**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ**

**УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ** **ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-**

**аналітики**

**КУРСОВА РОБОТА**

**з навчальної дисципліни**

**“Проектування та адміністрування БД і СД”**

**Тема:**

**«Інформаційна система для готелю»**

**Науковий керівник: Виконавець:**

к.ф.-м.н., доц.Ярема О.Р.Кирчей Я.Ю

(прізвище, ім’я, по-батькові) (прізвище, ім’я, по-батькові)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) УФЕ-31с група

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис)

**“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 р. “\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2020 р.**

**Загальна кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підписи, ПІП членів комісії)

**Львів 2020**

**Зміст**

**Вступ**

**1 Аналіз вимог:**

1.1. Постановка завдання;

1.2. Розробка моделі варіантів використання веб-сайту;

1.3. Аналіз засобів реалізації (техніко-економічне обґрунтування

вибору);

**2 Розробка бази даних:**

2.1 Опис моделі даних;

2.2 Опис таблиць бази даних;

2.3 Нормалізація відношень;

2.4 Визначення типів даних;

2.5 Обмеження цілісності даних;

2.6 Реалізація SQL-скрипту;

**Висновки**

**Вступ**

В наш час інформаційні системи відіграють важливу роль. Сучасною об’єктивною реальністю є широке впровадження у всі сфери життєдіяльності особи, суспільства, підприємств та держави сучасних інформаційних технологій, розгортання на їх основі різноманітної складності локальних і глобальних інформаційних систем, призначених для прискорення обміну інформацією та доступу до різноманітних інформаційних джерел.

Основною ідеєю сучасних інформаційних технологій є концепція про те, що всі дані повинні бути організовані в бази даних. Відповідно до даної концепції основою інформаційної технології є дані, організовані в БД, які адекватно відображають реалії дійсності в тій чи іншій предметної області і забезпечують користувача актуальною інформацією. Створення і функціонування таких баз даних управляються спеціальними програмними комплексами - системами управління базами даних (СУБД). Ядром будь-якої бази даних є модель даних. Модель даних являє собою безліч структур даних, обмежень цілісності і маніпулювання даними. За допомогою моделі даних можуть бути представлені об'єкти предметної області та взаємозв'язки між ними.

**Актуальність теми дослідження**. На сьогоднішній день неможливо уявити ефективне функціонування підприємства готельного бізнесу без використання спеціалізованого програмного забезпечення, що дає можливість автоматизувати всі бізнес-процеси. Сучасні темпи розвитку висувають найвищі вимоги до автоматизованих систем управління для підприємствах індустрії гостинності. Потреби клієнтів і конкурентна боротьба прагнуть від закладів швидкого реагування на всі коливання ринкової кон'юнктури. Використання комп'ютерних мереж, Інтернету та Інтернет-технологій, програмних продуктів наскрізної автоматизації всіх бізнесів-процесів готелю сьогодні не просто питання лідерства і створення конкурентних переваг, але і виживання на ринку.

Мета і завдання дослідження. Метою курсової робити є дослідження та аналіз актуальності інформаційної системи підприємства готельного бізнесу.

Реалізації поставленої мети передбачає вирішення наступних завдань:

* Дослідити історію створення реляційної моделі, структуру даних та відношення в реляційній моделі, а також основні принципи побудови РБД;
* визначити особливості створення бази для готелю;
* перевірити програмний продукт на наявність дефектів та знайти шляхи покращення;
* Розкрити модель сутність-зв'язок, описати характеристику зв'язків, класифікацію сутностей, структуру первинних і зовнішніх ключів, визначити поняття цілісності даних.
* Проаналізувати доцільність створення інформаційної системи(веб-сайту, бази даних) для підприємств готельного бізнесу.

Об’єктом дослідження виступає конкретний готель, для якого планується створення інформаційної системи та процес розробки відповідного програмного забезпечення.

Предметом є теоретичні, методичні та практичні аспекти розробки програмного забезпечення мовою розмітки: мовою запитів MySql для створення бази даних для готелю.

**Практичне значення отриманих результатів**. Програма може використовуватись юридичною особою, яка є власником готелю, з метою спрощення процесів обробки даних.

**Використане програмне забезпечення**. Для створення програмного продукту використовувалось програмний продукт DataGrip та мова запитів MySql.

**Структура роботи**. Курсова робота складається з двох розділів (теоретичного – „Аналіз вимог’’ та практичного – “Розробка бази даних’’), висновків, списку використаних джерел та додатків.

Загальний обсяг роботи − 29 сторінок.

**Розділ 1 Аналіз вимог**

* 1. **Постановка завдання**

У ході роботи необхідно провести аналіз туристично-готельного підприємства ”Luxury hotel”. У разі позитивною оцінки аналізу потрібно створити базу даних та веб-сайт для даного готелю.

В курсовому проекті за мету взято створення бази даних, яка має надати можливість зберігання великої кількості інформації, забезпечувати швидкий пошук інформації для клієнта. База даних повинна реалізовувати змогу обновлення та добавлення, або видалення даних які в ній зберігаються, а також зручне та надійне користування, та доступ усіх необхідних програм до неї.

При розробленні бази даних необхідно створити таблиці в SQL Workbanch, та заповнити їх необхідними полями(атрибутами).

Але до реалізації таблиць необхідно розробити модель в model.erwin, в якій буде зображено зв’язки між таблицями та основну схему побудови таблиць.

Після цього треба створити запити завдяки мові mysql, та перевірити їх роботу на видання різної інформації.

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин : збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Дані – це інформація, відомості, показники, необхідні для ознайомлення з ким-, чим-небудь, для характеристики когось, чогось або для прийняття певних висновків, рішень. В базах даних важливу роль відіграє інформація - Інформація — абстрактне поняття, що має різні значення залежно від контексту.

Кожна створена база даних має свою предметну область. Предметна область – це необхідний для розробки бази даних об’єкт, який має в собі дані, які будуть зберігатися в базі даних.

Модель даних — абстрактне представлення реального світу, що відображає тільки ті об'єкти, що безпосередньо стосуються програми. Це, як правило, визначає специфічну групу об'єктів, їх атрибутивне значення і відношення між ними.

Відомі два підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується рядом недоліків. Характерна риса файлового підходу - вузька спеціалізація як обробних програм, так і файлів даних, що служить причиною великої надлишковості, тому що ті самі елементи даних зберігаються в різних системах. Оскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше області. Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів передання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зв'язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань. Вказані причини обумовили появу баз даних. База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні "файл-програма", вона явним чином декларується в базі.

База даних орієнтована на інтегровані запити, а не на одну програму, яку випадку файлового підходу, і використовується для інформаційних потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

Системи управління базами даних – це програмні засоби, за допомогою яких можна створювати бази даних, заповнювати їх та працювати з ними. У світі існує багато різноманітних систем управління базами даних. Багато з них насправді є не закінченими продуктами, а спеціалізованими мовами програмування, за допомогою яких кожний, хто вивчить дану мову, може сам створювати такі структури, які йому потрібні, і вводити в них необхідні елементи управління. До таких мов відносяться Clipper, Paradox, FoxPro та інші.

**1.2. Розробка моделі варіантів використання веб-сайту**

Для того, щоб більш точно зрозуміти як повинна працювати система, все частіше використовується опис функціональності системи через варіанти використання (Use Case або прецеденти). Варіанти використання це – опис послідовності дій, які може здійснювати система у відповідь на зовнішні впливи користувачів або інших програмних систем. Варіанти використання відображають функціональність системи з точки зору отримання відчутного результату для користувача, тому вони точніше дозволяють ранжувати функції за значимістю одержуваного результату.

Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи і керують усім процесом розробки. Всі основні види діяльності такі як аналіз, проектування, тестування виконуються на основі варіантів використання. Під час аналізу і проектування варіанти використання дозволяють зрозуміти як результати, які хоче отримати користувач впливають на архітектуру системи і як повинні поводитися компоненти системи, для того щоб реалізувати потрібну для користувача функціональність. Варіанти використання ілюструються за допомогою діаграми варіантів використання. Діаграма варіантів використання складається з акторів, для яких система виробляє дію і власне варіанти використання (Use Case), які описують те, що актор хоче отримати від системи. Актор позначається значком чоловічка, а варіанти використання - овалом. Додатково в діаграми можуть бути додані коментарі.

Відповіді на такі питання дозволять визначити акторів, що взаємодіють з системою:

* хто взаємодіє з системою або використовує систему;
* хто передає чи приймає інформацію в/з системи;
* хто є зовнішнім по відношенню до системи.

Кожен варіант використання показує, як конкретний актор використовує

систему і надалі розширюється діаграмами станів і послідовності дій.

Між акторами і варіантами використання можуть бути різні види взаємодії. Основні види взаємодії наступні: − Проста асоціація - відображається лінією між актором і варіантом використання (без стрілки). Відображає зв'язок актора і варіанта використання. − Спрямована асоціація - те ж що і проста асоціація, але показує, що варіант використання ініціалізуєтся актором. Позначається стрілкою. − Спадкування - показує, що нащадок успадкує атрибути і поведінку свого прямого предку. Може застосовуватися як для акторів, так для варіантів використання. − Розширення (extend) - показує, що варіант використання розширює базову послідовність дій і вставляє власну послідовність. При цьому на відміну від типу відносин "включення" розширена послідовність може здійснюватися в залежності від певних умов. − Включення (include) - показує, що варіант використання включається в базову послідовність і виконується завжди (на рисунку не показаний). Існують і інші види взаємодії, але вони менш важливі і рідше застосовуються. Для створення діаграми використання можна використовувати редактор діаграм UML StarUML, або інший. В проекті також необхідно деталізувати кожний варіант використання, тобто виконати його уточнення за допомогою специфікації варіанту використання. Основний потік описує «ідеальний» хід розвитку подій у варіанті використання. Альтернативні потоки можуть перехоплювати помилки, відгалуження і переривання основного потоку. Основний потік завжди починається з дій головного актора, спрямованих на ініціацію варіанту використання.

Вдалим способом початку потоку можна визначити таку форму запису:

Прецедент починається, коли <актор> <дія>.

Кожен етап потоку варіанту використання повинен бути виражений в такій формі: <Номер> <хтось> <робить деяку дію>. Використання пасивного стану для опису етапу є невірним: «Вводяться дані покупця».

Альтернативні потоки можуть бути замінені за допомогою ключового

слова «Якщо».

Основний потік:

1 Варіант використання починається коли Клієнт вибирає заявку зі списку.

2 Якщо Клієнт вибирає "видалити заявку".

2.1Система призначає заявці статус "Відхилена".

**1.3. Аналіз засобів реалізації (техніко-економічне обґрунтування**

**вибору);**

Було вирішено обрати наступні технології для створення бази даних для готелю:

* мова запитів Sql.

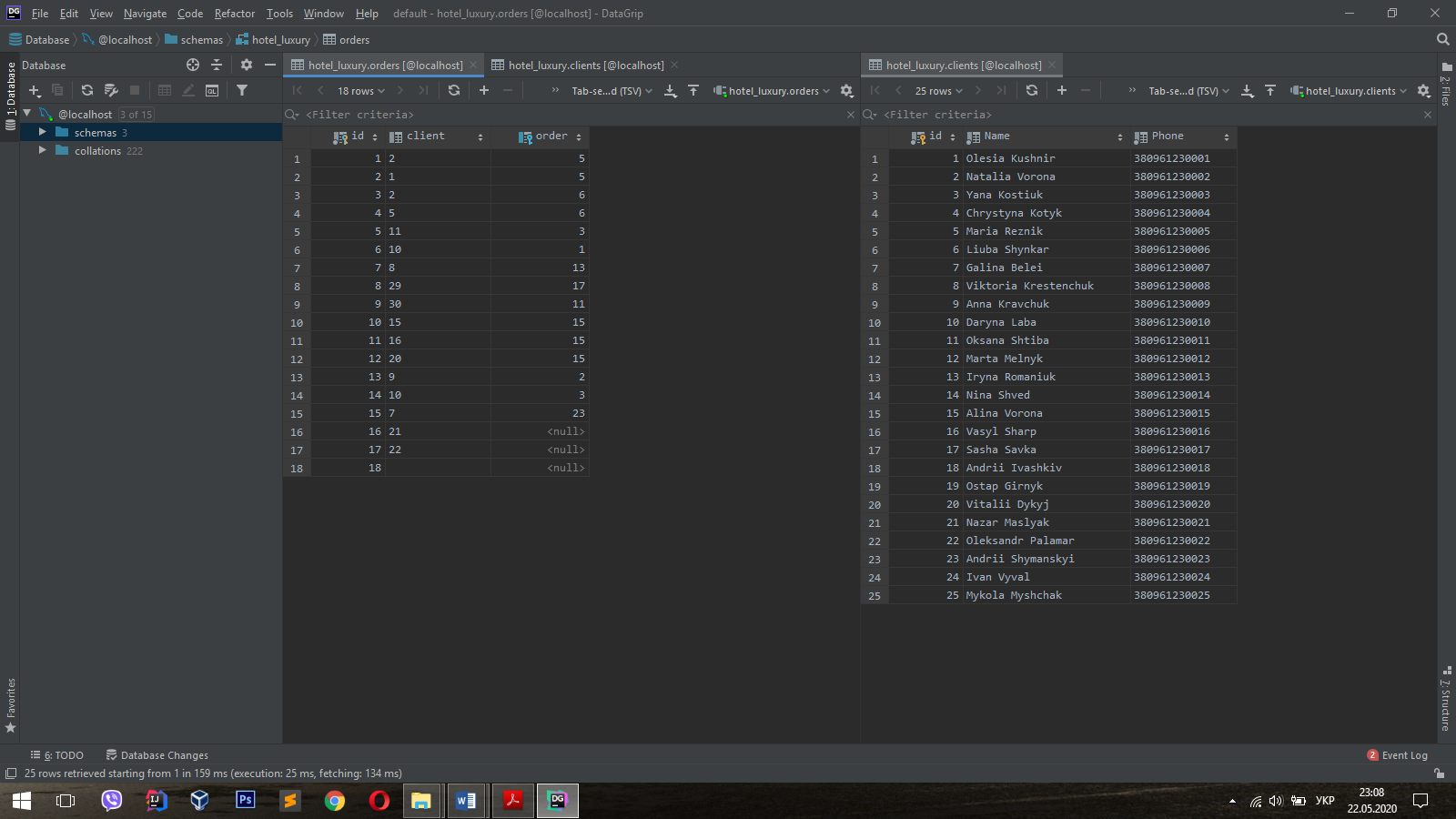
Чому саме MySql? Відповідь очевидна.

Переваги  даної мови[:](http://www.znannya.org/?view=concept:302)

* [простота у встановленні та використанні;](http://www.znannya.org/?view=concept:302" \o "<strong id=)
* [підтримується необмежена кількість користувачів, що одночасно працюють із БД;](http://www.znannya.org/?view=concept:302" \o "<strong id=)
* [кількість рядків у таблицях може досягати 50 млн.;](http://www.znannya.org/?view=concept:302" \o "<strong id=)
* [висока швидкість виконання команд;](http://www.znannya.org/?view=concept:302" \o "<strong id=)
* [наявність простої і ефективної системи безпеки.](http://www.znannya.org/?view=concept:302" \o "<strong id=)
* Програмне середовище DataGrip

Переваги даної програми:

* + Зручний інтерфейс прграми;
  + Можливість створювати таблиці та наповняти їх без використання жодної лінійки коду.
  + Можливість візуально оглядати таблиці та переглядати звязки між ними;
  + Можливість одночасно працювати з декількома базами даних та поєднювати їх між собою.
  + Можливість імортувати та експортувати файли.
  + Можливість імпортувати системні настройки та налаштовувати систему індивідуально під власні потреби.
  + Дана програма надає логічні підказки щодо синтаксису MySQl та будування структури бази даних.
  + В даній програмі з легкістю можна зберегти схему чи таблицю в потрібному форматі.



*Рис.1. програмний інтерфейс DataGrip*

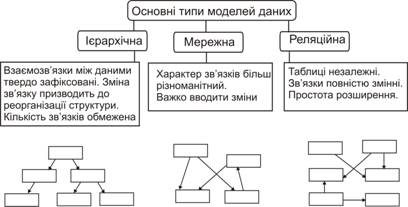
У зв’язку з вище перерахованими перевагами було проийнято рішення обирати дані технології.

**Розділ 2 Розробка бази даних**

**2.1 Опис моделі даних;**

Основою бази даних є модель даних — фіксована система понять і правил для представлення даних структури, стану і динаміки проблемної області в базі даних.

Моде́ль да́них (англ. Data model) — абстрактне представлення реального світу, що відображає тільки ті об'єкти, що безпосередньо стосуються програми. Це, як правило, визначає специфічну групу об'єктів, їх атрибутивне значення і відношення між ними. У випадку ГІС, використовується механізм представлення і організації просторової моделі даних, або растрової моделі даних. Вона не залежить від комп'ютерної системи і пов'язана тільки зі структурою даних.



*Рис.2.1. Схема типів моделей даних*

Ієрархічна модель

Ієрархічна структура (модель) будується у вигляді ієрархічної деревоподібної структури, у якій для кожного головного об’єкта існує кілька підлеглих, а для кожного підлеглого об’єкта може бути тільки один головний. На найвищому рівні ієрархії перебуває кореневий об’єкт. Прикладом ієрархічної структури даних може бути організація каталогів на диску, різного роду класифікації, структура державної влади тощо.

Концептуальна схема ієрархічної моделі являє собою сукупність типів записів, пов'язаних типами зв'язків в одним чи кількома деревами. Усі типи зв'язків цієї моделі належать до виду «один до декількох» і зображуються у вигляді стрілок.

Таким чином, взаємозв'язки між об'єктами нагадують взаємозв'язки в генеалогічному дереві, за єдиним винятком: для кожного породженого (підлеглого) типу об'єкта може бути тільки один вхідний (головний) тип об'єкта. Тобто ієрархічна модель даних допускає тільки два типи зв'язків між об'єктами: «один до одного» і «один до декількох». Ієрархічні бази даних є навігаційними, тобто доступ можливий тільки за допомогою заздалегідь визначених зв'язків.

При моделюванні подій, як правило, необхідні зв'язки типу «багато до декількох». Як одне з можливих рішень зняття цього обмеження можна запропонувати дублювання об'єктів. Однак дублювання об'єктів створює можливості неузгодженості даних.

Достоїнство ієрархічної бази даних полягає в тому, що її навігаційна природа забезпечує швидкий доступ при проходженні вздовж заздалегідь визначених зв'язків. Однак негнучкість моделі даних і, зокрема , неможливість наявності в об'єкта декількох батьків, а також відсутність прямого доступу до даних роблять її непридатною в умовах частого виконання запитів, не запланованих заздалегідь. Ще одним недоліком ієрархічної моделі даних є те, що інформаційний пошук з нижніх рівнів ієрархії не можна спрямувати по вище розміщених вузлах.

Мережна модель

У мережній моделі один і той же об’єкт може одночасно виступати як у ролі головного, так і підлеглого елемента. Це означає, що кожний об’єкт може брати участь у довільній кількості зв’язків. Зв’язок у цьому випадку може встановлюватися явно, коли значення деяких полів є посилання на дані, що містяться в іншому файлі. Прикладом мережної структури БД може бути структура автобусних маршрутів (із будь-якого населеного пункту існують маршрути в інші).

Подібно до ієрархічної, мережну модель також можна подати у вигляді орієнтованого графа. Але в цьому випадку граф може містити цикли, тобто вершина може мати кілька батьківських вершин.

Така структура набагато гнучкіша і виразніша від попередньої і придатна для моделювання більш ширшого класу завдань.

Ієрархічні і мережні бази даних часто називають базами даних з навігацією. Ця назва відбиває технологію доступу до даних, використовувану при написанні програм обробки мовою маніпулювання даними. При цьому доступ до даних по шляхах, не передбачених при створенні бази даних, може потребувати нерозумно тривалого часу.

Підвищуючи ефективність доступу до даних і скорочуючи таким чином час відповіді на запит, принцип навігації разом з цим підвищує і ступінь залежності програм і даних. Програми обробки даних виявляються жорстко прив'язаними до поточного стану структури бази даних і повинні бути переписані при її змінах. Операції модифікації і видалення даних вимагають переустановлення покажчиків, а маніпулювання даними залишається записоорієнтованим. Крім того, принцип навігації не дозволяє істотно підвищувати рівень мови маніпулювання даними, щоб зробити його доступним користувачу-непрограмісту чи навіть програмісту-непрофесіоналу. Для пошуку запису-мети в ієрархічній або мережній структурі програміст повинен спочатку визначити шлях доступу, а потім — крок за кроком переглянути всі записи, що трапляються на цьому шляху.

Наскільки складними є схеми представлення ієрархічних і мережних баз даних, настільки і трудомістким є проектування конкретних прикладних систем на їхній основі. Як показує досвід, тривалі терміни розроблення прикладних систем нерідко призводять до того, що вони постійно перебувають на стадії розроблення і доопрацювання. Складність практичної реалізації баз даних на основі ієрархічної і мережної моделей визначила створення реляційної моделі даних.

Реляційна модель

У Реляційна модель даних|реляційній моделі дані й взаємозв’язки між ними подаються за допомогою прямокутних таблиць. Рядки в реляційній базі даних називають записами, а стовпці — полями. Модель реляційної бази даних була вперше розроблена доктором Е.Ф. Коддом на початку 70-х років XX ст. як більш зручний засіб збереження, вибірки й маніпулювання даними, ніж ієрархічні й мережні бази даних. Модель двовимірної таблиці дозволяє звертатися до даних як по рядках, так і по стовпцях, що є значною перевагою.

Назва «реляційна» (relational) пов'язана з тим, що кожен запис у таблиці даних містить інформацію, яка стосується (related) якогось конкретного об'єкта. Крім того, зв'язані між собою (тобто такі, що знаходяться в певних відношеннях — relations) дані навіть різних типів в моделі можуть розглядатися як одне ціле.

Таблиця має такі властивості:

* кожний елемент таблиці являє собою один елемент даних;
* повторювані групи відсутні;
* усі стовпці в таблиці однорідні; це означає, що елементи стовпця мають однакову природу;
* стовпцям присвоєні унікальні імена;
* у таблиці немає двох однакових рядків

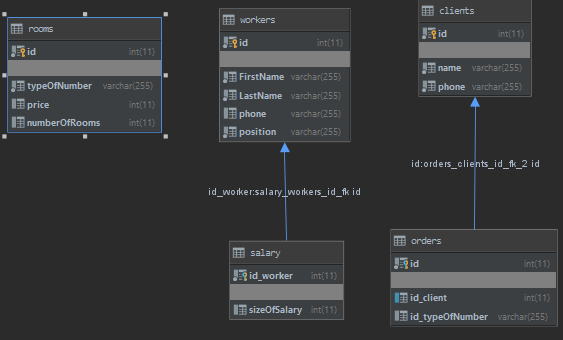
Порядок розміщення рядків і стовпців у таблиці довільний; таблиця такого типу називається відношенням. У сучасній практиці для рядка використовується термін «запис», а для стовпця термін «поле».

Основною відмінністю пошуку даних в ієрархічних, мережних і реляційних базах даних є те, що ієрархічні і мережні моделі даних здійснюють зв'язок і пошук між різними об'єктами за структурою, а реляційні — за значенням ключових атрибутів (наприклад, можна знайти всі записи, значення яких у полі «номер будинку» дорівнює 3, але не можна знайти 3-й рядок).

Оскільки реляційна структура концептуально проста, вона дозволяє реалізовувати невеликі і прості (і тому легкі для створення) бази даних, навіть персональні, сама можливість реалізації яких ніколи навіть і не розглядалася в системах з ієрархічною чи мережною моделлю.

Недоліком реляційної моделі даних є надмірність по полях (для створення зв'язків між різними об'єктами бази даних).

У ході практичного завдання була побудована наступна модель бази даних (Рис.2.2).



*Рис.2.2 Модель бази даних*

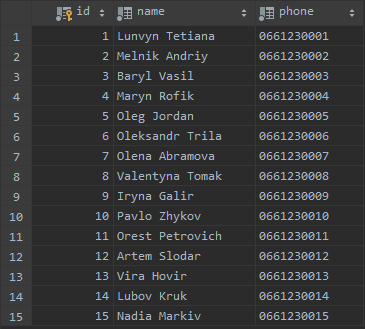
**2.2 Опис таблиць бази даних**

Також на рисунку (рис.2.1) можна побачити, що дана модель складається з п’яти таблиць: clients, orders, workers salary, rooms.

Таблиця clients orders з таких стовпців:

* id;
* clients
* order.

Для прикладу в даній таблиці внесано наступні поля:

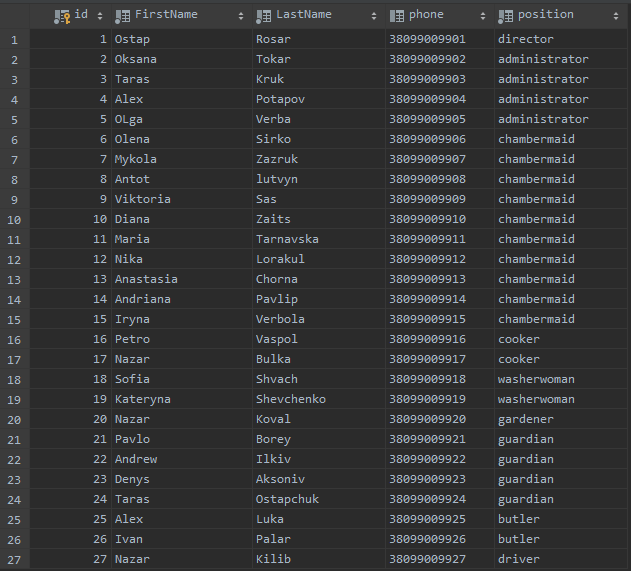


*Рис. 2.3. вигляд таблиці clients*

Таблиця workers скадається з таких стовпців:

* id;
* firstName;
* lastName;
* age
* position;
* phone.

Для прикладу в даній таблиці внесано наступні поля:

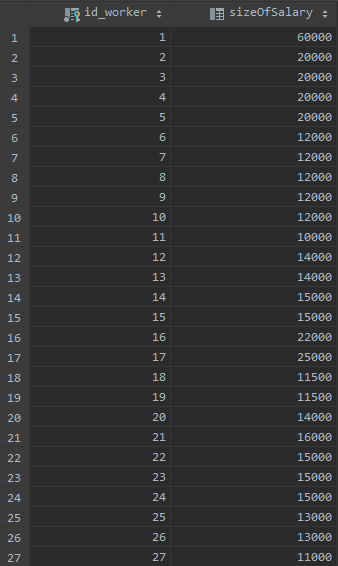


*Рис. 2.4. вигляд таблиці workers*

Таблиця salary скадається з наступних стовпців:

* id;
* size\_Salary.

Для прикладу в даній таблиці внесано наступні поля:

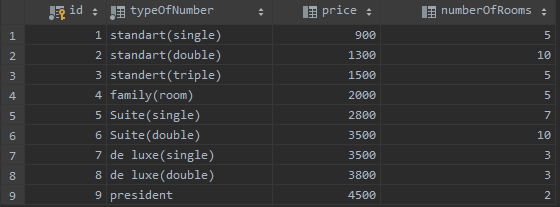


*Рис. 2.5. вигляд таблиці salary*

Таблиця rooms скадається з наступних стовпців:

* id;
* typeOfNumber;
* praice.

Для прикладу в даній таблиці внесано наступні поля:



*Рис. 2.5. вигляд таблиці salary*

**2.3 Нормалізація відношень**

Нормалізація відношень- покроковий процес розділення (декомпозиції) початкових відношень БД на простіші. Кроки цього процесу переводять схему відношення БД в послідовні нормальні форми. Кожна наступна форма володіє кращими властивостями ніж попередня. Кожній нормальній формі відповідає певний набір обмежень. При переведенні структури відношення у форми вищого порядку досягають видалення з таблиць надмірної описової інформації. Процес нормалізації заснований на понятті функціональної залежності атрибутів.

Перша нормальна форма. Відношення відповідає 1NF тоді, коли на перетині кожного стовпця і кожного рядка знаходяться тільки елементарні (неподільні) значення атрибутів і не містяться групи, що повторюються.

Друга нормальна форма. Відношення знаходиться в 2NF, якщо виконуються обмеження 1NF і кожен описовий атрибут функціонально повно залежить від первинного ключа (у тому числі і складеного).

Третя нормальна форма. Відношення знаходиться у ЗNF, якщо виконуються обмеження 2NF і всі описові атрибути відношення взаємно незалежні і повністю залежать від первинного ключа, тобто кожний описовий атрибут не транзитивно залежить від ключа.

Четверта нормальна форма (4НФ, 4NF) вимагає, аби в схемі баз даних не було нетривіальних багатозначних залежностей множин атрибутів від будь чого, окрім надмножини ключа-кандидата. Вважається, що таблиця знаходиться у 4НФ тоді, і тільки тоді, коли вона знаходиться в НФБК, та багатозначні залежності є функціональними залежностями. Четверта нормальна форма усуває небажані структури даних — багатозначні залежності.

П'ята нормальна форма (5НФ, 5NF, PJ/NF) вимагає, аби не було не тривіальних залежностей об'єднання, котрі б не витікали із обмежень ключів. Вважається, що таблиця в п'ятій нормальній формі, тоді, і тільки тоді, коли вона знаходиться в 4НФ, та кожна залежність об'єднання зумовлена її ключами-кандидатами.

Розглянемо нашу модель даних(Рис.2.2). Можна побачити що база даних „розділена” на декілька логічних за змістом таблиць. Є окрема таблиця з переліком клієнтів, окрема для замовлень, окрема з номерами готелю. Також окремо відділено таблицю із заробітньою платою від таблиці з працівників, але поєднато ці дві таблиці за допомогою вторинного ключа, аналогічно за допомогою вторинного ключа поєднано таблицю замовленя та клієнтів.

**2.4.Визначення типів даних;**

При створенні таблиці для всіх її стовпців необхідно вказати певний тип даних. Тип даних визначає, які значення можуть зберігатися в стовпці, скільки вони будуть займати місця в пам'яті.

Мова SQL надає безліч різних типів. Залежно від характеру значень все їх можна розділити на групи.

*Числові типи даних:*

BIT: зберігає значення 0 або 1. Фактично є аналогом булевого типу в мовах програмування. Займає 1 байт.

TINYINT: зберігає числа від 0 до 255. Займає 1 байт. Добре підходить для зберігання невеликих чисел.

SMALLINT: зберігає числа від -32 768 до 32 767. Займає 2 байта

INT: зберігає числа від -2 147 483 648 до 2 147 483 647. Займає 4 байта. Найбільш використовуваний тип для зберігання чисел.

BIGINT: зберігає дуже великі числа від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, які займають в пам'яті 8 байт.

DECIMAL: зберігає числа c фіксованою точністю. Займає від 5 до 17 байт в залежності від кількості чисел після коми.

Даний тип може приймати два параметри precision і scale: DECIMAL (precision, scale).

Параметр precision представляє максимальну кількість цифр, які може зберігати число. Це значення повинно знаходитися в діапазоні від 1 до 38. За замовчуванням воно дорівнює 18.

Параметр scale представляє максимальну кількість цифр, які може містити число після коми. Це значення повинно знаходитися в діапазоні від 0 до значення параметра precision. За замовчуванням воно дорівнює 0.

NUMERIC: даний тип аналогічний типу DECIMAL.

SMALLMONEY: зберігає дробові значення від -214 748.3648 до 214 748.3647. Призначено для зберігання грошових величин. Займає 4 байти. Еквівалентний типу DECIMAL (10,4).

MONEY: зберігає дробові значення від -922 337 203 685 477.5808 до 922 337 203 685 477.5807. Являє грошові величини і займає 8 байт. Еквівалентний типу DECIMAL (19,4).

FLOAT: зберігає числа від -1.79E + 308 до 1.79E + 308. Займає від 4 до 8 байт в залежності від дробової частини.

Може мати форму опредеенія у вигляді FLOAT (n), де n представляє число біт, які використовуються для зберігання десяткової частини числа (мантиси). За замовчуванням n = 53.

REAL: зберігає числа від -340E + 38 to 3.40E + 38. Займає 4 байти. Еквівалентний типу FLOAT (24).

*Типи даних, що представляють дату і час:*

DATE: зберігає дати від 0001-01-01 (1 січня 0001 року) до 9999-12-31 (31 грудня 9999 року). Займає 3 байта.

TIME: зберігає час в діапазоні від 00: 00: 00.0000000 до 23: 59: 59.9999999. Займає від 3 до 5 байт.

Може мати форму TIME (n), де n представляє кількість цифр від 0 до 7 в дробової частини секунд.

DATETIME: зберігає дати і час від 01/01/1753 до 31/12/9999. Займає 8 байт.

DATETIME2: зберігає дати і час в діапазоні від 01/01/0001 00: 00: 00.0000000 до 31/12/9999 23: 59: 59.9999999. Займає від 6 до 8 байт в залежності від точності часу.

Може мати форму DATETIME2 (n), де n представляє кількість цифр від 0 до 7 в дробової частини секунд.

SMALLDATETIME: зберігає дати і час в діапазоні від 01/01/1900 до 06/06/2079, тобто найближчі дати. Займає від 4 байта.

DATETIMEOFFSET: зберігає дати і час в діапазоні від 0001-01-01 до 9999-12-31. Зберігає детальну інформацію про час з точністю до 100 наносекунд. Займає 10 байт.

Поширені формати дат:

- yyyy-mm-dd - 2017-07-12

- dd / mm / yyyy - 12/07/2017

- mm-dd-yy - 07-12-17

Поширені формати часу:

- hh: mi - 13:21

- hh: mi am / pm - 1:21 pm

- hh: mi: ss - 1:21:34

- hh: mi: ss: mmm - 1: 21: 34: 12

- hh: mi: ss: nnnnnnn - 1: 21: 34: 1234567

*Рядкові типи даних:*

CHAR: зберігає рядок довжиною від 1 до 8 000 символів. На кожен символ виділяє по 1 байту. Не підходить для багатьох мов, так як зберігає символи не в кодуванні Unicode.

Кількість символів, яке може зберігати стовпець, передається в дужках. Наприклад, для стовпця з типом CHAR (10) буде виділено 10 байт. І якщо ми збережемо в стовпці рядок менше 10 символів, то вона буде доповнена пробілами.

VARCHAR: зберігає рядок. На кожен символ виділяється 1 байт. Можна вказати конкретну довжину для стовпця - від 1 до 8 000 символів, наприклад, VARCHAR (10). Якщо рядок повинен мати більше 8000 символів, то задається розмір MAX, а на зберігання рядка може виділятися до 2 Гб: VARCHAR (MAX).

Не підходить для багатьох мов, так як зберігає символи не в кодуванні Unicode.

На відміну від типу CHAR якщо в стовпець з типом VARCHAR (10) буде збережена рядок в 5 символів, то в Столц буде збережено саме п'ять символів.

NCHAR: зберігає рядок в кодуванні Unicode завдовжки від 1 до 4 000 символів. На кожен символ виділяється 2 байти. Наприклад, NCHAR (15).

Ще два типу TEXT і NTEXT є застарілими і тому їх не рекомендується використовувати. Замість них застосовуються VARCHAR і NVARCHAR відповідно.

BINARY: зберігає бінарні дані у вигляді послідовності від 1 до 8 000 байт.

VARBINARY: зберігає бінарні дані у вигляді послідовності від 1 до 8 000 байт, або до 2 ^ 31-1 байт при використанні значення MAX (VARBINARY (MAX)).

Ще один бінарний тип - тип IMAGE є застарілим, і замість нього рекомендується застосовувати тип VARBINARY.

*Решта типів даних*

UNIQUEIDENTIFIER: унікальний ідентифікатор GUID (по суті рядок з унікальним значенням), який займає 16 байт.

TIMESTAMP: певна кількість, яке зберігає номер версії рядки в таблиці. Займає 8 байт.

CURSOR: представляє набір рядків.

HIERARCHYID: представляє позицію в ієрархії.

SQL\_VARIANT: може зберігати дані будь-якого іншого типу даних T-SQL.

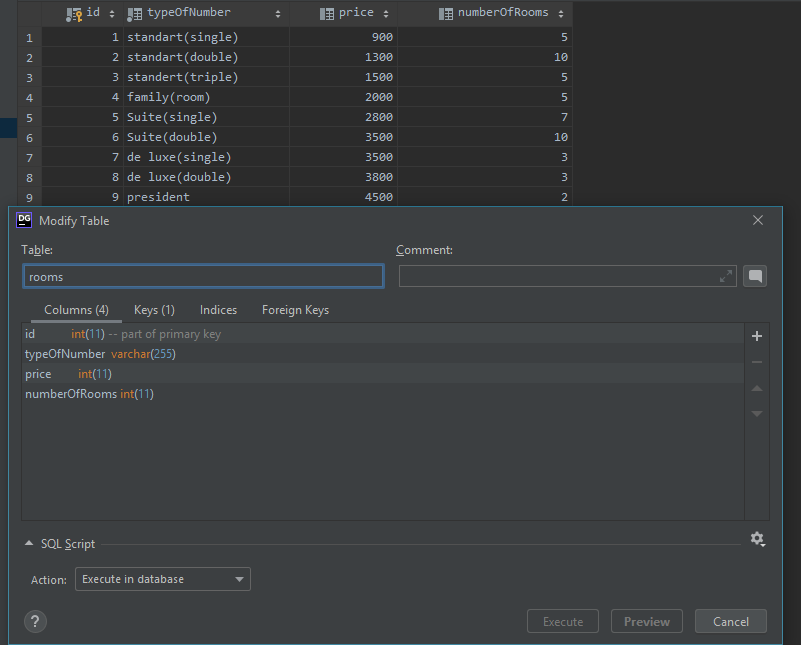
XML: зберігає документи XML або фрагменти документів XML. Займає в пам'яті до 2 Гб.

TABLE: представляє визначення таблиці.

GEOGRAPHY: зберігає географічні дані, такі як широта і довгота.

GEOMETRY: зберігає координати місцезнаходження на площині.

Для прикладу розглянемо типи даних таблиці rooms. Поле id – int, typeOf-Number – varchar, price – int, numberOfRooms – int(Рис. 2.6).

****

*Рис.2.6 Типи даних таблиці rooms*

**2.5 Обмеження цілісності даних;**

Нормалізовані реляційні відношення отримані в попередній главі не містять атрибутів, які є зовнішніми ключами, тобто реалізують зв'язки між відношеннями. Зовнішні ключі слід позначати як FK і пам'ятати, що тип данних зовнішнього ключа має співпадати з типом даних первинного ключа пов'язаного відношення. Назви зовнішніх ключів також варто давати у відповідності до назв пов'язаних відношень.

Для атрибутів реляційних відношень необхідно визначити:

* зазначення типів даних та їхніх розмірів (обов'язкова властивість);
* чи є обов'язковим значення атрибута;
* чи може значення атрибута дублюватися;
* можливі значення за замовчуванням.

Ці данні є важливим елементом забезпечення цілісності атрибутів рляційних відношень.

При роздобці бази даних для готелю використовувалось: primary key, foregion key, unique.

2.6 Реалізація SQL-скрипту;

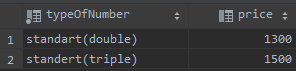
Sql-скрипт – це програмний код, який реалізує запитити, оновлення, видалення, тощо в базу даних.

Sql-скрипти можуть бути різноманітні. Ось декілка для прикладу:

# 1)Показати всі номери які коштують дороще 1000 і дешевше 1500

select typeOfNumber, price from rooms

where price between 1000 and 1500;



*Рис.2.7 Результат запиту* *1*

# 2)Показати всіх працівників готелю;

SELECT \* from workers;

#3)Показати імя і телефон адміністраторів і директора готелю

select FirstName, phone, Position from workers where Position ='administrator' or Position='director';

#4) Показати перших 10 клієнтів по алфафіну

select \* from clients order by Name asc limit 10;

#5)Показати клієнта чий номер закінчується на 15

select \*from clients where Phone like '%15';

#6)Показати імя номер телефону працівників:Ostap Rosar і OLga Verba

select FirstName, phone from workers

where FirstName like 'Ostap'and LastName like 'Rosar'

or FirstName like 'OLga'and LastName like 'Verba';

#7)показати ті номери і ціни на них, які коштуюте дороще 3000;

select typeOfNumber, price from rooms where price> 3000;

#8) Показати найдорощий номер готелю

select \*, Max(price) from rooms;

# 9)Показати зп всіх працівників

elect FirstName, position, sizeOfSalary from workers

join salary s on workers.id = s.id\_worker

order by id asc ;

# 10)Показати середню зарплату покоївок

select avg(sizeOfSalary) from workers

join salary s on workers.id = s.id\_worker

where position like 'chambermaid'

order by id;

#11)Показати замовлення клієнтів

select name , phone, id\_typeOfNumber from clients

join orders o on clients.id = o.id

order by name;

**Висновки**

У процесі виконання дослідження мета була досягнута, а завдання вирішені, а саме успішно виконано теоретичні і практичне завдання, (побудова бази даних для готелю.)

В першому розділі роботи було проаналізовано теоретичні питання дослідження, а саме:

* моделі варіантів використання веб-сайту;
* техніко-економічне обґрунтування вибору технологій для бази даних.

У другій главі даної роботи було розглянуто проаналізовано та описано основні принципи та особливості створення бази даних для туристично-готельного підприємства.

Результати проведеного дослідження дають підставу зробити такі висновки:

* На основі власного аналізу та опрацтованих джерел літератури, можна стверджувати про актуальність створення інформаційної системи для готелю;
* Інофрмаційна система повинна місти в собі не тільки фронтенд частину, але й бекенд, зокрема варто приділити увагу базі даних, оскільки це зручний, автоматизований спосіб обробки та маніполювання даними.

Варто зазначити, що у ході виконання роботи мета роботи досягнена, а поставлені завдання виконані. Оскільки у даній роботі було проведено комплексний аналіз бази даних для конкретного готелю, а саме - описано модель сутність-зв'язок, описно характеристику зв'язків, класифікацію сутностей, структуру первинних і зовнішніх ключів, визначено поняття цілісності даних.

Підсумовуючи усе вище зазначене, можна стверджувати, що доцільно створювати веб-сайт(інформаційну систему для готелю), оскільки це надасть безліч можливостей та переваг.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Линн Бейли: Изучаем SQL; Видавництво «O'Reilly», 2012. -623 с.
2. Paul Weinberg, James Groff, Andrew Oppel SQL The Complete Reference, Third Edition; « The McGraw-Hill Companies», 2010. – 882 c.
3. Переваги MySQl [Електронний ресурс]. – Режим доступу: <http://www.znannya.org/?view=concept:304>.
4. Моделі даних та їх класифікація [Електронний ресурс]. – Режим доступу: <https://wiki.cuspu.edu.ua/index.php/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%96_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_%D1%82%D0%B0_%D1%97%D1%85_%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F.>.
5. Нормалізація відношень БД [Електронний ресурс]. – Режим доступу: <https://life-prog.ru/ukr/1_331_normalizatsiya-vidnoshen.html>.
6. Типи данихSQL [Електронний ресурс]. – Режим доступу: <https://metanit.com/sql/sqlserver/3.3.php>.