ЗМІСТ

ВСТУП …...................................................................................................................... 6

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ …..................................................................... 7

1.1 Аналіз предметної області і коротка постановка задачі …....................... 7

1.2 Визначення основних бізнес-функцій високонавантаженої інформаційної системи ..................................................................................... 7

1.3 Визначення функцій інтерфейсу клієнтської частини інформаційної системи …............................................................................................................ 8

2 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ВИСОКОНАВАНТАЖЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ….............................................................................. 9

2.1 Логічне та фізичне моделювання даних …................................................ 9

2.2 Розробка підтримки цілісності даних ...................................................... 10

2.3 Реалізація бізнес-функцій інформаційної системи на стороні сервера MYSQL .............................................................................................................. 17

ВИСНОВКИ ................................................................................................................ 20

ПЕРЕЛІК ПОСИЛАНЬ ….......................................................................................... 21

ВСТУП

Обчислювальна техніка в наш час розвивається стрімкими темпами. Комп’ютери використовуються в усіх сферах життя суспільства: спілкування, пошук інформації в мережі Інтернет, придбання товарів та послуг, Інтернет-банкінг та інше. Сучасна людина не уявляє свого життя без електронних гаджетів.

В той же час вимоги до апаратного та програмного забезпечення невпинно розширюються. Якщо, наприклад, статичні веб-сторінки ще кілька років тому були досить поширеними, то зараз важко знайти сайт, де б не було інтерактивної складової.

Саме розвиток обчислювальної техніки, а особливо веб-технологій, дозволив перенести в Інтернет навчальний процес: в наш час майже кожна університет або школа має свій сайт. Це дозволяє економити кошти та час, залучати учасників навчального процесу до більш детального ознайомлення із навчальним закладом.

З поширенням мережі Інтернет виникли електронні сервіси, що надають різні можливості по інтерактивній взаємодії з навчальними закладами. У порівнянні із закладами які не мають власних сервісів вони мають безліч переваг, які сприяють зростанню популярності та доходів.

Метою даного курсового проекту є реалізація бази даних інформаційної системи «Вступна кампанія». Розроблена база даних має всі необхідні компоненти для виконання будь-яких завдань пов’язаних з пошуком і обробкою інформації.

Використання інтернету в навчальній сфері є досить перспективним напрямом, так як дозволяє за мінімальні затрати істотно підвищити рівень комунікація адміністрації навчального закладу та обліки, які використовуються в цьому напрямі.

Все вищезазначене підтверджує актуальність обраної теми - «Вступна кампанія».

# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

* 1. Аналіз предметної області і коротка постановка задачі

В якості підприємства було обрано Університет, головними функціями якого представленні :

* здійсненні освітньої;
* наукової діяльності.

Після аналізу підприємства було виділено наступні бізнес процеси, які можна автоматизувати з використанням програмних засобів ІС:

* управління розкладом;
* електронний журнал відвідування;
* онлайн портал для керування навчальним матеріалом;
* ведення обліку в бібліотеці по читачам\книгам;
* вступна кампанія яку проводить університет для вступу абітурієнтів до ВНЗ.

Серед вищезгаданих бізнес процесів було обрано саме бізнес процес проведення вступної кампанія. На сьогоднішній день вже існує аналог - «Інформаційна система Конкурс «Вступна кампанія»», але під час проведення цього конкурсу виникають проблеми в роботі, оскільки занадто велика кількість людей намагаються отримати доступ до порталу. Але якщо у кожного університету буде своя вступна кампанія це зменшить навантаження на кожний із серверів що в результаті приведе до стійкої роботи ІС «Вступна кампанія». В майбутньому буде розглядатися можливість синхронізації даних в спільній ноді(спільний сервер для всіх університетів, в якому будуть зберігатися вже оброблені дані), а сервер кожного університету буде виступати в якості під нод.

Об'єктом дослідження є розробка серверної частина інформаційної системи «Вступна кампанія».

Користувачі даної системи зможуть самостійно у будь-який час переглянути актуальну інформацію по стану проведення конкурсу на різні факультети, самостійно подати заявку для участі у конкурсі після реєстрації та авторизації та інше. Можливість приймати участь у «Вступній кампанія» не виходячи з дому економить час користувачів, оскільки для використання системи непотрібно навіть виходити з дому. Системою можуть користуватися як представники вузу (педагогічний колектив, деканат) так і інші особи (абітурієнти; викладачі різних шкіл; різні організації, які займаються збором статистики). Також, система буде корисна абітурієнтам, які живуть в смт або селах.

Також використання даної системи вигідне ВНЗ – відсутність необхідних витрат на персонал, який буде проводити реєстрацію вручну в університеті.

В системі має бути можливість реєстрації. Зареєстровані користувачі матимуть свої переваги.

1.2 Визначення основних бізнес-функцій високонавантаженої інформаційної системи

У системі існує 3 ролі користувачів:

* незареєстрований користувач(guest);
* зареєстрований користувач(enrollee);
* головний адміністратор(admin);
* адміністратор факультету (facultyAdmin);

Головний адміністратор має можливість додавати\редагувати\видаляти факультет\користувачів, але головною бізнес функцією є формування відомості по обраному факультету. Також він має можливість керувати адміністраторами факультетів (додавати, редагувати, видаляти).

Адміністратор факультету має ті ж самі можливості що і головний адміністратор, окрім можливості керувати адміністраторами факультетів. Адміністратор факультетів має можливість редагувати інформацію про факультет, а також видаляти абітурієнтів з конкурсу на факультет, за яким він закріплений.

Незареєстровані користувачі можуть переглядати інформацію про факультети, інформацію про конкурс.

Зареєстровані користувачі можуть виконувати всі ті самі дії, що і незареєстровані плюс мають наступні переваги після виконання авторизації:

* створити заявку для участі у конкурсі;
* додати власну інформацію;
* додати оцінки.

1.3 Визначення функцій інтерфейсу клієнтської частини інформаційної системи

Панель навігації містить наступні елементи:

* форма пошуку – текстове поле, в яке користувач вводить повну або часткову назву для факультету та кнопка «Пошук».
* посилання на каталог факультетів;
* кнопки : вхід, реєстрація.

Каталог факультетів містить список факультетів та панель фільтрів (за кількістю бюджетних місць, за загальною кількістю місць).

Сторінка факультету містить інформацію про факультет та кнопку «Подати заявку».

Форма додавання власних даних містить поля з даними користувача та кнопка «Зберегти».

Форма реєстрації містить текстові поля для введення імені, логіна, електронної пошти, пароля і кнопку «Реєстрація».

Клієнтська частина ІС «Вступна кампанія», яка базується на бізнес логікі відповідно до серверної частини повинна виконувати функції, які будуть надавати користувачеві необхідну інформацію і давати можливість взаємодіяти з системою:

* функція виведення результатів пошуку;
* функція виведення списку факультетів за фільтрами;
* функція виведення детальної інформації про факультет;
* функція оформлення заявки для участі у конкурсі;
* функція виведення списку всіх заявок;
* функція перегляду інформації про користувача.

# РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Фізичне моделювання даних

Основою для інформаційної системи є база даних. Саме БД дозволяє експлуатувати ІС, виконувати її поточне обслуговування, модифікувати і обробляти інформацію про роботу підприємства, приймати інформаційні потоки.

Для предметної області «Вступна кампанія» була розроблена модель бази даних у вигляді ER-діаграми (Рисунок 2.1)

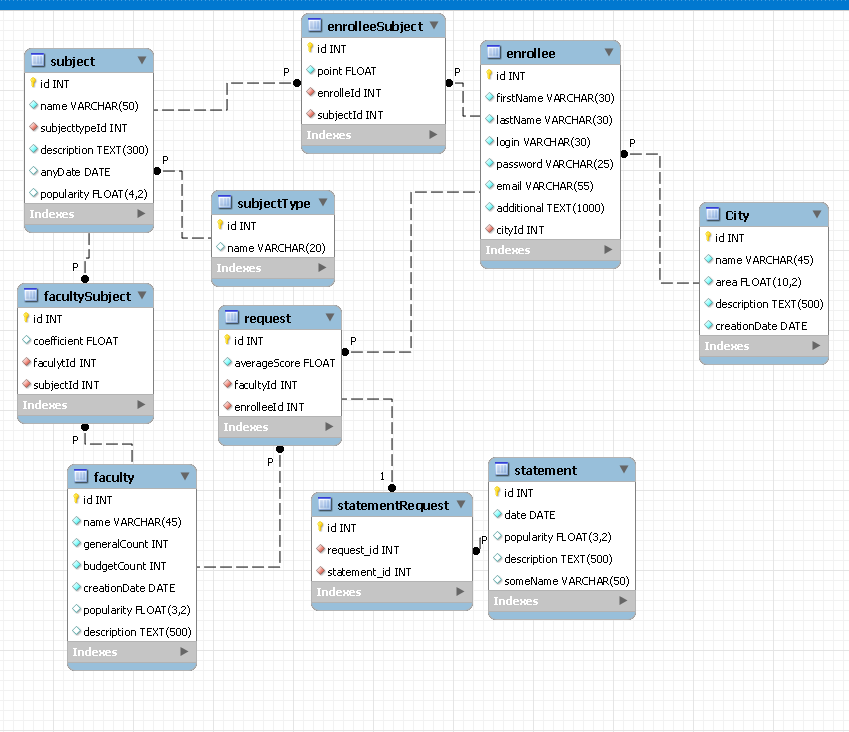


Рисунок 2.1 – EER діаграма бази даних з типом таблиць InnoDB

На EER діаграмі присутні наступні сутності:

* enrollee – містить наступну інформацію про абітурієнта: ім’я, прізвище, логін, пароль, емейл, додаткова інформація, id міста, **дата реєстрації**;
* city – містить інформацію про міста: назва міста, площа, опис, дата створення;
* subject – містить інформацію про предмети: назва предмета, id типу предмета, додаткова інформація про предмет, **популярність предмету**;
* subjectType – містить інформацію про тип предметів: назва (пр. екзамен, оцінка з атестату);
* faculty – містить інформацію про факультети: назва факультету, загальна кількість місць, бюджетна кількість місць, дата створення факультету, популярність факультету, опис факультету;
* request – містить інформацію про заявки на участь у конкурсі: середній бал, id факультету, id абітурієнт, **дата створення**;
* statement – містить інформацію про сформовані відомості: дата формування, **кількість абітурієнтів, назва відомості (пр. Відомість для факультету …), середній бал**.

2.2 Розробка підтримки цілісності даних

Логічна модель даних була перетворена в фізичну шляхом вибору системи управління базами даних і заданням правил підтримки цілісності. Вони необхідні для опису змін відбуваються в сутності, якщо пов'язана з нею інша сутність була змінена або видалена.

Після побудови схеми фізичної моделі необхідно задати правила цілісності даних, тобто характеристики для кожного атрибута кожної сутності, щоб забезпечити відповідність інформації її внутрішній логіці, структурі і всім заданим правилам. Цілісність БД не гарантує достовірності інформації, що міститься в ній, але забезпечує, принаймні, узгодженість даних, відкидаючи свідомо неймовірні, неможливі значення.

SQL код створення таблиць та задання правил цілісності для рушія InnoDB наведено нижче.

CREATE TABLE IF NOT EXISTS `courseproject`.`categories` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `courseproject`.`products` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`categoryId` INT NOT NULL,

`price` DECIMAL(10,2) NOT NULL,

`description` TEXT(1000) NOT NULL,

`stock` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `productCategory\_idx` (`categoryId` ASC),

INDEX `name` (`name` ASC),

INDEX `description` (`description`(15) ASC),

CONSTRAINT `productCategory`

FOREIGN KEY (`categoryId`)

REFERENCES `courseproject`.`categories` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `courseproject`.`roles` (

`id` INT NOT NULL AUTO\_INCREMENT,

`title` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `courseproject`.`users` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`phone` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`city` VARCHAR(45) NOT NULL,

`address` VARCHAR(45) NOT NULL,

`cardInfo` VARCHAR(45) NULL,

`roleId` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `roleId\_idx` (`roleId` ASC),

INDEX `email` (`email` ASC),

CONSTRAINT `userRole`

FOREIGN KEY (`roleId`)

REFERENCES `courseproject`.`roles` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `courseproject`.`orders` (

`id` INT NOT NULL AUTO\_INCREMENT,

`date` DATETIME NOT NULL,

`userId` INT NULL,

`phone` VARCHAR(45) NULL,

`city` VARCHAR(45) NULL,

`address` VARCHAR(45) NULL,

`cardInfo` VARCHAR(45) NULL,

PRIMARY KEY (`id`),

INDEX `orderUser\_idx` (`userId` ASC),

INDEX `userId` (`userId` ASC),

CONSTRAINT `orderUser`

FOREIGN KEY (`userId`)

REFERENCES `courseproject`.`users` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `courseproject`.`productOrders` (

`orderId` INT NOT NULL,

`productId` INT NOT NULL,

`count` INT NOT NULL,

PRIMARY KEY (`orderId`, `productId`),

INDEX `product\_idx` (`productId` ASC),

CONSTRAINT `order`

FOREIGN KEY (`orderId`)

REFERENCES `courseproject`.`orders` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `product`

FOREIGN KEY (`productId`)

REFERENCES `courseproject`.`products` (`id`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

Також було розроблено БД з типом таблиць MyISAM. Схема БД наведена на рисунку 2.2.

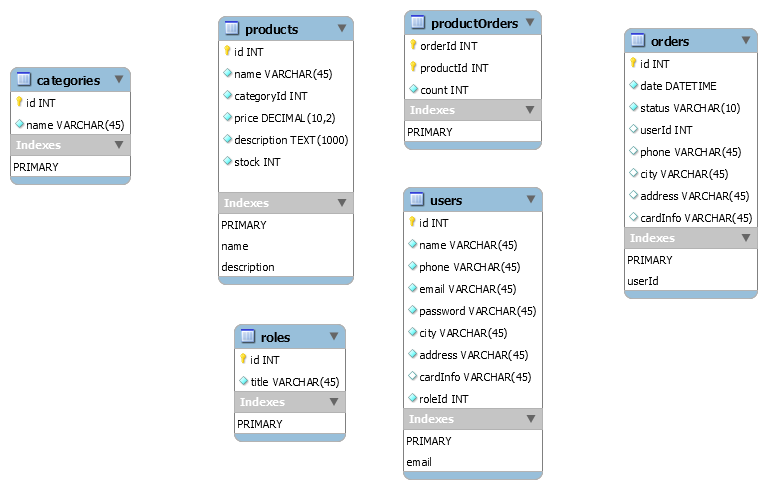


Рисунок 2.2 – EER діаграма бази даних з типом таблиць MyISAM

Оскільки таблиці типу MyISAM не підтримують зовнішні ключі, цілісність даних було реалізовано за допомогою тригерів.

Тригери для таблиці Categories:

Для забезпечення цілісності даних з таблицею products при видаленні запису:

CREATE TRIGGER `courseprojectmyisam`.`categories\_BEFORE\_DELETE` BEFORE DELETE ON `categories` FOR EACH ROW

BEGIN

declare product int;

SET product = (select id from products where categoryId = OLD.id limit 1);

IF not isnull(product) then

signal sqlstate '45000'

set message\_text = 'There are products with this category';

end if;

END

Для забезпечення цілісності даних з таблицею products при оновленні запису:

CREATE TRIGGER `courseprojectmyisam`.`categories\_BEFORE\_UPDATE` BEFORE UPDATE ON `categories` FOR EACH ROW

BEGIN

declare product int;

SET product = (select id from products where categoryId = OLD.id limit 1);

IF (not isnull(product) and old.id <> new.id) then

signal sqlstate '45000'

set message\_text = 'There are products with this category';

end if;

END

Тригери для таблиці Products:

Для забезпечення цілісності даних з таблицею productOrders при видаленні запису:

CREATE TRIGGER `courseprojectmyisam`.`products\_BEFORE\_DELETE` BEFORE DELETE ON `products` FOR EACH ROW

BEGIN

declare productOrder int;

SET productOrder = (select productId from productorders where productId = OLD.id limit 1);

IF not isnull(productOrder) then

signal sqlstate '45000'

set message\_text = 'There are orders for this product';

end if;

END

Для забезпечення цілісності даних з таблицею productOrders при оновленні запису:

CREATE TRIGGER `courseprojectmyisam`.`products\_BEFORE\_UPDATE` BEFORE UPDATE ON `products` FOR EACH ROW

BEGIN

declare productOrder int;

SET productOrder = (select productId from productorders where productId = OLD.id limit 1);

IF (not isnull(productOrder) and old.id <> new.id) then

signal sqlstate '45000'

set message\_text = 'There are orders for this product';

end if;

END

Тригери для таблиці Orders:

Для забезпечення цілісності даних з таблицею productOrders при видаленні запису:

CREATE TRIGGER `courseprojectmyisam`.`orders\_BEFORE\_DELETE` BEFORE DELETE ON `orders` FOR EACH ROW

BEGIN

delete from productorders

where orderId = old.id;

END

Для забезпечення цілісності даних з таблицею productOrders при оновленні запису:

CREATE TRIGGER `courseprojectmyisam`.`orders\_BEFORE\_UPDATE` BEFORE UPDATE ON `orders` FOR EACH ROW

BEGIN

declare orderId int;

SET orderId = (select orderId from productorders where orderId = OLD.id limit 1);

IF (not isnull(orderId) and old.id <> new.id) then

signal sqlstate '45000'

set message\_text = 'You can not change order id';

end if;

END

Тригери для таблиці Users:

Для забезпечення цілісності даних з таблицею Orders при видаленні запису:

CREATE TRIGGER `courseprojectmyisam`.`users\_BEFORE\_DELETE` BEFORE DELETE ON `users` FOR EACH ROW

BEGIN

declare buyer int;

SET buyer = (select userId from orders where userId = OLD.id limit 1);

IF not isnull(buyer) then

signal sqlstate '45000'

set message\_text = 'This user has orders!';

end if;

END

Для забезпечення цілісності даних з таблицею Orders при оновленні запису:

CREATE TRIGGER `courseprojectmyisam`.`users\_BEFORE\_UPDATE` BEFORE UPDATE ON `users` FOR EACH ROW

BEGIN

declare buyer int;

SET buyer = (select userId from orders where userId = OLD.id limit 1);

IF (not isnull(buyer) and old.id <> new.id) then

signal sqlstate '45000'

set message\_text = 'This user has orders';

end if;

END

Тригери для таблиці Roles:

Для забезпечення цілісності даних з таблицею Users при видаленні запису:

CREATE TRIGGER `courseprojectmyisam`.`roles\_BEFORE\_DELETE` BEFORE DELETE ON `roles` FOR EACH ROW

BEGIN

declare existinguser int;

SET existinguser = (select id from users where roleId = OLD.id limit 1);

IF not isnull(existinguser) then

signal sqlstate '45000'

set message\_text = 'There are users with this role';

end if;

END

Для забезпечення цілісності даних з таблицею Users при оновленні запису:

CREATE TRIGGER `courseprojectmyisam`.`roles\_BEFORE\_UPDATE` BEFORE UPDATE ON `roles` FOR EACH ROW

BEGIN

declare existinguser int;

SET existinguser = (select id from users where roleId = OLD.id limit 1);

IF (not isnull(existinguser) and old.id <> new.id) then

SET SQL\_SAFE\_UPDATES = 0;

update users set roleId = new.id where users.roleId = old.id;

end if;

END

2.3 Реалізація бізнес-функцій інформаційної системи на стороні сервера MYSQL

Для реалізації основних бізнес-функцій інформаційної системи було використано процедури та транзакції.

Процедура додання товару у кошик:

CREATE PROCEDURE `add\_to\_cart`(user int, productId int, count int)

BEGIN

declare orderId int;

declare currentStock int;

set orderId = (select id from orders where userId = user and status = 'new');

if isnull(orderId) then

insert into orders (date, status, userId)

values (

(select now()),

'new',

user

);

set orderId = LAST\_INSERT\_ID();

end if;

START TRANSACTION;

select stock into currentStock from products where id = productId;

if currentStock < count then

signal sqlstate '45000'

set message\_text = 'There are not enough products in stock';

end if;

insert into productorders values (orderId, productId, count);

update products set stock = currentStock - count where id = productId;

COMMIT;

END

Процедура підтвердження замовлення:

CREATE PROCEDURE `submit\_order`(user int)

BEGIN

update orders set status = 'confirmed' where userId = user and status = 'new';

END

Процедура пошуку товару за назвою або описом:

CREATE PROCEDURE `search\_product`(searchText varchar(255))

BEGIN

select \* from products where

name like concat('%', searchText, '%') or

description like concat('%', searchText, '%');

END

Процедура, що виводить список усіх замовлень, суму замовлення, кількість артикулів та ID користувача:

CREATE PROCEDURE `orders\_values`()

BEGIN

select orderId as `Order ID`, count(productId) as Listings, sum(price \* count) as Total, orders.id as `User ID`

from products inner join productorders on products.id = productorders.productId

inner join orders on orders.id = productorders.orderId

group by orderId

order by total desc;

END

Процедура реєстрації:

CREATE PROCEDURE `register`(

email varchar(45),

password varchar(45),

name varchar(45),

phone varchar(45),

city varchar(45),

address varchar(45),

cardInfo varchar(45)

)

BEGIN

declare existing int;

select id into existing from users where users.email = email;

if not isnull(existing) then

signal sqlstate '45000'

set message\_text = 'User with this email already exists';

end if;

insert into users values (DEFAULT, name, phone, email, password, city, address, cardInfo, DEFAULT);

END

ВИСНОВКИ

В ході виконання роботи було виконано проектування серверної частини високонавантаженої бази даних «Інтернет магазин електроніки». Для створення БД було використане середовище візуального проектування баз даних «MySQL Workbench» - інструмент, який дозволяє наочно уявити модель бази даних в графічному вигляді, а також є зручним редактором SQL запитів, що дозволяє відразу ж відправляти їх на сервер і отримати відповідь у вигляді таблиці.

Була розроблена база даних для двох типів таблиць «MyISAM» і «InnoDB». Для здійснення цілісності БД в «MyISAM» були реалізовані тригери, які забезпечать узгодженість даних, в «InnoDB» були описані обмеження у вигляді зв'язків (зовнішніх ключів). Для всіх типів таблиць були використані обмеження NOT NULL (виходячи з вимог системи).В ході виконання курсового проекту були поглиблені знання і закріплені практичні навички роботи з програмою Workbench, а також знайдені практичні навички роботи з високонавантажених базами даних.

Розроблене програмне забезпечення дозволяє створити власний інтернет магазин та володіє всіма необхідними компонентами для виконання необхідних завдань в рамках заданої предметної області.

Отримані знання та навички будуть використані при виконанні подальших робіт в програмі університету, а також у подальшій професійній діяльності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Методичні вказівки до міждисциплінарного курсового проекту по для студентів напряму підготовки 6.050101 - Комп'ютерні науки [Електронне видання] / Упоряд.: Ю.В. Міщеряков, З.А. Імангулова., Л.В. Колесник - Харків: ХНУРЕ, 2016. - 30 с.
2. Довідник по MySQL [Електронний ресурс] - Режим доступу URL: <https://dev.mysql.com>
3. Довідник по Workbench [Електронний ресурс] - Режим доступу URL:

https:// dev.mysql.com/doc/workbench/en/

1. Работа с базами данных: учебное пособие / уклад. О. Н. Євсєєва, О. Б. Шамшев. - Ульяновськ: УлГТУ, 2009. - 170 с.