МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Кафедра системотехніки

ЗВІТ

з виконання завдань практичного заняття № 1

дисципліни «Проектування високонавантажених систем зберігання даних»

|  |  |  |
| --- | --- | --- |
|  |  | Виконав  студент групи ITКНу-19-2  Суворов Максим Віталійович  Перевірив:  доцент кафедри СТ  Коваленко А.І. |

Харків, 2020

**1 Мета практичного заняття**

Набуття практичних навичок з розробки баз даних на платформі MySQL з таблицями типу MyIsam і InnoDB, з урахуванням особливостей роботи високонавантаженої інформаційної системи зберігання даних.

**2 Постановки задачі згідно з індивідуальним завданням**

Тема індивідуального завдання: ІС "Бронювання номерів готелю"

**Завдання 1.1**. Провести аналіз предметної області високонавантаженої інформаційної системи, що складається із серверної (бази даних) і клієнтської (інтерфейсу доступу до бази даних) частин. Для цього необхідно: – дати характеристику діяльності підприємства (організації), що визначається предметною областю; – описати основні бізнес-процеси, які можна автоматизувати з використанням програмних засобів високонавантаженої інформаційної системи; – визначити основні бізнес-функції високонавантаженої інформаційної системи, що відповідають завданням предметної області; – визначити бізнес-функції інтерфейсу клієнтської частини високонавантаженої інформаційної системи, у припущенні, що функціональна частина реалізується на стороні сервера баз даних MySQL; – визначити бізнес-функції інтерфейсу клієнтської частини високонавантаженої інформаційної системи для повнотекстового пошуку з різними параметрами, обумовленими предметною областю.

**Стисла характеристика діяльності підприємства, згідно з індивідуальним завданням.**

Готельна послуга є результатом безпосередньої взаємодії виконавця (готелю в особі його персоналу) та споживача в процесі її виробництва, а також власної діяльності виконавця, спрямованої на задоволення потреб споживача. Основну функцію виконує підрозділ, що забезпечує надання гостям готелю місця тимчасового проживання та організує їх обслуговування через допоміжні підрозділи готелю або стороні організації. Таким чином, послуги основного підрозділу можна охарактеризувати наступним чином:

− вартість послуг включається в плату за номер;

− споживачами послуг являються мешканці готелю.

**Перелік нормативних документів (законних і підзаконних актів), що регламентують діяльність підприємства:**

- регулювання основних аспектів господарської діяльності складової туристичної індустрії спрямована на поліпшення його тісних критеріїв функціонування і відображена у Законах України «Про туризм», «Про захист прав споживачів», «Про стандартизацію» та ін.; постановах Кабінету Міністрів України щодо програм розвитку цього напряму господарської діяльності;

- «Правила користування готелями й аналогічними засобами розміщення та надання готельних послуг», які затверджені наказом Державної туристичної адміністрації України від 16.03.04 р.

- «Правила обов’язкової сертифікації готельних послуг»;

- ДСТУ 4268:2003 «Послуги туристичні. Засоби розміщування. Загальні вимоги»;  
- ДСТУ4269:2003 «Послуги туристичні. Класифікація готелів»;  
- ДСТУ 4281:2004 «Заклади ресторанного господарства. Класифікація»;  
- ДСТУ 2296-93 «Національний знак відповідності. Форма, розміри, технічні вимоги та правила застосування»;  
- ДСТУ 2462-94 «Сертифікація. Основні поняття. Терміни та визначення»;  
- ДСТУ 3230-95 «Управління якістю та забезпечення якості. Терміни та визначення»;  
- ДСТУ 3498-96 «Система сертифікації УкрСЕПРО. Бланки документів. Форма та опис»;  
- ДСТУ ISO 9004-2-96 «Управління якістю та елементи системи якості. Частина 2. Настанови щодо послуг»;  
- ДК 016-97 «Державний класифікатор продукції та послуг» (ДКПП).

**Перелік програмного забезпечення, необхідного для реалізації трирівневої (триланкової) архітектури «клієнт–сервер» високонавантаженої інформаційної системи:** утиліта командного рядка MySQL Command Line Client; програмний пакет Workbench; програмний засіб Devart dbForge Studio for MySQL, CASE-засіб All Fusion Data Modeler (ERWin)

**Перелік можливих типів користувачів високонавантаженої інформаційної системи, їх статусів.**

Таблиця 1.1 – Перелік статусів можливих типів користувачів

|  |  |  |  |
| --- | --- | --- | --- |
| № | Користувач | Статус | Доступ до даних (Найменування таблиць) |
| 1 | Незареєстрований користувач | Немає | Має доступ до читання таблиці Rooms |
| 2 | Зареєстрований користувач | User | Має доступ до запису й читання таблиць …  Має доступ до читання таблиць … |
| 3 | Адміністратор системи | Admin | Має доступ до запису й читання всіх таблиць БД |

**Перелік бізнес-процесів, що підлягають автоматизації;:**

* Замовлення номеру на вказану кількість осіб, потрібного комфорту на вказані дати
* Реєстрація нових клієнтів

**Перелік бізнес-функцій для кожного типу (статусу) користувача:**

Бізнес-функції системи для незареєстрованих користувачів: реєстрація на сайті, перегляд усіх кімнат. Бізнес-функції системи для зареєстрованих користувачів: перегляд усіх доступних номерів для замовлення; пошук номерів по комфорту; замовлення номерів на вказану кількість осіб, потрібного комфорту на вказані дати; перегляд своїх замовлень та їх станів (нове, відхилено/підтверджено, оплачено); оплата бронювання. Бізнес-функції системи для адміністраторів: вхід в систему з визначенням статусу «admin»; перегляд вільних номерів; пошук номерів по даті, комфорту, ціні; підтвердження або відхилення бронювання; додавання нового номеру.

**Таблиця з переліком елементів інтерфейсу й пов’язаних з ним бізнесфункцій високонавантаженої інформаційної системи, що виконуються на стороні сервера MySQL (включаючи бізнес-функції повнотекстового пошуку).**

Таблиця 1.2 – Перелік елементів інтерфейсу й бізнес-функцій

|  |  |  |
| --- | --- | --- |
| № | Елементи інтерфейсу | Бізнес-функція високонавантаженої інформаційної системи |
| 1 | Текстове поле «Логін»  Текстове поле «Пароль»  Кнопка «Вхід» | Використовуються для введення логіна й пароля. Натискання кнопки «Вхід» дозволяє користувачу ввійти в систему й змінити свій статус «незареєстрований клієнт» на «Client» або «Admin». |
| 2 | Кнопка «Додати до кошика» | Використовується для додавання кімнати чи послуги до кошика. |
| 3 | Кнопка «Оформити» | Використовується для підтвердження бронювання. |
| 4 | Текстове поле «Дата прибуття»  Текстове поле «Дата від’їзду»  Кнопка «Замовити» | Використовуються для завершення процедури замовлення. |
| 5 | Кнопка «Прийняти»  Кнопка «Відмовити» | Використовуються адміном для зміни статусу замовлення з «Нове» на «Прийнято» або «Відмовлено» |
| 6 | Текстове поле «Номер кімнати». Текстове поле «Кількість місць». Текстове поле «Ціна». Текстове поле «Комфорт». Текстове поле «Фото». Кнопка «Додати» | Використовуються адміном для додавання нового номеру |

Таблиця з переліком основних особливостей використання таблиць СУБД MySQL типу MyIsam і InnoDB у високонавантажених базах даних, що визначають їх переваги й недоліки.

Таблиця 1.3 – Порівняльний аналіз таблиць типу MyIsam і InnoDB

|  |  |  |  |
| --- | --- | --- | --- |
| № | Опис | MyIsam | InnoDB |
| 1 | Підтримка транзакцій – Немає /Так | Немає | Так |
| 2 | Підтримка зв’язків за зовнішніми ключами – Немає /Є | Немає | Є |
| 3 | Підтримки посилальної цілісності зв’язків для інструкцій UPDATE, DELETE, INSERT | Немає | Є |
| 4 | Блокування – на рівні таблиць / на рівні записів / немає блокування | На рівні таблиць | На рівні записів |
| 5 | Одночасні запити до різних записів однієї таблиці – повільніше / швидше. | Повільніше | Швидше |
| 6 | Під час змішаного навантаження за запитами SELECT, UPDATE, DELETE, INSERT – повільніше / швидше | Повільніше | Швидше |
| 7 | Однотипні операції INSERT – повільніше / швидше | Швидше | Повільніше |
| 8 | Однотипні операції SELECT – повільніше / швидше | Швидше | Повільніше |
| 9 | Запит Count(\*) – повільніше / швидше | Швидше | Повільніше |
| 10 | Взаємне блокування операцій (Deadlock) – можлива / неможлива | Неможлива | Можлива |
| 11 | Підтримка повнотекстового пошуку – Немає /Є | Є | Немає |
| 12 | Підтримка індексування полів у запитах – Немає /Є | Є | Немає |
| 13 | Можливі типи індексів | Bree | Bree, Rtree |
| 14 | Можливість бінарного копіюванні таблиць – Немає /Є | Є | Немає |
| 15 | Розмір таблиці БД – укажіть максимальний розмір даних | 2 Гб | 4 Гб |
| 16 | Можливість відновлення у випадку збою – Немає /Є | Немає | Є |
| 17 | СУБД створює за замовчуванням для таблиці MyIsam / InnoDB (укажіть кількість файлів, їх розширення, папки для зберігання та їх призначення) | Кожна таблиця зберігається у вигляді двох файлів: файл з данними та файл з індексом. Файли мають розширення .MYD та .MYI відповідно. Дані зберігаються в /var/lib/mysql | База даних зберігається у вигляді одного файлу. Формат файлу залежить від версії MySQL та налаштувань. |

**Завдання 1.2**. Розробити серверну частину високонавантаженої інформаційної системи з таблицями СУБД MySQL типу MyIsam, виконавши такий перелік робіт, відповідних до етапів проектування: – створення логічної моделі бази даних у вигляді ER-діаграми згідно з нотацією IDEF1X; – створення бази даних з таблицями типу MyIsam. Для кожної таблиці мають бути розроблені й визначені: SQL-запит, що встановлює тип таблиці; первинний ключ (primary key); індексні поля, зовнішні ключі (foreign key), які надалі використовуватимуться для забезпечення цілісності даних; поля з описом доменів, відповідних до типів даних СУБД MySQL і визначеними атрибутами «NOT NULL» і «NULL». База даних має відповідати вимогам третьої нормальної форми, містити не менш п’яти основних таблиць, з кожною з яких має бути зв’язана хоча б одна підставкова таблиця. Кожна основна таблиця має містити не менш п’яти атрибутів з наступними обов’язковими типами даних: «INT», «VARCHAR», «TEXT», «FLOAT», «DATE»; – створення фізичної моделі бази даних з таблицями типу MyIsam у вигляді ER-діаграм у нотації IDEF1X з використанням інтерфейсу реверсінжинірингу (reverse engineering) середовища розробки (для програмного пакета Workbench – у вигляді EER-діаграм); – заповнення основних таблиць даними. До основних належать таблиці, що містять дані про клієнтів, товари, послуги тощо.

**Код SQL-запиту, на основі якого створена база даних з таблицями MyIsam:**

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema HotelDBMyISAM

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `HotelDBMyISAM` DEFAULT CHARACTER SET utf8 COLLATE utf8\_unicode\_ci ;

USE `HotelDBMyISAM` ;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`Status`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`Status` (

`id\_Status` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_Status`))

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`Role`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`Role` (

`id\_Role` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_Role`))

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`Client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`Client` (

`id\_Client` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`phoneNumber` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NOT NULL,

`fk\_role\_id` INT NOT NULL,

`orderHistory` TEXT NULL,

`registrationDate` DATE NOT NULL,

`moneySpent` FLOAT NOT NULL DEFAULT 0,

`age` INT NOT NULL,

PRIMARY KEY (`id\_Client`),

UNIQUE INDEX `Name\_UNIQUE` (`name` ASC) VISIBLE,

INDEX `fk\_Client\_Role1\_idx` (`fk\_role\_id` ASC) VISIBLE)

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`Order`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`Order` (

`id\_Order` INT NOT NULL AUTO\_INCREMENT,

`orderDate` DATE NOT NULL,

`dateOfArrival` DATE NOT NULL,

`dateDeparture` DATE NOT NULL,

`fk\_status\_id` INT NOT NULL,

`fk\_client\_id` INT NOT NULL,

`totalPrice` FLOAT NOT NULL,

`OrderINT` INT NULL,

`OrderVARCHAR` VARCHAR(45) NULL,

`individualDetails` TEXT NULL,

PRIMARY KEY (`id\_Order`),

INDEX `fk\_Order\_Status1\_idx` (`fk\_status\_id` ASC) VISIBLE,

INDEX `fk\_Order\_Client1\_idx` (`fk\_client\_id` ASC) VISIBLE)

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`AdditionalEstablishments`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`AdditionalEstablishments` (

`id\_AdditionalEstablishments` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`description` TEXT(45) NOT NULL,

`price` FLOAT NOT NULL,

`freeDay` DATE NULL,

`numberOfSeats` INT NOT NULL,

PRIMARY KEY (`id\_AdditionalEstablishments`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE)

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`OrderHasEstablishments`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`OrderHasEstablishments` (

`fk\_order\_id` INT NOT NULL,

`fk\_establishment\_id` INT NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`fk\_order\_id`, `fk\_establishment\_id`),

INDEX `fk\_OrderHasEstablishments\_AdditionalEstablishments1\_idx` (`fk\_establishment\_id` ASC) VISIBLE)

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`AdditionalServices`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`AdditionalServices` (

`id\_AdditionalServices` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`description` TEXT(45) NOT NULL,

`price` FLOAT NOT NULL,

`count` INT NOT NULL,

`freeDay` DATE NULL,

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE,

PRIMARY KEY (`id\_AdditionalServices`))

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`OrderHasAdditionalServices`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`OrderHasAdditionalServices` (

`fk\_order\_id` INT NOT NULL,

`fk\_service\_id` INT NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`fk\_order\_id`, `fk\_service\_id`),

INDEX `fk\_OrderHasAdditionalServices\_AdditionalServices1\_idx` (`fk\_service\_id` ASC) VISIBLE)

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`Quality`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`Quality` (

`id\_Quality` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`id\_Quality`))

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`Room`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`Room` (

`id\_Room` INT NOT NULL AUTO\_INCREMENT,

`description` TEXT NOT NULL,

`numberOfSeats` INT NOT NULL,

`img` VARCHAR(45) NOT NULL,

`roomNumber` INT NOT NULL,

`price` FLOAT NOT NULL,

`fk\_quality\_id` INT NOT NULL,

`cleaningDate` DATE NOT NULL,

PRIMARY KEY (`id\_Room`),

UNIQUE INDEX `roomNumber\_UNIQUE` (`roomNumber` ASC) VISIBLE,

UNIQUE INDEX `img\_UNIQUE` (`img` ASC) VISIBLE,

INDEX `fk\_Room\_Quality\_idx` (`fk\_quality\_id` ASC) VISIBLE)

ENGINE = MyISAM;

-- -----------------------------------------------------

-- Table `HotelDBMyISAM`.`OrderHasRooms`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelDBMyISAM`.`OrderHasRooms` (

`fk\_room\_id` INT NOT NULL,

`fk\_order\_id` INT NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`fk\_room\_id`, `fk\_order\_id`),

INDEX `fk\_OrderHasRooms\_Order1\_idx` (`fk\_order\_id` ASC) VISIBLE)

ENGINE = MyISAM;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

Скріншот схеми фізичної моделі бази даних з таблицями типу MyIsam у нотації IDEF1X з обов’язковим відображенням первинних і зовнішніх ключів, типів даних, атрибутів «NULL», «NOT NULL», кардинальності зв’язків (для програмного пакета Workbench – у вигляді EER-діаграм, для таблиць – нотація Workbench [Default], для зв’язків – нотація IDEF1X):

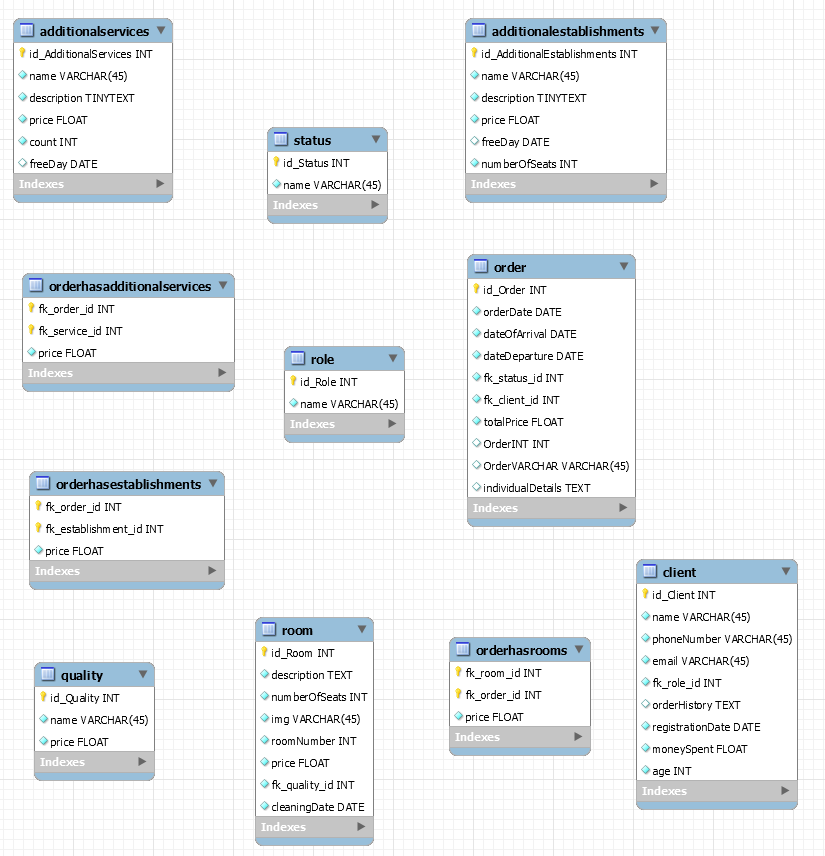


Рисунок 1.1 - Схема фізичної моделі бази даних з таблицями типу MyIsam

**Завдання 1.3.** На основі логічної моделі бази даних завдання 1.2 розробити серверну частину високонавантаженої інформаційної системи з таблицями СУБД MySQL типу InnoDB, виконавши такий перелік робіт, відповідних до етапів проектування: – створення бази даних з таблицями типу InnoDB. Для кожної таблиці мають бути розроблені й визначені: SQL-запит, що встановлює тип таблиці; первинний ключ (primary key); зовнішні ключові поля (foreign key) для забезпечення зв’язку з іншими таблицями; тип посилальної цілісності зв’язків за зовнішнім ключем (ON UPDATE, CASCADE, SET NULL, NO ACTION, RESTRICT для інструкцій UPDATE і DELETE); поля з описом доменів, відповідних до типів даних СУБД MySQL і визначеними атрибутами «NOT NULL» і «NULL». База даних має повністю відповідати логічній моделі завдання 1.2; – створення фізичної моделі бази даних з таблицями типу InnoDB у вигляді ER-діаграм у нотації IDEF1X з використанням інтерфейсу реверс- 11 інжинірингу (reverse engineering) середовища розробки (для програмного пакета Workbench – у вигляді EER-діаграм); – заповнення основних таблиць даними.

**Код SQL-запиту, на основі якого створена база даних з таблицями InnoDB:**

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema HotelInnoDB

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `HotelInnoDB` DEFAULT CHARACTER SET utf8 COLLATE utf8\_unicode\_ci ;

USE `HotelInnoDB` ;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`Status`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`Status` (

`id\_Status` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_Status`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`Role`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`Role` (

`id\_Role` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_Role`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`Client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`Client` (

`id\_Client` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`phoneNumber` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NOT NULL,

`fk\_role\_id` INT NOT NULL,

`orderHistory` TEXT NULL,

`registrationDate` DATE NOT NULL,

`moneySpent` FLOAT NOT NULL DEFAULT 0,

`age` INT NOT NULL,

PRIMARY KEY (`id\_Client`),

UNIQUE INDEX `Name\_UNIQUE` (`name` ASC) VISIBLE,

INDEX `fk\_Client\_Role1\_idx` (`fk\_role\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Client\_Role1`

FOREIGN KEY (`fk\_role\_id`)

REFERENCES `HotelInnoDB`.`Role` (`id\_Role`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`Order`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`Order` (

`id\_Order` INT NOT NULL AUTO\_INCREMENT,

`orderDate` DATE NOT NULL,

`dateOfArrival` DATE NOT NULL,

`dateDeparture` DATE NOT NULL,

`fk\_status\_id` INT NOT NULL,

`fk\_client\_id` INT NOT NULL,

`totalPrice` FLOAT NOT NULL,

`OrderINT` INT NULL,

`OrderVARCHAR` VARCHAR(45) NULL,

`individualDetails` TEXT NULL,

PRIMARY KEY (`id\_Order`),

INDEX `fk\_Order\_Status1\_idx` (`fk\_status\_id` ASC) VISIBLE,

INDEX `fk\_Order\_Client1\_idx` (`fk\_client\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Order\_Status1`

FOREIGN KEY (`fk\_status\_id`)

REFERENCES `HotelInnoDB`.`Status` (`id\_Status`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_Order\_Client1`

FOREIGN KEY (`fk\_client\_id`)

REFERENCES `HotelInnoDB`.`Client` (`id\_Client`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`AdditionalEstablishments`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`AdditionalEstablishments` (

`id\_AdditionalEstablishments` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`description` TEXT(45) NOT NULL,

`price` FLOAT NOT NULL,

`freeDay` DATE NULL,

`numberOfSeats` INT NOT NULL,

PRIMARY KEY (`id\_AdditionalEstablishments`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`OrderHasEstablishments`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`OrderHasEstablishments` (

`fk\_order\_id` INT NOT NULL,

`fk\_establishment\_id` INT NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`fk\_order\_id`, `fk\_establishment\_id`),

INDEX `fk\_OrderHasEstablishments\_AdditionalEstablishments1\_idx` (`fk\_establishment\_id` ASC) VISIBLE,

CONSTRAINT `fk\_OrderHasEstablishments\_Order1`

FOREIGN KEY (`fk\_order\_id`)

REFERENCES `HotelInnoDB`.`Order` (`id\_Order`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_OrderHasEstablishments\_AdditionalEstablishments1`

FOREIGN KEY (`fk\_establishment\_id`)

REFERENCES `HotelInnoDB`.`AdditionalEstablishments` (`id\_AdditionalEstablishments`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`AdditionalServices`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`AdditionalServices` (

`id\_AdditionalServices` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`description` TEXT(45) NOT NULL,

`price` FLOAT NOT NULL,

`count` INT NOT NULL,

`freeDay` DATE NULL,

UNIQUE INDEX `name\_UNIQUE` (`name` ASC) VISIBLE,

PRIMARY KEY (`id\_AdditionalServices`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`OrderHasAdditionalServices`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`OrderHasAdditionalServices` (

`fk\_order\_id` INT NOT NULL,

`fk\_service\_id` INT NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`fk\_order\_id`, `fk\_service\_id`),

INDEX `fk\_OrderHasAdditionalServices\_AdditionalServices1\_idx` (`fk\_service\_id` ASC) VISIBLE,

CONSTRAINT `fk\_OrderHasAdditionalServices\_Order1`

FOREIGN KEY (`fk\_order\_id`)

REFERENCES `HotelInnoDB`.`Order` (`id\_Order`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_OrderHasAdditionalServices\_AdditionalServices1`

FOREIGN KEY (`fk\_service\_id`)

REFERENCES `HotelInnoDB`.`AdditionalServices` (`id\_AdditionalServices`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`Quality`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`Quality` (

`id\_Quality` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`id\_Quality`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`Room`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`Room` (

`id\_Room` INT NOT NULL AUTO\_INCREMENT,

`description` TEXT NOT NULL,

`numberOfSeats` INT NOT NULL,

`img` VARCHAR(45) NOT NULL,

`roomNumber` INT NOT NULL,

`price` FLOAT NOT NULL,

`fk\_quality\_id` INT NOT NULL,

`cleaningDate` DATE NOT NULL,

PRIMARY KEY (`id\_Room`),

UNIQUE INDEX `roomNumber\_UNIQUE` (`roomNumber` ASC) VISIBLE,

UNIQUE INDEX `img\_UNIQUE` (`img` ASC) VISIBLE,

INDEX `fk\_Room\_Quality\_idx` (`fk\_quality\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Room\_Quality`

FOREIGN KEY (`fk\_quality\_id`)

REFERENCES `HotelInnoDB`.`Quality` (`id\_Quality`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HotelInnoDB`.`OrderHasRooms`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HotelInnoDB`.`OrderHasRooms` (

`fk\_room\_id` INT NOT NULL,

`fk\_order\_id` INT NOT NULL,

`price` FLOAT NOT NULL,

PRIMARY KEY (`fk\_room\_id`, `fk\_order\_id`),

INDEX `fk\_OrderHasRooms\_Order1\_idx` (`fk\_order\_id` ASC) VISIBLE,

CONSTRAINT `fk\_OrderHasRooms\_Room1`

FOREIGN KEY (`fk\_room\_id`)

REFERENCES `HotelInnoDB`.`Room` (`id\_Room`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `fk\_OrderHasRooms\_Order1`

FOREIGN KEY (`fk\_order\_id`)

REFERENCES `HotelInnoDB`.`Order` (`id\_Order`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

Таблиця 1.4 – Перелік типів посилальної цілісності зв’язків за зовнішнім ключем для всіх таблиць типу InnoDB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ім’я таблиці 1, зовнішній ключ | Ім’я таблиці 2, первинний ключ | SQL-інструкція для таблиці 1 | Тип посилальної цілісності |
| 1 | Client,  role\_id | Role,  id | UPDATE | RESTRICT |
| 2 | Client,  role\_id | Role,  id | DELETE | RESTRICT |
| 3 | Order,  client\_id | Client,  id | UPDATE | RESTRICT |
| 4 | Order,  client\_id | Client,  id | DELETE | RESTRICT |
| 5 | Order,  status\_id | Status,  id | UPDATE | RESTRICT |
| 6 | Order,  status\_id | Status,  id | DELETE | RESTRICT |
| 7 | Room,  quality\_id | Quality,  id | UPDATE | RESTRICT |
| 8 | Room,  quality\_id | Quality,  id | DELETE | RESTRICT |
| 9 | OrderHasRooms,  order\_id | Order,  id | UPDATE | RESTRICT |
| 10 | OrderHasRooms,  order\_id | Order,  id | DELETE | RESTRICT |
| 11 | OrderHasRooms,  room\_id | Room,  id | UPDATE | RESTRICT |
| 12 | OrderHasRooms,  room\_id | Room,  id | DELETE | RESTRICT |
| 13 | OrderHasEstablishments, order\_id | Order,  id | UPDATE | RESTRICT |
| 14 | OrderHasEstablishments, order\_id | Order,  id | DELETE | RESTRICT |
| 15 | OrderHasEstablishments, establishmant\_id | AdditionalEstablishments,  id | UPDATE | RESTRICT |
| 16 | OrderHasEstablishments, establishmant\_id | AdditionalEstablishments,  id | DELETE | RESTRICT |
| 17 | OrderHasAdditionalServices,  order\_id | Order,  id | UPDATE | RESTRICT |
| 18 | OrderHasAdditionalServices,  order\_id | Order,  id | DELETE | RESTRICT |
| 19 | OrderHasAdditionalServices,  services\_id | AdditionalServices,  id | UPDATE | RESTRICT |
| 20 | OrderHasAdditionalServices,  services\_id | AdditionalServices,  id | DELETE | RESTRICT |

Скріншот схеми фізичної моделі бази даних з таблицями типу InnoDB у нотації IDEF1X з обов’язковим відображенням первинних і зовнішніх ключів, типу даних, атрибутів «NULL», «NOT NULL», кардинальності зв’язків (для програмного пакета Workbench – у вигляді EER-діаграм, для таблиць – нотація Workbench [Default], для зв’язків – нотація IDEF1X):

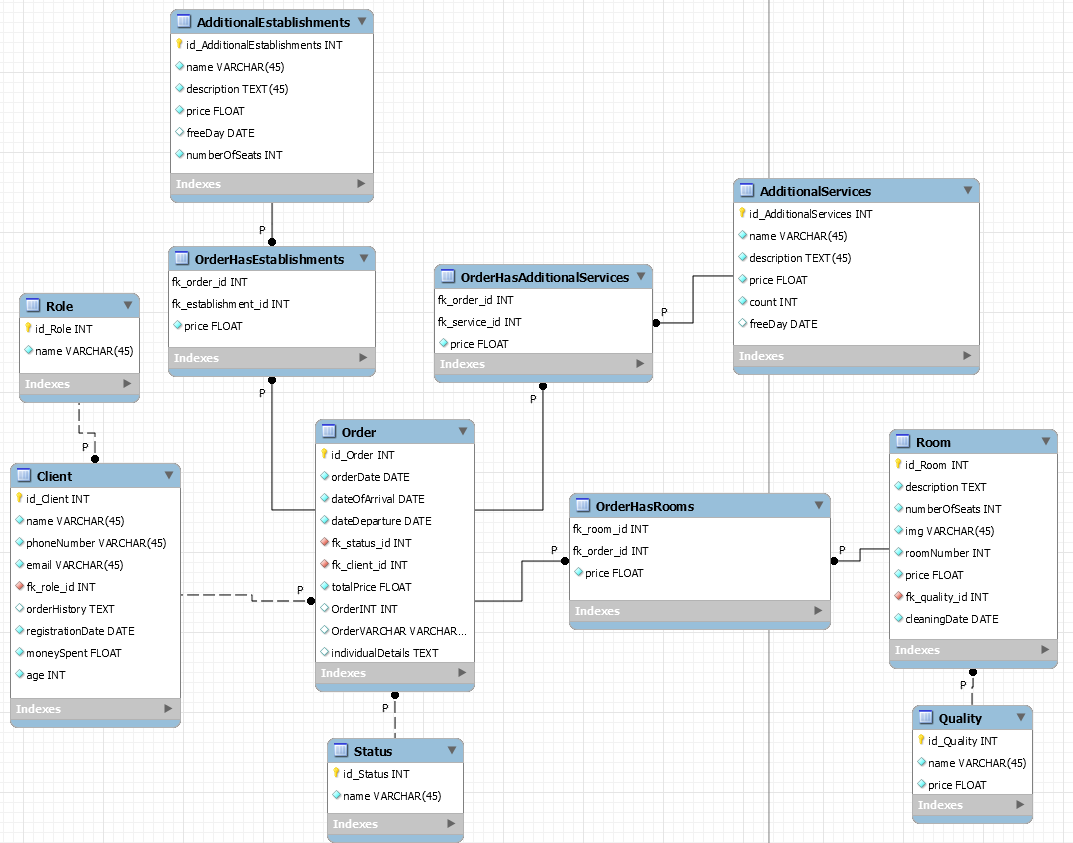


Рисунок 1.2 – Схема фізичної моделі бази даних з таблицями типу InnoDB у нотації IDEF1X

**3 Висновки**

Виконуючи дане практичне завдання було набуто практичних навичок з розробки баз даних на платформі MySQL з таблицями типу MyIsam і InnoDB, з урахуванням особливостей роботи високонавантаженої інформаційної системи зберігання даних.

Після порівняння InnoDB та MyISAM можна зробити такі висновки:

1. Краще використовувати MyISAM в таблицях, де переважають один тип доступу: читання (сайт новин) або написання (наприклад, журналювання) ;
2. Використання InnoDB має сенс у всіх інших випадках і випадках підвищених вимог безпеки даних.