Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра інтелектуальних та інформаційних систем

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО КУРСОВОЇ РОБОТИ

з дисципліни «Організація баз даних і знань»

на тему: «ІС менеджера консалтингової компанії»

Студента 2 курсу групи КН-21
спеціальності
6.050101 Комп'ютерні науки
Астахова А. К.

Керівник:
Гайна Г. А.

Національна шкала
_____

Кількість балів: _____

Члени комісії

_____
_____
(підпис) (прізвище та ініціали)

_____
_____
(підпис) (прізвище та ініціали)

Київ – 2017 рік

Зміст

Зміст 2
Вступ 3
Частина 1. Дослідження предметної області 4
1.1. Критерії відповідності інформаційної системи вимогам ринку 4
1.2. Опис вхідних даних 5
1.3. Опис процесів перетворення даних 6
1.4. Опис вихідних даних 7
Частина 2. Розробка інфологічної моделі 8
2.1. Створення інформаційних об’єктів та атрибутів 8
2.2. Створення структурних зв’язків 10
2.3. Побудова інфологічної моделі 11
2.4. Діаграми потоків даних 12
Частина 3. Розробка даталогічної моделі 13
3.1. Зв’язки 13
3.2 Нормалізація 14
3.3. Даталогічна модель 15
Частина 4. Проектування та реалізація БД на фізичному рівні 16
4.1. Обмеження полів 16
4.2. Фізична модель 18
4.3. Мобільний застосунок 19
Висновки 20
Джерела 21
Додатки 22
Словник бази даних 22
Знімки екрану 24
Лістинг коду 27

Вступ

Мета цієї курсової роботи полягає в тому, щоб спроектувати та реалізувати базу даних на мобільному застосунку під операційну систему Android. Спочатку здійснюється проектування на інфологічному та даталогічному рівнях. Потім спроектована база даних реалізується на фізичному рівні з використанням реляційної СКБД SQLite. Зрештою, розроблена система імплементується в мобільний застосунок для планшетних пристроїв під ОС Android. Потрібно передбачити засоби для взаємодії оператора з базою даних через графічний інтерфейс корисувача, а також можливість для клієнта отримувати звітність у зручній для нього формі. Для цього розроблюється система тригерів та обмежень, які мають допомогти користувачеві контролювати зміни даних.

Обрана мною тема називається «Інформаційна система менеджера консалтинтової компанії». Ця предметна область відноситься до управління проектами. Головним завданням проектного управління є досягнення всіх цілей та виконання завдань проекту, одночасно виконуючи зобов'язання щодо наперед визначених обмежень проекту. Розроблена мною БД має охоплювати задачі контролю та аналізу роботи відділів, іх керівників та підлеглих, а також виконання завдань на різних етапах.

В якості програмної платформи я обрав операційну систему Android, як одну з найпоширеніших на сучасному ринку. Її використання дасть клієнту змогу працювати із базою даних у мобільному режимі, не обмежуючи себе одним місцезнаходженням.

Частина 1. Дослідження предметної області

1.1. Критерії відповідності інформаційної системи вимогам ринку

Вибір інформаційної системи для управління задачами і персоналом — актуальне питання, яке постає практично перед кожним менеджером-початківцем. Це дуже важливий інструмент, який має відповідати таким критеріям:

1) простота встановлення та впровадження (у невеликих компаній немає часу та фінансових ресурсів на те, щоб протягом місяців займатись впровадженням нових систем);
2) гнучкіть налаштування (система має підходити для максимальної кількості суміжних за свєю сутністю завдань);
3) мобільність у використанні (у сучасному світі менеджеру конче необхідно преміщатись між різними локаціями, при цьому не втрачаючи контроль над підлеглими);
4) надійність (база даних має бути вміло зпроектована та нормалізована; в ній мають бути передбачені засоби для безпечного маніпулювання з даними, у тому числі тригери для відслідковування видалення чи зміни даних у різних таблицях);
5) адаптивніть до середовища використання (застосунок має бути придатним для користування якумога ширшим колом потенційних клієнтів);
6) швидкіть доступу до інформації (база даних має оперативно відповідати на запити та не заважати комфортній роботі користувача);
7) можливість аналізувати за впорядковувати дані (мають бути передбачені механізми для ведення статистики та аналізу роботи працівників).

1.2. Опис вхідних даних

Вищеописана сукупніть критеріїв була врахована при розробці моєї бази даних з метою стровення сучасного конкурентноспроможного продукту.

Враховуючи необхідність створення рішення для компаній різного розміру, було вирішено відштовхуватись від необхідності розробки системи управління з можливістю розширення меж компанії та утримання деякого числа менеджерів, що управляють декількома відділами. Проаналізувавши цільову аудиторію користувачів застосунку, я дійшов висновку, що в базі даних необідно реалізувати принцип єдиновладдя (у підлеглого має бути один начальник), тому в базі даних було передбачено можливіть призначення в голові відділу лише одного менеджера, однак, один керівник може управляти багатьма відділами.

Це рішення є конче вдалим із двох причин. По-перше, вдалося реалізували один с одновних принципів побудови прозорої організаційної структури на підприємтсві. По-друге, це дало змогу на повну використати потенціал реляційної бази даних за допомою зв’язку «один до багатьох».

Вхідні дані для відділів були сформовані на основі аналізу відкритих документів таких компаній, як ABInBev та Amazon (див. джерела [3], [4]). Вони включають інформацію про технічні дані кожного з відділів.

Інформація про робітників береться із корпоративного довідника та місить дані про вік, стать, заробітну платню, посаду, досвід, банківські реквізити та контактні дані.

Вхідні дані про задачі надходять із журналу та містять інформацію про дату початку, очікувану кількість годин на виконання та орієнтовну дату завершнення.

1.3. Опис процесів перетворення даних

Основну частину процесів перетворення даних складають операції введення, корегування та видалення інформації, а також групування та розрахунку сум, середніх, максимальних та мінімальних значень по обраних угрупуваннях, підрахунку кількості об’єктів групування.

В наших вихідних документах використовуються такі розрахункові атрибути:

1) підрахування кількості завдань з визначеним пріоритетом, що були виконані певним робітником зі зривом графіка;
2) зміна статусу задачі при виконнані всіх її етапів на 100%;
3) зміна статусу робітника при призначенні йому нової задачі;
4) підрахування кількості завдань за визначений час, що були виконані певним департаментом зі зривом графіка;
5) підрахування кількості завдань на заданий час, що були делеговані пеному менеджеру.

1.4. Опис вихідних даних

В якості вихідних документів використовуються два звіти:

1) звіт про стан виконання завдань — «Відомість про стан виконання завдань по відділам»;

2) звіт по роботі працівників — «Відомість про роботу працівників у підпорядкуванні менеджерам».

Звіт про стан виконання завдань друкується для подальшої звітності і аналізу ефективності вирішення задач та їх етапів. Цей документ містить дані про департамент, ім’я його керівника, кількість завдань, що було упішно виконано, кількіть зривів графіка, кількість поточних та майбутніх завдань, а також інформацію про виконання епатів кожного з завдань, назву, статус, прогресс, ступінь готовності.

Звіт по роботі працівників друкується для подальшої звітності і аналізу ефективності роботи менеджерів та їх підлеглих. Цей документ містить дані про ім’я менеджера, відділ, яким він керує, імена підлеглих, кількість вільних та зайнятих підлеглих, а також робітників, які зривали графік.

Частина 2. Розробка інфологічної моделі

2.1. Створення інформаційних об’єктів та атрибутів

Метою інфологічного проектування є створення структурованої інформаційної моделі предметної області, для якої розроблятиметься база даних. Основною складовою інфологічної моделі є атрибути, які потрібно проаналізувати та певним чином згрупувати для подальшого зберігання в БД.

Було виділено п’ять інформаційних об’єктів, до яких було додано певинні ключі та розподілено атрибути.

Інформаційний об’єкт	*Назва атрибута*
Менеджер	Первинний ключ
	ПІБ
	Вік
	Стать
	Дата народження
	Телефон
	Електронна пошта
	Адреса
	Досвід роботи
	Номер паспорта
	Зарплата
	Банківський рахунок
Відділ	Первинний ключ
	Назва
	Телефон
	Електронна пошта
Працівник	Первинний ключ
	ПІБ
	Вік
	Стать

	Дата народження
	Телефон
	Електронна пошта
	Адреса
	Посада
	Досвід роботи
	Номер паспорта
	Зарплата
	Банківський рахунок
	Статус
Завдання	Первинний ключ
	Назва
	Дата початку
	Дата здачі
	Статус
Етап	Первинний ключ
	Назва
	Дата початку
	Дата здачі
	Важливість
	Прогрес виконання
	Час на виконання

2.2. Створення структурних зв’язків

Усі таблиці поєднуться зв’язком «один до багатьох», що робить кожний первинний ключ простим та незалежним. Розглянемо детальніше:

1) між об’єктами «Менеджер» та «Відділ» існує наступний зв’язок: менеджер управляє відділом, тобто один менеджер може керувати багатьма відділами, але відділ завжди управляється одним менеджером;
2) між об’єктами «Відділ» та «Робітник» існує наступний зв’язок: відділ містить робітника, тобто один відділ може містити багато робітників, але робітник завжди закріплений за одним відділом;
3) між об’єктами «Робітник» та «Завдання» існує наступний зв’язок: робітник виконує завдання, тобто один робітник може виконувати багато завдань, але завдання завжди виконується одним робітником;
4) між об’єктами «Завдання» та «Етапи» існує наступний зв’язок: завдання містить етапи, тобто одне завдання може містити багато етапів, але етап завжди належить одному завданню.

2.3. Побудова інфологічної моделі

На рисунку 1 представлена інфологічна модель

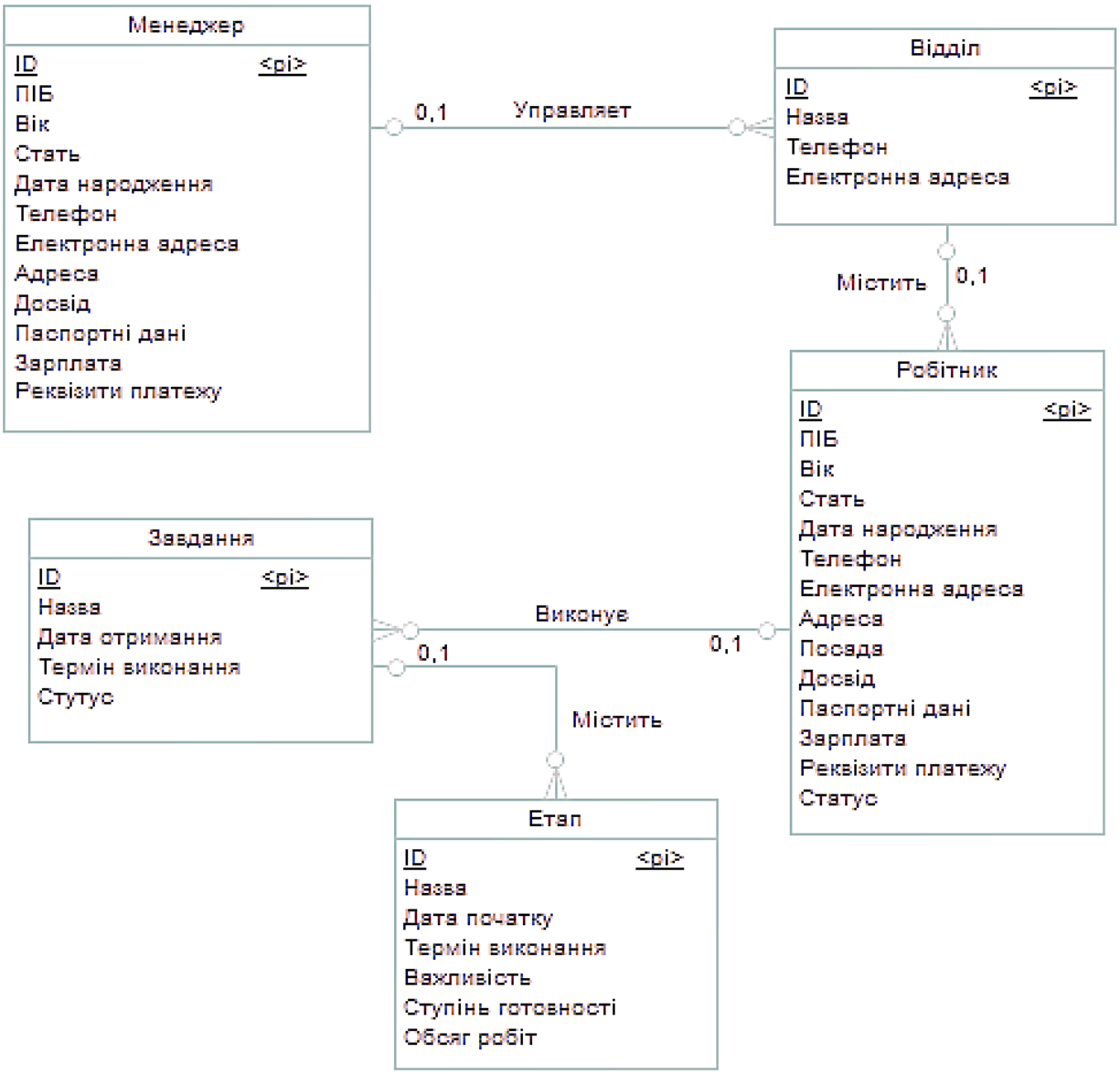


Рисунок 1 — побудована інфологічна модель

Біля первинних ключів проставлений атрибут <pi> (primary identificator).

2.4. Діаграми потоків даних

На схемах 1—3 зображено рівні 0—2 діграми потоків даних в ІС відповідно.

Схема 1

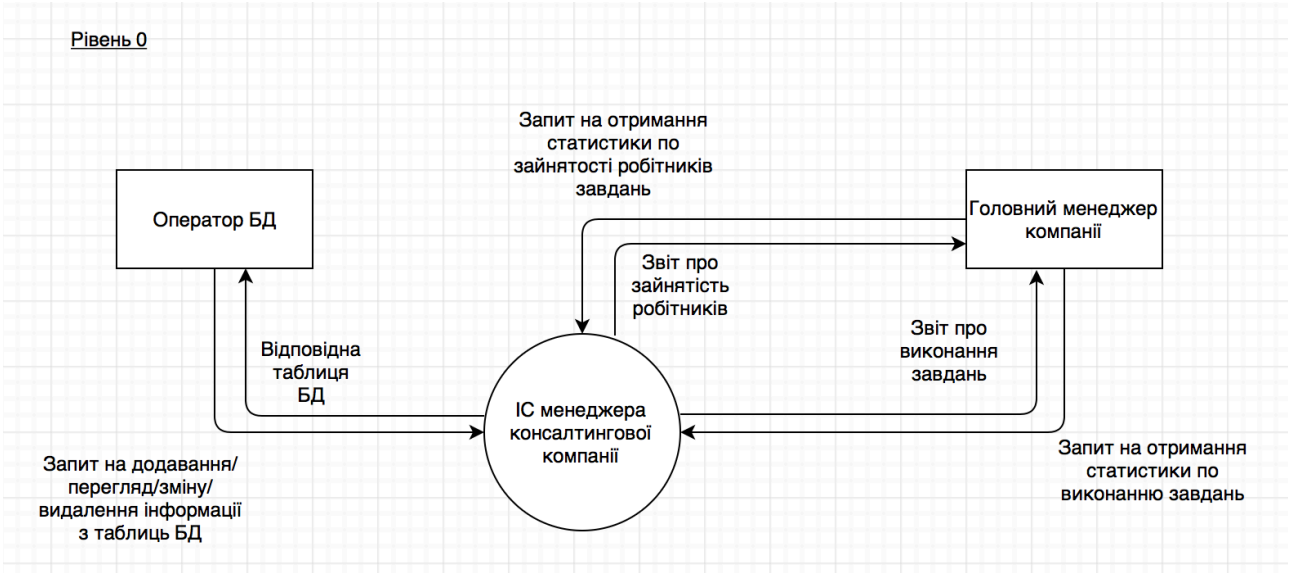


Схема 2

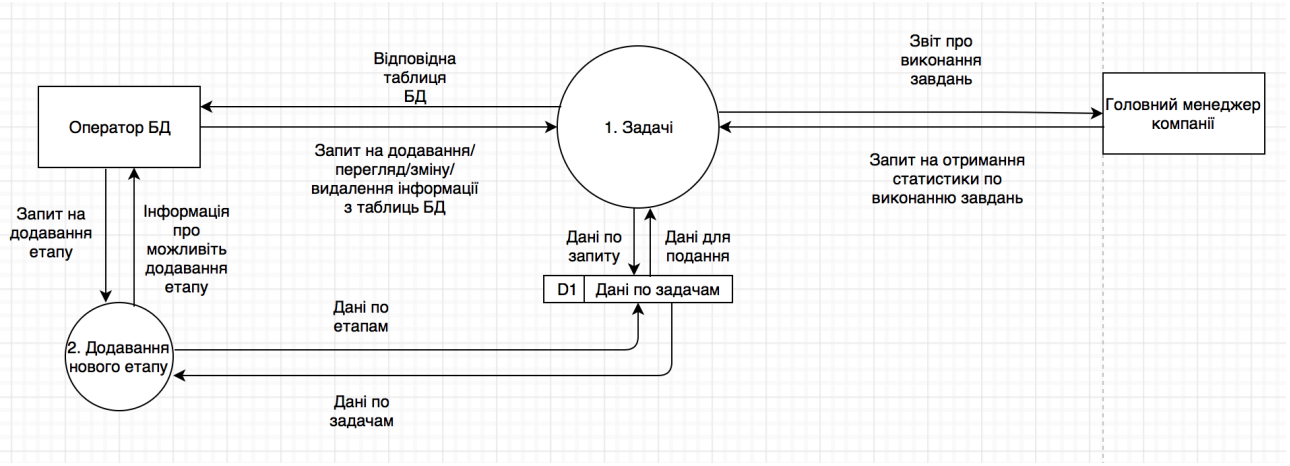
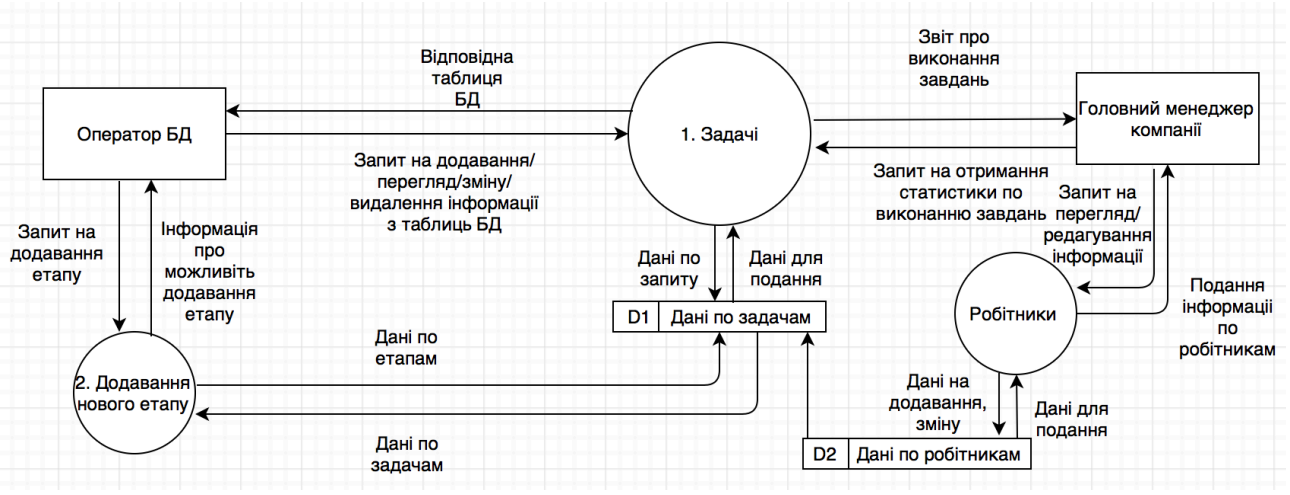


Схема 3



Частина 3. Розробка даталогічної моделі

3.1. Зв’язки

Як вже зазначалося, усі таблиці поєднуться зв’язком «один до багатьох», що робить кожний первинний ключ простим та незалежним. Через відсутність зв’язків «багато до багатьох» немає необхідності створювати проміжні таблиці для зберігання зовнішніх ключів, а через відсутність зв’язків «один до одного» не потрібно позбуватися таблиць за їх надмірністю.

Далі зв’яжемо таблиці зовнішніми ключами. Для цього до таблиць «Відділ», «Робітник», «Завдання», «Етап» додамо поля, що будуть зовнішніми ключами (таблиці «Менеджер» таке поле не потрібно, адже вона не ссилається на іншу таблицю). Розглянемо детальніше:

1) таблиці «Відділ» та «Менеджер» зв’яжемо зовнім ключем з відповідного поля табліці «Відділ» до поля первинного ключа таблиці «Менеджер»;
2) таблиці «Робітник» та «Відділ» зв’яжемо зовнім ключем з відповідного поля табліці «Робітник» до поля первинного ключа таблиці «Відділ»;
3) таблиці «Завдання» та «Робітник» зв’яжемо зовнім ключем з відповідного поля табліці «Завдання» до поля первинного ключа таблиці «Робітник»;
4) таблиці «Етап» та «Завдання» зв’яжемо зовнім ключем з відповідного поля табліці «Етап» до поля первинного ключа таблиці «Завдання».

3.2 Нормалізація

Нормалізація відношень – це ітераційний зворотній процес декомпозиції початкового відношення на декілька більш простих відношень меншої розмірності. Склад атрибутів відношень нормалізованої БД повинен відповідати таким вимогам:

1) між атрибутами не повинно бути небажаних функціональних залежностей;

2) групування атрибутів має забезпечувати мінімальне дублювання даних, їх обробку і поновлення без ускладнень та аномалій;

3) отримані в результаті декомпозиції відношення не повинні втратити функціональних залежностей початкового відношення, бо це може призвести до спотворення семантики даного відношення.

Розроблювана база даних має унікальні та атомарні атрибути. Всі відношення перебувають у 1НФ. У кожному відношенні не ключові атрибути функціонально повно залежать від своїх ключів — залежать від усього ключа і не залежать від його складових. Це свідчить про те, що відношення даної бази даних перебувають і в 2НФ. Жодне з відношень бази даних не має транзитивних залежностей, всі неключові атрибути в кожному з них взаємно незалежні. Отже, відношення перебувають у 3НФ.

3.3. Даталогічна модель

На рисунку 2 представлена даталогічна модель

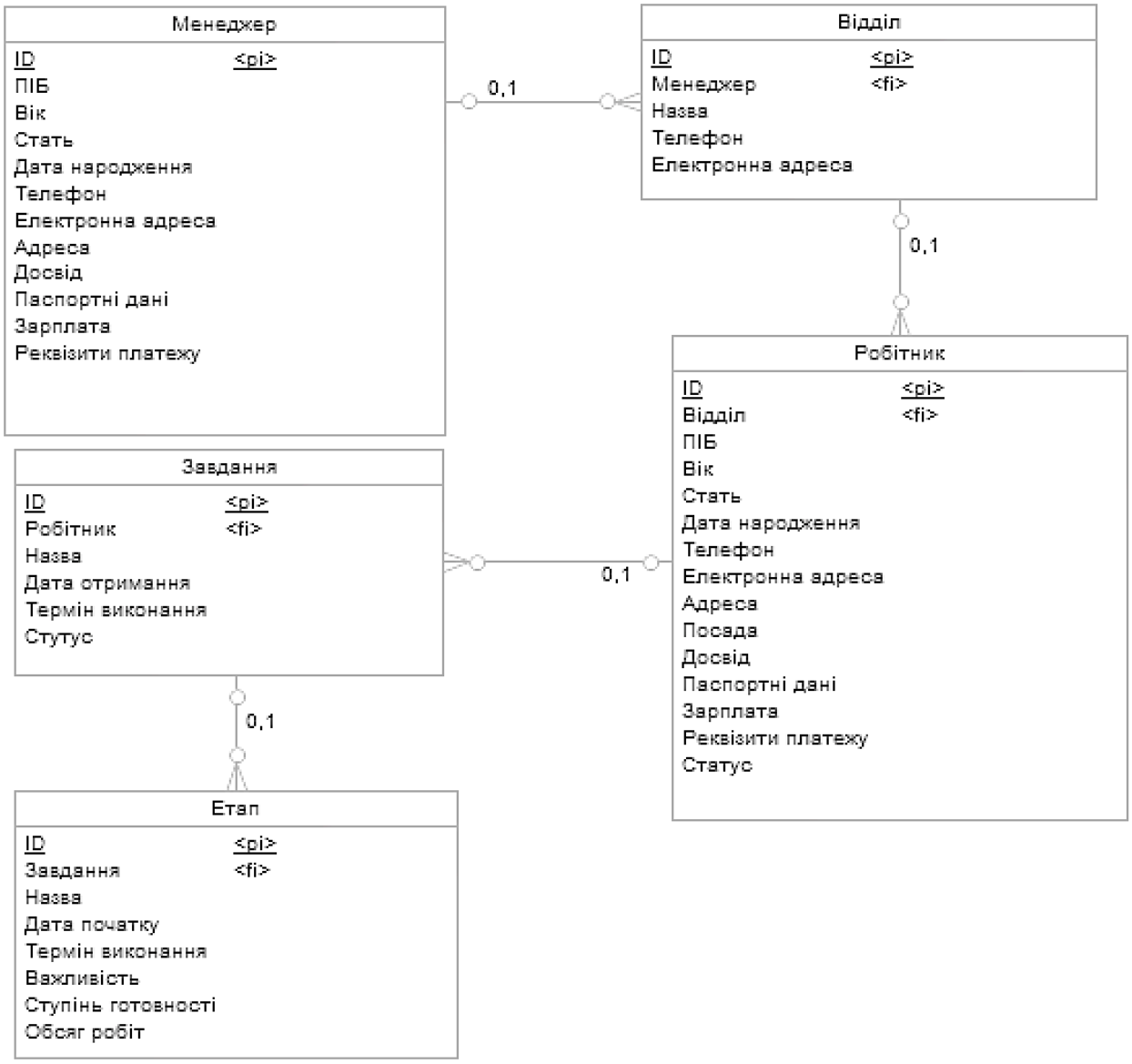


Рисунок 2 — побудована інфологічна модель

Біля зовнішніх ключів проставлений атрибут <fi> (foreign identificator).

Частина 4. Проектування та реалізація БД на фізичному рівні

4.1. Обмеження полів

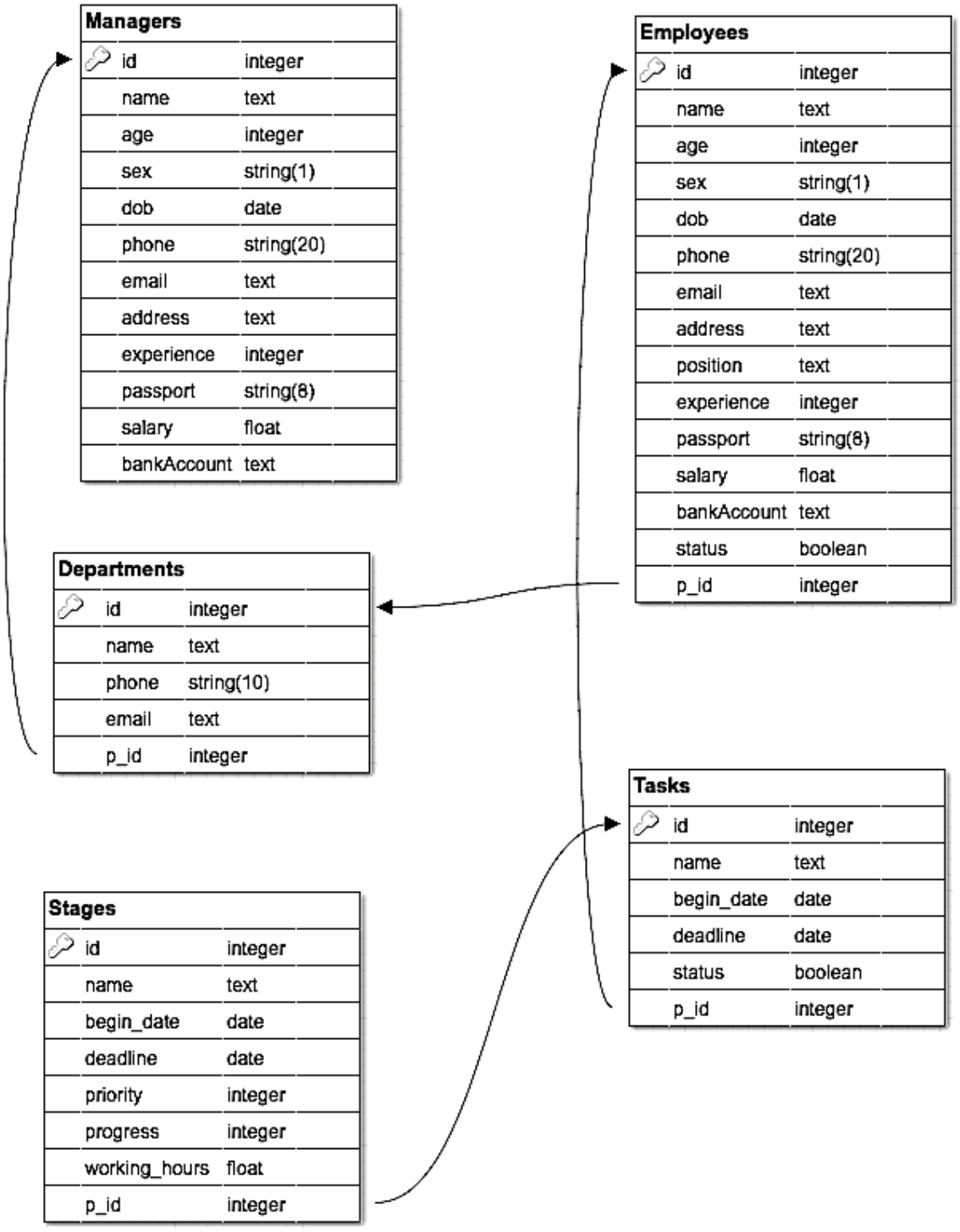
На етапі проектування фізичної моделі необхідно передбачити обмеження на значення полів в таблицях бази даних. Тлумачення назв полів та таблиць наведено у словнику (див. додаток).

Таблиця	*Поле БД*	*Обмеження, коментар*
Managers	id	автоінкрементне поле
	name	
	age	18 — 100
	sex	
	dob	
	phone	
	email	
	address	
	experience	0 — 92
	passport	
	salary	> 0
	bankAccount	
Departments	id	автоінкрементне поле
	name	
	phone	
	email	
	p_id	зовнішній ключ
Employees	id	автоінкрементне поле
	name	
	age	18 — 100
	sex	
	dob	
	phone	

	email	
	address	
	position	
	experience	0 — 92
	passport	
	salary	> 0
	bankAccount	
	status	0 — 10
	p_id	зовнішній ключ
Tasks	id	автоінкрементне поле
	name	
	begin_date	
	deadline	> begin_date
	status	0 — 10
	p_id	зовнішній ключ
Stages	id	автоінкрементне поле
	name	
	begin_date	
	deadline	> begin_date
	priority	0 — 10
	progress	0 — 100
	working_hours	> 0
	p_id	зовнішній ключ

4.2. Фізична модель

Рисунок 3 — фізична модель



4.3. Мобільний застосунок

Оскільки проект включає не просто розробку бази даних, а й використання її в застосунку під операційну систему Andorid, необхідно розробити інтерфейс (як програмний, так і графічний) для зручного користування мобільним додатком. Це передбачає два сценарії використання:

1) маніпулювання адміністратором напряму базою даних;
2) виведення даних, що містять узагальнення та розрахункові атрибути у зручному для менеджера вигляді.

Перший пункт передбачає створення інтерфейсів для додавання, перегляду, коригування та видалення записів. Потрібно розробити систему тригерів для безпечного видалення даних із залежних таблиць, а також накласти додаткові умови на введення та зміни даних:

1) для введення дати та часу використовується спеціальний віджет, який унеможливлює введення користувачем некорректних даних;
2) для введення інших числових значень використовується спеціальний класс, що контролює вхідні символи та пропускає лише придатні символи (в цьому випадку, числові значення);
3) вибір зовнішнього ключа для зв’язування таблиць реалізовано шляхом використання контекстного меню, переліком полів якого є можливі значення зовнішнього ключа; такий підхід не дає змогу користувачеві некорректно зв’язати таблиці, що могло би призвести до непередбачуваних наслідків.

Другий пункт передбачає створення звітів, що матимуть статистично важливі для менеджера дані, та розрахунок параметрів, що дають змогу проаналізувати ефективність роботи персоналу та актуальний стан виконання завдань (для детальних прикладів див. додаток).

Висновки

Як бачимо, база даних для даної предметної області вийшла не надто складною, але вельми корисною для практичного застосування. В поєднанні з мобільним застосунком вона дозволяє розподіляти завдання між робітниками, контролювали та аналізувати їх виконання, контролювати робочий процес на підприємстві та поліпшувати ефективність роботи компанії.

Завдяки курсовому проекту я набув навичок проектування прикладних баз даних із застосуванням сучасних CASE-засобів і СКБД, що відповідають вимогам нормалізації і добре підходять для використання в програмних продуктах.

Крім того, я дізнався, як можна на практиці використовувати бази даних у сучасних мобільних застосунках під операційну систему Android. Створена програма стане мені в нагоді як засіб управління задачами у власних проектах.

Джерела

1) https://www.wikipedia.org
2) https://developer.android.com
3) https://www.forbes.com
4) http://www.ab-inbev.com/investors
5) Бадрак Ю.А., Донченко М.В., Журавська І.М., Фисун М.Т. Методичні вказівки до оформлення звітної текстової документації та кваліфікаційних робіт з дисциплін, закріплених за факультетом комп'ютерних наук. – Миколаїв: Вид-во ЧДУ ім. Петра Могили, 2009. – 42 с.
6) Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – [Електронний ресурс]. – Режим доступу: http://www.citforum.ru.
7) Дейт К.Дж. Введение в системы баз данных. – Киев–Москва: Диалектика, 1998. – 787 с.
8) ДСТУ 2874-94. Системи оброблення інформації. Бази даних. Терміни та визначення. – К.: Держстандарт України.
9) Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е издание. – Москва-Санкт-Петербург-Киев: «Вильямс», 2000. – 1112 с.
10) Маклаков С.В. BPWin и ERWin. CASE-средства разработки информационных систем. – М.: ДИАЛОГ-МИФИ, 2000. – 256 с.
11) Пасічник В.В. Організація баз даних та знань, 2006. – 384 с.
12) Фісун М.Т., Ніколенко С.Г. Створення та ведення баз даних засобами мови Jet SQL: методичні вказівки до виконання лабораторних робіт з дисципліни «Організація баз даних». – Миколаїв: вид-во ЧДУ ім. Петра Могили, 2009. – 89 с.

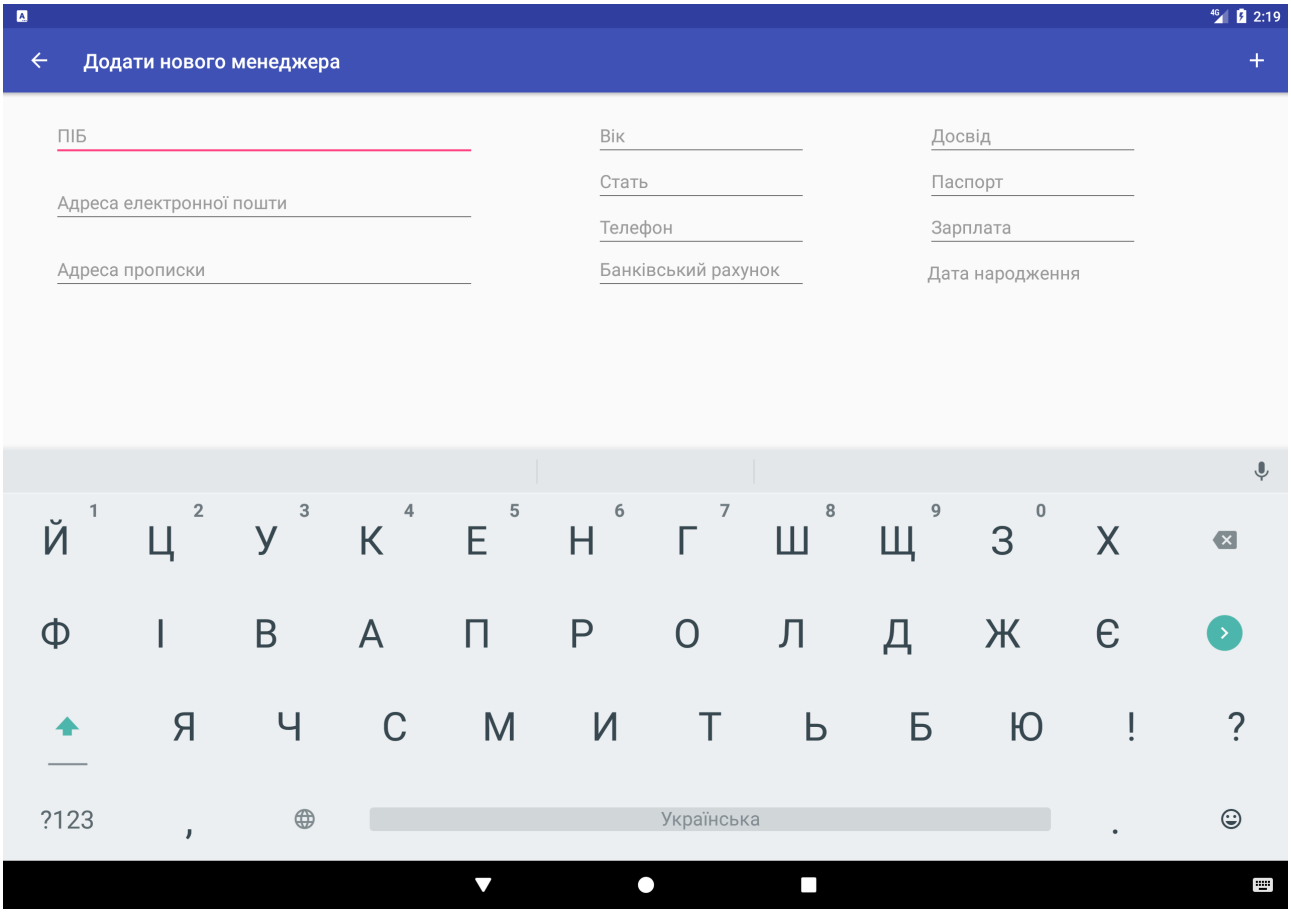
Додатки

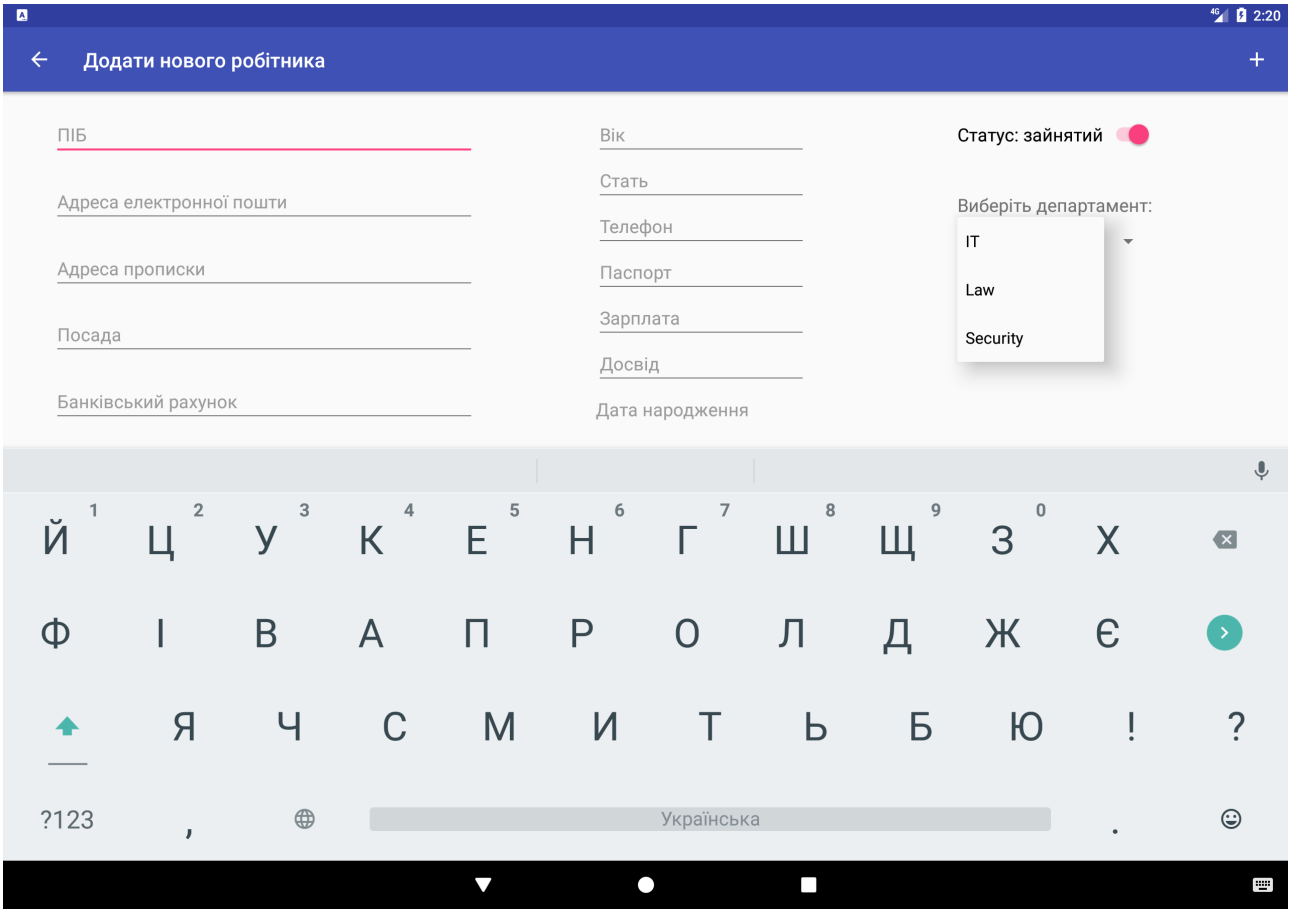
Словник бази даних

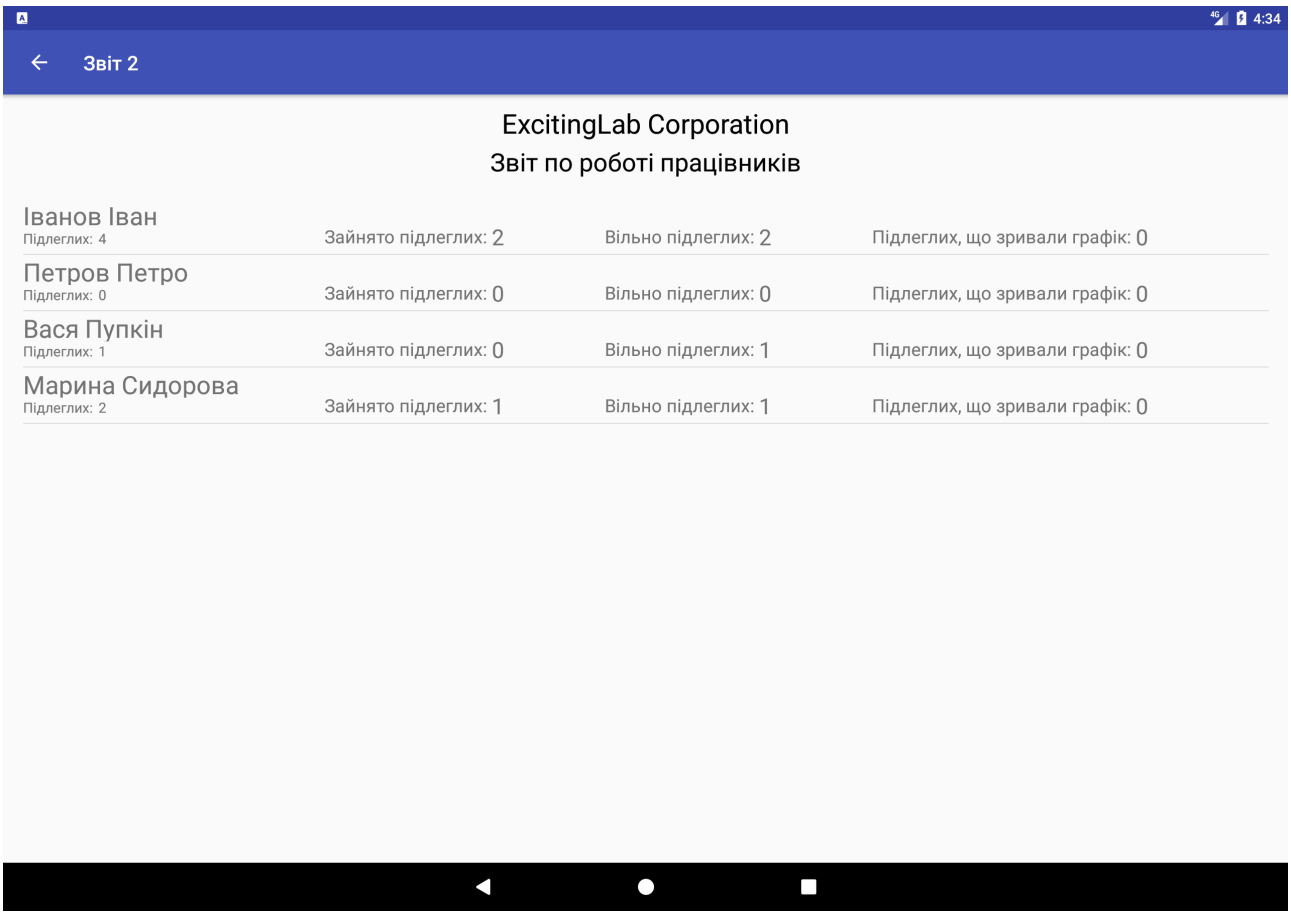
Таблиця	*Поле БД*	*Назва українською*
Managers (Менеджери)	id	Первинний ключ
	name	ПІБ
	age	Вік
	sex	Стать
	dob	Дата народження
	phone	Телефон
	email	Електронна пошта
	address	Адреса
	experience	Досвід роботи
	passport	Номер паспорта
	salary	Зарплата
	bankAccount	Банківський рахунок
Departments (Відділи)	id	Первинний ключ
	name	Назва
	phone	Телефон
	email	Електронна пошта
	p_id	Зовнішній ключ
Employees (Працівники)	id	Первинний ключ
	name	ПІБ
	age	Вік
	sex	Стать
	dob	Дата народження
	phone	Телефон
	email	Електронна пошта
	address	Адреса
	position	Посада
	experience	Досвід роботи

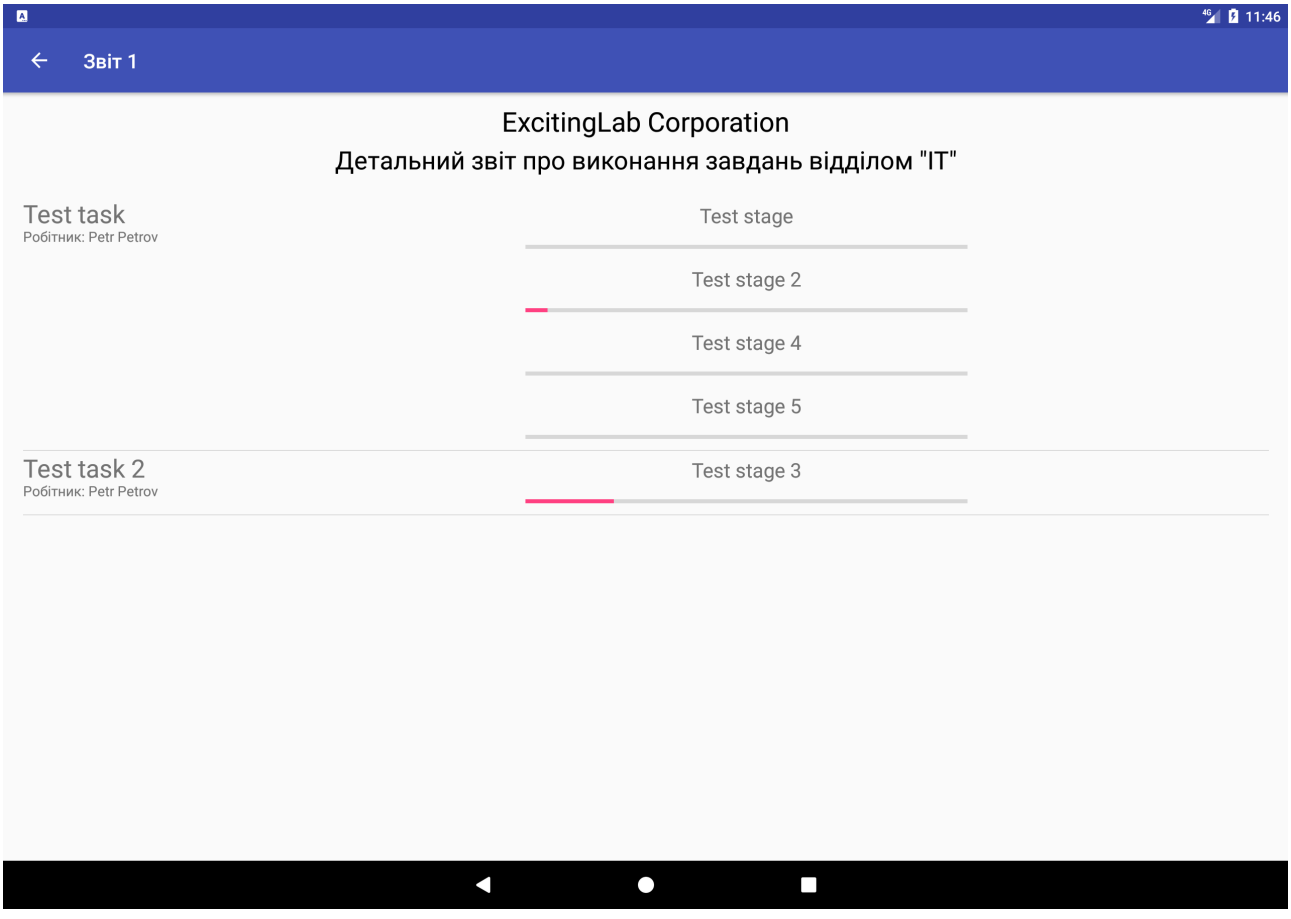
	passport	Номер паспорта
	salary	Зарплата
	bankAccount	Банківський рахунок
	status	Статус
	p_id	Зовнішній ключ
Tasks (Завдання)	id	Первинний ключ
	name	Назва
	begin_date	Дата початку
	deadline	Дата здачі
	status	Статус
	p_id	Зовнішній ключ
Stages (Етапи)	id	Первинний ключ
	name	Назва
	begin_date	Дата початку
	deadline	Дата здачі
	priority	Важливість
	progress	Прогрес виконання
	working_hours	Час на виконання
	p_id	Зовнішній ключ

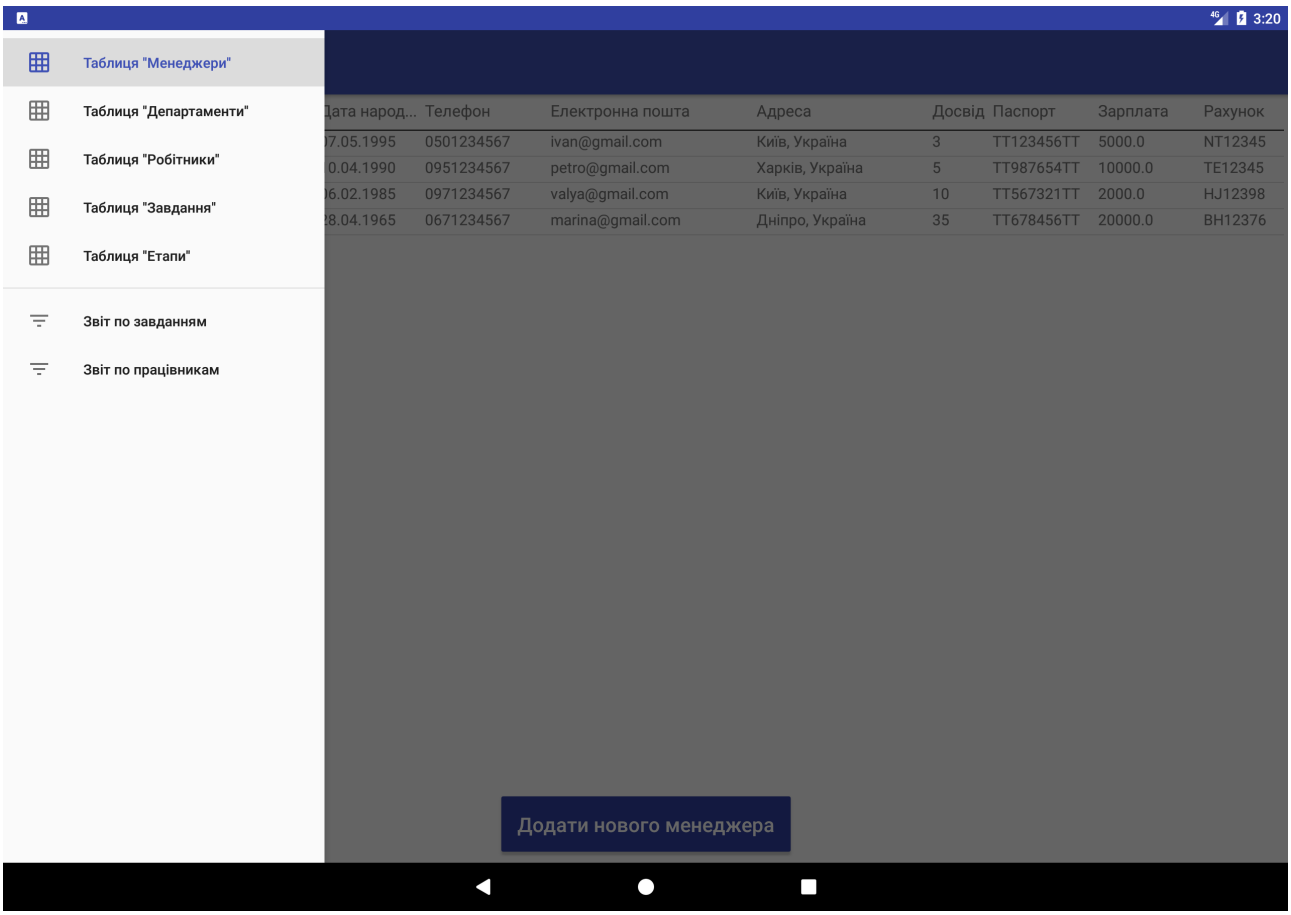
Знімки екрану

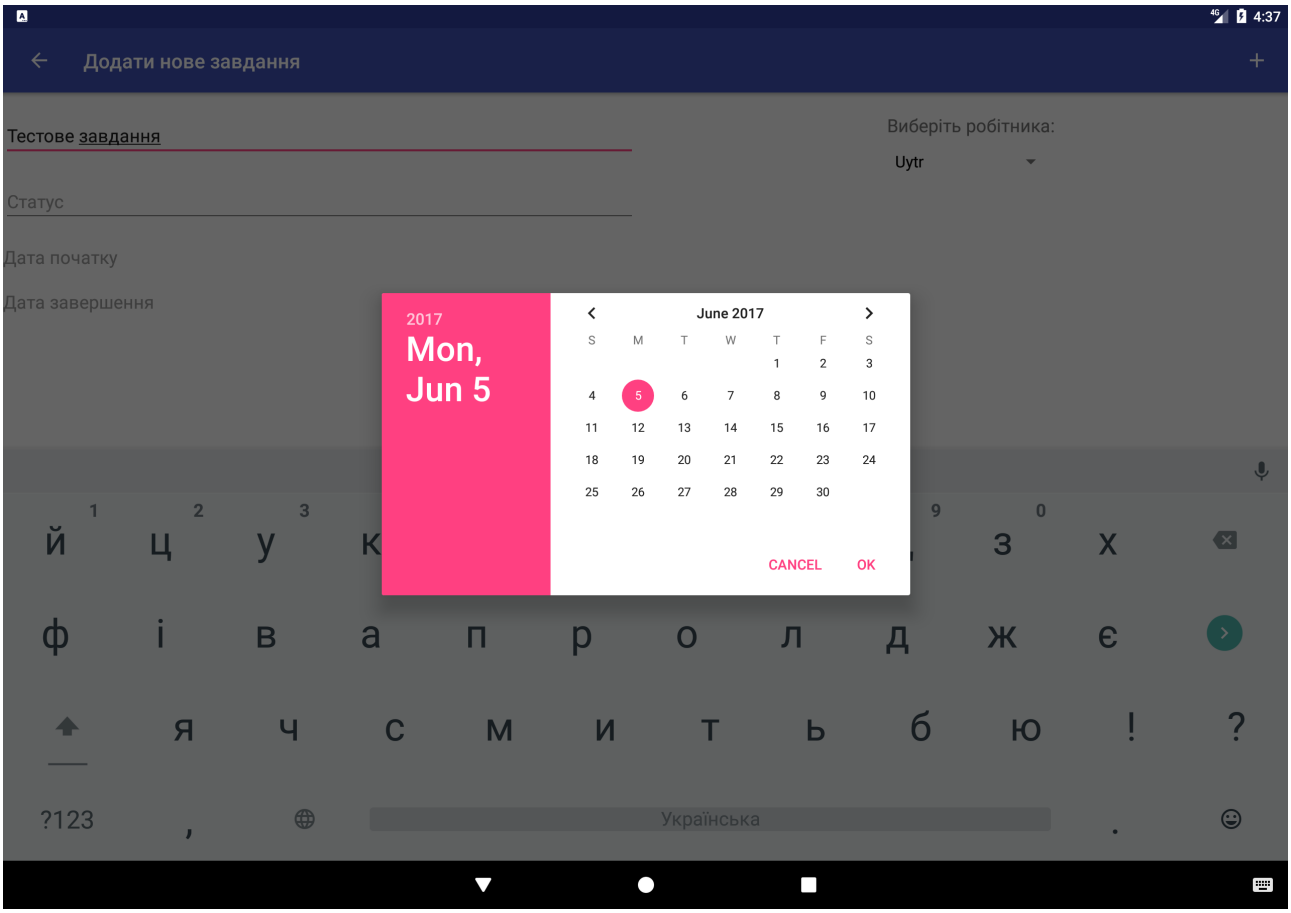





4:34
Звіт 2
ExcitingLab Corporation
Звіт по роботі працівників
Іванов Іван
Підлеглих: 4
Зайнято підлеглих: 2
Вільно підлеглих: 2
Підлеглих, що зривали графік: 0
Петров Петро
Підлеглих: 0
Зайнято підлеглих: 0
Вільно підлеглих: 0
Підлеглих, що зривали графік: 0
Вася Пупкін
Підлеглих: 1
Зайнято підлеглих: 0
Вільно підлеглих: 1
Підлеглих, що зривали графік: 0
Марина Сидорова
Підлеглих: 2
Зайнято підлеглих: 1
Вільно підлеглих: 1
Підлеглих, що зривали графік: 0


11:46
Звіт 1
ExcitingLab Corporation
Детальний звіт про виконання завдань відділом "IT"
Test task
Робітник: Petr Petrov
Test stage
Test stage 2
Test stage 4
Test stage 5
Test task 2
Робітник: Petr Petrov
Test stage 3


3:20
Таблиця "Менеджери"
Таблиця "Департаменти"
Таблиця "Робітники"
Таблиця "Завдання"
Таблиця "Етапи"
Звіт по завданням
Звіт по працівникам
Дата народ... Телефон Електронна пошта Адреса Досвід Паспорт Зарплата Рахунок
7.05.1995 0501234567 ivan@gmail.com Київ, Україна 3 TT123456TT 5000.0 NT12345
0.04.1990 0951234567 petro@gmail.com Харків, Україна 5 TT987654TT 10000.0 TE12345
6.02.1985 0971234567 valya@gmail.com Київ, Україна 10 TT567321TT 2000.0 HJ12398
8.04.1965 0671234567 marina@gmail.com Дніпро, Україна 35 TT678456TT 20000.0 BH12376
Додати нового менеджера


4:37
Додати нове завдання
Тестове завдання
Статус
Дата початку
Дата завершення
Виберіть робітника:
Uytr
2017
Mon, Jun 5
June 2017
S M T W T F S
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
CANCEL
OK
Українська

Лістинг коду

Створення таблиць та зв’яків на чистому SQL

```
CREATE TABLE `Departments` (
	`id` INT NOT NULL,
	`name` TEXT NOT NULL,
	`phone` VARCHAR(10) NOT NULL UNIQUE,
	`email` TEXT NOT NULL,
	`p_id` INT NOT NULL,
	PRIMARY KEY (`id`)
);
CREATE TABLE `Employees` (
	`id` INT NOT NULL,
	`name` TEXT NOT NULL,
	`age` INT NOT NULL,
	`sex` VARCHAR(1) NOT NULL,
	`dob` DATE NOT NULL,
	`phone` VARCHAR(20) NOT NULL,
	`email` TEXT NOT NULL,
	`address` TEXT NOT NULL,
	`position` TEXT NOT NULL,
	`experience` INT NOT NULL,
	`passport` VARCHAR(8) NOT NULL UNIQUE,
	`salary` FLOAT NOT NULL,
	`bankAccount` TEXT NOT NULL,
	`status` BOOLEAN NOT NULL,
	`p_id` INT NOT NULL,
	PRIMARY KEY (`id`)
);
CREATE TABLE `Tasks` (
	`id` INT NOT NULL,
	`name` TEXT NOT NULL,
	`begin_date` DATE NOT NULL,
	`deadline` DATE NOT NULL,
	`status` BOOLEAN NOT NULL,
	`p_id` INT NOT NULL,
	PRIMARY KEY (`id`)
);
CREATE TABLE `Stages` (
	`id` INT NOT NULL,
	`name` TEXT NOT NULL,
	`begin_date` DATE NOT NULL,
	`deadline` DATE NOT NULL,
	`priority` INT NOT NULL,
	`progress` INT NOT NULL,
	`working_hours` FLOAT NOT NULL,
	`p_id` INT NOT NULL,
	PRIMARY KEY (`id`)
);
CREATE TABLE `Managers` (
	`id` INT NOT NULL,
	`name` TEXT NOT NULL,
	`age` INT NOT NULL,
	`sex` VARCHAR(1) NOT NULL,
	`dob` DATE NOT NULL,
	`phone` VARCHAR(20) NOT NULL,
	`email` TEXT NOT NULL,
	`address` TEXT NOT NULL,
	`experience` INT NOT NULL,
	`passport` VARCHAR(8) NOT NULL UNIQUE,
	`salary` FLOAT NOT NULL,
	`bankAccount` TEXT NOT NULL,
	PRIMARY KEY (`id`)
);
ALTER TABLE `Departments` ADD CONSTRAINT `Departments_fk0` FOREIGN KEY (`p_id`) REFERENCES
`Managers`(`id`);
ALTER TABLE `Employees` ADD CONSTRAINT `Employees_fk0` FOREIGN KEY (`p_id`) REFERENCES
`Departments`(`id`);
ALTER TABLE `Tasks` ADD CONSTRAINT `Tasks_fk0` FOREIGN KEY (`p_id`) REFERENCES `Employees`(`id`);
ALTER TABLE `Stages` ADD CONSTRAINT `Stages_fk0` FOREIGN KEY (`p_id`) REFERENCES `Tasks`(`id`);
```

```
private static DatabaseHelper sInstance;
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_NAME = "Coursework.db";
private static final String TABLE_MANAGERS = "managers";
private static final String TABLE_DEPARTMENTS = "departments";
private static final String TABLE_EMPLOYEES = "employees";
private static final String TABLE_TASKS = "tasks";
private static final String TABLE_STAGES = "stages";

private static final String MANAGER_ID = "id";
private static final String MANAGER_NAME = "name";
private static final String MANAGER_AGE = "age";
private static final String MANAGER_SEX = "sex";
private static final String MANAGER_DOB = "dob";
private static final String MANAGER_PHONE = "phone";
private static final String MANAGER_EMAIL = "email";
private static final String MANAGER_ADDRESS = "address";
private static final String MANAGER_EXPERIENCE = "experience";
private static final String MANAGER_PASSPORT = "passport";
private static final String MANAGER_SALARY = "salary";
private static final String MANAGER_BANK_ACCOUNT = "bankAccount";

private static final String DEPARTMENT_ID = "id";
private static final String DEPARTMENT_NAME = "name";
private static final String DEPARTMENT_PHONE = "phone";
private static final String DEPARTMENT_EMAIL = "email";
private static final String DEPARTMENT_P_ID = "p_id";

private static final String EMPLOYEE_ID = "id";
private static final String EMPLOYEE_NAME = "name";
private static final String EMPLOYEE_AGE = "age";
private static final String EMPLOYEE_SEX = "sex";
private static final String EMPLOYEE_DOB = "dob";
private static final String EMPLOYEE_PHONE = "phone";
private static final String EMPLOYEE_EMAIL = "email";
private static final String EMPLOYEE_ADDRESS = "address";
private static final String EMPLOYEE_POSITION = "position";
private static final String EMPLOYEE_EXPERIENCE = "experience";
private static final String EMPLOYEE_PASSPORT = "passport";
private static final String EMPLOYEE_SALARY = "salary";
private static final String EMPLOYEE_BANK_ACCOUNT = "bankAccount";
private static final String EMPLOYEE_STATUS = "status";
private static final String EMPLOYEE_P_ID = "p_id";

private static final String TASK_ID = "id";
private static final String TASK_NAME = "name";
private static final String TASK_BEGIN_DATE = "beginDate";
private static final String TASK_DEADLINE = "deadline";
private static final String TASK_STATUS = "status";
private static final String TASK_P_ID = "p_id";

private static final String STAGE_ID = "id";
private static final String STAGE_NAME = "name";
private static final String STAGE_BEGIN_DATE = "beginDate";
private static final String STAGE_DEADLINE = "deadline";
private static final String STAGE_PRIORITY = "priority";
private static final String STAGE_PROGRESS = "progress";
private static final String STAGE_WORKING_HOURS = "workingHours";
private static final String STAGE_P_ID = "p_id";

private static final String CREATE_TABLE_MANAGERS = "CREATE TABLE "
        + TABLE_MANAGERS + "(" + MANAGER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + MANAGER_NAME + " TEXT, " +
        MANAGER_AGE + " INTEGER, " +
        MANAGER_SEX + " TEXT, " +
        MANAGER_DOB + " LONG, " +
        MANAGER_PHONE + " TEXT, " +
        MANAGER_EMAIL + " TEXT, " +
        MANAGER_ADDRESS + " TEXT, " +
        MANAGER_EXPERIENCE + " INTEGER, " +
        MANAGER_PASSPORT + " TEXT, " +
        MANAGER_SALARY + " REAL, " +
        MANAGER_BANK_ACCOUNT + " TEXT)";

private static final String CREATE_TABLE_DEPARTMENTS = "CREATE TABLE " + TABLE_DEPARTMENTS
```

```
        + "(" + DEPARTMENT_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + DEPARTMENT_NAME + " TEXT, "
+
        DEPARTMENT_PHONE + " TEXT, " +
        DEPARTMENT_EMAIL + " TEXT, " +
        DEPARTMENT_P_ID + " INTEGER)";

private static final String CREATE_TABLE_EMPLOYEES = "CREATE TABLE "
        + TABLE_EMPLOYEES + "(" + EMPLOYEE_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + EMPLOYEE_NAME + " TEXT, " +
        EMPLOYEE_AGE + " INTEGER, " +
        EMPLOYEE_SEX + " TEXT, " +
        EMPLOYEE_DOB + " LONG, " +
        EMPLOYEE_PHONE + " TEXT, " +
        EMPLOYEE_EMAIL + " TEXT, " +
        EMPLOYEE_ADDRESS + " TEXT, " +
        EMPLOYEE_POSITION + " TEXT, " +
        EMPLOYEE_EXPERIENCE + " INTEGER, " +
        EMPLOYEE_PASSPORT + " TEXT, " +
        EMPLOYEE_SALARY + " REAL, " +
        EMPLOYEE_BANK_ACCOUNT + " TEXT, " +
        EMPLOYEE_STATUS + " INTEGER, " +
        EMPLOYEE_P_ID + " INTEGER)";

private static final String CREATE_TABLE_TASKS = "CREATE TABLE "
        + TABLE_TASKS + "(" + TASK_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + TASK_NAME + " TEXT, " +
        TASK_BEGIN_DATE + " LONG, " +
        TASK_DEADLINE + " LONG, " +
        TASK_STATUS + " INTEGER, " +
        TASK_P_ID + " INTEGER)";

private static final String CREATE_TABLE_STAGES = "CREATE TABLE "
        + TABLE_STAGES + "(" + STAGE_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + STAGE_NAME + " TEXT, " +
        STAGE_BEGIN_DATE + " LONG, " +
        STAGE_DEADLINE + " LONG, " +
        STAGE_PRIORITY + " INTEGER, " +
        STAGE_PROGRESS + " INTEGER, " +
        STAGE_WORKING_HOURS + " REAL, " +
        STAGE_P_ID + " INTEGER)";

public static synchronized DatabaseHelper getInstance(Context context) {
    if (sInstance == null) {
        sInstance = new DatabaseHelper(context.getApplicationContext());
    }
    return sInstance;
}

private Context context;
private DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
    this.context = context;
}

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE_MANAGERS);
    db.execSQL(CREATE_TABLE_DEPARTMENTS);
    db.execSQL(CREATE_TABLE_EMPLOYEES);
    db.execSQL(CREATE_TABLE_TASKS);
    db.execSQL(CREATE_TABLE_STAGES);

    Log.e("MANAGER: ", CREATE_TABLE_MANAGERS);
    Log.e("DEPARTMENT: ", CREATE_TABLE_DEPARTMENTS);
    Log.e("EMPLOYEE: ", CREATE_TABLE_EMPLOYEES);
    Log.e("TASK: ", CREATE_TABLE_TASKS);
    Log.e("STAGE: ", CREATE_TABLE_STAGES);
}
```

Приклади запитів на чистому SQL

```
SELECT managers.name, departments.name, tasks.name,
tasks.status, stages.name, stages.priority, stages.progress
FROM managers
JOIN departments ON managers.id = departments.p_id
JOIN employees ON departments.id = employees.p_id
JOIN tasks ON employees.id = tasks.p_id
JOIN stages ON tasks.id = stages.p_id
WHERE tasks.status = 0
ORDER BY tasks.beginDate DESC;


SELECT COUNT (employees.name) FROM managers
LEFT JOIN departments ON managers.id = departments.p_id
RIGHT JOIN employees ON departments.id = employees.p_id
JOIN tasks ON employees.id = tasks.p_id
INNER JOIN stages ON tasks.id = stages.p_id
WHERE tasks.status = 1
AND stages.progress > 0
AND stages.progress < 20
ORDER BY tasks.beginDate ASC;


SELECT name, SUM (salary)
FROM employees
GROUP BY name
HAVING SUM (salary) > 1000;

SELECT id, SUM (salary)
FROM managers
GROUP BY id
HAVING SUM (salary) < 5000;

SELECT * FROM managers
CROSS JOIN departments;

SELECT tasks.name FROM tasks
UNION ALL
SELECT stages.name FROM stages;
```

Приклади Java-функцій із застосунку, що використовують SQL запити

```
public Cursor getData(int status){
        SQLiteDatabase db = this.getWritableDatabase();

        String query = "SELECT " +
                TABLE_MANAGERS + "." + MANAGER_NAME + ", " +
                TABLE_DEPARTMENTS + "." + DEPARTMENT_NAME + ", " +
                TABLE_TASKS + "." + TASK_NAME + ", " +
                TABLE_TASKS + "." + TASK_STATUS + ", " +
                TABLE_STAGES + "." + STAGE_NAME + ", " +
                TABLE_STAGES + "." + STAGE_PRIORITY + ", " +
                TABLE_STAGES + "." + STAGE_PROGRESS +
                " FROM " + TABLE_MANAGERS +
                " JOIN " + TABLE_DEPARTMENTS + " ON " +
                TABLE_MANAGERS + "." + MANAGER_ID + " = " + TABLE_DEPARTMENTS + "." +
DEPARTMENT_P_ID +
                " JOIN " + TABLE_EMPLOYEES + " ON " +
                TABLE_DEPARTMENTS + "." + DEPARTMENT_ID + " = " + TABLE_EMPLOYEES + "." +
EMPLOYEE_P_ID +
                " JOIN " + TABLE_TASKS + " ON " +
                TABLE_EMPLOYEES + "." + EMPLOYEE_ID + " = " + TABLE_TASKS + "." + TASK_P_ID +
                " JOIN " + TABLE_STAGES + " ON " +
                TABLE_TASKS + "." + TASK_ID + " = " + TABLE_STAGES + "." + STAGE_P_ID +
                " WHERE " + TABLE_TASKS + "." + TASK_STATUS + " = " + status +
                " ORDER BY " + TABLE_TASKS + "." + TASK_BEGIN_DATE + " DESC;";

        return db.rawQuery(query, null);
}
```

```
public int getEmployeesCount(int id) {
        String selectQuery = "SELECT " + "COUNT (" + TABLE_EMPLOYEES + "." + EMPLOYEE_ID + ")" +
                " FROM " + TABLE_MANAGERS +
                " JOIN " + TABLE_DEPARTMENTS + " ON " +
                TABLE_MANAGERS + "." + MANAGER_ID + " = " + TABLE_DEPARTMENTS + "." +
DEPARTMENT_P_ID +
                " JOIN " + TABLE_EMPLOYEES + " ON " +
                TABLE_DEPARTMENTS + "." + DEPARTMENT_ID + " = " + TABLE_EMPLOYEES + "." +
EMPLOYEE_P_ID +
                " WHERE " + TABLE_MANAGERS + "." + MANAGER_ID + " = " + id;

        Log.e("COUNT: ", selectQuery);

        SQLiteDatabase db = this.getReadableDatabase();
        Cursor c = db.rawQuery(selectQuery, null);

        int count = 0;

        if (c.moveToFirst()) {
            count = c.getInt(0);
        }

        c.close();
        db.close();

        return count;
}



public List<Employee> getEmployees(int id) {
        List<Employee> employees = new ArrayList<>();
        String selectQuery = "SELECT * FROM " + TABLE_MANAGERS +
                " JOIN " + TABLE_DEPARTMENTS + " ON " +
                TABLE_MANAGERS + "." + MANAGER_ID + " = " + TABLE_DEPARTMENTS + "." +
DEPARTMENT_P_ID +
                " JOIN " + TABLE_EMPLOYEES + " ON " +
                TABLE_DEPARTMENTS + "." + DEPARTMENT_ID + " = " + TABLE_EMPLOYEES + "." +
EMPLOYEE_P_ID +
                " WHERE " + TABLE_MANAGERS + "." + MANAGER_ID + " = " + id;

        Log.e("ALL: ", selectQuery);

        SQLiteDatabase db = this.getReadableDatabase();
        Cursor c = db.rawQuery(selectQuery, null);

        if (c.moveToFirst()) {
            do {
                Employee t = new Employee();
                t.setId(c.getInt((c.getColumnIndex(EMPLOYEE_ID))));
                t.setName(c.getString(c.getColumnIndex(EMPLOYEE_NAME)));
                t.setAge(c.getInt(c.getColumnIndex(EMPLOYEE_AGE)));
                t.setSex(c.getString(c.getColumnIndex(EMPLOYEE_SEX)));
                t.setDob(c.getLong(c.getColumnIndex(EMPLOYEE_DOB)));
                t.setPhone(c.getString(c.getColumnIndex(EMPLOYEE_PHONE)));
                t.setEmail(c.getString(c.getColumnIndex(EMPLOYEE_EMAIL)));
                t.setAddress(c.getString(c.getColumnIndex(EMPLOYEE_ADDRESS)));
                t.setPosition(c.getString(c.getColumnIndex(EMPLOYEE_POSITION)));
                t.setExperience(c.getInt(c.getColumnIndex(EMPLOYEE_EXPERIENCE)));
                t.setPassport(c.getString(c.getColumnIndex(EMPLOYEE_PASSPORT)));
                t.setSalary(c.getDouble(c.getColumnIndex(EMPLOYEE_SALARY)));
                t.setBankAccount(c.getString(c.getColumnIndex(EMPLOYEE_BANK_ACCOUNT)));
                t.setStatus(c.getInt(c.getColumnIndex(EMPLOYEE_STATUS)));
                t.setP_id(c.getInt(c.getColumnIndex(EMPLOYEE_P_ID)));
                employees.add(t);
            } while (c.moveToNext());
        }

        c.close();
        db.close();

        return employees;
    }
```

```
public boolean updateStage(Stage stage) {
        SQLiteDatabase db = this.getWritableDatabase();

        int id = stage.getId();
        String selectQuery = "SELECT  * FROM " + TABLE_STAGES + " WHERE " + STAGE_ID + " = " + id;

        ContentValues values = new ContentValues();
        values.put(STAGE_NAME, stage.getName());
        values.put(STAGE_BEGIN_DATE, stage.getBeginDate());
        values.put(STAGE_DEADLINE, stage.getDeadline());
        values.put(STAGE_PRIORITY, stage.getPriority());
        values.put(STAGE_PROGRESS, stage.getProgress());
        values.put(STAGE_WORKING_HOURS, stage.getWorkingHours());
        values.put(STAGE_P_ID, stage.getP_id());

        long i = db.update(TABLE_STAGES, values, STAGE_ID + " = ?",
                new String[]{String.valueOf(stage.getId())});

        db.close();

        if (i == -1){
            return false;
        }
        else {
            return true;
        }
}
```

(Знімки екранів з формами, що реалізуються наведеними запитими, див. вище у розілі «знімки екрану»)