Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Прикладна математика та менеджмент

(повна назва)

Кафедра Інформатика

(повна назва)

Напрям 6.040302 – «Інформатика»

(код, назва)

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до курсової роботи

з дисципліни “Бази даних та інформаційні системи”

за темою

Розробка інформаційної системи „Готель” з використанням СУБД MS SQL Server та мови запитів SQL

Виконав: Перевірив:

студент гр. ІНФ-12-1    доц.О.В. Яковлева

              (шифр групи) (посада, ініціали, прізвище)

  О.А. Астаппєв

                     (ініціали, прізвище) (підпис)

2014 р.

Харківський національний університет радіоелектроніки

(найменування вищого учбового закладу)

Кафедра Інформатики

Дисципліна Бази даних та інформаційні системи

Спеціальність Інформатика

Курс 2 Группа ИНФ-12-1 Семестр 4

ЗАВДАННЯ

на курсовий проект (роботу) студента

Астаппєв Олег Олександрович

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) “Розробка інформаційної системи „Готель” з використанням СУБД MІ SQL Server та мови запитів SQL”

2. Строк здачі студентом закінченого проекту (роботи) 10.06.2014

3. Вихідні данні проекту (роботи) опис предметної області «Готель»

мова запитів SQL

мова програмування C#

CASE-засіб візуальнго проектування ERwin

Microsoft SQL Server 2008

SQL Server Management Studio

середа розробки Microsoft Visual Studio 2013

методика проектування РМД

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1 Принципи проектування та реалізації реляційних баз даних

2 Проектування бази даних предметної області «Готель»

3 Розробка та реалізація функціональності інформаційної системи для предметної області «Готель»

4 Тестування інформаційної системи

Висновки

5. Перелік графічного матеріала (з точним зазначенням обов’язкових креслень).

Рисунок 1.1, 1.2, 1.3 – моделі інформаційних систем.

Рисунок 2.1, 2.2 – концептуальна та логічна модель по Чену.

Рисунок 2.3, 2.4 – концептуальна та логічна модель у ERwin.

Рисунок 2.5 – схема даних у SQL Server Management Studio.

Рисунки 3.1 − 3.14 – ієрархія форм та форми програми.

Рисунки 4.1 – 4.6 – тестування програми.

Рисунки Б.1 – Б.12 – відношення екземпляри розробленої БД

6. Дата видачі завдання 3.02.14 .

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Найменування етапів курсового проекту | Термін виконання етапів проекту | Примітки |
| 1 | Отримання завдання на курсове проектування. | 3.02.2014 | Виконано |
| 2 | Аналіз завдання, робота з літературою. | 4.02.2014 – 23.02.2014 | Виконано |
| 3 | Розробка програмної системи. | 24.02.2014 – 16.04.2014 | Виконано |
| 4 | Тестування програмної системи. | 17.04.2014 – 06.05.2014 | Виконано |
| 5 | Оформлення пояснювальної записки. | 07.05.2014 – 27.05.2014 | Виконано |
| 6 | Оформлення програмної документації. | 28.05.2014 – 4.06.2014 | Виконано |
| 7 | Оформлення графічної частини. | 5.06.2014 – 10.06.2014 | Виконано |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (підпис)

РЕФЕРАТ

Курсова робота: 60 с., 40 рис., 2 додатка, 7 джерел.

В курсовій роботі вирішується завдання створення інформаційної системи для предметної області «Готель». Основне завдання системи автоматизувати та впорядкувати збереження інформації про клієнтів, створити систему бронювання та управління рахунками. Середа застосування системи — готельний бізнес.

Під час розробки інформаційної системи були використані: теорія проектування реляційних баз даних на базі будування ER- моделі; основні принципи нормалізації реляційних баз даних; мова запитів SQL; CASE- засіб візуального проектування даних ERwin, середовище розробки програмних систем MS Visual Studio, мова програмування C#.

ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, МОВА SQL, КОНЦЕПТУАЛЬНА, ФІЗИЧНА ТА ЛОГІЧНА МОДЕЛЬ, СИНТАКСИС ЧЕНА, НОРМАЛІЗАЦІЯ ДАНИХ.

РЕФЕРАТ

Курсовая работа: 60 с., 40 рис., 2 приложения, 7 источников.

В курсовой работе решается задача создания информационной системы для предметной области «Отель». Основная задача системы автоматизировать и упорядочить хранение информации о клиентах, создать систему бронирования и управления счетами. Среда применения системы - гостиничный бизнес.

При разработке информационной системы были использованы: теория проектирования реляционных баз данных на базе построения ER-модели; основные принципы нормализации реляционных баз данных; язык запросов SQL; CASE-средство визуального проектирования данных ERwin, среда разработки программных систем MS Visual Studio, язык программирования C#.

ИНФОРМАЦИОННАЯ СИСТЕМА, БАЗА ДАННЫХ, ЯЗЫК SQL, КОНЦЕПТУАЛЬНАЯ, ФИЗИЧЕСКАЯ И ЛОГИЧЕСКАЯ МОДЕЛЬ, СИНТАКСИС ЧЕНА, НОРМАЛИЗАЦИЯ ДАННЫХ.

ABSTRACT

Term paper: 60 p., 40 img., 2 appendices, 7 sources.

In this course work solving the problem of creating an information system for subject area "Hotel". The main task of this system is to automate and streamline a storage of clients information, create a system of booking and account management. Scope of the system is hotel business.

With creating this information system were using: the theory of projecting relational database on the base of construction the entity-relationship model; query language SQL ; CASE-mean of visual projecting the data of ERwin, development environment of the softwary system Visual Studio, programming language C#.

INFORMATION SYSTEM, DATABASE, QUERY LANGUAGE SQL, CONCEPTUAL, PHYSICAL AND LOGICAL MODELS, SYNOPSIS CHEN, DATA NORMALIZATION.

ЗМІСТ

Перелік посилань 8

Вступ 9

1 Принципи проектування та реалізації реляційних баз даних 10

1.1 Основні етапи життєвого циклу інформаційних систем 10

1.2 Основні властивості реляційних БД 13

1.3 Можливості мови SQL 15

1.4 CASE-засоби моделювання БД 18

1.5 Постановка завдання курсової роботи 20

2 Проектування бази даних предметної області «Готель» 23

2.1 Специфікація вимог 23

2.2 Розробка бізнес правил і глосарію БД 24

2.3 Розробка концептуальної моделі БД 26

2.4 Побудова та перевірка логічної моделі 29

2.5 Побудова моделі даних за допомогою ERwin 31

2.6 Фізичне проектування 33

3 Розробка та реалізація функціональності предметної області «Готель» 35

3.1 Дизайн ІС 35

3.2 Забезпечення необхідної функціональності ІС 42

Висновки 54

Перелік посилань 55

Додаток А – DDL запити на створення фізичної структіри БД 56

Додаток Б − Відносини екземпляри розробленої БД 58

# ПЕРЕЛІК ПОСИЛАНЬ

БД – база даних

БП – бізнес правила

ЖЦ – життєвий цикл

ЗК – зовнішній ключ

ІС – інформаційна система

ОС – операційна система

ПЗ – програмне забезпечення

ПК – первинний ключ

ПО – предметна область

СКБД – система керування базами даних

# ВСТУП

Бази даних виконують функцію систематизації знань. На основі цієї систематизації можуть створюватися нові дані. Будь-яка база даних служить для опису подій, що відбулися та були занесені до бази даних, на основі цих даних, що містить база даних в результаті їх систематизації та обробки створюємо нові дані.

База даних - набір об'єктів даних, тобто набір можливих понять або подій, що описуються базою даних, з можливістю пошуку цих об'єктів за ознаками та маніпулювання ними об’єднуючи та створюючи нові таблиці.

Згідно з завданням предметної області «Готель» для курсового проекту, в ІС необхідно розробити функціональність, що дозволяє вести облік гостей готелю, реєструвати клієнта, поселити клієнта до номера та забезпечити бронювання номерів.

Згодом, використовуючи ці знання ми отримуємо інформацію щодо зайнятості номерів, можливості їх бронювання та отримуємо інформацію щодо діяльності готелю.

Розроблена таким чином ІС може забезпечити автоматизацію безлічі функцій, виконуваних менеджером готелю, що за підсумком значно скоротить час, необхідний для пошуку інформації щодо гостей готелю, зайнятості номерів та наданих послуг.

Основною ціллю курсового проекту є закріплення, систематизація та поглиблення знань, отриманих під час вивчення дисципліни, а також розвиток практичних навичок з аналізу об’єктів дослідження, проектування баз даних, розробки та налагодження програмного забезпечення для організації роботи зі спроектованою базою даних.

# 1 ПРИНЦИПИ ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЇ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

### 1.1 Основні етапи життєвого циклу інформаційних систем

Життєвий цикл інформаційної системи – період часу, який починається з моменту прийняття рішення про необхідність створення інформаційної системи і закінчується в момент її повного вилучення з експлуатації [1].

Для кожного етапу визначаються склад і послідовність виконуваних робіт, отримані результати, методи і засоби, необхідні для виконання робіт, ролі та відповідальність учасників і т.д. Такий життєвий цикл інформаційної системи дозволяє спланувати та організувати процес колективної розробки і забезпечити управління цим процесом.

Повний життєвий цикл інформаційної системи включає в себе, як правило, стратегічне планування, аналіз, проектування, реалізацію, впровадження та експлуатацію. У загальному випадку життєвий цикл можна в свою чергу розбити на ряд стадій.

Етапи життєвого циклу – послідовні відрізки часу, упродовж якого виконуються роботи. Протягом етапу можуть виконуватися роботи, що належать до різних процесів. В основі діяльності із створення й використання автоматизованої інформаційної системи управління лежить поняття її життєвого циклу. Життєвий цикл є моделлю створення і використання інформаційної системи, що відображає різні її стани, починаючи з моменту виникнення необхідності і закінчуючи моментом його повного виходу з використання всіх, без винятку, користувачів.

Традиційно виділяють такі основні етапи життєвого циклу інформаційних систем: аналіз вимог, проектування, програмування (впровадження), тестування і налагодження, експлуатація і супровід.

Життєвий цикл утворюється відповідно до принципу низхідного проектування і зазвичай має ітераційний характер: реалізовані етапи, починаючи з найперших, циклічно повторюються відповідно до змін вимог і зовнішніх умов, введення обмежень тощо. На кожному етапі життєвого циклу породжується певний набір документів і технічних рішень, при цьому для кожного етапу початковими є документи і рішення, отримані на попередньому етапі. Кожний етап завершується верифікацією породжених документів і рішень з метою перевірки відповідності їх вихідним.

Існуючі моделі життєвого циклу визначають порядок виконання етапів у ході розробки, а також критерії переходу від етапу до етапу. Найбільше поширення отримали три моделі життєвого циклу.

Каскадна модель (рис. 1.1) передбачає послідовне виконання всіх етапів проекту в строго фіксованому порядку. Перехід на наступний етап означає повне завершення робіт на попередньому етапі.



Рисунок 1.1 – Каскадна модель

Поетапна модель з проміжним контролем (рис. 1.2) – ітераційна модель розробки з циклами зворотного зв’язку між етапами. Перевага такої моделі – в тому, що між етапні коригування забезпечують меншу трудомісткість порівняно з каскадною моделлю; з іншого боку, час життя кожного з етапів розтягується на весь період розробки.

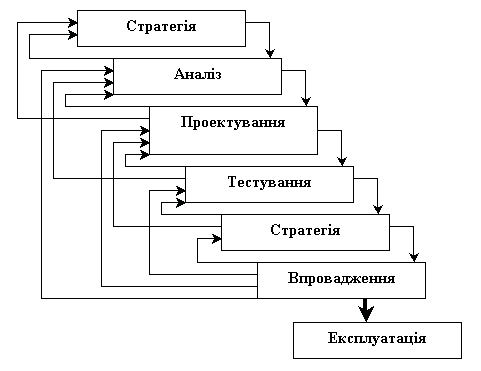


Рисунок 1.2 – Поетапна модель з проміжним контролем

Спіральна модель (рис. 1.3) загострює увагу на початкових етапах життєвого циклу: аналізі вимог, проектуванні специфікацій, попередньому й детальному проектуванні. На цих етапах перевіряється і обгрунтовується реалізовуваність технічних рішень створенням прототипів. Кожний виток спіралі відповідає поетапній моделі створення фрагмента або версії системи, на ньому уточняються цілі й характеристики проекту, визначається його якість, плануються роботи наступного витка спіралі. Таким чином поглиблюються і послідовно конкретизуються деталі проекту і в результаті обирається обґрунтований варіант, який доводиться до реалізації.

Фахівці відзначають такі переваги спіральної моделі: накопичення і повторне використання програмних засобів, моделей і прототипів; орієнтація на розвиток і модифікацію системи в ході її проектування; аналіз ризику і витрат у процесі проектування.

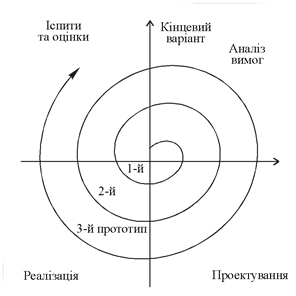


Рисунок 1.3 – Спіральна модель життєвого циклу

### 1.2 Основні властивості реляційних БД

Реляційною називається база даних, в якій всі дані, доступні користувачеві, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями. У кожної таблиці є своє унікальне ім'я.

Кожна таблиця складається із стовпців (їх називають полями або атрибутами) та рядків (їх називають записами або кортежами). Кожне поле запису містить одну характеристику об'єкта і має строго певний тип даних (наприклад, текстовий рядок, число, дата). Графічно зображено схему на рисунку 1.4.

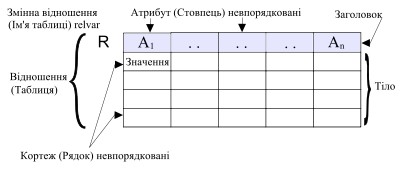


Рисунок 1.4 – Терміни реляційної моделі

Реляційна БД має такі властивості:

* Кожен елемент таблиці – один елемент даних
* Усі стовпці таблиці є однорідними, тобто мають однаковий тип
* Кожний стовпець (поле) має унікальне ім’я
* Однакові рядки в таблиці відсутні
* Порядок слідування рядків в таблиці може бути довільним

Кожна реляційна БД повинна мати потенційні ключі, один з яких відіграє роль первинного ключа.

Ключ – це набір стовпчиків, може складатися з одного стовпчика, чи охоплювати всі стовпчики таблиці. Ключ призначений для ідентифікації рядків таблиці. У теорії БД це єдиний спосіб послатися на рядок. Ключі бувають різні - потенційні, первинні, альтернативні, зовнішні, індексні, хеш-ключі, ключі сортування, вторинні ключі, ключі шифрування і розшифрування та ін. Але ми будемо розглядати тільки такі, що нам знадобиться в роботі над проектами [4].

Стовпець однієї таблиці, значення в якої збігається зі значенням стовпця, що є первинним ключем другої, називається зовнішнім ключем. Таблиця може містити кілька зовнішніх ключів. Пара первинний ключ – зовнішній ключ створюють відносини предок - нащадок між таблицями, що містять їх.

Перевага реляційної моделі даних полягає в простоті, зрозумілості та зручність фізичної реалізації на ЕОМ. Саме простота і зрозумілість для користувача є основною причиною їх широкого використання. Проблеми ж ефективності обробки даних цього типу виявилися технічно цілком вирішальні.

### 1.3 Можливості мови SQL

Мова SQL, призначена для взаємодії з базами даних, з'явилася в середині 70-х рр. (перші публікації датуються 1974 р.) і була розроблена у компанії IBM у рамках проекту експериментальної реляційної СУБД System R. Вихідна назва мови SEQUEL (Structured English Query Language) тільки частково відбиває суть цієї мови. Звичайно, мова була орієнтований головним чином на зручну й зрозумілу користувачам формулювання запитів до реляційних БД [7].

Спочатку, SQL був основним способом роботи користувача з базою даних і дозволяв виконувати наступний набір операцій:

* Створення в базі даних нової таблиці;
* Додавання в таблицю нових записів;
* Зміна записів;
* Видалення записів;
* Вибірка записів з однієї або декількох таблиць (відповідно до заданих умовою);
* Зміна структури таблиці.

З часом, SQL ставав все більш складним – збагатився новими конструкціями, забезпечив можливість опису та управління новими збереженими об'єктами (наприклад, індекси, подання, тригери і процедури) – і став набувати рис, властиві мовам програмування.

В даний час мова SQL підтримується багатьма десятками СУБД різних типів, розроблених для найрізноманітніших обчислювальних платформ.

У SQL існує кілька основних категорій (типів) команд, що реалізують виконання різних функцій. Серед таких функцій – побудова об'єктів бази даних, управління об'єктами, поповнення таблиць бази даних новими даними, оновлення даних, вже наявних в таблицях, виконання запитів, управління доступом користувачів до бази даних, а також здійснення загального адміністрування бази даних [4].

Такими категоріями (типами) команд є:

* DDL (Data Definition Language — мова визначення даних);
* DML (Data Manipulation Language — мова маніпуляцій даними);
* DQL (Data Query Language — мова запитів до даних);
* DCL (Data Control Language — мова керування даними);
* команди адміністрування даних;
* команди керування транзакціями.

Мова SQL являє собою сукупність

* операторів;
* інструкцій;
* та обчислюємих функцій.

Мова визначення даних (DDL) є частиною SQL, що дає користувачеві можливість створювати різні об'єкти бази даних і перевизначати їх структуру, наприклад, створювати або видаляти таблиці [3].

Серед основних команд DDL наступні команди:

1. CREATE TABLE
2. ALTER TABLE
3. DROP TABLE
4. CREATE INDEX
5. ALTER INDEX
6. DROP INDEX

Відбір даних (DQL). Хоча цей розділ мови представлений тільки однією командою, для користувача реляційної бази даних мова запитів до даних (DQL) є найголовнішою частиною SQL. Цією командою є команда

1. SELECT

Ця команда, що має безліч опцій і необов'язкових параметрів, використовується для побудови запитів до реляційних баз даних. З її допомогою можна конструювати запити будь-якої складності - від найзагальніших до дуже спеціальних і від найпростіших до неймовірно складних [3].

Запит - це вимога на отримання інформації з бази даних.

Мова маніпуляцій даними (DML) є частиною SQL, що дає користувачеві можливість маніпулювати даними всередині об'єктів реляційної бази даних.

Ось три основні команди DML:

1. INSERT
2. UPDATE
3. DELETE

Команди керуванням даними в SQL дозволяють здійснювати контроль над можливістю доступу до даних усередині бази даних. Команди DCL зазвичай використовуються для створення об'єктів, що відносяться до управління доступом користувачів до бази даних, а також для призначення користувачам підходящих рівнів привілеїв доступу [3].

Основні команди DCL:

1. ALTER PASSWORD
2. GRANT
3. REVOKE
4. CREATE SYNONYM

### 1.4 CASE-засоби моделювання БД

Для підвищення швидкості розробки програмного забезпечення широко використовуються CASE-засоби на етапах програмування (створення БД), інтеграції та тестування (рис. 1.5).

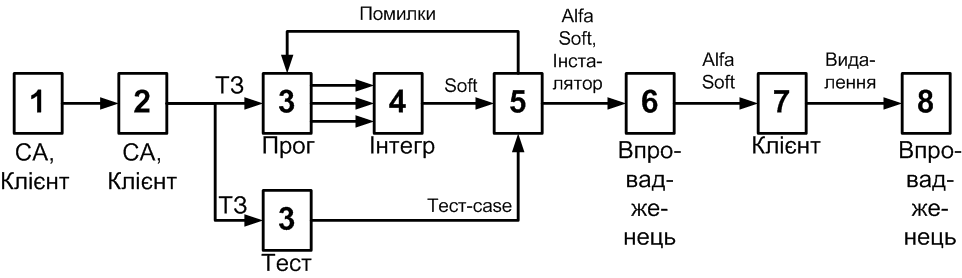


Рисунок 1.5. – Етапи створення програмного забезпечення

Варіанти застосування сучасних CASE-засобів:

1. перепроектування бізнес-процесу (реінжиніринг);
2. системний аналіз і проектування ІС:

* функціональне моделювання – відображає функції з DFD;
* інформаційне моделювання – показує які об'єкти даних використовуються в технології;
* подійне моделювання - показує на що ми реагуємо.

CASE-засоби дозволяють побудувати модель бізнес-процесу.

Другий аспект - системний аналіз і проектування: проектування потрібно починати з аналізу існуючої технології та reengineering-а. Існує багато програмних продуктів (CASE-засобів), це не означає, що всі вони конкурентно-здатні. Широко використовуються CASE-системи:

* PowerDesigner (Sybase);
* CA ERwin Data Modeler Erwin (Logic Works);
* Oracle Designer (Oracle Corp);
* ER/Studio (Embarcadero);
* System Architect (Popkin Software);
* Visible Analyst (Visible Systems);
* Visio Enterprise (Microsoft).

Сімейство програмних продуктів Powersoft PowerDesigner є сучасним засобом моделювання і побудови баз даних і додатків в архітектурі клієнт-сервер, який можуть використовувати в своїй роботі аналітики, проектувальники, розробники додатків і баз даних, а також адміністратори баз даних. Модульна структура PowerDesigner дозволяє організаціям вибрати конфігурацію, що відповідає саме їхнім вимогам і розмірам розроблювального проекту. Широкі можливості по аналізу і проектуванню інформаційних систем забезпечують гнучкий структурний підхід, що підвищує ефективність роботи створюваних баз даних і додатків без необхідності слідування вибраної одного разу методології.

CA ERwin Data Modeler – CASE-засіб для проектування та документування баз даних, який дозволяє створювати, документувати і супроводжувати бази даних, сховища і вітрини даних. Моделі даних допомагають візуалізувати структуру даних, забезпечуючи ефективний процес організації, управління і адміністрування таких аспектів діяльності підприємства, як рівень складності даних, технологій баз даних та середовища розгортання.

CA ERwin Data Modeler (ERwin) призначений для всіх компаній, що розробляють і використовують бази даних, для адміністраторів баз даних, системних аналітиків, проектувальників баз даних, розробників, керівників проектів. CA ERwin Data Modeler дозволяє управляти даними в процесі корпоративних змін, а також в умовах, коли технології стрімко змінюються.

ERwin дозволяє наочно відображати складні структури даних. Зручне у використанні графічне середовище ERwin Data Modeler спрощує розробку бази даних і автоматизує множину трудомістких завдань, зменшуючи терміни створення високоякісних і високопродуктивних транзакційних баз даних і сховищ даних. Дане рішення покращує комунікацію організації, забезпечуючи спільну роботу адміністраторів і розробників баз даних, багаторазове використання моделі, а також наочне уявлення комплексних активів даних в зручному для розуміння і обслуговування форматі.

Oracle Designer являє собою універсальний CASE-засіб, що дозволяє моделювати бізнес-процеси, створювати діаграми потоків даних і функціональні моделі. Засіб проектування даних і створення ER-діаграм є лише однією зі складових частин цього досить складного продукту і надає можливість зберігати створені моделі даних і описані бізнес-правила в призначеному для цього репозитарії.

Oracle Designer призначений для використання головним чином з Oracle 11, підтримує всі особливості даної СКБД, включаючи об'єктні типи даних, так само як і специфічні особливості фізичної реалізації бази даних Oracle. Для Oracle 11 і Oracle 12 цей CASE-засіб дозволяє створити визначення ролей, згенерувати тригери, що реалізують бізнес-логіку, яка описана в моделях, використовуваних при генерації бази даних, а також згенерувати об'єкти для розподілених баз даних. Крім того, за допомогою Oracle Designer можна створювати фізичні моделі та здійснювати зворотне проектування і для інших СКБД.

Інші засоби мают схожий функціонал та характер дії.

### 1.5 Постановка завдання курсової роботи

Метою курсової роботи є створення ІС для ПО «Готель».

В якості ПО курсової роботи обрана ІС «Готель». Нижче наведені опис даної ПО, користувальницькі вимоги до даних і функціональності ІС.

Готель – будинок із кімнатами для короткочасного проживання приїжджих. [https://uk.wikipedia.org/wiki/Готель]

Передбачимо що в нашій ІС готель має номери з різним рівнем сервісу, комфортності і відповідно, вартості. Однією з характеристик номера є його тип. Існують такі типи номерів: люкс - багатокімнатний номер з високим рівнем сервісу, комфортності та обслуговування; напівлюкс - номер менший, ніж люкс, площі, але з достатнім рівнем сервісу і комфортності; стандартні одномісні, двомісні та багатомісні номери з мінімальним рівнем сервісу; та дешеві номери типу хостела з мінімальним рівнем сервісу та комфорту.

Вартість для номерів типу люкс і напівлюкс встановлюється як вартість всього номера (за добу), незалежно від кількості проживаючих у номері. Вартість проживання в інших номерах встановлюється для однієї людини (за добу). Номери та місця в номерах можуть бронюватися.

Всі додаткові послуги такі як їжа, напої, обслуговування та інші записуються на поточний рахунок клієнта.

Для всіх громадян що розміщуються в готелі при вселенні створюється окремий запис до БД з особистою інформацією наданою клієнтом. Крім того, для розрахунків з клієнтами адміністрація готелю заводить картки реєстрації, які містять дані про розміщення клієнта, наданих послугах і всіх розрахунках з ним. Для номерів типу люкс і напівлюкс особиста інформаціє зберігається лише для одного клієнта, що оплачує номер, також на нього реєструється номер.

При вибутті клієнта дані про нього зберігаються в архіві.

Якщо клієнт приїжджає до одного готелю другий раз, протягом останніх півроку вартість номера зменшується на 10%.

ІС повинна мати систему бронювання номерів, за їх типом. Відстеження бронювання при заселенні клієнтів. При бронюванні користувач повинен надати особисту інформацію, та інформацію щодо його візиту (дата прибуття, дата вибуття, тип номера).

При бажанні клієнта продовжити перебування в готелі по вичерпанню передбачуваної дати відбуття, він повинен розрахуватися за поточний номер та зареєструватися по новій.

Створювана інформаційна система призначена для адміністрації готелю, яка на підставі інформації про номери займається розміщенням клієнтів відповідно до їх запитами. При вибутті клієнта інформація про клієнта повинна видалятися з робочих таблиць та поміщатися в архівну таблицю зберігаючи мінімум даних про візит. Особиста інформація клієнта залишається в ІС на випадок повторного візиту.

Для досягнення мети курсової роботи необхідно вирішити наступні завдання:

1. розробити ІС для ПО «Готель»:
2. розробити бізнес-правила і глосарій;
3. розробити ER-модель і відобразити її за допомогою ER-діаграми в синтаксисі Чена;
4. розробити модель даних за допомогою CASE-засоби візуального проектування даних ERwin;
5. сформувати структуру БД в СУБД;
6. розробити дизайн ІС;
7. розробити програмний код для забезпечення необхідної функціональності ІС;
8. провести тестування ІС;
9. відобразити процес і результати розробки ІС в пояснювальній записці до курсової роботи.

Для розробки ІС необхідно використовувати такі інструменти:

* ERwin - CASE-засіб проектування БД;
* MS SQL Server - СУБД;
* MS Visual Studio, C # - середа і мова розробки програмного забезпечення.

# 2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ПРЕДМЕТНОЇ ОБЛАСТІ «ГОТЕЛЬ»

### 2.1 Специфікація вимог

ІС «Готель» повинна забезпечувати зберігання та редагування наступної інформації:

* дані про клієнтів (Номер клієнта, прізвище, ім'я, по-батькові, стать, день народження, документ що засвідчує особу та його номер, телефон);
* дані про номери (Номер кімнати, тип номера, назва номера, поверх будівлі, телефон (якщо є), вартість, кількість місць);
* дані про бронювання (Номер клієнта, дата початку, дата закінчення);
* дані про реєстрацію (Номер реєстрації, номер клієнта, номер кімнати, дата заїзду, дата виїзду, вартість);
* дані про додаткові послуги (Номер реєстрації, назва послуги, вартість, дата надання);
* дані в архіві (Номер клієнта, номер кімнати, дата заселення, дата виселення, вартість послуг);

Система повинна забезпечувати наступну функціональність:

* необхідно забезпечити виведення поточних мешканців готелю. Передумовою для цього є створення кімнати, створення клієнту та реєстрація клієнта в номеру.
* необхідно забезпечити можливість бронювання номеру на визначений термін;
* перегляд інформації по готелю;
* забезпечити зберігання даних про надані послуги для поточного мешканця готелю;
* забезпечити розрахунок вартості враховуючи бронювання, надані послуги та можливі знижки;
* формувати список клієнтів які повинні сьогодні виселитися;
* створення та використання шаблонів послуг;
* Також можуть вирішуватися завдання статистичної обробки даних. Наприклад, підрахунок кількості мешканців і доходів готелю.

Користувачами даної інформаційної системи є:

* адміністратори готелю;
* ресепшен готелю;

### 2.2 Розробка бізнес правил і глосарію БД

В результаті аналізу функціональності інформаційної системи були розроблені наступні бізнес правила:

1. Клієнт має один документ, один тип документа може бути у багатьох клієнтів.
2. У різних типів документів, є вірогідність повтору номера.
3. Кожна кімната в готелю має унікальний номер, не залежно від її типу.
4. Кожна кімната має тип, кімнат одного типу може бути декілька.
5. Номер телефона в кімнаті унікальний, є кімнати без телефонів.
6. Лише кімнати типу «Люкс» та «Полулюкс» мають тарифи за номер, інші кімнати сплачуються за місце в номері (ліжко).
7. Особиста інформація про клієнта створюється під час його першої взаємодії з системою (бронювання чи реєстрація) та зберігається.
8. При кожному заселенні створюється нова реєстраційна карта.
9. Якщо клієнт відвідує готель не вперше за останні пів року, йому надається знижка на вартість номера в розмірі 10%.
10. При заселені обов'язково сплачується перший день проживання.
11. Клієнт може відвідувати готель безліч раз.
12. Клієнт може жити в одній кімнаті декілька раз.
13. Бронювання номера відбувається за типом, сплачується під час заселення. Відміна бронювання безкоштовна.
14. В разі відсутності клієнта в зазначену дату заїзду (зазначаються під час бронювання), броня анулюється в наступний день.
15. Вартість бронювання номера не залежно від кількості днів становить 30% вартості одного дня проживання.
16. Додаткові послуги записуються надані клієнту під час його перебування в готелі записуються на номер його реєстрації, їх загальна сума додається до загальної суми в реєстраційній карті.
17. Надану послугу неможливо видалити.
18. Після виселення клієнту, інформація щодо візиту в готель видаляється з основних таблиць та додається до архіву.
19. Під час виселення з номеру клієнт зобов'язується сплатить за всі дні проживання та нараховані рахунки за послуги.
20. Клієнт є однієї статі, клієнтів однієї статі може бути безліч.
21. Клієнт може бути створений менеджером готелю, або власноруч користувачем. У випадку наявності інтернет-ресурсу що дозволяє клієнтам бронювати номери в готелі.

Під час подальшої розробки інформаційної системи бізнес правила можуть уточнюватися, але основна логіка розроблювального додатка, представлена ​​бізнес правилами, зазначеними вище, залишиться незмінною.

Глосарій для ПО «Готель» зазначена нижче:

Реєстраційна карта – карта реєстрації клієнта в готелі, містить інформацію про поточний візит клієнта до готелю (відомості про поточне проживання клієнту).

Дата заселення – дата починаючи з якої клієнт фізично перебуває в готелі.

Дата виселення – дата до якої клієнт може перебувати в номері згідно з реєстраційною карткою.

Загальна сума – загальна вартість проживання клієнта, включає до себе: вартість бронювання, вартість проживання та суму всіх додаткових послуг що були надані клієнту.

Додаткові послуги – послуги надані клієнтові під час його проживання в готелі, такі як додаткове харчування, напої, телефонні розмови та інше. Їх додає менеджер готелю та/або автоматичні системи (наприклад, за користування телефоном, система білінгу може автоматично ставити рахунок).

### 2.3 Розробка концептуальної моделі БД

Концептуальна модель представляє собою опис основних таблиць і зв'язків між ними без урахування прийнятої моделі БД та синтаксису цільової СУБД. Її побудова є першим кроком при проектуванні БД після фіксації основних бізнес правил.

При розробці концептуальної моделі ІС «Готель», були визначені наступні таблиці:

* Ролі\_користувача(Номер\_ролі, Назва\_ролі) – сильна сутність з потенційним ключем Номер\_ролі.
* Користувач(Номер\_користувача, Номер\_ролі, Логін\_користувача, Пароль\_користувача, Дата\_реєстрайії, Дата\_останьої\_авторизації) – сильна сутність з потенційним ключем Номер\_користувача.
* Клієнт(Номер\_клієнта, Номер\_користувача, ПІБ, Стать, Дата народження, Тип документа, Серійний\_номер\_документа, Телефон, Адреса) – сильна сутність з потенційним ключем Номер\_клієнта.
* Тип\_номера(Номер\_типу, Назва\_типу) – сильна сутність з потенційним ключем Номер\_типу.
* Номер\_готелю(Номер\_кімнати, Номер\_типу, Назва\_номера, Поверх, Телефон, Кількість\_місць, Вартість\_номера) – сильна сутність з потенційним ключем Номер\_кімнати.
* Бронювання(Номер\_типу, Номер\_клієнта, Дата\_заїзду, Дата\_виїзду, Дата\_бронювання) – слаба сутність з потенційним ключем Номер\_типу, Номер\_клієнта, Дата\_заїзду.
* Архів(Номер\_типу, Номер\_клієнта, Дата\_заселення, Дата\_виселення, Повна\_вартість) – слаба сутність з потенційним ключем Номер\_типу, Номер\_клієнта, Дата\_заселення.
* Реєстраційна\_карта(Номер\_типу, Номер\_клієнта, Дата\_заселення, Дата\_виселення, Вартість\_бронювання, Кількість\_сплачених\_днів, Сплачена\_сума, Всього\_нараховано) – слаба сутність з потенційним ключем Номер\_типу, Номер\_клієнта, Дата\_заселення.
* Додаткові\_послуги(Номер\_послуги, Назва\_послуги, Вартість\_послуги, Номер\_типу, Номер\_клієнта, Дата\_заселення) – сильна сутність з потенційним Номер\_послуги.
* Шаблон\_послуги(Номер\_шаблона, Назва\_послуги, Вартість\_послуги) – сильна сутність з потенційним Номер\_ шаблона.

Визначено такі типи зв'язків між сутностями:

1. Зв'язок 1:1:
   * + Користувач – Клієнт.
2. Зв'язок 1:M:
   * + Ролі\_користувача – Користувач.
     + Клієнт – Бронювання.
     + Клієнт – Архів.
     + Клієнт – Реєстраційна\_карта.
     + Тип\_номера – Клієнт.
     + Тип\_номера – Номер\_готелю.
     + Номер\_готелю – Архів.
     + Номер\_готелю – Реєстраційна\_карта.
     + Реєстраційна\_карта – Додаткові\_послуги.

При побудові концептуальної моделі представленої на рисунку 2.1, використовувався наступний алгоритм: спочатку з вихідних даних були виділені сутності, які явно були властиві цій БД. Далі кожної сутності були приписані атрибути, явно входили до її складу без порушення бізнес правил, після чого в кожної сутності були встановлені потенційні ключі. Потім, слідуючи логікі встановленій бізнес правилами, були встановлені типи зв'язків між сутностями і їх кардинальність. Результатом стала схема, що має складові первинні ключі та зв'язки з атрибутами.

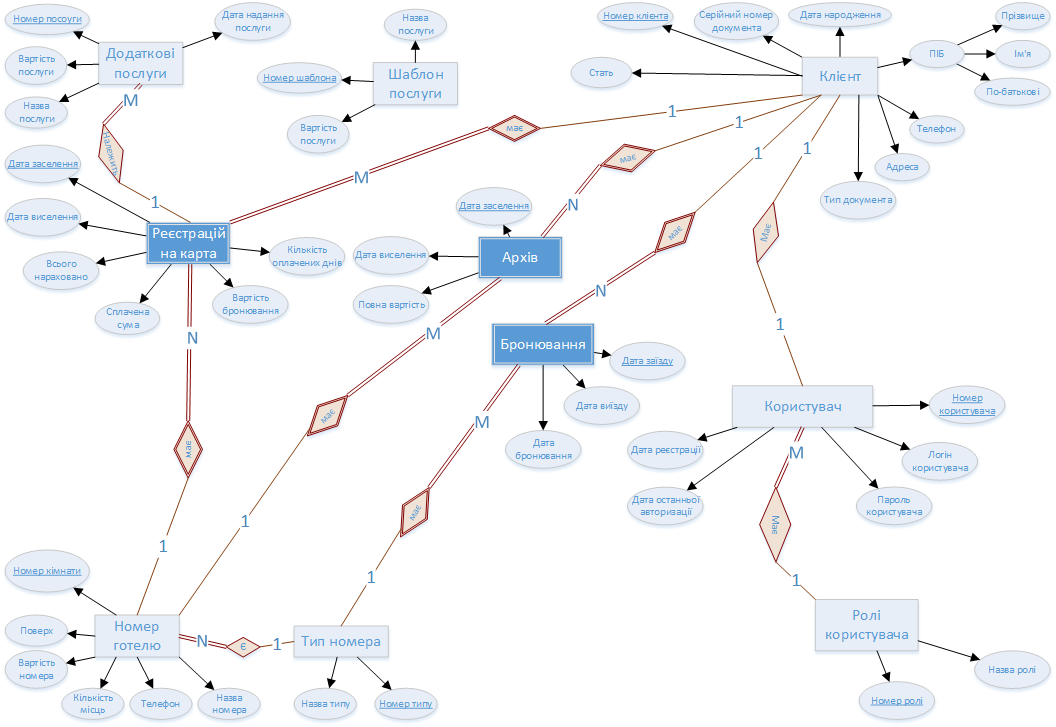


Рисунок 2.1 – Концептуальна модель в синтаксисі Чена

Проаналізувавши отримані дані, можна прийти до висновку, що отримана концептуальна модель готова до подальшого перетворенню в логічну.

### 2.4 Побудова та перевірка логічної моделі

Для перетворення концептуальної моделі в логічну, необхідно виконати ряд перетворень моделі даних.

Наявність зв'язку 1:1 пояснюється тим що, згідно бізнес-правилу х) Клієнт може бути створений менеджером готелю, або власноруч користувачем. У випадку наявності інтернет-ресурсу що дозволяє клієнтам бронювати номери в готелі. Тобто таблиця користувач використовується для авторизації в системі, клієнт може мати такий запис, але не обов'язково.

В нашій схемі є атрибути які можуть мати заздалегідь прогнозований перелік значень, винесемо їх до окремих сукупностей.

Також ми позбавилися складних атрибутів, замінивши їх на прості.

Окрім цього ми добавили атрибути що будуть виступати в якості первинних ключів, задля полегшення маніпулювання таблицями.

Основна ідея нормалізації полягає в тому, щоб кожен факт зберігався в одному місці, тобто щоб не було дублювання даних.

У результаті ми отримали відносини, в яких на перетині кожного рядка і кожного стовпця міститься тільки одне атомарне значення, дублюючі групи даних відсутні і в кожному відношенні визначений первинний ключ. Відповідно можемо зробити висновок, що відносини знаходиться в першій нормальній формі. Так як в моделі присутні відносини з складеним первинним ключем, необхідно перевірити їх на відсутність часткових функціональних залежностей. В результаті перевірки виявилося, що неключові атрибути перебувають у повній функціональної залежності від складеного первинного ключа згідно розробленої логічної моделі. Переходимо до перевірки на третю нормальну форму. Третя нормальна форма виключає наявність транзитивних залежностей усередині одного відношення. Оскільки при аналізі логічної моделі не було виявлено транзитивних залежностей, то розроблена модель знаходиться в третій нормальній формі, що є необхідним і достатнім для побудови по даній моделі реляційної БД. Таким чином, після проведених перетворень концептуальної моделі, можемо визначити наступний набір відносин логічної моделі:

* Ролі\_користувача(Номер\_ролі, Назва\_ролі).
* Користувач(Номер\_користувача, Номер\_ролі (ЗК), Логін\_користувача, Пароль\_користувача, Дата\_реєстрайії, Дата\_останьої\_авторизації).
* Тип\_документа(Номер\_документа, Назва\_документа).
* Стать\_клієнта(Номер\_статі, Назва\_статі).
* Клієнт(Номер\_клієнта, Номер\_користувача (ЗК), Прізвище, Ім'я, По-батолькові, Номер\_статі (ЗК), Дата народження, Номер\_документа (ЗК), Серійний\_номер\_документа, Телефон, Адреса).
* Тип\_номера(Номер\_типу, Назва\_типу).
* Номер\_готелю(Номер\_кімнати, Номер\_типу (ЗК), Назва\_номера, Поверх, Телефон, Кількість\_місць, Вартість номера).
* Бронювання(Номер\_броні, Номер\_типу (ЗК), Номер\_клієнта (ЗК), Дата\_заїзду, Дата\_виїзду, Дата\_бронювання).
* Архів(Номер\_архіва, Номер\_типу (ЗК), Номер\_клієнта (ЗК), Дата\_заселення, Дата\_виселення, Повна\_вартість).
* Реєстраційна\_карта(Номер\_реєстрації, Номер\_типу (ЗК), Номер\_клієнта (ЗК), Дата\_заселення, Дата\_виселення, Вартість\_бронювання, Кількість\_сплачених\_днів, Сплачена\_сума, Всього\_нараховано).
* Додаткові\_послуги(Номер\_послуги, Назва\_послуги, Вартість\_послуги, Номер\_реєстрації (ЗК)).
* Шаблон\_послуги(Номер\_шаблона, Назва\_послуги, Вартість\_послуги).

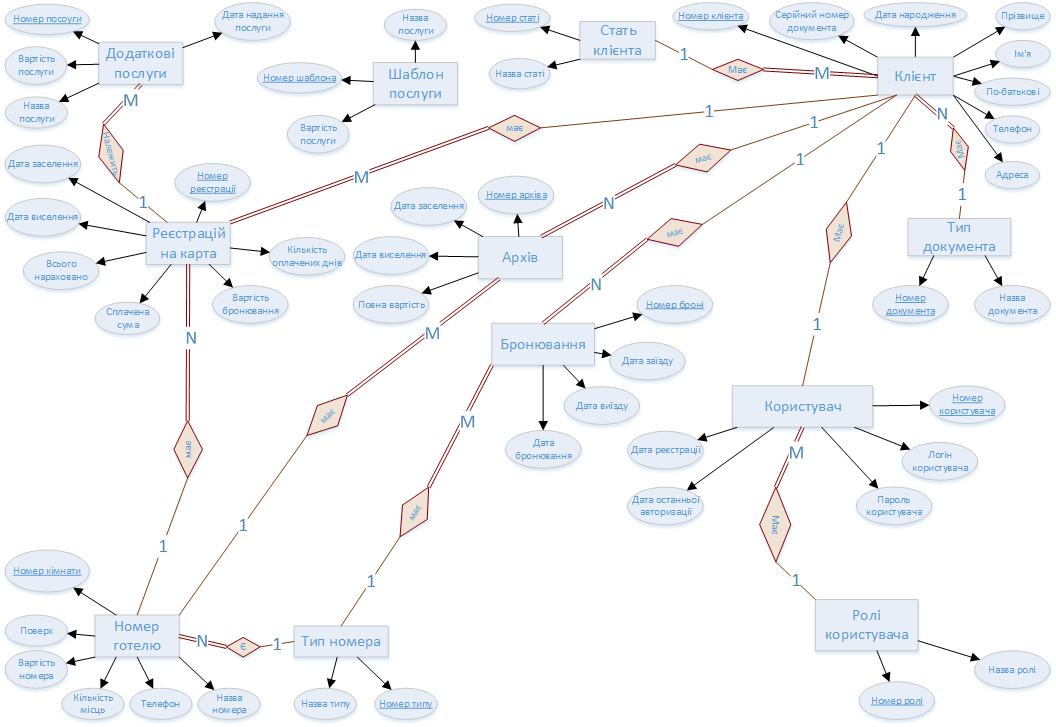


Рисунок 2.2 – Логічна модель в синтаксисі Чена

### 2.5 Побудова моделі даних за допомогою ERwin

У ERwin існують два рівня уявлення і моделювання - логічний і фізичний. Логічний рівень означає пряме відображення фактів з реального життя. Вони в своїй назві можуть містити будь-які символи, включно з розділовими. На логічному рівні не розглядається використання конкретної СУБД, не визначаються типи даних і не визначаються індекси для таблиць. Логічна модель даних представлена ​​на рисунку 2.3.

Цільова СУБД, імена об'єктів і типи даних, індекси складають другий (фізичний) рівень моделі ERwin. Фізична модель даних показана на рисунку 2.4. Фактично, вона в даному програмному продукті лише злегка спрощує наочний вид готової схеми, але при цьому зміни у функціональності програмного продукту, стосуються лише функцій, що дозволяють згенерувати створену модель в реальну БД.

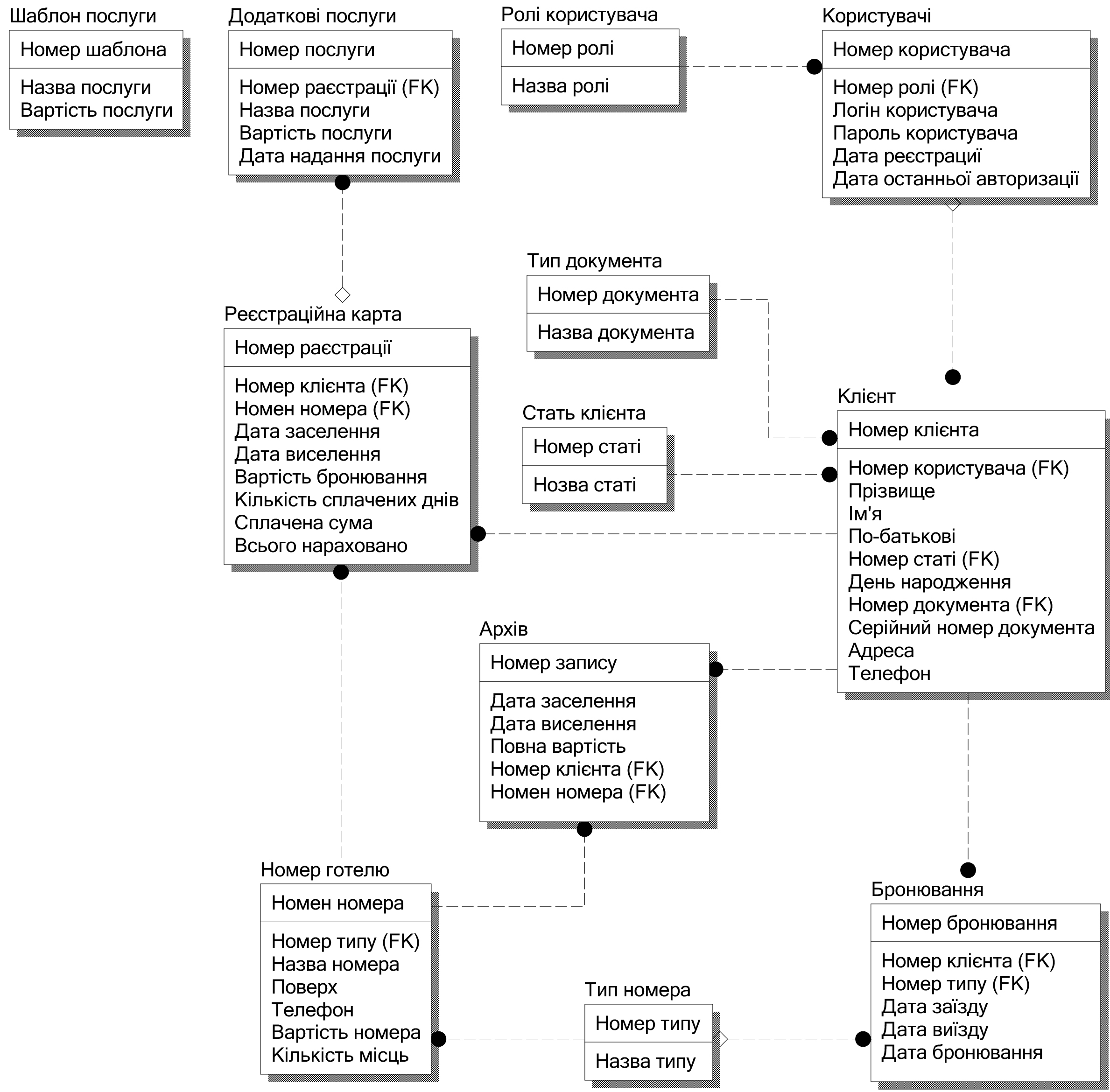


Рисунок 2.3 – Логічна модель в ERWin

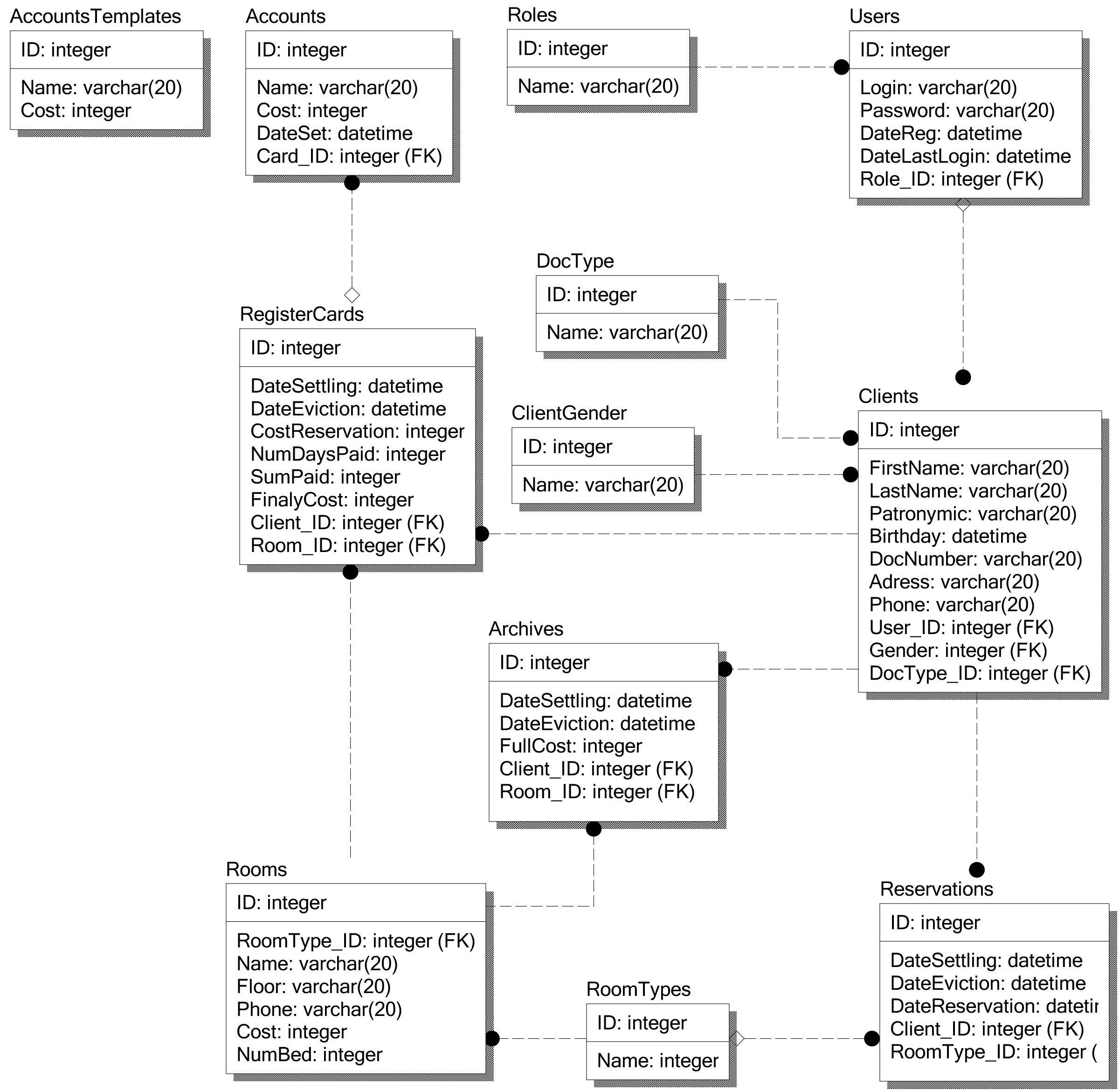


Рисунок 2.4 – Фізична модель в ERWin

### 2.6 Фізичне проектування

Відповідно до фізичної моделі даних, яка була представлена ​​в пункті 2.5 була створена БД на основі СУБД MS SQL Server. Схема даних, отримана в MS SQL Server, представлена ​​на рисунку 2.4.

При створенні БД, для текстових атрибутів було створено обмеження в розмірі та для деяких полів встановлено значення за замовчуванням.

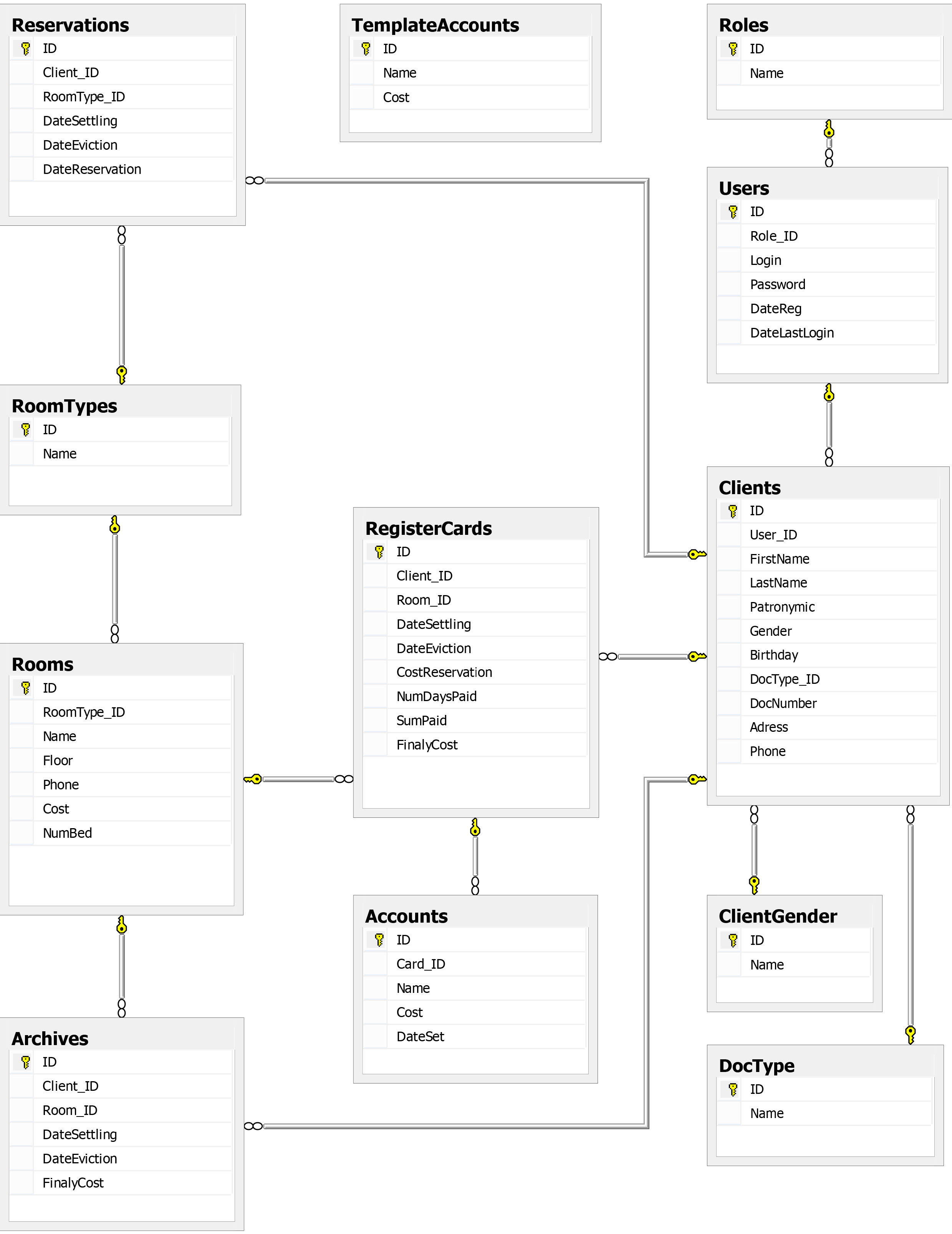


Рисунок 2.5 – Схема даних SQL Server

# 3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ФУНКЦІОНАЛЬНОСТІ предметної області «Готель»

### 3.1 Дизайн ІС

Під час розробки ІС «Готель» була створена структура меню, що дозволяє зорієнтуватися в програмі. Вона зображена на рисунку 3.1.

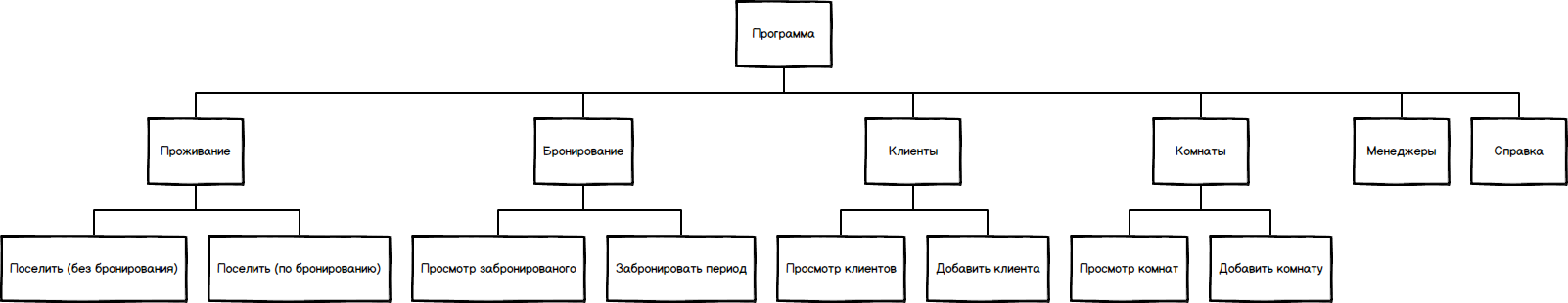


Рисунок 3.1 – Структура меню

Всього створено 15 інтерфейсів програми, кожен з яких виконує певний функціонал. Ознайомимося з ними ближче.

Під час запуску розробленого додатку відкривається вікно ауторизації, зображене на рисунку 3.2.

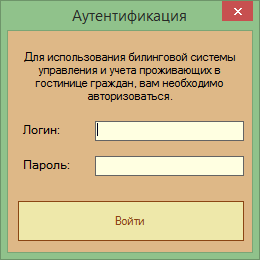


Рисунок 3.3 – Вікно авторизації

В залежності від рівня доступу, в програмі буде відмінність в функціоналу. Існує 3 рівня доступу: користувач – доступ до програми заборонений цей тип використовується клієнтами, менеджер – основний режим доступу та адміністратор – створення та видалення менеджерів, а також управління номерами готелю.

Головне вікно «Hotel Booking System» користувач відкриває під час запуску програми, пройшовши авторизацію. На цій формі відображені короткі статистичні дані про готель та список поточних проживаючих та елементи керування за допомогою яких можливо: виселити клієнта, додати рахунок за послугу, внести оплату.

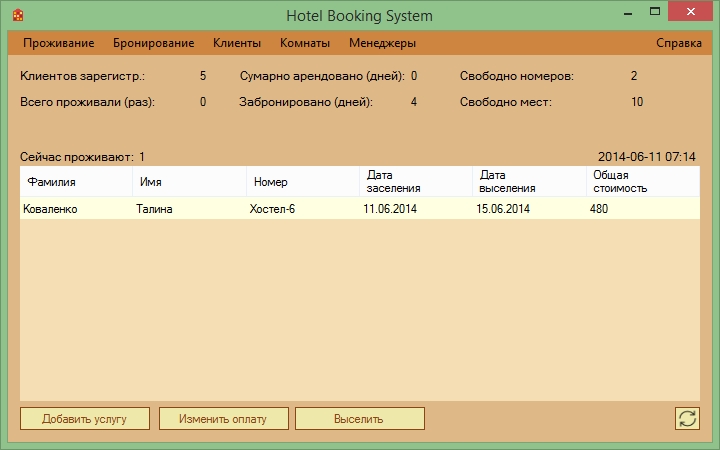


Рисунок 3.3 – Головне вікно «Hotel Booking System»

Напевно, найпопулярніший розділ програми «Проживание» складається з двух частин, в одній розглядується випадок якщо користувач бронював номер, в іншій якщо ні.

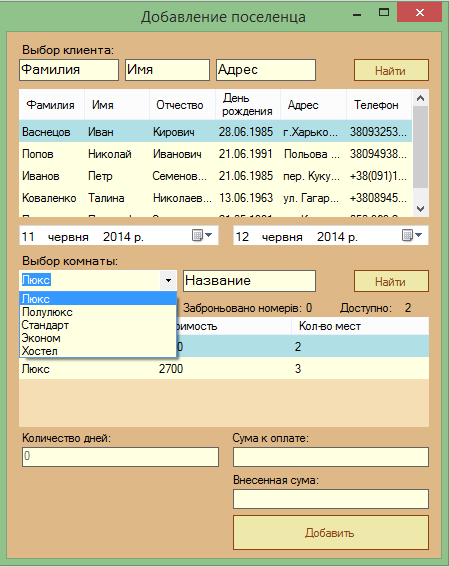


Рисунок 3.4 – Головне вікно «Hotel Booking System»

Вікно складається з області в якій ми вибираємо клієнта, згодом дати, а далі номер готелю. Система підраховує й жде відмітки про внесену суму передплати. До речі інтерфейс створення клієнта зображено на рисунку 3.5.

При створенні клієнту менеджер заповнює інтерактивну анкету, яка зображена на рисунку 3.5.

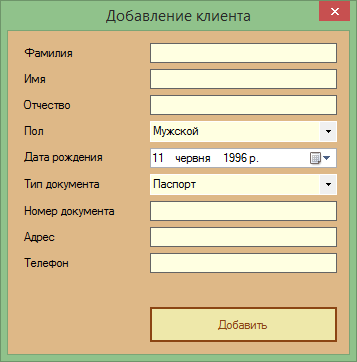


Рисунок 3.5 – Створення клієнта

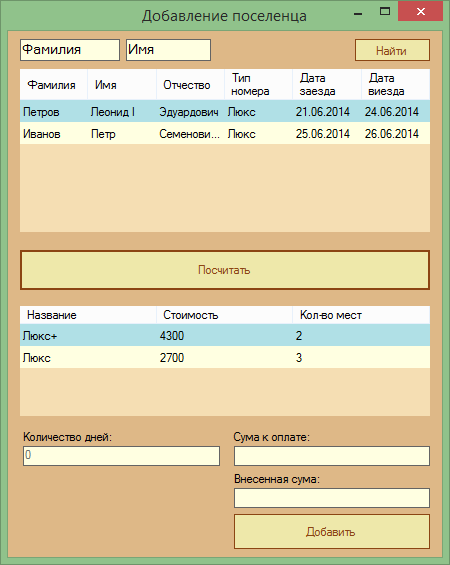


Рисунок 3.6 – Поселення використовуючи бронювання

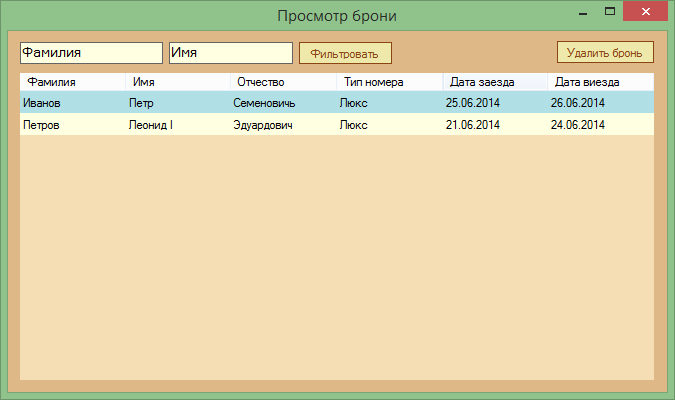


Рисунок 3.7 – Перегляд інформації про поточне бронювання

В більшості таблиць використовується фільтрування по імені та прозвищу для швидкого пошуку клієнта. Також можливо сортувати.

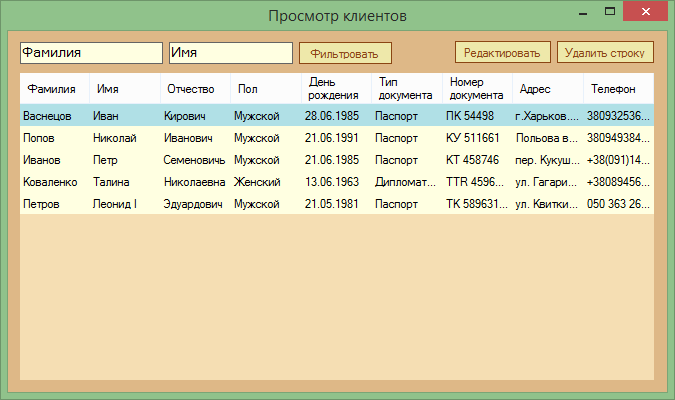


Рисунок 3.8 – Перегляд інформації про клієнтів

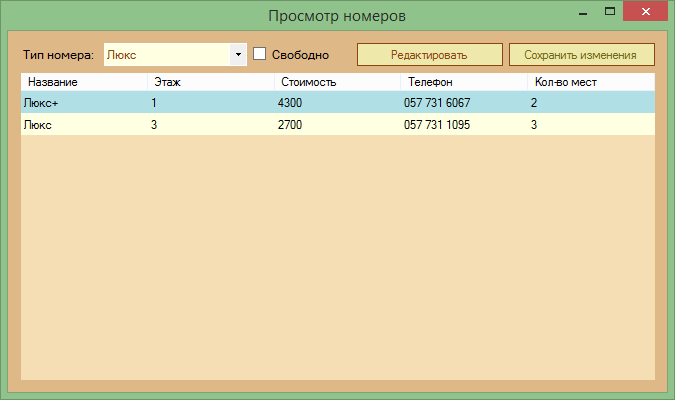


Рисунок 3.9 – Перегляд інформації про номери готелю

Щоб переглянути список кімнат необхідно вибрати тип кімнат, також можливо примінити фільтр та відобразити лише ті, які наразі не зайняті. При створені кімнати, в залежності від її типу змінюється опис полів.

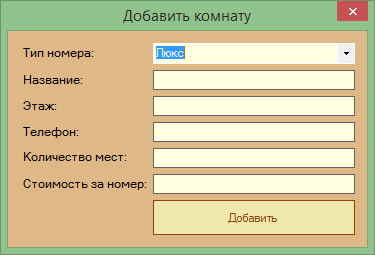


Рисунок 3.10 – Вікно створення кімнати в готелі

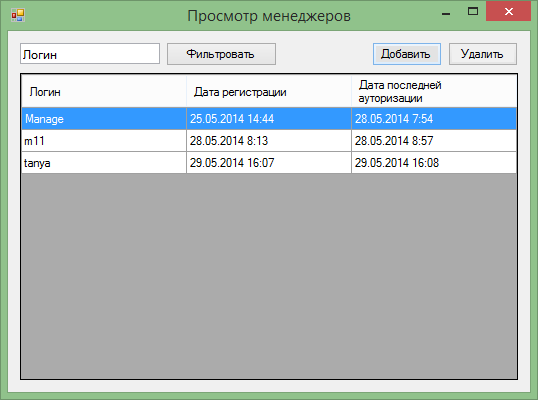


Рисунок 3.11 – Сторінка інформації про менеджерів, видалення та створення

Інформація про менеджерів, та їх керування необхідне для адміністратора на випадок звільнення менеджера чи додаванні нових.

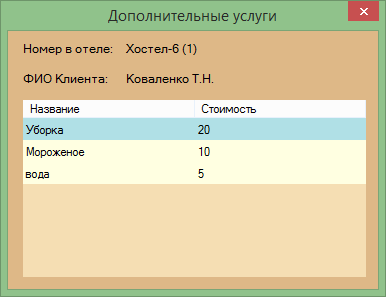


Рисунок 3.12 – Перегляд додаткових послуг що належать поточному клієнту

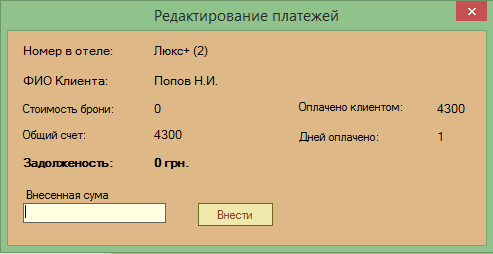


Рисунок 3.13 – Відомості про рахунок

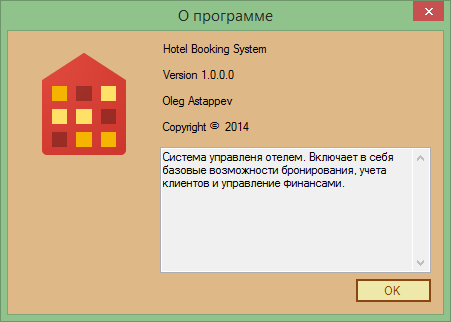


Рисунок 3.14 – Відомості про програму

### 3.2 Забезпечення необхідної функціональності ІС

Інформаційна система забезпечує виконання поставлених завдань за допомогою реалізації запитів, написаних мовою SQL. Функціональність програми можна розбити на кілька груп, кожна з яких відноситься до певного оператора SQL, синтаксис якого наведено в пункті 1.3.

Додавання даних здійснювалося за допомогою оператора INSERT. Для прикладу візьмемо сутність «Кімната готелю» (лістинг 3.1).

Лістинг 3.1 – Приклад заповнення таблиці «Кімната готелю»

String query = "INSERT INTO Rooms (RoomType\_ID,Name,Floor,Phone,Cost,NumBed) VALUES('" + type + "','" + FormName.Text + "','" + FormFloor.Text + "','" + FormPhone.Text + "','" + FormCost.Text + "','" + FormNum.Text + "')";

Для редагування даних використовуємо оператор UPDATE. Приклад обновлення даних зображено в лістингу 3.2.

Лістинг 3.2 – Приклад редагування даних сутності «Кімната готелю»

String query = "UPDATE Rooms SET Name = '" + FormName.Text + "', Cost = '" + FormFloor.Text + "' WHERE ID = " + id;

Для видалення данних використовується оператор DELETE. Приклад використання наведено в лістингу 3.3.

Лістинг 3.3 – Приклад видалення даних із таблиці «Кімната готелю»

String query = " DELETE FROM Rooms WHERE ID = " + id;

Для других таблиць запити використовуються аналогічно.

Для відображення табличних даних, наприклад візьмемо клієнтів використовуємо оператор SELECT, повний запит відображено в лістингу 3.4.

Лістинг 3.4 – Приклад отримання даних про клієнтів

String query = "SELECT Clients.ID AS ID, FirstName AS [Фамилия], LastName AS [Имя], Patronymic AS [Отчество], ClientGender.Name AS [Пол], Birthday AS [День рождения], DocType.Name AS [Тип документа], DocNumber AS [Номер документа], Adress AS [Адрес], Phone AS [Телефон] FROM Clients, DocType, ClientGender WHERE Clients.DocType\_ID = DocType.ID AND Clients.Gender = ClientGender.ID";

В лістингу 3.4 використовується з'єднання декількох таблиць та перейменування стовбців.

Для фільтрації користувачів використаємо оператор LIKE, наприклад в лістингу 3.5 ми отримали інформацію щодо клієнтів з ім'ям «Петро».

Лістинг 3.5 – Приклад фільтрування даних

String query = "SELECT Clients.ID AS ID, FirstName AS [Фамилия], LastName AS [Имя], Patronymic AS [Отчество], ClientGender.Name AS [Пол], Birthday AS [День рождения], DocType.Name AS [Тип документа], DocNumber AS [Номер документа], Adress AS [Адрес], Phone AS [Телефон] FROM Clients, DocType, ClientGender WHERE Clients.DocType\_ID = DocType.ID AND Clients.Gender = ClientGender.ID AND LastName LIKE '%" + LastNameBox.Text + "%'";

Для перегляду клієнтів які зараз мешкают в отелі використовуємо лістинг 3.7.

Лістинг 3.7 – Отримання даних про поточних клієнтів

string query = "SELECT RegisterCards.ID AS ID, Clients.FirstName AS Фамилия, Clients.LastName AS Имя, Rooms.Name AS Номер, RegisterCards.DateSettling AS [Дата заселения], RegisterCards.DateEviction AS [Дата выселения], FinalyCost AS [Общая стоимость] FROM RegisterCards INNER JOIN Rooms ON RegisterCards.Room\_ID = Rooms.ID INNER JOIN Clients ON RegisterCards.Client\_ID = Clients.ID WHERE GetDate() >= DateSettling";

Для отримання статистичних даних ми використовуємо з'єднання таблиць та функції SQL.

Лістинг 3.8 – Отримання інформації про кількість зареєстрованих клієнтів, загальну кількість проживань в готелі, загальну кількість орендованих днів, кількість поточно заброньованих номерів, кількість вільних номерів та кількість вільних місць.

string queryInfo = "SELECT (SELECT COUNT(\*) FROM Clients) AS [Кол-во клиентов], (SELECT COUNT(\*) FROM Archives) AS [Всего проживали], IsNull((SELECT (SELECT SUM(DATEDIFF(day, DateSettling, DateEviction)) FROM Archives) + (SELECT SUM(DATEDIFF(day, DateSettling, DateEviction)) FROM RegisterCards WHERE DateSettling < GETDATE())), 0) AS [Дней прожито], IsNull((SELECT SUM(DATEDIFF(day, DateSettling, DateEviction)) FROM Reservations), 3) AS [Всего забронировано], (SELECT COUNT(\*)FROM Rooms WHERE RoomType\_ID IN(1,2) AND NOT EXISTS(SELECT Room\_ID FROM RegisterCards WHERE Room\_ID = Rooms.ID)) AS [Свободно номеров], IsNull((SELECT SUM(NumBed)-(SELECT COUNT(\*) FROM RegisterCards WHERE Room\_ID IN(SELECT ID FROM Rooms WHERE RoomType\_ID NOT IN(1,2))) FROM Rooms), 3) AS [Свободно мест], (SELECT COUNT(\*) FROM RegisterCards WHERE DateSettling <= GetDate() AND DateEviction >= GetDate()) AS [Сейчас проживает]";

Для перевірки входження дати до діапазонів зайнятих дат, використовуємо оператор BETWEEN.

Лістинг 3.9 – Отримання списку номерів що не є заянятими на проміжку часу (але можуть бути заброньованими), для типів люкс та напівлюкс.

String query = "SELECT ID, Name AS [Название], Cost AS [Стоимость], NumBed AS [Кол-во мест] FROM Rooms WHERE Rooms.RoomType\_ID = " + type + " AND NOT EXISTS(SELECT Room\_ID FROM RegisterCards WHERE RegisterCards.Room\_ID = Rooms.ID AND ('" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction-1 OR '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling+1 AND DateEviction OR ('" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction) OR (DateSettling BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND DateEviction BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "')))";

Лістинг 3.10 – Отримання списку номерів що не є занятими на проміжку часу (але можуть бути заброньованими), для інших типів.

String queryGrid = "SELECT ID, Name AS [Название], Cost AS [Стоимость], NumBed AS [Кол-во мест], NumBed-ISNULL(Busy, 0) AS [Свободно мест] FROM Rooms LEFT JOIN (SELECT Room\_ID, COUNT(\*) AS [Busy] FROM RegisterCards WHERE '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction-1 OR '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling+1 AND DateEviction OR ('" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction) OR (DateSettling BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND DateEviction BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "') GROUP BY Room\_ID) AS AlredyExists ON AlredyExists.Room\_ID = Rooms.ID WHERE Rooms.RoomType\_ID = " + type + " AND (Busy IS NULL OR Busy < NumBed)";

Лістинг 3.11 – Підрахунок кількості заброньованих номерів на проміжку, для будь-якого типу.

Select query = "SELECT COUNT(\*) AS [Забронировано] FROM Reservations WHERE RoomType\_ID = " + type + " AND (DateSettling BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.AddDays(-1).ToString("yyyy-MM-dd") + "' OR DateEviction BETWEEN '" + DateSettlingTimePicker.Value.Date.AddDays(1).ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' OR (DateSettling BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND DateEviction BETWEEN '" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "') OR ('" + DateSettlingTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction AND '" + DateEvictionTimePicker.Value.Date.ToString("yyyy-MM-dd") + "' BETWEEN DateSettling AND DateEviction))";

4 ТЕСТУВАННЯ ІС

Тестування - важлива стадія розробки програмного забезпечення, це перевірка відповідності між реальною та очікуваною поведінкою програми, здійснюється на кінцевому етапі розробки.

Проведемо тестовий приклад ІС «Готель» під менеджером готелю, так як це цільовий користувач. Поточний стан таблиць представлено в Додатку Б.

Сценарій тестування буде наступним:

а) реєстрація нового користувача;

б) заселення його до номеру люкс;

г) додамо йому рахунок за напої;

д) переглянемо виставлені рахунки;

е) виселимо його;

Додамо клієнта з такими даними: Иванов Петр Иванович, чоловічої статі, який мешкає за адресою: вул. Гамарника, 7/9, Харків, Харківська область. Його номер телефону: 057 731 2094, дата народження 13 червня 1972 року, представлений документ – паспорт з номером КК 11144455. На рисунку 4.1 заповнена анкета користувача. Для того щоб відкрити таку анкету необхідно нажати на розділ «Клиенты» та «Добавить клиента».

Наступний крок, реєстрація в готелі. Для реєстрації нам необхідно перейти до розділу «Проживание», а згодом до «Поселить (без брони)». В відкрившомуся вікні знайти нашого клієнта (за допомогою фільтрів), вибрати дати та номер. Внести оплату та натиснути «Добавить». Цей процес зображено на рисунку 4.2.

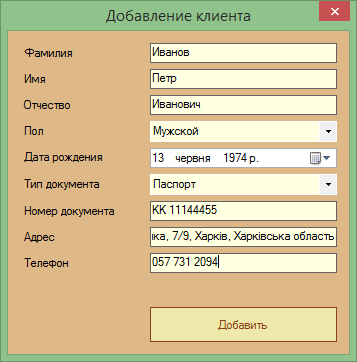


Рисунок 4.1 – Заповнена анкета додавання користувача

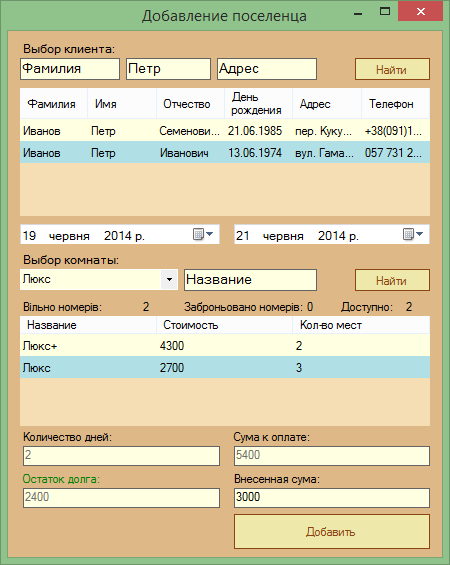


Рисунок 4.2 – Додавання мешканця готелю без бронювання

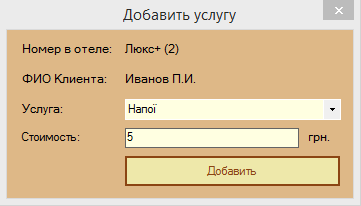


Рисунок 4.3 – Додавання послуги

Після додавання послуги, збільшується вартість проживання, на суму наданих послуг, перед видаленням перевіряємо (рис 4.5) чи не є він боржником. Виправляємо цей недолік (рис 4. 6), тепер можемо видалити клієнта (рис 4.6).

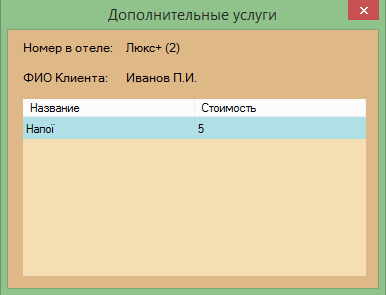


Рисунок 4.4 – Перелік наданих послуг

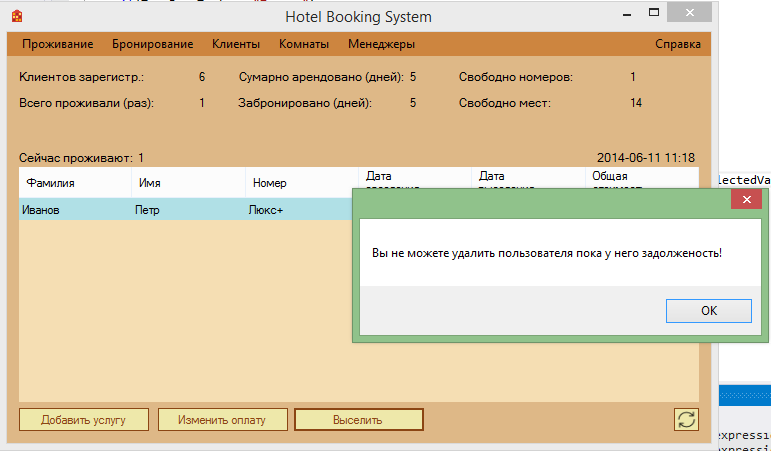


Рисунок 4.5 – Додавання послуги

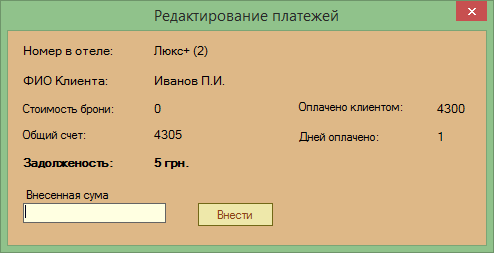


Рисунок 4.5 – Додавання послуги

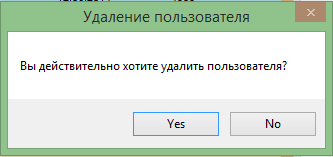


Рисунок 4.6 – Додавання послуги

# ВИСНОВКИ

Завдання курсової роботи, а саме, створення інформаційної системи «Готель» було виконане. В результаті виконання біла розроблена система що дозволяє автоматизувати роботу готелю та покращити якість обслуговування клієнтів.

Було виконано такі завдання:

* Забезпечити можливість зберігання та редагування інформації;
* Пошук інформації про клієнтів, номери, рахунки;
* Завдання статистичної обробки даних.

В процесі виконання роботи ми ознайомилися з таким матеріалом:

* Основи проектування моделі БД, використовуя CASE-засіб ERWin;
* Робота з СУБД MS SQL Server;
* Можливостями мови C# для взаємодії с БД та створення віконних додатків.

До базового, існуючого функціоналу додатку в майбутньому можливо додати більш складні частини. Наприклад такі як автоматичний підбір проміжку часу для бронювання, підбір номеру. Також можливо ускладнення системи за допомогою додання умови що існують однакові номери в системі готелю.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Боуман, Дж. Практическое руководство по SQL пер. с англ. – Л.: Сизай; М.: Суперайт, 2007. – 262 с.
2. Бошемин, Б. Основы ADO .Net: пер. с англ. – К.: Пинчер; М.: СофтБук, 2009. – 552 с.
3. Виейра, Р. Программирование баз данных Microsoft SQL Server 2008. Базовый курс: пер. с англ. – К.: Птицын; М.: Диалектика, 2010. – 816 с.
4. ДСТУ 3918-99 (ISO/IEC 12207:1995) Інформаційні технології. Процеси життєвого циклу програмного забезпечення. – К.: Держстандарт України, 2002. – 49 с.
5. Історія мови SQL » Навчальні матеріали з інформатики [Электронный ресурс] / Навчальні матеріали з інформатики. – Режим доступа : www/ URL: http://www.ua5.org/osnprog/111-storja-movi-sql.html − 10.06.2014 г. – Загл. с экрана.
6. Тернстрем, Т. Microsoft SQL Server 2008. Разработка баз данных. Учебный курс Microsoft: пер. с англ. – Т.: Коротяева; М.: Русская Редакция, 2010. – 816 с.
7. Типы команд и команды SQL « АСУТП блог [Электронный ресурс] / АСУТП блог. – Режим доступа : www/ URL: http://asutpblog.ru/2010/03/13/tipy-komand-i-komandy-sql/ − 10.06.2014 г. – Загл. с экрана.

# Додаток А – DDL ЗАПИТИ НА СТВОРЕННЯ ФІЗИЧНОЇ СТРУКТІРИ БД

CREATE TABLE RoomTypes

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_RoomTypes PRIMARY KEY,

Name VARCHAR(120) NOT NULL

);

CREATE TABLE Rooms

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Rooms PRIMARY KEY,

RoomType\_ID INT NOT NULL CONSTRAINT FK\_RoomType\_Rooms REFERENCES RoomTypes(ID) ON UPDATE CASCADE ON DELETE CASCADE,

Name VARCHAR(50) NOT NULL,

Floor INT NOT NULL,

Phone VARCHAR(14),

Cost INT NOT NULL,

NumRoom INT,

NumBed INT

);

CREATE TABLE Roles

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Roles PRIMARY KEY,

Name VARCHAR(60) NOT NULL

);

CREATE TABLE Users

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Users PRIMARY KEY,

Role\_ID INT NOT NULL CONSTRAINT FK\_Roles\_Users REFERENCES Roles(ID) ON UPDATE CASCADE ON DELETE CASCADE,

Login VARCHAR(40) NOT NULL CONSTRAINT UC\_Login UNIQUE,

Password VARCHAR(255) NOT NULL,

DateReg DateTime NOT NULL DEFAULT GetDate(),

DateLastLogin DateTime

);

CREATE TABLE DocType

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_DocType PRIMARY KEY,

Name VARCHAR(120) NOT NULL

);

CREATE TABLE ClientGender

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_ClientGender PRIMARY KEY,

Name VARCHAR(120) NOT NULL

);

CREATE TABLE Clients

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Clients PRIMARY KEY,

User\_ID INT CONSTRAINT FK\_Users\_Clients REFERENCES Users(ID) ON UPDATE CASCADE ON DELETE SET NULL,

FirstName VARCHAR(90) NOT NULL,

LastName VARCHAR(90) NOT NULL,

Patronymic VARCHAR(90) NOT NULL,

Gender INT NOT NULL CONSTRAINT FK\_Clients\_ClientGender REFERENCES ClientGender(ID) ON UPDATE CASCADE,

Birthday DateTime NOT NULL,

DocType\_ID INT NOT NULL CONSTRAINT FK\_Clients\_DocType REFERENCES DocType(ID) ON UPDATE CASCADE,

DocNumber VARCHAR(30) NOT NULL,

Adress VARCHAR(128) NOT NULL,

Phone VARCHAR(14) NOT NULL

);

CREATE TABLE RegisterCards

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_RegisterCards PRIMARY KEY,

Client\_ID INT NOT NULL CONSTRAINT FK\_Clients\_RegisterCards REFERENCES Clients(ID) ON UPDATE CASCADE ON DELETE CASCADE,

Room\_ID INT NOT NULL CONSTRAINT FK\_Rooms\_RegisterCards REFERENCES Rooms(ID) ON UPDATE CASCADE ON DELETE CASCADE,

DateSettling DateTime NOT NULL,

DateEviction DateTime NOT NULL,

CostReservation INT NOT NULL DEFAULT '0'

NumDaysPaid INT,

SumPaid INT NOT NULL,

FinalyCost INT NOT NULL DEFAULT '0'

);

CREATE TABLE Accounts

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Accounts PRIMARY KEY,

Card\_ID INT NOT NULL CONSTRAINT FK\_RegisterCards\_Accounts REFERENCES RegisterCards(ID) ON UPDATE CASCADE ON DELETE CASCADE,

Name VARCHAR(60) NOT NULL,

Cost INT NOT NULL,

DateSet DateTime NOT NULL DEFAULT(GetDate())

);

CREATE TABLE AccountsTemplate

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_AccountsTemplate PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Cost INT

);

CREATE TABLE Archives

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Archives PRIMARY KEY,

Client\_ID INT NOT NULL CONSTRAINT FK\_Clients\_Archives REFERENCES Clients(ID) ON UPDATE CASCADE ON DELETE CASCADE,

Room\_ID INT NOT NULL CONSTRAINT FK\_Rooms\_Archives REFERENCES Rooms(ID) ON UPDATE CASCADE ON DELETE CASCADE,

DateSettling DateTime NOT NULL,

DateEviction DateTime NOT NULL,

FinalyCost INT NOT NULL

);

CREATE TABLE Reservations

(

ID INT IDENTITY(1,1) CONSTRAINT PK\_Reservations PRIMARY KEY,

Client\_ID INT NOT NULL CONSTRAINT FK\_Clients\_Reservations REFERENCES Clients(ID) ON UPDATE CASCADE ON DELETE CASCADE,

RoomType\_ID INT NOT NULL CONSTRAINT FK\_RoomTypes\_Reservations REFERENCES RoomTypes(ID) ON UPDATE CASCADE ON DELETE CASCADE,

DateSettling DateTime NOT NULL,

DateEviction DateTime NOT NULL,

DateReservation DateTime NOT NULL DEFAULT(GetDate()));

# Додаток Б − Відносини екземпляри розробленої БД



Рисунок Б.1 – Відношення «Accounts»



Рисунок Б.2 – Відношення «Archives»



Рисунок Б.3 – Відношення «ClientGender»

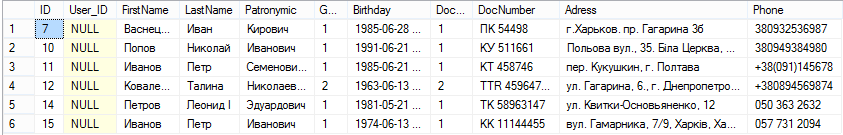


Рисунок Б.4 – Відношення «Clients»

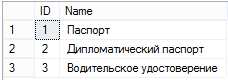


Рисунок Б.5 – Відношення «DocType»

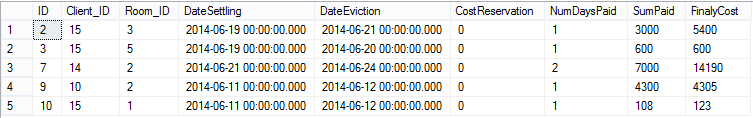


Рисунок Б.6 – Відношення «RegisterCards»

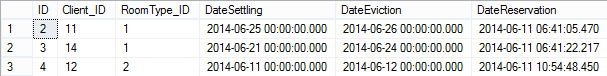


Рисунок Б.7 – Відношення «Reservations»

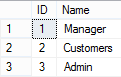


Рисунок Б.8 – Відношення «Roles»

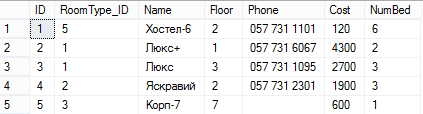


Рисунок Б.9 – Відношення «Rooms»

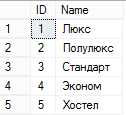


Рисунок Б.10 – Відношення «RoomTypes»

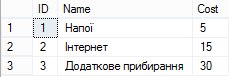


Рисунок Б.11 – Відношення «TemplateAccounts»

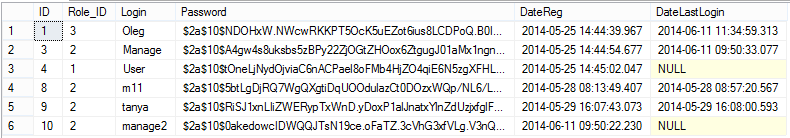


Рисунок Б.12 – Відношення «Users»