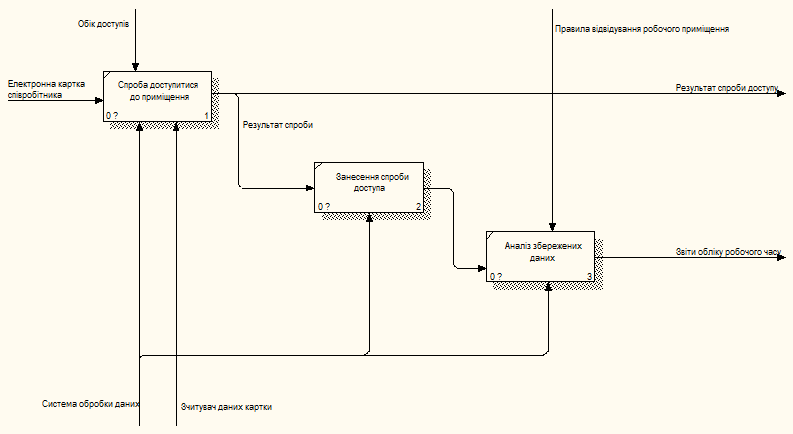


Рисунок 1.2 – Декомпозиція контекстної діаграми обліку робочого часу за електронними картками

З декомпозованої діаграми більш детально видно процес обліку робочого часу за електронними картками.

При спробі доступу співробітником до приміщення, зчитувач даних картки зчитує дані. Надалі ці дані передаються до системи обробки даних, яка, в свою чергу, використовуючи управління «Облік доступів» вирішує чи надати доступ, чи ні. Результат виконання цієї операції видно одразу.

Надалі результат операції, в сукупності із часом, коли операція буда виконана, та ідентифікатором картки передається на збереження. Система обробки даних заносить цю сукупність до бази даних.

Надалі, при необхідності, можливо сформувати аналіз, облік робочого часу за електронними картками на основі зібраної інформації про спроби доступу, під управлінням «Правила відвідування робочого приміщення».

Декомпозуємо блоки діаграми. Декомпозуємо блок «Спроба доступу до приміщення».

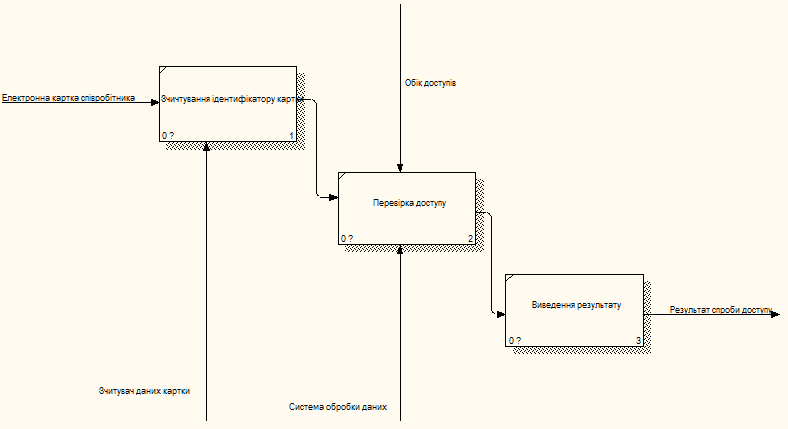


Рисунок 1.3 – Декомпозиція блоку «Спроба доступу до приміщення»

На цій декомпозиції більш детально показано процес спроби доступу до приміщення. Як вже було зазначено, співробітник намагається доступитися до робочого приміщення, прикладаючи картку, ідентифікатор якої зчитується апаратом.

Дані зчитування передаються на перевірку доступу. Перевірка доступу проходить наступним чином: для встановленого апарату обирається список карток, які він «пропускає». Надалі, йде пошук зчитаної картки у цьому списку. Якщо збіг є, то співробітнику надається доступ, інакше – забороняється.

Після цих операцій виводиться результат.

Виконаємо декомпозицію блоку «Занесення спроби доступу». Занесення спроби доступу проходить доволі тривіально.

Спочатку формується група значень: поточний апарат, поточна дата та ідентифікатор картки. Також зберігається інформація про те, чи це був вхід чи вихід, а також результат спроби: вдало, чи ні.

Після формування, ці дані зберігаються локально, або відправляються на сервер для подальшої обробки.

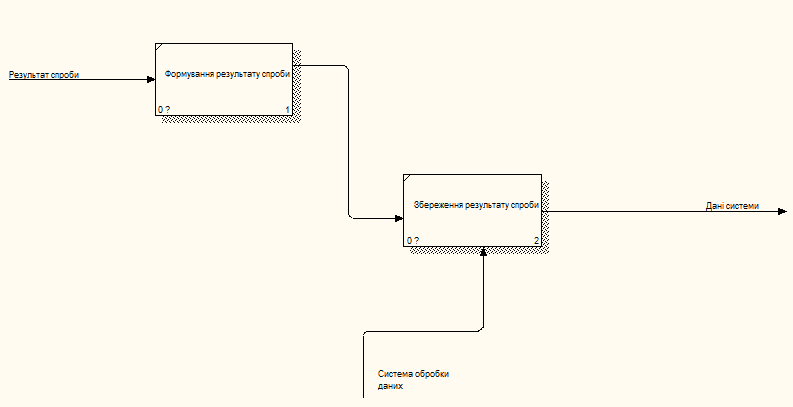


Рисунок 1.4 – Декомпозиція блоку «Занесення спроби доступу»

Основним для розроблюваної системи є блок «Аналіз збережених даних», який виконує основні цілі системи обліку робочого часу за електронними картками. Даний блок розглянемо найбільш детально. Декомпозуємо блок «Аналіз збережених даних» та опишемо основні запити, які необхідні для обліку робочого часу за електронними картками.

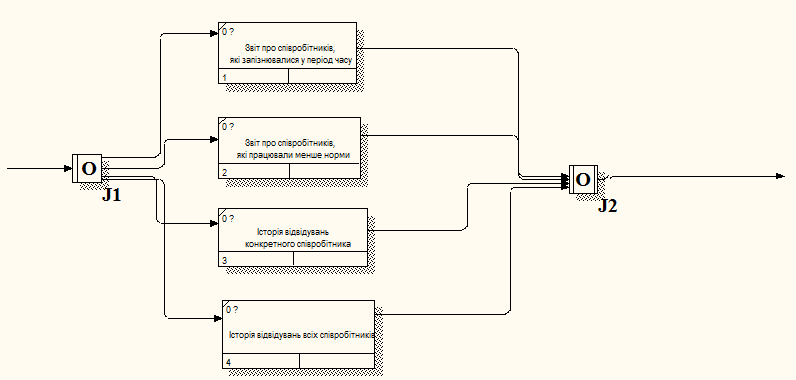


Рисунок 1.5 – Декомпозиція блоку «Аналіз збережених даних» за допомогою нотації IDEF3

Декомпозуємо блок «Аналіз збережених даних» за допомогою нотації IDEF3. У декомпозиції цієї діаграми показано, які звіти система має генерувати для обліку робочого часу за електронними картками.

На вибір користувачу надається чотири основних звіти (не включаючи) стандартні переліки об’єктів та їх властивостей. Як користувачу, та аналізатору робочого часу співробітників, я хотів би знати наступне:

* Історію відвідувань всіх співробітників – для представлення реального статусу відвідування у кампанії
* Історію відвідування конкретного співробітника – для аналізу, наскільки сумлінно він ставиться до роботи
* Звіт про співробітників які працювали менше зазначеної норми – для перегляду заробітної плати цього співробітника, або висунення догани, або отримання аргументованої причини відсутності
* Звіт про співробітників, які спізнювалися – адже вчасний прихід на роботи важливий у кампанії.

Більш детально опишемо кожний із запитів.

Історія відвідувань всіх співробітників: необхідно співпоставити ідентифікатори зчитаних карток з їх власниками, вивести результат у вигляді «Співробітник – коли він з’явився на роботу – коли пішов з роботи».

Історію відвідування конкретного співробітника: необхідно вивести інформацію, аналогічну попередній, але для конкретного співробітника.

Звіт про співробітників які працювали менше зазначеної норми: у правилах відвідування робочого приміщення для кожного співробітника вказано в який день тижня та о котрій годині він має з’явитися на роботі та скільки годин має працювати. Необхідно співпоставити ідентифікатор картки з її власником – співробітником, отримати інформацію про його правила відвідування робочого приміщення та співпоставити фактичні дані (скільки він пробув на роботі в конкретний день) і теоретичні (скільки мав пробути на роботі в конкретний день). Необхідно врахувати те, що у співробітник у той день міг бути відсутній з поважних причин, що також збережено. Врахувати те, що операція «виходу» з приміщення має бути виключена з аналізу.

Звіт про співробітників, які спізнювалися: у правилах відвідування робочого приміщення для кожного співробітника вказано в який день тижня та о котрій годині він має з’явитися на роботі та скільки годин має працювати. Необхідно співпоставити ідентифікатор картки з її власником – співробітником, отримати інформацію про його правила відвідування робочого приміщення та співпоставити час його фактичного приходу та час, у який він мав бути теоретично. Якщо він спізнився – занести до результату. Додаткові примітки аналогічні приміткам у попередньому запиті.

## 1.2 Архітектура та засоби реалізації бази даних формування обліку робочого часу за електронними картками

Основною операцією для розроблюваної бази даних є запис нової спроби доступитися до робочого місця за допомогою електронної картки. Так, майже кожні п’ять хвилин до офісного центру заходить/співробітник і користується системою. На кожну операцію проводиться звернення до бази даних.

Проте, немає необхідності постійного зв’язку апаратної частини із базою даних. Апаратна частина буде проводити усі операції через сервер, який вже, в свою чергу, буде взаємодіяти із базою даних. Тому, вибір СУБД, найбільш сумісний із серверною частиною, є доцільним.

Серверна частина буде реалізована як Web API, та може бути розгорнутою на платформі під керівництвом операційної системи Windows.

Порівняємо системи управління базами даних MS SQL SERVER та Oracle. Результати порівняння занесемо до таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика СУБД Microsoft SQL Serve та Oracle

|  |  |  |
| --- | --- | --- |
| Характеристика | Microsoft SQL Server | Oracle |
| Адміністративне керування | Відмінно | Відмінно |
| Графічні інструменти | Відмінно | Добре |
| Простота обслуговування | Відмінно | Відмінно |
| Механізм данних | Добре | Відмінно |
| Робота с декількома ЦП | Задовільно | Відмінно |
| Функції з’єднання і вибір індексів | Відмінно | Відмінно |
| Одночасний доступ декількох користувачів | Добре | Відмінно |
| Обробка даних мультимедіа | Не задовільняє | Відмінно |
| Подключення к Web | Відмінно | Відмінно |
| Повнотекстовий пошук | Добре | Відмінно |
| Функціональна сумісність | Добре | Добре |
| Інтеграція з іншими СУБД | Добре | Добре |
| Єдина реєстрація | Добре | Добре |
| Робота під керівництвом ОС | Задовільно | Добре |
| Можливості програмування | Задовільно | Відмінно |
| Процедури, що зберігаються та тригери | Добре | Відмінно |
| Вбудована мова програмування | Задовільно | Відмінно |
| Побудова БД | Відмінно | Відмінно |
| Мова SQL | Відмінно | Відмінно |
| Підтримка об’єктно-орієнтованої парадигми | Задовільно | Відмінно |
| Робота у режимі віддаленого доступу | Відмінно | Відмінно |
| Тиражування | Відмінно | Відмінно |
| Розподілена обробка транзакцій | Відмінно | Відмінно |
| Дистанційне адміністрування | Добре | Відмінно |
| Організація сховищ даних і підготовка звітів | Відмінно | Відмінно |
| Засоби завантаження | Відмінно | Відмінно |
| Засоби аналізу | Відмінно | Відмінно |

Також, порівняємо системи управління базами даних за вимогами технічного завдання.

Таблиця 1.2 – Порівняння СУБД за вимогами технічного завдання

|  |  |  |
| --- | --- | --- |
| Необходимые требования | MS SQL Server | Oracle |
| Локалізація інтерфейсу користувача, можливість побудови і сортування полів БД, що містять символи кирилиці | + | + |
| Підтримка структури відносних даних | + | + |
| Підтримка технології клієнт/сервер | + | + |
| Підтримка багатопроцесорної архітектури | + | + |
| Підтримка кластерної архітектури | + | + |
| Наявність засобів для створення індексів і кластерів для підвищення ефективності використання даних | + | + |
| Відновлення баз даних із використанням журнала трансакцій | + | + |
| Механізм блокування трансакцій під час запису або на рівні сторінок | + | + |
| підтримка ANSI SQL | + | + |
| Підтримка стандарту SQL-3 (новое название – SQL99) | + | + |
| підтримка ODBC | + | + |
| Контроль цілісності БД | + | + |
| Підтримка утиліт резервування БД | + | + |
| Імпорт/експорт таблиць БД | + | + |
| Сумісність з ОС модулів користувача та сервера | + | + |
| Підтримка визначених мережевих протоколів | + | + |
| Наявність графічного інтерфейсу для адміністраторів БД | + | + |
| Контроль доступу до даних. Аутентифікація засобами СУБД | + | + |
| Централізоване керування користувачами | + | + |
| Наявність оптимізатора запитів для оптимізації планів виконання | + | + |
| Підтримка великих двійкових об’єктів (BLOB) | + | + |
| Підтримка OLAP технологій, спеціалізованих засобів OLAP-аналізу | + | + |
| Підтримка протоколів VI SAN (Virtual Interface System Area Network) | + | + |
| Відлагоджений механізм реплікації даних | + | + |
| Підтримка служби єдиного каталогу | + | + |

З порівняльної таблиці бачимо, що обидві СУБД добре вправляються із необхідними задачами. Хоча Oracle в деяких моментах є кращим за Microsoft SQL Server, остання легша в користуванні та простіше інтегрується з ASP Web API, тому для реалізації бази даних буде обрано Microsoft SQL Server.

1.3 Опис Microsoft SQL Server

Це система керування базами даних, розроблена корпорацією Microsoft.

Основний використовуваний мову запитів - Transact-SQL, створений Microsoft. Transact-SQL є реалізацією стандарту по структурованій мові запитів (SQL) з розширеннями. Використовується для роботи з невеликими і середніми за розміром базами даних.

Завдяки інтеграції з середовищем розробки Microsoft Visual Studio SQL

Server дозволяє розробляти керовані додатками дані з широкими можливостями, які забезпечують поліпшену безпеку сховищ і швидке розгортання.

У своєму складі система має засоби створення баз даних, роботи з інформацією баз даних, перенесення даних з інших систем і в інші системи, резервного копіювання та відновлення даних, розвинену систему транзакцій, систему реплікації даних, реляційну підсистему для аналізу, оптимізації та виконання запитів клієнтів, систему безпеки для управління правами доступу до об'єктів бази даних та ін. Система не містить засобів розробки клієнтських

додатків.

SQL Server будується за структурою таблиці на основі рядків, а також з'єднує пов'язані елементи даних у різних таблицях, уникаючи необхідності резервного зберігання даних у кількох місцях в межах бази даних. Реляційна модель також забезпечує дотримання принципів атомності, узгодженості, ізоляції та довговічності - загальновідомі як властивості ACID і призначені для гарантування того, що транзакції бази даних надійно обробляються.

Основним компонентом Microsoft SQL Server є SQL Server Database Engine, який керує збереженням, обробкою даних та безпекою даних. Вона включає в себе реляційний движок, який обробляє команди та запити, а також двигун зберігання, який керує файлами, таблицями, сторінками, індексами, буферами даних та транзакціями. Збережені процедури, активатори, перегляди та інші об'єкти бази даних також створюються та виконуються за допомогою двигуна бази даних.

# 2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

## 2.1 Аналіз інформаційних та датологічних процесів

Основні потоки даних, які наявні у розроблюваній системі були описані у попередньому пункті при описанні предметної області системи. Найбільш основні деталі були показані за допомогою діаграм IDEF0 та IDEF3.

У цьому пункті більшу увагу приділимо опису датологічних процесів та поступово будемо зв’язувати їх із майбутньою базою даних.

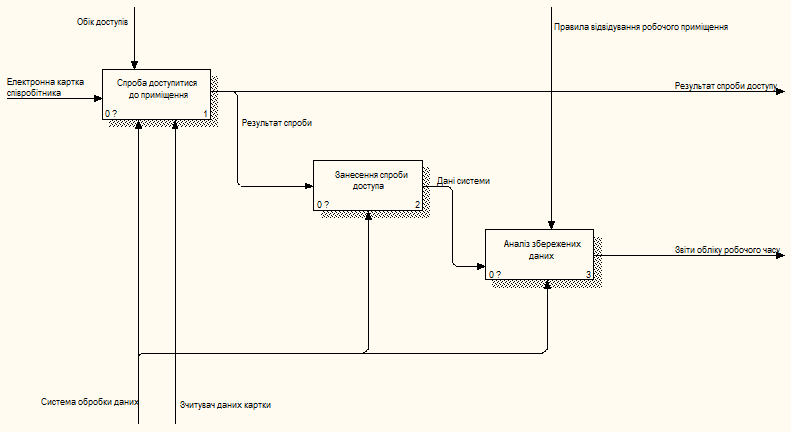


Рисунок 2.1 – Основні інформаційні стани системи та датологічні потоки

Опишемо основний потік даних, який буде проходити у системі.

Коли співробітник піднесе карту до зчитувального апарату, дані про картку будуть зчитані та збережені локально.

Надалі збережені дані проходять обробку для пропуску співробітника, або відмови у пропуску. Зчитуються усі картки які мають доступ до «проходу» на даному апараті, та порівнюються із зчитаною карткою.

Після вищезазначених операцій проводиться збереження даних. Для збереження формується група даних: поточна дата, поточний автомат перепустки, ідентифікатор зчитаної картки.

Сформовані дані переправляються на сервер для обробки.

Надалі, при необхідності, зібрані дані аналізуються. Проводяться запити на формування різник звітів, статистик. Для цих цілей проводиться суміщення власників карток та їх карток, їх правил відвідування офісу.

## 2.2 Проектування структури бази даних обліку робочого часу за електронними картками

Аналізуючи предметну область, виведемо необхідні дані, які на необхідно зберігати. Обрані дані для збереження надалі будуть прес давлені у схемі бази даних для проектуємої системи.

Основними даними для збереження є:

* Співробітник. Для співробітника необхідно зберігати його ім’я, прізвище та назву роботи для легшого представлення даних
* Електронна картка. Електронна картка, яка видається співробітнику, за допомогою якої він зможе пройти до робочого місця
* Список перепусток. Список перепусток з ідентифікаторами, на яких встановлено автомат для зчитування даних електронної картки
* Список доступів. Список доступів окремих карток до окремої перепустки.
* Правило відвідування офісу. Правило відвідування офісу складається з декількох окремих правил відвідування офісу за день.
* Спроби пройти перепустку

Зобразимо логічну структуру бази даних за допомогою Entity Relationship діаграми (рисунок 2.2).

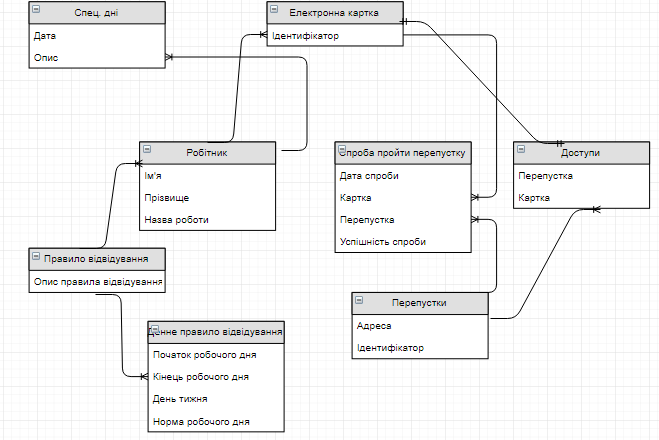


Рисунок 2.2 – Логічна структура бази даних, описана за допомогою Entity Relationship діаграми

Опишемо дану діаграму більш детально. «Основним» елементом даної діаграми можна виділити елемент Entity (далі сутність) «Спроба пройти перепуску». Він зберігає дату спроби, картку за якою була здійснена спроба, та інформацію про саму перепустку.

Як показано на діаграмі за допомогою відношення «Один до багатьох», робітник може мати декілька карток (для доступу до різних офісів) та може мати особливі дні, в які він не має бути в офісі.

«Доступи» реалізовані за допомогою наступних відношень: одна перепустка може мати декілька доступів, але одна картка може мати один доступ.

Один робітник має одне правило відвідування (наприклад відвідування по буднім дням з 10 до 18). Таке правило містить в собі декілька більш детальних правил, що розписані по днях тижня.

Денне правило включає в себе час початку робочого дня, час кінця робочого дня, день тижня та норму робочого дня. Причиною виділення такої окремої сутності є гнучкість настроювань правил відвідування для різних співробітників за їх потребами.

Зобразимо фізичну

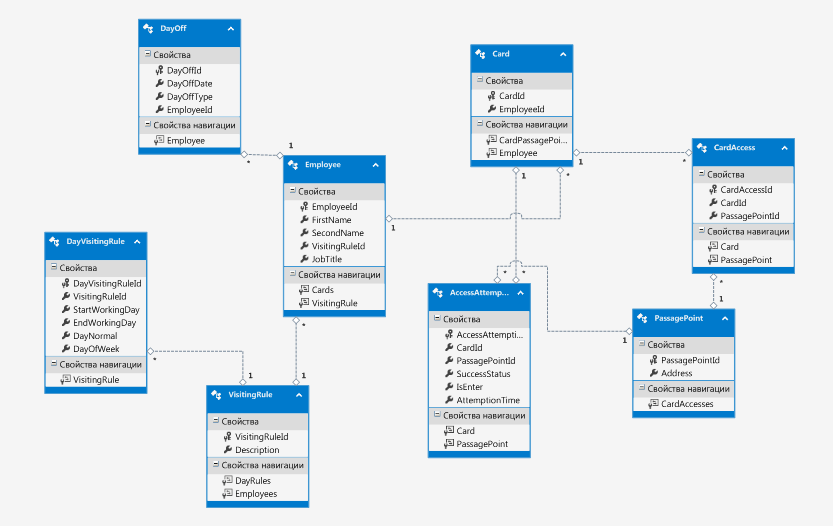


Рисунок 2.3 – Фізична діаграма бази даних

Більш детально опишемо кожну таблицю

Таблиця Employee – описує інформацію про співробітника:

* EmployeeId – уныкальний ідентифікатор співробітника
* FirstName – зберігається інформація про Ім’я співробітника
* SecondName – зберігається інформація про Прізвище співробітника
* JobTitle – зберігається інформація про назву роботи співробітника

Таблиця Card – описує інформацію про карту співробітника

* CardId – унікальний ідентифікатор картки
* EmployeeId – власник картки. Employee та Card знаходяться у выдношенні «один до одного» адже один співробітник може мати лише одну картку.

Таблиця VisitingRule – описує правила відвідування робочого приміщення на різні дні. Містить набір DayVisitingRule.

* VisitingRuleId – унікальний ідентифікатор правила відвідування
* Description – короткий опис правила відвідування

Таблиця DayVisitingRule – описує правило відвідування для конкретного дня тижня.

* DayVisitingRuleId – унікальний ідентифікатор
* VisitingRuleId – ідентифікатор правила відвідування до якого відноситься денне правило
* StartWorkingDay – час початку робочого дня
* EndWorkingDay – час кінця робочого дня
* DayNormal – норма часу, який співробітник має провести на роботі
* DayOfWeek – день тижня, до якого відносить дане правило денного відвідування

Таблиця AccessAttemption – описує спробу пройти до робочого приміщення використовуючи електронну картку

* AccessAttemptionId – унікальний ідентифікатор спроби
* CardId – ідентифікатор картки, для якої було виконано спробу
* PassagePointId – ідентифікатор місця, до якого намагалися пройти
* SuccessStatus – показує, чи успішна була операція
* IsEnter – показує, чи операція була входом чи виходом із приміщення
* AttemptionTime – час стровення операції

Таблиця PassagePoint – описує місця-перепустки до робочих приміщень

* PassagePointId
* Address

Таблиця DayOff – описує дату та причину відсутності співробітника у місці роботи

* DayOffId – унікальний ідентифікатор
* DayOffDate – дата відсутності співробітника
* DayOffType – тип (причина) відсутності співробітника
* EmployeeId – ідентифікатор співробітника, який був (або буде) відсутній у вказану дату

Таблиця CardAccess – описує доступи карток до перепусток

* CardAccessId – унікальний ідентифікатор
* CardId – ідентифікатор картки, для якої надано доступ
* PassagePointId – ідентифікатор перепустки, у яку надано доступ карткою

# 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ

## 3.1 Проектування інтерфейсу обробки даних

Опишемо структуру інтерфейсу системи та засобів, які було використано в процесі реалізації функцій системи.

Згідно з предметною областю основна частина взаємодії з користувачем: тобто співробітником, буде проводитись через апаратну частину, тому, для облегшення доступу до системи було прийнято рішення: створити сервер для взаємодії із базою даних. Апаратна частина лише надсилатиме запити до нього та не буде обробляти багато зайвої логіки.

Опишемо логіку взаємодії через Web API.

Основний потік інформації проходить, як показано на рисунку 3.1.

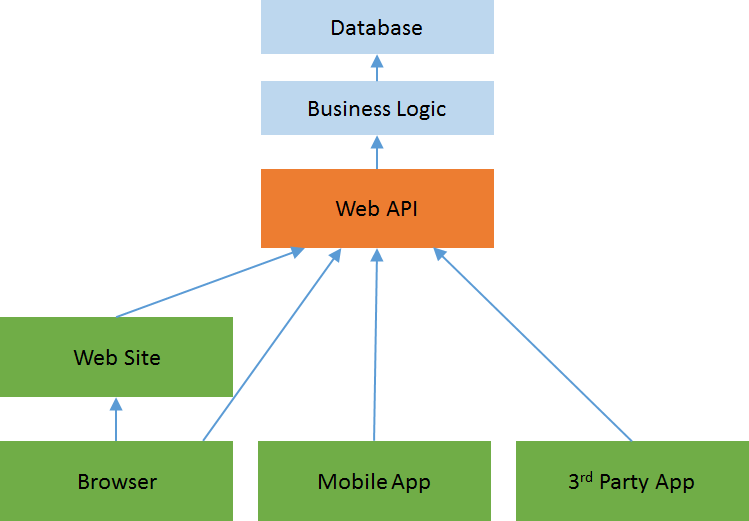


Рисунок 3.1 – Потік інформації при взаємодії із Web API

Розглянемо даний малюнок детальніше. У нашому випадку для відправлення даних виступає апаратна частина (тобто 3rd party App), а для перегляду результатів – браузер.

Для того, щоб записати нові дані, або отримати їх – необхідно лише надіслати HTTP запит, а у відповідь прийдуть готові результати.

Уля логіка обробки запитів лягає на сторону сервера (що в нашому випадку є вигідним), а також уся взаємодія із базою даних проходить крізь нього.

Для взаємодії із базою даних був обраний Entity Framework, який легко взаємодіяти із базо даних та виконувати основні операції більш типово-залежно, тобто більш безпечно.

Також, в Entity Framework вбудований захист від SQL Injections та підтримує міграції бази даних.

## 3.2 Фізична реалізація бази даних

У даному пункті опишемо фізичну реалізацію бази даних. Буде наведено SQL запити для створення схеми бази даних.

Згідно з предметною областю та фізичною моделлю бази даних (та ER діаграмою) створимо SQL запит для створення визначених таблиць.

create table [dbo].[AccessAttemptions] (

[AccessAttemptionId] [int] not null identity,

[CardId] [int] not null,

[PassagePointId] [int] not null,

[SuccessStatus] [bit] not null,

[IsEnter] [bit] not null,

[AttemptionTime] [datetime] not null,

primary key ([AccessAttemptionId])

);

create table [dbo].[Cards] (

[CardId] [int] not null identity,

[EmployeeId] [int] not null,

primary key ([CardId])

);

create table [dbo].[CardAccesses] (

[CardAccessId] [int] not null identity,

[CardId] [int] not null,

[PassagePointId] [int] not null,

primary key ([CardAccessId])

);

create table [dbo].[DayOffs] (

[DayOffId] [int] not null identity,

[DayOffDate] [datetime] not null,

[DayOffType] [int] not null,

[EmployeeId] [int] not null,

primary key ([DayOffId])

);

create table [dbo].[DayVisitingRules] (

[DayVisitingRuleId] [int] not null identity,

[VisitingRuleId] [int] not null,

[StartWorkingDay] [time](7) not null,

[EndWorkingDay] [time](7) not null,

[DayNormal] [int] not null,

[DayOfWeek] [int] not null,

primary key ([DayVisitingRuleId])

);

create table [dbo].[Employees] (

[EmployeeId] [int] not null identity,

[FirstName] [nvarchar](max) null,

[SecondName] [nvarchar](max) null,

[VisitingRuleId] [int] not null,

[JobTitle] [nvarchar](max) null,

primary key ([EmployeeId])

);

create table [dbo].[PassagePoints] (

[PassagePointId] [int] not null identity,

[Address] [nvarchar](max) null,

primary key ([PassagePointId])

);

create table [dbo].[VisitingRules] (

[VisitingRuleId] [int] not null identity,

[Description] [nvarchar](max) null,

primary key ([VisitingRuleId])

);

alter table [dbo].[AccessAttemptions] add constraint [AccessAttemption\_Card] foreign key ([CardId]) references [dbo].[Cards]([CardId]) on delete cascade;

alter table [dbo].[AccessAttemptions] add constraint [AccessAttemption\_PassagePoint] foreign key ([PassagePointId]) references [dbo].[PassagePoints]([PassagePointId]) on delete cascade;

alter table [dbo].[CardAccesses] add constraint [CardAccess\_Card] foreign key ([CardId]) references [dbo].[Cards]([CardId]) on delete cascade;

alter table [dbo].[DayOffs] add constraint [DayOff\_Employee] foreign key ([EmployeeId]) references [dbo].[Employees]([EmployeeId]) on delete cascade;

alter table [dbo].[DayVisitingRules] add constraint [DayVisitingRule\_VisitingRule] foreign key ([VisitingRuleId]) references [dbo].[VisitingRules]([VisitingRuleId]) on delete cascade;

alter table [dbo].[Cards] add constraint [Employee\_Cards] foreign key ([EmployeeId]) references [dbo].[Employees]([EmployeeId]) on delete cascade;

alter table [dbo].[CardAccesses] add constraint [PassagePoint\_CardAccesses] foreign key ([PassagePointId]) references [dbo].[PassagePoints]([PassagePointId]) on delete cascade;

alter table [dbo].[Employees] add constraint [VisitingRule\_Employees] foreign key ([VisitingRuleId]) references [dbo].[VisitingRules]([VisitingRuleId]) on delete cascade;

У пункті 1.2 як система управління базами даних був обраний MS SQL SERVER. За допомогою зазначеної СУБД виконаємо даний скрипт. При виконанні даного скрипту отримаємо: Script has been executed. Return code: 0, що позначає успішне виконання.

Як підтвердження успішності виконання скрипту, переглянемо обігрівач SQL SERVER. Результат SQL SERVER показано на рисунку 3.2.

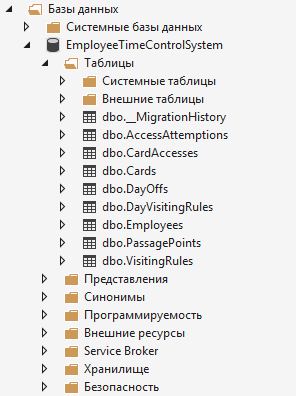


Рисунок 3.2 – Створена база даних та таблиці після успішного виконання скрипту

## 3.3 Виконання основних запитів

У даному розділі опишемо основні операції над базою даних, які необхідні для коректної роботи системи. Також, опишемо складні запити для генерації звітів, вказаних як функціональна вимога до системи.

Для реалізації доволі тривіальних засобів використаємо вбудовані до Entity Framework засоби, які автоматично генерують Sql команди та виконують їх.

Для підключення до бази даних через Entity Framework, нам потрібний посередник - контекст даних. Контекст даних представляє собою клас, похідний від класу DbContext. Контекст даних містить один або кілька властивостей типу DbSet <T>, де T являє собою тип об'єкта, що зберігається в базі даних. Для того, щоб зв'язати додаток, контекст даних та БД, необхідно додати в файл web.config (app. config) строку підключення до даної бази даних.

Прикладом використання автогенероравних операцій показано у коді нижче.

private EmployeeTimeControlDataContext db = new EmployeeTimeControlDataContext();

// GET: api/Employees

public IQueryable<Employee> GetEmployeeSet()

{

return db.EmployeeSet;

}

Де db містить варіації DbSet. Вираз db.EmployeeSet буде перетворено у наступний SQL запит та виконано:

SELECT

[Extent1].[EmployeeId] AS [EmployeeId],

[Extent1].[FirstName] AS [FirstName],

[Extent1].[SecondName] AS [SecondName],

[Extent1].[VisitingRuleId] AS [VisitingRuleId],

[Extent1].[JobTitle] AS [JobTitle]

FROM [dbo].[Employees] AS [Extent1]

Інші операції, як додавання, оновлення, видалення виконуються аналогічним образом. Наприклад, код видалення буде виглядати наступним чином: db.EmployeeSet.Remove(employee).

Більш складні запити потребують написання власних SQL запитів. Реалізуємо основні запити, сказані в предметній області.

* Історія відвідування конкретного співробітника (Результат виконання показано на рисунку 3.3)

SELECT \*

FROM AccessAttemptions AS aas

WHERE aas.CardId IN

(SELECT Cards.CardId

FROM Cards

WHERE Cards.EmployeeId = @id);

Розглянемо скрипт більш детально.

SELECT \* FROM AccessAttemptions AS aas виконує вибірку усіх полів з таблиці AccessAttemptions (спроби доступу).

WHERE aas.CardId IN

(SELECT Cards.CardId

FROM Cards

WHERE Cards.EmployeeId = @id); На усі спроби доступу накладається умова, що картка має принадлежати досліджуваному співробітнику.

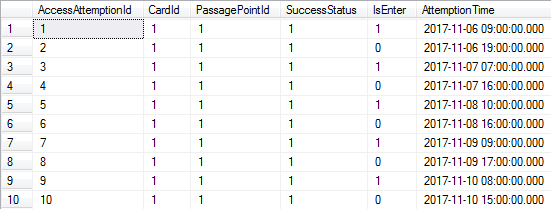


Рисунок 3.3 – Результат виконання запиту «Історія відвідування конкретного співробітника»

* Історія відвідування всіх співробітників. Історію відвідування всіх співробітників можна отримати засобами Entity Framework, написавши db.AccessAttemptionsSet (результат виконання зображено на рисунку 3.4)

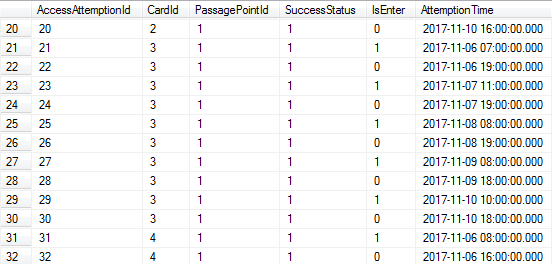


Рисунок 3.3 – Результат виконання запиту «Історія відвідування всіх співробітників»

* Звіт про співробітників, які спізнювалися у конкретний період часу (результат виконання наведено на рисунку 3.4)

SELECT empl.EmployeeId, empl.FirstName, empl.SecondName, empl.JobTitle, empl.VisitingRuleId

FROM Employees AS empl

JOIN Cards ON Cards.EmployeeId = empl.EmployeeId

JOIN AccessAttemptions AS aas ON aas.CardId = Cards.CardId

WHERE cast(aas.AttemptionTime as time) >

(

SELECT cast(dvr.StartWorkingDay as time)

FROM DayVisitingRules as dvr

JOIN VisitingRules AS vis ON dvr.VisitingRuleId = vis.VisitingRuleId

JOIN Employees AS empl ON empl.VisitingRuleId = vis.VisitingRuleId

JOIN Cards AS ca ON ca.CardId = empl.EmployeeId

WHERE ca.CardId = aas.CardId AND dvr.DayOfWeek + 1 = DATEPART(weekday, aas.AttemptionTime)

)

AND aas.IsEnter = 1

AND NOT aas.AttemptionTime IN

(

SELECT do.DayOffDate

FROM DayOffs AS do

JOIN Employees ON Employees.EmployeeId = do.EmployeeId

JOIN Cards ON Cards.CardId = aas.CardId

)

AND aas.AttemptionTime BETWEEN @from AND @to

Розглянемо скрипт більш детально:

SELECT empl.EmployeeId, empl.FirstName, empl.SecondName, empl.JobTitle, empl.VisitingRuleId

FROM Employees AS empl – робимо вибірку по всім полям із таблиці Employees.

JOIN Cards ON Cards.EmployeeId = empl.EmployeeId

JOIN AccessAttemptions AS aas ON aas.CardId = Cards.CardId – робимо додання до запиту даних із таблиць Cards та Employees по відповідним полям.

WHERE cast(aas.AttemptionTime as time) >

(

SELECT cast(dvr.StartWorkingDay as time)

FROM DayVisitingRules as dvr

JOIN VisitingRules AS vis ON dvr.VisitingRuleId = vis.VisitingRuleId

JOIN Employees AS empl ON empl.VisitingRuleId = vis.VisitingRuleId

JOIN Cards AS ca ON ca.CardId = empl.EmployeeId

WHERE ca.CardId = aas.CardId AND dvr.DayOfWeek + 1 = DATEPART(weekday, aas.AttemptionTime)

) - створення умови на те, що співробітник прийшов пізніше, ніж мав бути.

SELECT cast(dvr.StartWorkingDay as time)

FROM DayVisitingRules as dvr – отримуємо час дня, у який співробітник мав прийти

JOIN VisitingRules AS vis ON dvr.VisitingRuleId = vis.VisitingRuleId

JOIN Employees AS empl ON empl.VisitingRuleId = vis.VisitingRuleId

JOIN Cards AS ca ON ca.CardId = empl.EmployeeId

WHERE ca.CardId = aas.CardId AND dvr.DayOfWeek + 1 = DATEPART(weekday, aas.AttemptionTime – робимо взаємозв’язок між картою співробітника та часом, у який він мав прийти.

AND aas.IsEnter = 1 – накладаємо умову на те, що співробітник має входити в приміщення, а не виходити

AND NOT aas.AttemptionTime IN

(

SELECT do.DayOffDate

FROM DayOffs AS do

JOIN Employees ON Employees.EmployeeId = do.EmployeeId

JOIN Cards ON Cards.CardId = aas.CardId

) – враховуємо те, що співробітник не мав приходити у цей день, але якщо прийшов – не вважати його, як того, який запізнився (згідно вимогам).

AND aas.AttemptionTime BETWEEN @from AND @to умова на час, у який необхідно дослідити.

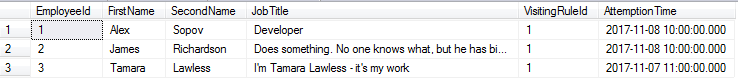


Рисунок 3.3 – Результат виконання запиту «Звіт про співробітників, які спізнювалися у конкретний період часу (2017-11-06.. 2017-11-09)»

* Звіт про співробітників, які працювали менше зазначеної норми (результат виконання наведено на рисунку 3.5)

SELECT empl.EmployeeId, empl.FirstName, empl.SecondName, empl.JobTitle, empl.VisitingRuleId

FROM Employees AS empl

JOIN Cards ON Cards.EmployeeId = empl.EmployeeId

JOIN AccessAttemptions AS aas ON aas.CardId = Cards.CardId

WHERE aas.IsEnter = 0

AND DATEPART(hour, aas.AttemptionTime -

(

SELECT DATEPART(hour, aasg.AttemptionTime) FROM AccessAttemptions AS aasg

WHERE aasg.CardId = aas.CardId

AND aasg.IsEnter = 1

AND aasg.PassagePointId = aas.PassagePointId

AND CONVERT(date, aasg.AttemptionTime) = CONVERT(date, aas.AttemptionTime)

))

<

(

SELECT dvr.DayNormal

FROM DayVisitingRules as dvr

JOIN VisitingRules AS vis ON dvr.VisitingRuleId = vis.VisitingRuleId

JOIN Employees AS empl ON empl.VisitingRuleId = vis.VisitingRuleId

JOIN Cards AS ca ON ca.CardId = empl.EmployeeId

WHERE ca.CardId = aas.CardId AND dvr.DayOfWeek + 1 = DATEPART(weekday, aas.AttemptionTime)

)

AND aas.AttemptionTime BETWEEN @from AND @to.

Розглянемо скрипт більш детально:

SELECT empl.EmployeeId, empl.FirstName, empl.SecondName, empl.JobTitle, empl.VisitingRuleId

FROM Employees AS empl – робимо вибірку по всім полям із таблиці Employees.

JOIN Cards ON Cards.EmployeeId = empl.EmployeeId

JOIN AccessAttemptions AS aas ON aas.CardId = Cards.CardId – робимо додання до запиту даних із таблиць Cards та Employees по відповідним полям.

WHERE aas.IsEnter = 0 – робимо умову, що був вихід (для віднімання від часу входу)

SELECT DATEPART(hour, aasg.AttemptionTime) FROM AccessAttemptions AS aasg

WHERE aasg.CardId = aas.CardId

AND aasg.IsEnter = 1

AND aasg.PassagePointId = aas.PassagePointId

AND CONVERT(date, aasg.AttemptionTime) = CONVERT(date, aas.AttemptionTime) – знаходимо до виходу відповідний час входу та знаходимо різницю у годинах.

SELECT dvr.DayNormal

FROM DayVisitingRules as dvr

JOIN VisitingRules AS vis ON dvr.VisitingRuleId = vis.VisitingRuleId

JOIN Employees AS empl ON empl.VisitingRuleId = vis.VisitingRuleId

JOIN Cards AS ca ON ca.CardId = empl.EmployeeId

WHERE ca.CardId = aas.CardId AND dvr.DayOfWeek + 1 = DATEPART(weekday, aas.AttemptionTime) – знаходимо норму, яку співробітник мав провести на роботі



Рисунок 3.5 – Результат виконання запиту «Звіт про співробітників, які працювали менше зазначеної норми (2017-11-06.. 2017-11-09)»

## 3.4 Підтримка міграцій бази даних

За допомогою засобів Entity Framework можна підтримувати міграції бази даних: вказати що зміниться при підвищенні версії бази даних та що відбудеться при її пониженні.

Наприклад, однією з необхідних міграцій у проекті буда заміна типу DateTime на Time. Кожна міграція у Entity Framework має бути екземпляром класу DbMigration. Для реалізації функціоналу мігрування, реалізовуються два методи: Up() та Down(), які, в свою чергу, відповідають за дії, які необхідно виповнити при підвищенні версії бази даних та її пониженні відповідно.

Приклад міграції бази даних показано на рисунку 3.6.

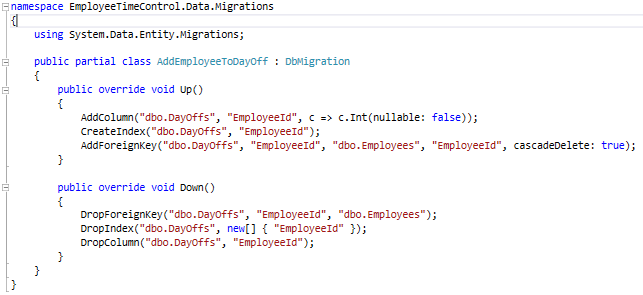


Рисунок 3.6 – Вихідний код міграції бази даних

# 4 АДМІНІСТРУВАННЯ БАЗИ ДАНИХ

## 4.1 Порядок налаштування сервера

Запускаємо інталятор MS SQL SERVER і у вітальному діалозі вибираємо Installation -> New installation or Add features to an existing installation :

При запуску інсталяції в першу чергу проводяться перевірки сумісності операційної системи, чи вистачає прав у користувача для установки SQL Server і т.п. Якщо у вас вже стоїть екземпляр SQL Server 2008, то ви побачите попередження про те, що загальні компоненти ( SQL Server Management Studio, Integration Services, Books Online тощо) будуть оновлені до версії до SQL Server 2008 R2. Якщо всі перевірки пройдено, можна переходити до наступного кроку. І після розпакування файлів, необхідних для інсталяції та ще одного етапу перевірок, почнеться найцікавіше - конфігурація установки :

Першим етапом якої буде введення ліцензійного ключа продукту, або вибору безкоштовної редакції ( Evaluation, Express, Express with Advanced Services ). І якщо ви ставите редакцію SQL Server, де ключ вже вбитий в полі « Enter the product key » (наприклад Developer Edition ), то раджу зберегти його.

Наступним кроком буде вибір формату установки, де пропонується 3 варіанти :

* SQL Server Feature Installation - тут всі налаштування доведеться робити.
* SQL Server PowerPivot for SharePoint - окрім самого SQL Server, буде поставлений і сконфігурований PowerPivot плагін до SharePoint.
* All Features With Defaults - для установки будуть вибрані всі фічі (з можливістю прибрати те, що не потрібно ) і проставлені акаунти за замовчуванням для сервісів:

На наступному екрані вибираємо ті компоненти SQL Server, які хочемо встановити. Тут доцільно вибрати все. Перелік елементі, які можна вибрати для установки ( більш детальний опис компонентів можна отримати натиснувши F1 на поточному кроці):

* Database Engine Services - SQL Server.
* SQL Server Replication - компоненти реплікації SQL Server використовуються для синхронізації баз даних.
* Full - Text Search - компонент повнотекстового пошуку дозволяє організувати ефективний пошук по текстових полям бази з обліків різних мов і різних форм слова.
* Analysis Services - дозволяє будувати багатовимірні ( OLAP ) сховища даних і DataMining моделі для проведення аналізу та побудови прогнозів..
* Reporting Services - сервіси та інструменти для побудови та управління звітами.
* Shared Features (вони ставляться 1 раз, і будуть доступні всім екземплярам, які встановлені на машині).
* Business Intelligence Development Studio - якщо стоїть Visual Studio, то до неї додаються нові типи проектів для розробки рішень Analysis Services, Reporting Services і Integration Services. Якщо Visual Studio немає, то ставиться «міні» Visual Studio, в якій доступні тільки ці, перераховані вище типи проектів.
* Client Tools Connectivity - провайдери для з'єднання клієнтів з сервером.
* Integration Services - сервіси, що дозволяють організувати отримання, перетворення і перенесення даних з різних джерел.
* Client Tools Backwards Compatibility - SQL Distributed Management Objects (SQL - DMO ), Decision Support Objects ( DSO ), Data Transformation Services ( DTS ).
* Client Tools SDK - SDK для розробників.
* SQL Server Books Online - документація по SQL Server.
* Management Tools - Basic - базовий варіант Management Studio, SQLCMD і SQL Server PowerShell provider.
* Management Tools - Complete - повноцінна Management Studio (підтримка Analysis Serveices, Integration Services, Reporting Services ), Profiler, Database Engine Tuning Advisor, SQL Server Utility.
* SQL Client Tools Connectivity SDK - на Microsoft Connect є баг щодо опису цього елемента - SQL Client Connectivity SDK and Client Tools SDK DOCUMENTATION.
* Microsoft Sync Framework - багатофункціональна платформа синхронізації, що дозволяє інтегрувати будь-який додаток з будь-якими даними з будь-якого сховища, за будь-якого протоколу і в будь-якій мережі.

## 4.2 Налаштування прав доступу

Згідно з предметною областю, дві основних сутності будуть користуватися базою даних: апаратне управління та аналітик, який буде проводити облік робочого часу за електронними картками.

Для першої сутності необхідним є можливість створювати нові записи у базі даних, а для дружньої достатньо доступ тільки для читання.

Для цього використаємо наступні скрипти:

CREATE LOGIN Analytic WITH PASSWORD=' Analytic ';

USE EmployeeTimeControlSystem;

CREATE USER Analytic FOR LOGIN Analytic;

EXECUTE sp\_addrolemember db\_datareader, " Analytic ";

CREATE LOGIN Apparat WITH PASSWORD=' Apparat ';

USE EmployeeTimeControlSystem;

CREATE USER Apparat FOR LOGIN Apparat;

EXECUTE sp\_addrolemember db\_ db\_datawriter, " Analytic ";

ВИСНОВКИ

Отже, у ході виконання даної курсової роботи, буда спроектована та створена база даних для обліку робочого часу за електронними картками. Описані особливості предметної області, створені запити до бази даних, які нададуть можливість легко провести обліку.

База даних має надає дані для повного аналізу робочого часу персоналу, контролю доступу карток до приміщень, ідентифікації персони на основі даних картки.

Для фізичної реалізації бази даних була вибрана система управління базою даних MS SQL SERVER – на основі порівняння з іншими СУБД. Описано основні етапи налаштування сервера та надання прав користувачам. Взаємодія між апаратною частиною, користувачем та базою даних була полегшена за допомогою Web API, який в свою чергу взаємодіє із базою даних за допомогою Entity Framework.

У ході роботи надано сніпети із вихідного кода, надано SQL скрипт основних запитів до бази даних.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології. Процеси життєвого циклу програмного забезпечення (ISO/IEC 12207:1995): ДСТУ 3918-1999. – [Чинний від 2000-01-01]. – К.: Держстандарт України, 2000. – 50 c. – (Національний стандарт України).
2. Інформаційні технології. Основні напрямки оцінювання та відбору CASE-інструментів (ISO/IEC 14102:1995) – ДСТУ 3919-1999 [Чинний від 2000-01-01]. – К.: Держстандарт України, 2000. – 470 c. – (Національний стандарт України).
3. Вендров А. М. CASE-технологии – современные методы и средства проектирования информационных систем / Вендров А. М. – М.: Финансы и статистика, 1998 – 171 с.
4. Кальянов Г. Н. CASE. Структурный системный анализ (автоматизация и применение) / Кальянов Г. Н. – М.: Лори, 1996. – 242 с.
5. Марка Д., Методология структурного анализа и проектирования / Д. Марка, К. МакГоуэн. – М.: МетаТехнология, 1993. – 240 с.
6. Черемных С. В. Структурный анализ систем: IDEF-технологии / Черемных С. В., Семенов И. О., Ручки В. С. – М.: Финансы и статистика, 2003. – 208 с.
7. Кватрани Терри. Визуальное моделирование с помощью Rational Rose 2002 и UML / Кватрани Т.; пер. с англ. – М.: Издательский дом „Вильямс”, 2003. – 192 с.
8. Новоженов Ю. В. Объектно-ориентированные технологии разработки сложных программных систем / Ю. В. Новоженов. – М.: Наука, 1996 – 356 с.
9. Фаулер М, Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования / М. Фаулер, К. Скотт. – М: «Мир», 1999 – 191 с.
10. Дейт К. Введение в системы баз данных / Дейт К. – [8-е изд. ] – СПб.: Вильямс, 2005 – 1328 с.
11. Праг К. Н. Access 2002 / К. Н. Праг, М. Р. Ирвин; пер. с англ. – М. Диалектика, 2003. – 1216 с. – (Серия “Библия пользователя”).
12. Фаронов В. Программирование баз данных Delphi 7. Учебный курс / Фаронов В.В. – СПб: Питер, 2005. – 459 с.
13. Чкалов А. П. Базы данных: от проектирования до разработки приложений / Чкалов А. П. – СПб.: БХВ-Петербург, 2003. – 384 с.
14. Искусственный интеллект: справочник в 3 кн. / [под ред. Э. В. Попова, Д. А. Поспелова, В. Н. Захарова, В. Ф. Хорошевского] – М.:. ‑ Радио и связь, 1990. ‑

Кн. 2: Модели и методы / [под ред. Д. А. Поспелова]. – 1990. – 304 с.

Кн. 3: Программные и аппаратные средства: справочник / [под ред. В. Н. Захарова, В. Ф. Хорошевского] – 1990. – 368 с.

1. Глущенко В. В. Исследование множеств и разработка одного типа представления и метода планирования испытания сложных технических систем / В. В. Глущенко // Кибирнетика и системный анализ. – 1992. – № 2. – С. 27-28
2. Сугоняк І. І. Структура та сценарії роботи системи підтримки прийняття рішень з оптимального керування інноваційними процесами підприємств / І. І. Сугоняк // Наукові проблеми модернізації та застосування інформаційних систем: ХVII наук. – тех. конф: 24–25 квіт. 2008 р.: тези доп. Ч. І – Житомир, 2008. – С. 86 ‑ 87.
3. Кириллов В. В. Структуризованный язык запросов (SQL): учебн. пособ.: [Електронный ресурс] / В. В. Кириллов, Г. Ю. Громов. – СПб: [Санкт-Петерб. госуд. техн. универ.](http://www.ifmo.ru/) , [каф. выч. техники](http://www.cs.ifmo.ru), 1998. – Режим доступа к пособию:

http://www.citforum.ru/database/sql\_kg/.

1. Иванова В. Б. Алгоритм принятия решения о внедрении инновационного проекта [Электронный ресурс] / В. Б. Иванова // Вісн. міжнар. слов`ян. ун-ту. Серія: Екон. науки. – 2006. – 9, № 1. – С. 20-24 – Режим доступу до статті: [http://www.nbuv.gov.ua/articles/2006/06ivboip. html](http://www.nbuv.gov.ua/articles/2006/06ivboip.%20html)
2. Шилдт Г. C# 4.0 Полное руководство [Текст] / Г. Шилдт. — М.: Издательский

дом «Вильямс», 2011. — 1056 с.

1. Троелсен Е. Язык программирования C# 5.0 и платформа .NET 4.5 [Текст] / Е. Троелсен – Москва: Издательский дом “Вильямс”, 2013. — 810 с.
2. Сайт Microsoft Developer Network [Електронний ресурс]. — Режим доступу: <https://msdn.microsoft.com/en-us/library/system.runtime.serialization>
3. Черемних С.В Моделирование и анализ систем [Текст] / С.В. Черемних – Москва: Финансы и статистика, 2006. – 192 с.
4. Сайт Microsoft Docs [Електронний ресурс]. — Режим доступу: <https://docs.microsoft.com/en-us/ef>
5. Сайт веб-API ASP.NET [Електронний ресурс]. — Режим доступу: <https://msdn.microsoft.com/ru-ru/library/hh833994(v=vs.108).aspx>
6. Дейт К. Дж. Введение в системы баз данных / [Дейт К. Дж] 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с: ил.
7. Харрингтон Дж. Л. Проектирование реляционных баз данных/ [Харрингтон Дж. Л.] – М.: Издательство "Лори", 2006 – 232 с.

# ДОДАТОК А

Діаграми бази даних

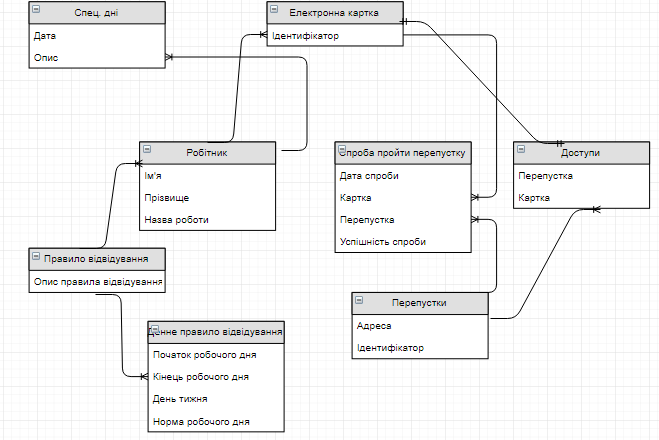


Рисунок А.1 – ER-діаграма бази даних

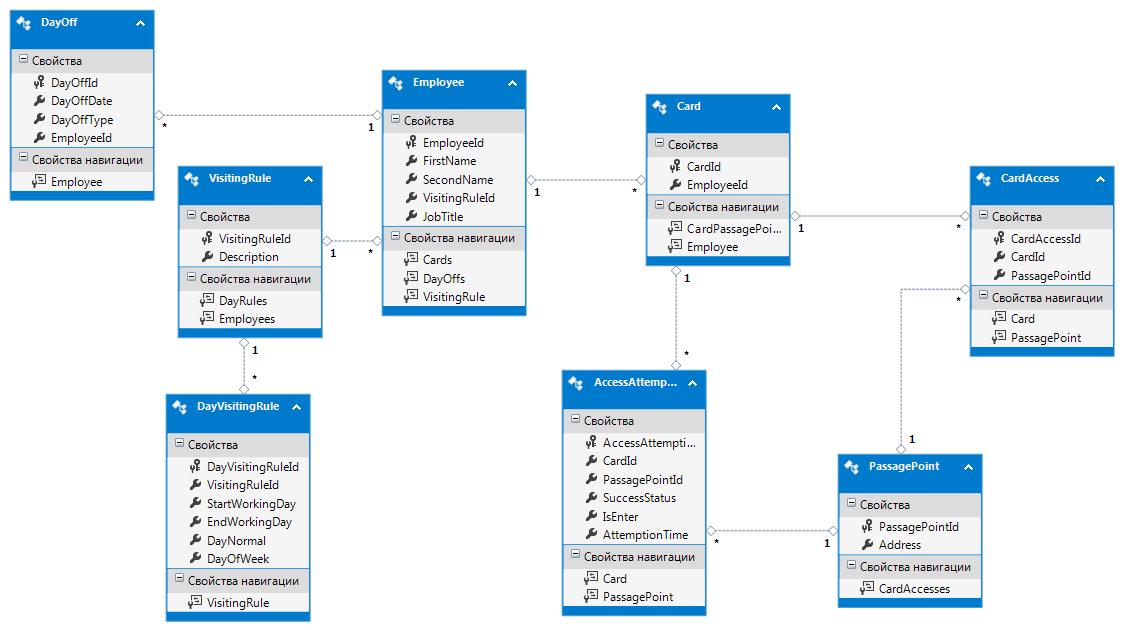


Рисунок А.2 – Фізична діаграма бази даних