**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”**

**на тему:**

**«Інформаційна система страхової компанії»**

Галузь знань: 05 «Соціальні та поведінкові науки»

Спеціальність: 051 «Економіка»

Спеціалізація: «Інформаційні технології в бізнесі»

Освітній ступінь: бакалавр

**Науковий керівник:         Виконавець:**

к.ф.-м.н., доц. Депутат Б.Я.Труш Андрій Григорович

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

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

                                                                                \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис)

**“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.                               “\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.**

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

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

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

(підпис, ПІП членів комісії)

**Зміст**

[РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 5](#_Toc73007070)

[1.1. Постановка задачі 5](#_Toc73007071)

[1.2. Розробка моделі 5](#_Toc73007072)

[1.3. Аналіз засобів реалізації 5](#_Toc73007073)

[РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ 8](#_Toc73007074)

[2.1. Що таке реляційні бази даних? 8](#_Toc73007075)

[2.2. Принципи нормалізації баз даних та логічні зв’язки 9](#_Toc73007076)

[2.3. Опис моделі даних 12](#_Toc73007077)

[2.4. Визначення типів даних 13](#_Toc73007078)

[ВИСНОВОК 18](#_Toc73007084)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc73007085)

Вступ

**Актуальність проблеми.**

В даний час швидко розвиваються нові інформаційні технології (ІТ), які тягнуть за собою корінні зміни не тільки у виробничих галузях, але і в сфері страхування.Страхування це одна з форм охорони майнових й особистих інтересів [застрахованих](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA), які сплачують [внески (премії)](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BF%D0%BB%D0%B0%D1%82%D1%96%D0%B6) до [страхових організацій](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%B8%D0%BA) і дістають [відшкодування втрат](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%B5_%D0%B2%D1%96%D0%B4%D1%88%D0%BA%D0%BE%D0%B4%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), що виникають у наслідок [стихійного лиха](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B8%D1%85%D1%96%D0%B9%D0%BD%D0%B5_%D0%BB%D0%B8%D1%85%D0%BE) і [нещасних випадків](https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D1%89%D0%B0%D1%81%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B8%D0%BF%D0%B0%D0%B4%D0%BE%D0%BA) (страхування від [недуг](https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B4%D1%83%D0%B3%D0%B0), [страхування від вогню](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B2%D1%96%D0%B4_%D0%B2%D0%BE%D0%B3%D0%BD%D1%8E) ([пожеж](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B6%D0%B5%D0%B6%D0%B0)), [крадіжок](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D0%B4%D1%96%D0%B6%D0%BA%D0%B0), [страхування життя](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B6%D0%B8%D1%82%D1%82%D1%8F) та ін.). З другого боку, страхування є формою [ризик-менеджменту](https://uk.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B7%D0%B8%D0%BA-%D0%BC%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%BC%D0%B5%D0%BD%D1%82), що використовується для [хеджування](https://uk.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B4%D0%B6%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) проти ризику фінансових втрат, та, в ідеалі, страхування може бути визначене як справедлива передача [ризику](https://uk.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B7%D0%B8%D0%BA) потенційної втрати від однієї сторони до іншої за відповідну сплату. Важливою умовою при цьому є дотримання [правил страхування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D0%B0_%D1%81%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F).

У країнах вільної [ринкової економіки](https://uk.wikipedia.org/wiki/%D0%A0%D0%B8%D0%BD%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B5%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D1%96%D0%BA%D0%B0) існують тепер головні два роди страхування:

* обов'язкове публічно-правове, організація якого зосереджена в руках публічно-правових, а подекуди й державних страхових організацій (усе [соціальне страхування](https://uk.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D1%81%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), а в деяких країнах також страхування [будівель](https://uk.wikipedia.org/wiki/%D0%91%D1%83%D0%B4%D1%96%D0%B2%D0%BB%D1%8F) від [вогню](https://uk.wikipedia.org/wiki/%D0%92%D0%BE%D0%B3%D0%BE%D0%BD%D1%8C) тощо);
* приватне, яким опікуються [акціонерні товариства](https://uk.wikipedia.org/wiki/%D0%90%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B5%D1%80%D0%BD%D0%B5_%D1%82%D0%BE%D0%B2%D0%B0%D1%80%D0%B8%D1%81%D1%82%D0%B2%D0%BE) й [товариства взаємного страхування](https://uk.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B2%D0%B0%D1%80%D0%B8%D1%81%D1%82%D0%B2%D0%BE_%D0%B2%D0%B7%D0%B0%D1%94%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D1%81%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (страхування майна, життя та ін.) на базі [приватної угоди](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%B3%D0%BE%D0%B2%D1%96%D1%80_%D1%81%D1%82%D1%80%D0%B0%D1%85%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) між застрахованим і страхувальною організацією, якій застрахований зобов'язаний виплачувати встановлені [угодою](https://uk.wikipedia.org/wiki/%D0%A3%D0%B3%D0%BE%D0%B4%D0%B0) внески, а страхувач при виникненні зазначеного у договорі випадку відшкодувати застрахованому втрату або виплатити обумовлену договором суму. З уваги на те, що [страховий ризик](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%B8%D0%B9_%D1%80%D0%B8%D0%B7%D0%B8%D0%BA) не можна передбачити, страхувальні організації повинні створити відповідні [резерви](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D0%B5%D1%80%D0%B2), встановлюючи належну висоту внесків.

Кожна страхова компанія повинна зберігати свої дані тому виникає потреба у створенні інформаційної системи для неї.

**Мета та завдання.** Метою даної роботи є обгрунтування теоретичних основ та реалізації інформаційної системи. Для досягнення мети курсової роботи, необхідно вирішити такі завдання:

* коротко охарактеризувати інформаційну систему для страхової компанії;
* проаналізувати предметну область;
* спроектувати структуру бази даних;

**Об’єкт дослідження**. Особливість розробки інформаційної системи для страхової компанії за допомогою системи управління базами даних MySQL.

**Практичне значення отриманих результатів.** База даних може використовуватись юридичною особою, яка хоче розмістити інформацію в комп’ютері для подальшого використання.

**Використане програмне забезпечення**. В процесі розробки використовуватимуться наступні технології:

* MySQL - вільна система керування реляційними базами даних.

**Структура роботи**. Курсова робота складається з двох розділів(«Аналіз предметної області», «Розробка бази даних»), висновків, списку використаних джерел та додатків.

Загальний обсяг роботи − сторінок

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

**1.1. Постановка задачі**

Основним завданням даної роботи є створення повноцінної інформаційної системи для страхової компанії. Функціональна структура ІС має орієнтуватися на ті інформаційні потреби кінцевих користувачів, які змінюються в умовах ринку, та відображати зміст і специфіку функцій управління конкретним економічним об’єктом. Вимоги, які пред’являються користувачами до ІС, це: гнучка структура і відкритість системи, тобто можливість вносити необхідні зміни у розроблену модель і забезпечувати нарощування функціональних можливостей в міру необхідності. Отож, в даній роботі нам слід створити ІС для страхової компанії у вигляді бази даних.

**1.2. Розробка моделі**

База даних страхової компанії має один тип керування :

* Адміністратор;

Він має змогу наповнювати БД системи новою інформацією, тобто даними про курси, що з’явилися, їх опис тощо, а також додавати чи видаляти користувачів.

**1.3. Аналіз засобів реалізації**

Для реалізації даної роботи було вибрано технологію MySQL:

MySQL — вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам.

Зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. MySQL надає багатий набір функціональних можливостей, які підтримують безпечне середовище для зберігання, обслуговування і отримання даних.

MySQL — характеризується великою швидкістю, стійкістю і простотою використання, була розроблена для підвищення швидкодії обробки великих баз даних.

Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNІХ-системах, де є підтримка багатоканальності, що підвищує продуктивність системи в цілому.

Для некомерційного використання MySQL є безкоштовним.

Можливості сервера MySQL:

1. простота у встановленні та використанні;
2. підтримується необмежена кількість користувачів, що одночасно працюють із БД;
3. кількість рядків у таблицях може досягати 50 млн.;
4. висока швидкість виконання команд;
5. наявність простої і ефективної системи безпеки.

Основні переваги MySQL. Серед основних переваг MySQL відзначають наступні:

* Масштабованість. MySQL може підтримувати роботу БД значних розмірів, що підтверджують її реалізації в Yahoo!, Google,HP, Associated Press. Згідно документації, що додається до MySQL, деякі БД, що використовуються компанією MySQL АВ (розробником MySQL), зберігають до 50 млн. записів.
* Переносність. MySQL працює на різних платформах, серед яких Unix, Linux, Windows, OS/2, Solaris, Mac OC. Окрім того, MySQL працює на різних платформах.
* Зв’язаність. MySQL має мережеву структуру. До MySQL можна одержувати доступ із будь-якої точки Internet кільком користувачам одночасно. MySQL має цілий ряд програмних інтерфейсів додатків (Application Programming Interface –API ), які дозволяють встановлювати з’єднання з MySQL із додатків, написаних на таких мовах як С, С++, Perl, PHP, Java, Python.
* Бeзпека. MySQL має систему контроля доступу до даних, забезпечує шифрування даних при передаванні.
* Швидкість функціонування.
* Зручність експлуатації. MySQL досить зручно встановлюється та реалізується, легко адмініструється.
* Відкритий код.

**РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ**

**2.1. Що таке реляційні бази даних**

**База даних** — це інтегрована сукупність структурованих і взаємозалежних даних, організована за певними правилами, які передбачають загальні принципи опису, зберігання і обробки даних.

**Реляційна (від анг. relation) база даних** — це тип бази даних, що зберігає інформацію в електронних таблицях і здійснює пошук даних в одній таблиці на підставі визначених ключових полів іншої таблиці. Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних. Використання реляційних БД було запропоноване Едгаром Коддом в 1970 році.

Для того щоб створити власну СУБД, слід скористатися одним з інструментів моделювання, продумати, з якою інформацією вам необхідно працювати, спроектувати таблиці і реляційні одно- і множинні зв'язки між даними, заповнити осередки сутностей і встановити первинний, зовнішні ключі.

Моделювання таблиць і проектування реляційних баз даних здійснюється за допомогою безкоштовних інструментів, таких як Workbench, PhpMyAdmin, Case Studio, dbForge Studio. Після детального проектування слід зберегти графічно готову реляційну модель і перевести її в готовий SQL-код. На цьому етапі можна починати роботу з сортуванням даних, їх обробку і систематизацію.

Реляційна база даних складається з наступних компонентів:

1. Сутність. Таблиця реляційної бази даних може бути одна, а може бути цілий набір з таблиць, які характеризують описані об'єкти. У них фіксована кількість полів і змінне число записів. Таблиця реляційної моделі баз даних складається з рядків, атрибутів і макета.
2. Запис - змінне число рядків, що відображають дані, що характеризують описуваний об'єкт. Нумерація записів проводиться системою автоматично.
3. Атрибути - дані, що демонструють собою опис стовпців суті.
4. Поле. Являє собою стовпець суті. Їх кількість - фіксована величина, що встановлюється під час створення або зміни таблиці.

Тепер, знаючи складові елементи таблиці, можна переходити до властивостей реляційної моделі баз даних:

* Суті реляційної БД двовимірні. Завдяки цій властивості з ними легко проробляти різні логічні і математичні операції.
* Порядок проходження значень атрибутів і записів в реляційній таблиці може бути довільним.
* Стовпець в межах однієї реляційної таблиці повинен мати свою індивідуальну назву.
* Всі дані в стовпці суті мають фіксовану довжину і однаковий тип.
* Будь-який запис по суті вважається одним елементом даних.
* Складові компоненти рядків єдині в своєму роді. У реляційної сутності відсутні однакові рядки.

Виходячи з властивостей реляційної СУБД, зрозуміло, що значення атрибутів повинні бути однакового типу, довжини.

**2.2. Принципи нормалізації баз даних та логічні зв’язки**

Нормалізація - це процес ефективної організації даних у базі даних. Є дві цілі процесу нормалізації: усунення зайвих даних (наприклад, зберігання одних і тих же даних у більш ніж одній таблиці) та забезпечення залежності даних має сенс (лише зберігання пов’язаних даних у таблиці). Обидва ці цілі - це гідні цілі, оскільки вони зменшують обсяг місця, який займає база даних, і забезпечують логічне зберігання даних.

Спільнота баз даних розробила низку вказівок щодо забезпечення нормалізації баз даних. Вони називаються нормальними формами і нумеруються від однієї (найнижча форма нормалізації, яка називається першою нормальною формою або 1NF) через п'ять (п'ята нормальна форма або 5NF). У практичних програмах часто зустрічаються 1NF, 2NF і 3NF, а також епізодичні 4NF. П'ята нормальна форма зустрічається дуже рідко .

Важливо зазначити, що принципи нормалізації є лише вказівками. Іноді виникає необхідність відхилятися від них, щоб задовольнити практичні вимоги бізнесу. Однак коли виникають зміни, надзвичайно важливо оцінити всі можливі наслідки, які вони могли б мати у вашій системі, та врахувати можливі невідповідності.

Перша нормальна форма (1NF) встановлює дуже основні правила для організованої бази даних:

* Видалити дублікати стовпців з тієї ж таблиці.
* Створити окремі таблиці для кожної групи пов’язаних даних та визначити кожен рядок унікальним стовпцем або набором стовпців (первинний ключ).

Друга нормальна форма (2NF) додатково стосується концепції видалення дублюючих даних:

* Відповідати всім вимогам першої нормальної форми.
* Видалити підмножини даних, які застосовуються до декількох рядків таблиці, і помістити їх в окремі таблиці.
* Створити зв’язки між цими новими таблицями та їх попередниками за допомогою використання зовнішніх ключів.

Третя нормальна форма (3NF) йде на один великий крок далі:

* Відповідати всім вимогам другої нормальної форми.
* Видалити стовпці, які не залежать від первинного ключа.

Нормальна форма Бойса-Кодда додає ще одну вимогу:

* Відповідати всім вимогам третьої нормальної форми.
* Кожна детермінанта повинна бути ключовим кандидатом.

Нарешті, четверта нормальна форма (4NF) має ще одну додаткову вимогу:

* Відповідати всім вимогам третьої нормальної форми.
* Відношення є в 4NF, якщо воно не має багатозначних залежностей.

Ці норми щодо нормалізації є сукупними. Щоб база даних знаходилася в 2NF, вона повинна спочатку виконати всі критерії бази 1NF.

Хоча нормалізація бази даних часто є хорошою ідеєю, це не є абсолютною вимогою. Насправді, є деякі випадки, коли навмисне порушення правил нормалізації є хорошою практикою.

Зв’язок встановлюється між двома загальними полями (стовпцями) двох таблиць. Існують зв’язки з відношенням «один-до-одного»*,* «один-до-багатьох»і«багато-до-багатьох».

Відносини, які можуть існувати між записами двох таблиць:

1. **один — до — одного** — кожному запису з однієї таблиці відповідає один запис у іншій таблиці;
2. **один — до — багатьох** — кожному запису з однієї таблиці відповідає кілька записів у іншій таблиці;
3. **багато — до — одного** — безлічі записів з однієї таблиці відповідає один запис у іншій таблиці;
4. **багато — до — багатьох** — безлічі записів з однієї таблиці відповідає кілька записів в іншій таблиці.

Відношення «один-до-багатьох» створюється в тому випадку, коли тільки одне з полів є полем первинного ключа або унікального індексу. Відношення «один-до-одного» створюється в тому випадку, коли обидва поля є ключовими або мають унікальні індекси. Відношення «багато-до-багатьох» фактично є двома відносинами «один-до-багатьох» з третьої таблицею, первинний ключ якої складається з полів зовнішнього ключа двох інших таблиць.

**Ключ** — це стовпець (може бути декілька стовпців), що додається до таблиці і дозволяє встановити зв’язок із записами в іншій таблиці. Існують ключі двох типів: первинні і вторинні (зовнішні). **Первинний ключ** — це одне або кілька полів (стовпців), комбінація значень яких однозначно визначає кожний запис у таблиці. Первинний ключ не допускає значень null і завжди повинен мати унікальний індекс. Первинний ключ використовується для зв’язування таблиці з зовнішніми ключами в інших таблицях. **Зовнішній (вторинний) ключ** — це одне або кілька полів (стовпців) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об’єднання таблиць.

З двох логічно пов’язаних таблиць одну називають таблицею первинного ключа або головною таблицею, а іншу таблицею вторинного (зовнішнього) ключа або підпорядкованою таблицею. СУБД дозволяють зіставити споріднені записи з обох таблиць і спільно вивести їх у формі, звіті або запиті.

**2.3. Опис моделі даних**

Для розробки своєї бази даних було обрано саме реляційну модель через ряд її переваг, а саме:

1. вона проста і доступна для розуміння користувачем; єдиною використовуваною інформаційною конструкцією є «таблиця»;
2. суворі правила проектування, які базуються на математичному апараті;
3. повна незалежність даних; зміни в прикладній програмі при зміні реляційної БД мінімальні;
4. для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті.

У ході практичного завдання була побудована наступна модель бази даних (Рис.2.3.1).

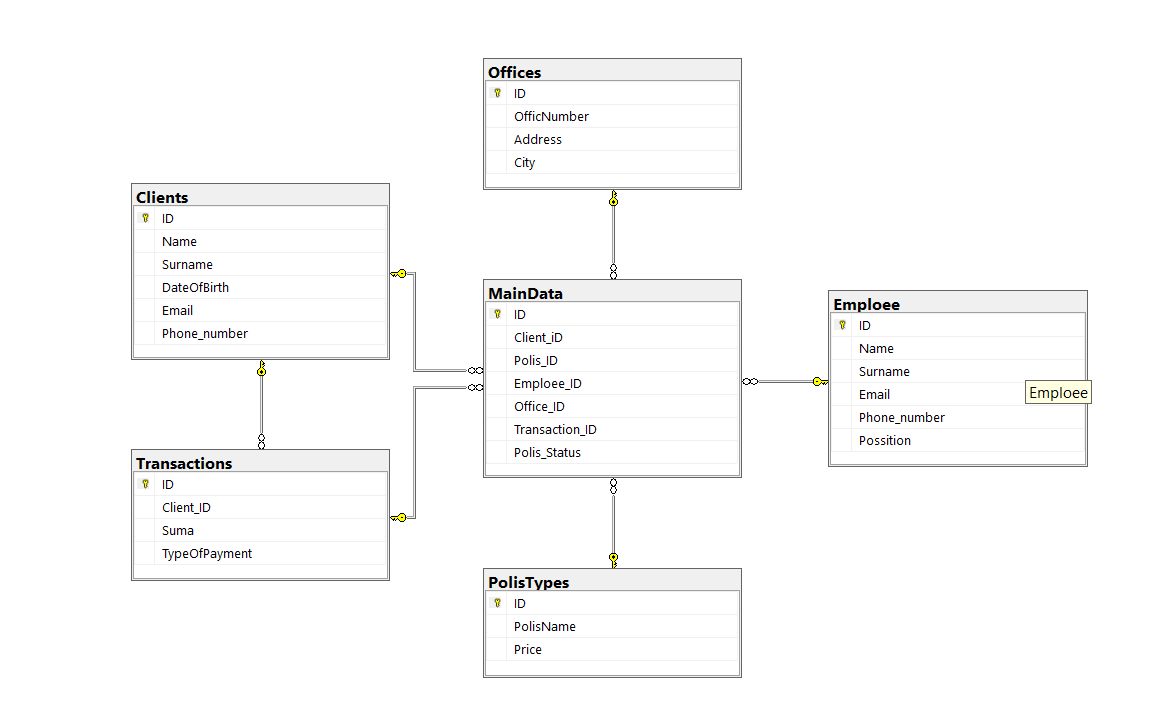


Рис. 2.3.1. Діаграма бази даних

Можна побачити, що дана модель складається з шістьох таблиць: MainData, Clients, Transactions, PolisTypes, Emploee, Offices.

**2.4. Визначення типів даних**

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

Числові типи даних:

* serial: представляє числове значення, яке автоінкрементується та займає 4 байти і може зберігати числа від 1 до 2147483647; значення даного типу утворюється шляхом автоінкремента значення попереднього рядка; тому, як правило, даний тип використовується для визначення ідентифікаторів рядків;
* smallserial: представляє числове значення, яке автоінкрементується та займає 2 байти і може зберігати числа від 1 до 32767; аналог типу serial для невеликих чисел;
* bigserial: представляє числове значення, яке автоінкрементується та займає 8 байтів і може зберігати числа від 1 до 9223372036854775807; аналог типу serial для великих чисел;
* smallint: зберігає числа від -32768 до +32767; займає 2 байти; має псевдонім int2;
* integer: зберігає числа від -2147483648 до +2147483647; займає 4 байти; має псевдоніми int і int4;
* bigint: зберігає числа від -9223372036854775808 до +9223372036854775807; займає 8 байтів; має псевдонім int8;
* numeric: зберігає числа з фіксованою точністю, які можуть мати до 131072 знаків в цілій частині і до 16383 знаків після коми; даний тип може приймати два параметри precision і scale: numeric (precision, scale); параметр precision вказує на максимальну кількість цифр, які може зберігати число; 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; наприклад, для числа 23.5141 precision дорівнює 6, а scale - 4.
* decimal: зберігає числа з фіксованою точністю, які можуть мати до 131072 знаків в цілій частині і до 16383 знаків у дробовій частині; те ж саме, що і numeric.
* real: зберігає числа з плаваючою точкою з діапазону від 1E-37 до 1E + 37. Займає 4 байти; має псевдонім float4.
* double precision: зберігає числа з плаваючою точкою з діапазону від 1E-307 до 1E + 308; займає 8 байт; має псевдонім float8.

Символьні типи:

* character (n): представляє рядок з фіксованої кількості символів; за допомогою параметра задається кількість символів в рядку; має псевдонім char (n);
* character varying (n): представляє рядок з фіксованої кількості символів; за допомогою параметра задається кількість символів в рядку; має псевдонім varchar (n);
* text: представляє текст довільної довжини.

Для роботи с грошовими одиницями визначений тип money, який може приймати значення в діапазоні від -92233720368547758.08 до +9223372036854758.07 і займає 8 байтів.

Для зберігання бінарних даних визначено тип bytea. Він зберігає дані у вигляді бінарних рядків, які представляють послідовність октетів або байт.

Типи для роботи з датами і часом:

* timestamp: зберігає дату і час; займає 8 байтів; для дат найнижче значення - 4713 р. до н.е., найбільш верхнє значення - 294 276 р. н.е.;
* timestamp with time zone: те ж саме, що і timestamp, тільки додає дані про часовий пояс;
* date: представляє дату від 4713 р. до н.е. до 5874897 р. н.е.; займає 4 байти;
* time: зберігає час з точністю до 1 мікросекунди без вказівки часового поясу; приймає значення від 00:00:00 до 24:00:00; займає 8 байтів;
* time with time zone: зберігає час з точністю до 1 мікросекунди із зазначенням часового поясу; приймає значення від 00:00:00 +1459 до 24:00:00-1459; займає 12 байт;
* interval: представляє часовий інтервал; займає 16 байт.

Логічний тип boolean може зберігати одне з двох значень: true або false. Замість true можна вказувати наступні значення: TRUE, 't', 'true', 'y', 'yes', 'on', '1'. Замість false можна вказувати наступні значення: FALSE, 'f', 'false', 'n', 'no', 'off', '0'.

Типи для подання інтернет-адрес:

* cidr: інтернет-адреса в форматі IPv4 і IPv6 (наприклад, 192.168.0.1.); займає від 7 до 19 байтів;
* inet: інтернет-адреса в форматі cidr / y, де cidr це адреса в форматі IPv4 або IPv6, а /y - кількість біт в адресі (якщо цей параметр не вказано, то використовується 34 для IPv4, 128 для IPv6) (наприклад, 192.168.0.1/24 або 2001: 4f8: 3: ba: 2e0: 81ff: ​​fe22: d1f1 / 128); займає від 7 до 19 байтів.
* macaddr: зберігає MAC-адресу; займає 6 байтів;
* macaddr8: зберігає MAC-адресу у форматі EUI-64; займає 8 байтів.

Геометричні типи:

* point: представляє точку на площині в форматі (x, y); займає 16 байтів;
* line: представляє лінію невизначеної довжини в форматі {A, B, C}; займає 32 байти;
* lseg: представляє відрізок в форматі ((x1, y1), (x2, y2)); займає 32 байти;
* box: представляє прямокутник в форматі ((x1, y1), (x2, y2)); займає 32 байти;
* path: представляє набір з’єднаних точок; у форматі ((x1, y1), ...) шлях є закритим (перша і остання точка з'єднуються лінією) і фактично являє багатокутник; у форматі [(x1, y1), ...] шлях є відкритим; займає 16 + 16n байтів;
* polygon: представляє многокутник в форматі ((x1, y1), ...); займає 40 + 16n байтів;
* circle: представляє круг в форматі <(x, y), r>; займає 24 байти.

Решта типів даних:

* json: зберігає дані json в текстовому вигляді;
* jsonb: зберігає дані json в бінарному форматі;
* uuid: зберігає універсальний унікальний ідентифікатор (UUID), наприклад, a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11, займає 32 байти;
* xml: зберігає дані в форматі XML.

Склад таблиць бази даних:

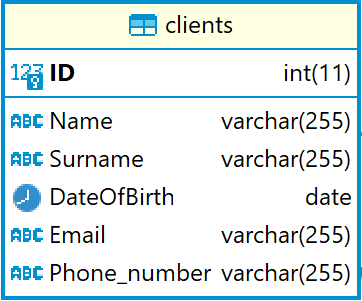


Рис. 1.1 Типи даних таблиці clients

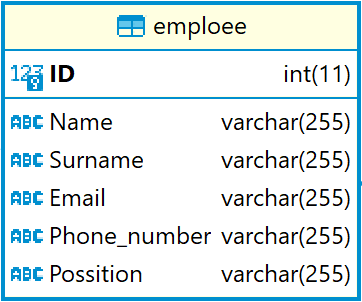


Рис. 1.2 Типи даних таблиці emploee

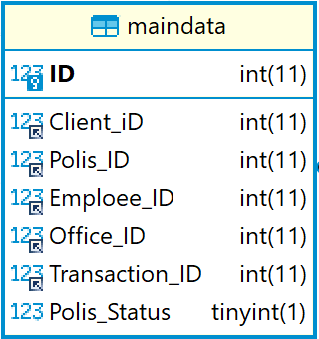


Рис.1.3 Типи даних таблиці maindata

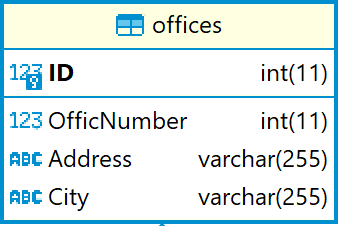


Рис. 1.4 Типи даних таблиці offices

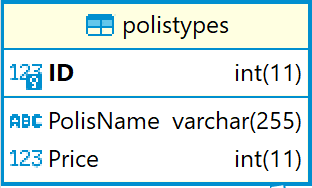


Рис. 1.5 Типи даних таблиці polistypes

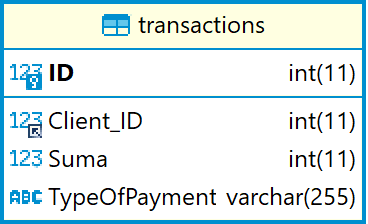


Рис. 1.6 Типи даних таблиці transactions

Реалізація SQL-скрипту

Код створення таблиць:

create table Clients(

ID INT PRIMARY KEY IDENTITY(1,1),

Name varchar(255),

Surname varchar(255),

DateOfBirth date,

Email varchar(255),

Phone\_number varchar(255)

)

create table PolisTypes(

ID int PRIMARY KEY IDENTITY(1,1),

PolisName varchar(255),

Price int

)

create table Offices(

ID int PRIMARY KEY IDENTITY(1,1),

OfficNumber int,

Address varchar(255),

City varchar(255)

)

create table Transactions(

ID int PRIMARY KEY IDENTITY(1,1),

Client\_ID int FOREIGN KEY REFERENCES Clients(ID),

Suma int,

TypeOfPayment varchar(255)

)

create table Emploee(

ID int PRIMARY KEY IDENTITY(1,1),

Name varchar(255),

Surname varchar(255),

Email varchar(255),

Phone\_number varchar(255),

Possition varchar(255)

)

create table MainData(

ID int PRIMARY KEY IDENTITY(1,1),

Client\_iD int FOREIGN KEY REFERENCES Clients(ID),

Polis\_ID int FOREIGN KEY REFERENCES PolisTypes(ID),

Emploee\_ID int FOREIGN KEY REFERENCES Emploee(ID) ,

Office\_ID int FOREIGN KEY REFERENCES Offices(ID),

Transaction\_ID int FOREIGN KEY REFERENCES Transactions(ID),

Polis\_Status bool

)

Код заповнення таблиць:

INSERT INTO Clients VALUES

('Sndrew', 'Priyma', '1999-01-01', 'fsdhgfjds@gmail.com', '156745164'),

('Petro', 'Gentosh', '1998-02-02', 'andjsqwe@gmail.com', '0938854122'),

('Maksym', 'Priymak', '1988-12-02', 'sadgiimjs@gmail.com', '0934444444'),

('Piter', 'Yrkosh', '2000-11-09', 'anshdweee@gmail.com', '4651123345'),

('Yanosh', 'Krop', '1999-02-02', 'ansdnsand@gmail.com', '465423488'),

('Denys', 'Smak', '2001-10-25', 'smmndjkaa@gmail.com', '0884545258'),

('Krap', 'Mircha', '1992-02-26', 'anminder@gmail.com', '0934445841'),

('Ilki', 'Mirosh', '1988-03-01', 'sdweeeer@gmail.com', '9884141312'),

('Misha', 'Perun', '2000-10-27', 'etygfdsa@gmail.com', '4412233548'),

('Maksym', 'Smakov', '2001-05-17', 'asddwer@gmail.com', '0993658213')

INSERT INTO PolisTypes VALUES

('Avangard', 1500),

('Kanon', 2000),

('Mialka', 1100),

('Lebidka', 2200),

('Korol', 800),

('Anelka', 2250),

('PrincAktov', 4000),

('VIP', 7800),

('Piden', 1900),

('Mars', 5500)

INSERT INTO Offices VALUES

(1, 'Striyska 11', 'Rivne'),

(2, 'Anishkova 88', 'Kharkiv'),

(3, 'Chornovola 177', 'Lviv'),

(4, 'Naukova 25', 'Lviv'),

(5, 'Lypynskogo 55', 'Lviv'),

(6, 'Agutina 11', 'Lviv'),

(7, 'Snopkivska 17', 'Lviv'),

(8, 'Alamova 77', 'Kyiv'),

(9, 'Shevchenka 1', 'Odessa'),

(10, 'Knyazha 7', 'Lviv')

INSERT INTO Transactions VALUES

(1, 1500, 'card'),

(2, 2500, 'card'),

(3, 5000, 'cash'),

(4, 900, 'cash'),

(5, 1100, 'cash'),

(6, 3400, 'card'),

(7, 500, 'card'),

(8, 1300, 'cash'),

(9, 4000, 'cash'),

(10, 4400, 'card')

INSERT INTO Emploee VALUES

('Sergey', 'Prush', 'amkjdsww@gmail.com', '0938854655', '1'),

('Nazar', 'Bereza', 'sadsadsadas@gmail.com', '093888888', '2'),

('Slava', 'Kalina', 'aswwwwthh@gmail.com', '455587412', '3'),

('Rakhim', 'Morkvyak', 'aytress@gmail.com', '088412365', '4'),

('Ignat', 'Suleyman', 'kurman@gmail.com', '251447899', '5'),

('Stas', 'Gabnodik', 'kubiyew@gmail.com', '099899999', '6'),

('John', 'Awekno', 'kalina@gmail.com', '098888889', '7'),

('Ikar', 'Musyk', 'samka@gmail.com', '097745556', '8'),

('Sam', 'Murtyk', 'sasews@gmail.com', '032547895', '9'),

('Ira', 'Bogach', 'irkabogach@gmail.com', '066547888', '10')

INSERT INTO MainData VALUES

(1, 1, 1, 1, 1, 'y'),

(2, 2, 2, 2, 2, 'y'),

(3, 3, 3, 3, 3, 'y'),

(4, 4, 4, 4, 4, 'n'),

(5, 5, 5, 5, 5, 'y'),

(6, 6, 6, 6, 6, 'n'),

(7, 7, 7, 7, 7, 'y'),

(8, 8, 8, 8, 8, 'y'),

(9, 9, 9, 9, 9, 'n'),

(10, 10, 10, 10, 10, 'y')

В даному коді було зображено заповнення кожної з таблиць нашої бази даних, а точніше : Clients, PolisTypes, Offices, Transactions, Emploee, MainData.

Тут зображено декілька запитів до коду:

select \* from clients

Select \* from emploee WHERE possition = 'Admin'

Select TOP 5 from offices where City = 'Lviv'

Select \* from clients where Email = '%@gmail.com' AND Name = 'Piter'

select \* FROM maindata where Client\_ID = 1 and Polis\_status = TRUE

Select TOP 5 from transactions where TypeOfPayment = 'card' DESC

Select \* from emploee where Name = 'John' or Surname = 'Prush'

Select \* from maindata WHERE Client\_ID = 9

Select \* from polistypes where Price < 5000 ORDER BY PolisName

Select clients.Name, clients.Surname ,transactions.Suma From clients INNER JOIN transactions on clients.ID = transactions.Client\_ID

select Polis\_ID from maindata left join clients on maindata.client\_ID = clients.ID

**ВИСНОВОК**

У процесі виконання дослідження мета була досягнута, а завдання вирішені, а саме успішно виконано теоретичні і практичне завдання (побудова інформаційної системи (бази даних) для страхової компанії). Результати проведеного дослідження дають підставу зробити такі висновки:

1) З вище опрацьованих джерел використаної літератури та з власного аналізу інформаційної системи, безумовно можна стверджувати про актуальність бази даних .

2) Особливість будь якої бази даних полягає в її функціоналі. Перш ніж створити базу необхідно детально проаналізувати і зрозуміти для чого вона буде використовуватись.

3) Правильно підібрані технологої для роботи значно полегшують розробку бази даних. В даному випадку, можна стверджувати, що обравши технології MySQL з економлено багато часу в створенні бази даних.

Отож, враховуючи усе вище сказане, можна стверджувати про важливу роль інформаційних систем, зокрема побудована інформаційна система позитивно впливає на розвиток бізнесу.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Мартин Грабер :SQL; Видавництво «Лори», 2016. -643 с.

2. Моррисон М. : Изучаем JavaScript; Видавництво «O`REILLY», 2012. – 606 c.

3. Моделі даних [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%

D1%96\_%D0%B1%D0%B0%D0%B7\_%D0%B4%D0%B0%D0%BD%D0%B8

%D1%85.

4. Spring документація [Електронний ресурс]. – Режим доступу: <https://spring.io/guides/gs/serving-web-content/>

5. MySQl[Електронний ресурс]. – режим доступу: <https://studfile.net/preview/5607354/page:3/> .