МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

Організація баз даних:

МЕТОДИЧНІ ВКАЗІВКИ

до самостійної роботи студентів за спеціальностями
6.050102/123 «Комп'ютерна інженерія»,
125 «Кібербезпека»

ЗАТВЕРДЖЕНО
кафедрою кібербезпеки та
програмного забезпечення,
протокол від 5липня 2017 року № 1

КРОПИВНИЦЬКИЙ
2017

Організація баз даних: Методичні вказівки до самостійної роботи студентів за спеціальностями 6.050102/123 «Комп'ютерна інженерія»,125 «Кібербезпека» / уклад. В.В. Сидоренко, Л.В. Константинова—Кропивницький: ЦНТУ, 2017. —88 с.

Укладач: СидоренкоВ. В., Константинова Л. В.

Рецензенти: СмірновО. А., д-р техн. наук, професор;
Дрєєв О. М., канд. техн. наук.

Теми самостійної роботи

з навчальної дисципліни “Організація баз даних” для студентів очної форми навчання за спеціальностями 6.050102/123 “Комп’ютерна інженерія”, 125 “Кібербезпека”

уклад. Сидоренко В.В.

№з. с.	Назва теми	Кі-сть год.		
		Д.ф.	З. ф.	
1	*2*	*3*	*4*	*5*
1	**Тема 1.** Загальні поняття інформаційних систем.	4	8	15
2	**Тема2.** Архітектура інформаційних систем.	4	8	15
3	**Тема3.** Моделювання даних. Модель «об’єкт-атрибут-зв’язок».	4	8	16
4	**Тема4.** Теоретичні мови запитів.	4	8	15
5	**Тема5.** Додаткові операції реляційної алгебри запропоновані Дейтом.	6	8	15
6	**Тема6.**Мова реляційного числення за зразком QBE.	6	8	16
7	**Тема7.** Побудова інтерфейсу. Звіти. Основні оператори мови SQL.	4	8	16
8	**Тема8.**Історія мови SQL та огляд її можливостей.	2	4	5
9	**Тема 9.** SQL-оператори групування, теоретико множинні оператори.	2	4	5
10	**Тема10.**Додаткові можливості мови SQL. Віртуальні таблиці.	2	4	5
11	**Тема11.** Проектування і використання баз даних.	2	4	5
12	**Тема12**. Нормальні форми.	1	2	5
13	**Тема 13.**Рекомендації по розробці структур.	1	2	5
14	**Тема14.**Семантичне моделювання даних. ER – діаграми.	1	4	6
15	**Тема15.**Проектування концептуальної схеми бази даних. ER – моделювання даних.	1	4	6
16	**Тема16.**Етапи проектування баз даних.	1	4	6
17	**Тема17.**Приклад побудови ER-моделі.	1	4	6
18	**Тема18.**Зберігання інформації у базах даних.	1	4	6
19	**Тема19.**Індексація даних.	1	4	6

1	*2*	*3*	*4*	*5*
20	**Тема20.**Методологія функціонального моделювання.	1	4	6
21	**Тема21.**Інформаційні системи в мережах.	1	4	6
	Разом:	50	108	186

Самостійна робота №1

Тема: Загальні поняття інформаційних систем.

1.1. Поняття про предметну область і об'єкти

Об'єктно-орієнтований підхід являє собою метод, який дає можливість при розв’язанні складних задач в певній предметній області виділити та описати конкретні об'єкти, їх характеристики, зв'язки та взаємодію між ними.

Під **предметною областю (ПО)** прийнято розуміти частину реального світу, щопідлягає вивченню з метою організації керування і у кінцевому рахунку - автоматизації.

Це може бути підприємство, міністерство, вуз і т. д. Предметна область представляється множиною більш-менш самостійних та незалежних одна від одної частин - **фрагментів:** наприклад, підприємство — бухгалтерією, відділом кадрів, планово-фінансовою службою і т. д.

У той жечас фрагмент предметної області характеризується множиною об’єктів,кожний з яких можна описати набором конкретних характеристик— **атрибутів.** Об'єкт являє собою один типовий, але невизначений екземпляр чогось у реальному світі, наприклад, будь-який типовий літак. Зокрема, для бухгалтерії об'єкти - це договір підприємства, контрагент (фірма, з якою працює підприємство), співробітник, трудова угода, листки непрацездатності співробітників та ін.

Кожний об'єкт має зв'язки з іншими об'єктами. Об'єкти та зв'язки в часі мають життєвий цикл, який визначається як стан об'єктів та зв'язків.

Приклад 1: Об'єкт Договір має зв'язок з об'єктом Контрагент, а стан об'єкта Договір може бути одним з: укладений, виконаний, вступив в силу і т. д.

Приклад 2: Об'єкт Співробітник пов'язаний з об'єктом Трудова угода, а стан об'єкта Співробітник може бути одним з: працює, у відпустці, у відрядженні і т. д.

З об'єктом в тому чи іншому стані можуть відбуватися ті чи інші дії - процеси, що можуть бути підпорядкованими конкретним правилам або лініям поведінки.

Приклад 1. Об'єкт — Договір. Його стан —"виконується". Процеси - "матеріальний облік товарів, що надійшли", "облік витрат на відрядження за цим договором", "банківські операції".

Приклад 2. Об'єкт — Співробітник. Його стан - "працює". Процес - "розрахунок заробітної плати".

Об'єкт — це такаабстракціямножинипредметівреального світу, щовсіпредметивціймножині — екземпляри:- маютьті ж характеристики: - підпорядкованіі погоджуютьсяз одним і тим же набором правилі лінійповедінки.

Для **довідки:**
Абстракція *— це відволікання в процесі пізнання від окремих, несуттєвих сторін досліджуваного явища з метою зосередження на загальних, основних, суттєвих його рисах.*

Предмети в реальному світі мають різні характеристики. Наприклад, висота, температура, реєстраційний номер або положення. Кожна окрема характеристика, що є загальною для всіх можливих екземплярів об'єкта, абстрагується як окремий атрибут.

Атрибут —це абстракціяоднієї характеристики, яка властивавсім абстрагованим як об'єкт сутностям.

Кожний атрибут забезпечується ім'ям, унікальним у межах об'єкта. Атрибут може набувати значення для будь-якого визначеного екземпляра. Припустимо, що ми маємо об'єкт Кішка з атрибутами: Ім'я, Стать, Вага, Забарвлення, Темперамент. Тоді:

Дмитрикова Кішка:
Ім'я: Барсик
Стать: Ч. 
Вага: 3,3 кг
Забарвлення: буре, смугасте
Темперамент:
неймовірно ледача

Петрикова Кішка:
Ім'я: Саллі
Стать: Ж. 
Вага: 2,8 кг
Забарвлення: черепахове
Темперамент:
агресивна

Доменомназиваєтьсядіапазонприпустимихзначень, якихможенабуватиатрибут

Ідентифікаторомназиваєтьсямножина з одногоабо декількохатрибутів, значення якиходнозначновизначаютькожнийекземпляроб'єкта.

Кожен об'єкт повинен обов'язково мати свій ідентифікатор.

Об'єкт може мати декілька ідентифікаторів, кожен з яких складається з одного або декількох атрибутів. Наприклад, об'єкт Аеропорт може мати наступні атрибути:

- Код Аеропорту.
- Широта.
- Довгота.
- Місто.
- Кількість пасажирських пропускних пунктів.

Атрибут Код Аеропорту — ідентифікатор об'єкта Аеропорт, і комбінація Широти та Довготи — другий ідентифікатор Аеропорту.

Але для спрощення ми не розглядатимемо випадки наявності декількох ідентифікаторів, розглядатимемо об'єкти, які можуть бути однозначно визначені тільки одним ідентифікатором.

Наприклад, об'єкт Кішка однозначно ідентифікується ім'ям, якщо це ім'я унікальне, а якщо імена можуть повторюватися, то атрибут Ім'я не зможе бути ідентифікатором. Не зможе бути ідентифікатором і атрибут Вага, тому що можуть зустрітися кішки з однаковою вагою.

Тоді краще за все ввести ще один атрибут - Ідентифікаційний код (ID), який може складатися, наприклад, з перших трьох букв імені і номера (САЛ1, БАР2 і т. д.).

Кожен об'єкт може бути інтерпретований як таблиця. Наприклад, об'єкт Кішка:

Кішка					
ГО	Ім'я	Стать	Вага	Забарвлення	Темперамент
Бар1	Барсик	Ч	3,3	Буре, смугасте	Неймовірно ледачий
Сал1	Саллі	Ж	2,8	Черепахове	Агресивний
Лау1	Лаура	Ж	2,8	Кремове	Рухливий
Арн1	Арнольд	Ч	4,5	Кремове	Стриманий
Бар2	Варті	ж	3,8	Чорне	Ледачий
Ласі1	Ласпі	Ч	4,5	Черепахове	Рухливий
Пуш1	Пушок	Ч	5,2	Біле	Ніжний

1. Завдання

Об'єднати за допомогою різнокольорових стрілок кожне поняття з визначенням, яке йому відповідає.

Атрибут	Така абстракція множини предметів реального світу, що всі предмети в цій множині — екземпляри: - мають тіжхарактеристики та зв'язки; - підпорядковані і погоджуються з одним і тим же набором правил і ліній поведінки.
Домен	Множина з одного або декількох атрибутів, значення яких однозначно визначають кожний екземпляр об'єкта.
Ідентифікатор	Абстракція однієї характеристики, яка властива всімабстрагованим як об'єкт сутностям.
Об'єкт	Життєвий цикл (в часі) об'єктів та зв'язків між ними.
Предметна область	Дії, які відбуваються з об'єктом в тому чи іншому стані
Стан об'єкта	Частина реального світу, що підлягає вивченню з метою організації керування і в кінцевому рахунку— автоматизації.
Процес	Діапазон припустимих значень, яких може набувати атрибут.

1.2. Зв'язки між об'єктами

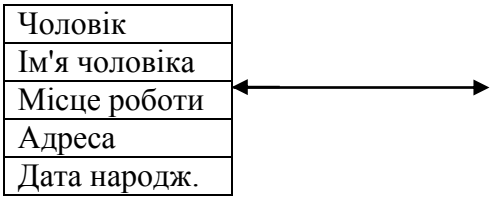
Між різноманітними предметами реального світу можуть існувати стосунки **— зв'язки.** Зв'язок — цеабстракціянаборустосунків, якісистематичновиникаютьміжрізнимипредметамиу реальномусвіті.

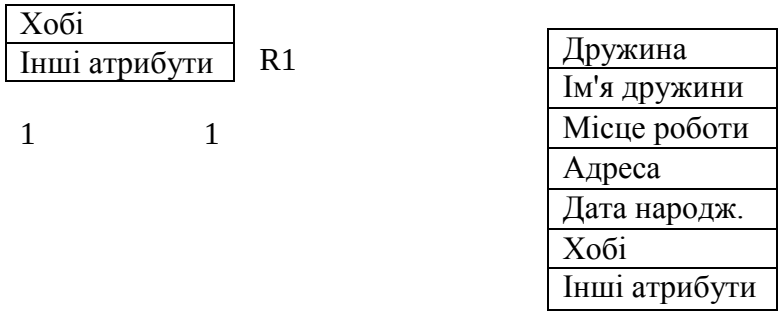
Реальні предмети, що беруть участь у стосунках, повинні бути самі абстраговані як об'єкти.

Зв'язок представляється графічно лінією між об'єктами, що беруть участь у відношенні. Зв'язок має ідентифікатор (далі К.1, К.2ДЗ і т. д.).

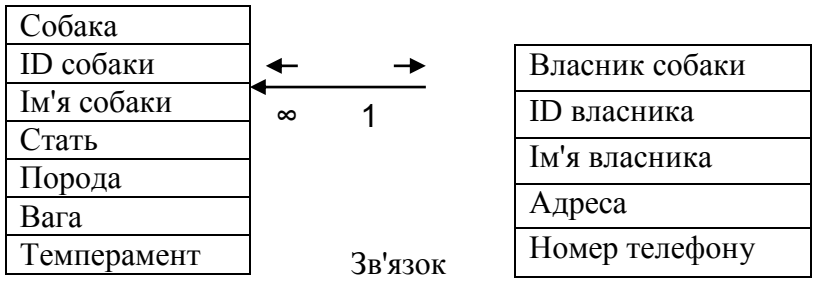
Існує три фундаментальні види зв'язку: **один-до-одного (1—1), один-до-багатьох(1—°°), багато-до-багатьох(°°— °°).**

Зв'язок **один-до-одного** існує, коли кожний екземпляр першого об'єкта пов'язаний з одним екземпляром іншого об'єкта і кожний екземпляр другого об'єкта пов'язаний з одним екземпляром першого.

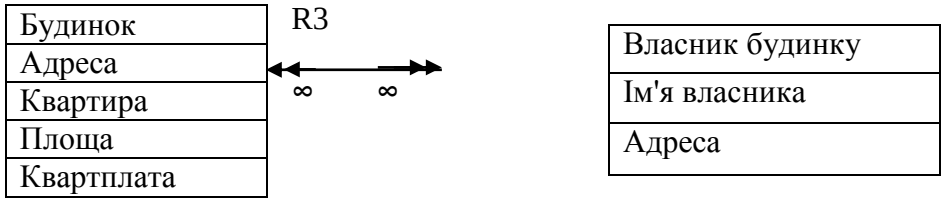




Зв'язок **один-до-багатьох** існує, коли кожний екземпляр першого об'єкта пов'язаний з багатьма екземплярами іншого і кожний екземпляр другого об'єкта пов'язаний тільки з одним екземпляром першого.



Зв'язок **багато-до-багатьох** існує, коли кожний екземпляр першого об'єкта пов'язаний з багатьма екземплярами другого і кожний екземпляр другого об'єкта пов'язаний з багатьма екземплярами першого.



Терміни "один-до-одного", "один-до-багатьох" і "багато-до-багатьох" є формулюваннями множинності зв'язку. Зауважимо, що множинність позначається графічно: одна стрілка на кінці означає один екземпляр, а подвійна стрілка — більше ніж один екземпляр.

Всі зв'язки поділяються на:

-умовні:

- безумовні.

Всі зв'язки, розглянуті вище, відносяться до безумовних.

В умовному зв'язку можуть існувати екземпляри об'єктів, що не беруть участі в зв'язку.

2. Завдання

1. Навести власний приклад, що демонструє між об'єктами зв 'язок "один-до-одного":

—умовний_____

- безумовний_____

2. Навести власний приклад, що демонструє між об'єктами зв'язок "один-до-багатьох":

—умовний;_____

безумовний'_____

3. Навести власний приклад, що демонструє між об'єктами зв'язок "багато-до-багатьох":

—умовний;_____

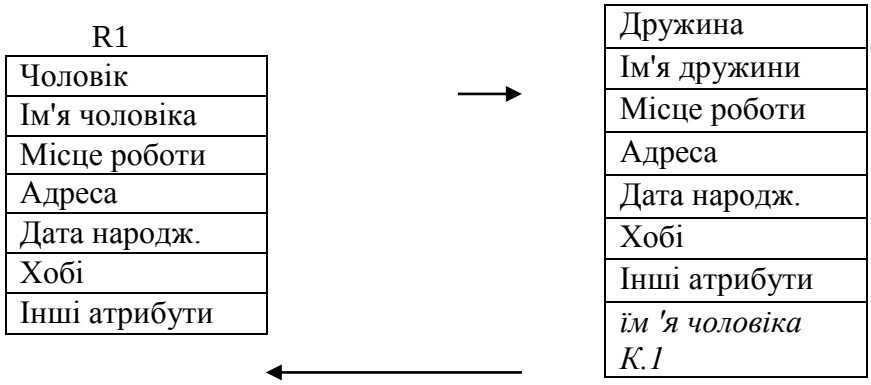
- безумовний_____

1.3. Формалізація зв'язків

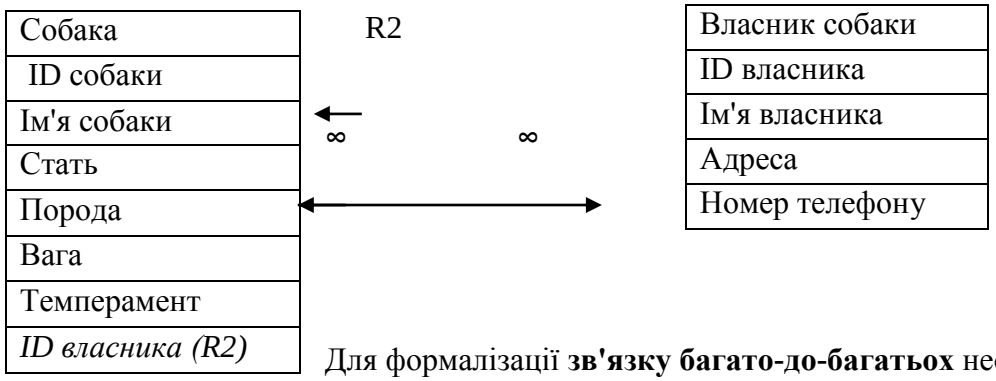
Мета зв'язку у тому, щоб дозволити нам встановити зв'язок екземпляра одного об'єкта з екземпляром іншого. Це виконується розміщенням допоміжних атрибутів у відповідних об'єктах. Коли це виконано, говорять, що зв'язок формалізований у даних.

Для формалізації зв'язку один-до-одного допоміжні атрибути можуть бути додані до будь-якого об'єкта (але не до обох). Допоміжні атрибути позначаються ярличком ідентифікатора зв'язку:

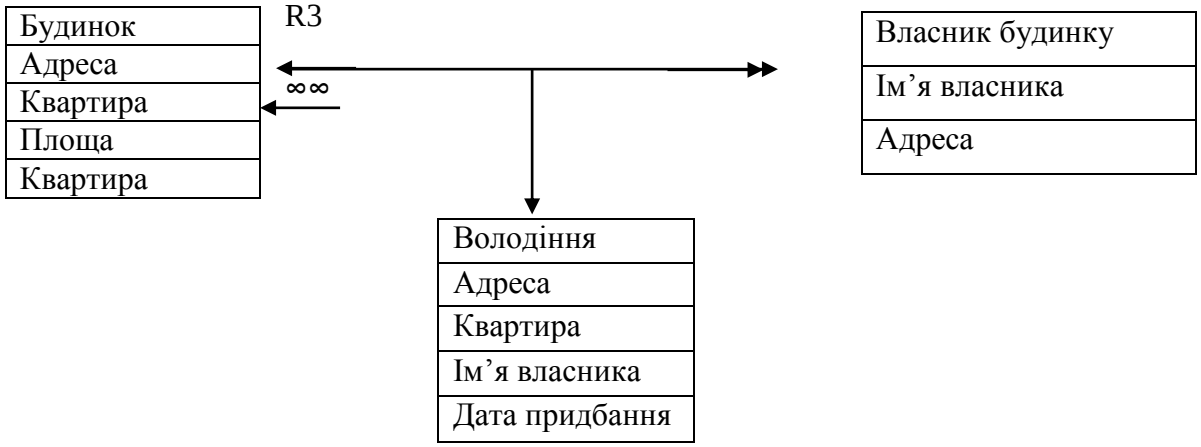
В умовному зв'язку можуть існувати екземпляри об'єктів, що не беруть участі в зв'язку.



Для формалізації зв'язку один-до-багатьох допоміжні атрибути повинні бути додані до об'єкта з боку "багатьох".



Для формалізації **зв'язку багато-до-багатьох** необхідно створити асоціативний об'єкт, що містить посилання на ідентифікатори кожного з об'єктів, що беруть участь у зв'язку.



Тоді асоціативний об'єкт **Володіння** оброблятиметься як окремий об'єкт. Подібно будь-якому іншому, асоціативний об'єкт може мати додаткові атрибути і брати участь у зв'язках з іншими об'єктами.

3. Завдання

Запишіть атрибути,які,на ваш погляд,можутьхарактеризувати наведені нижче об'єкти, визначіть тип зв'язку і формалізуйте його у даних.

1.4. Поняття "системи"

Поняття інформаційної системи логічно пов'язане з визначенням системи взагалі.

Система — це сукупність самостійних об'єктів (елементи системи), відповідним чином структурованих (структура системи), організованих (організація системи), що складають певну єдність (цілісність системи).

Елемент системи — самостійна частина системи, що має конкретну функціональність.

Структура системи — це сукупність елементів та внутрішніх зв'язків між елементами системи.

Організація системи — це:

- впорядкованість (внутрішня) елементів системи;
- узгодженість взаємодій елементів системи; обмеженість станів (життєвого циклу) елементів системи.

Цілісність системи — це залежність властивостей кожного елемента від його місця в структурі та організації системи. Тобто властивості кожного елемента визначаються його місцем в системі, але за властивостями кожного елемента не можна передбачити властивості системи в цілому.

Усі системи можна поділити на:

- абстрактні — продукт людського мислення (теорії, гіпотези);
- матеріальні - існуюча реальність.

Матеріальні системи можна поділити на: неорганічні (технічні, хімічні);

- органічні (біологічні);
- змішані.

Серед змішаних систем особливе місце займає підклас:

- людино-машинних систем (ерготехнічних) — систем, в які входять людина (ерготичний елемент) та машина (технічний елемент). У таких системах людина за допомогою машин здійснює певну діяльність;
- соціально-економічних — систем, що пов'язані із суспільними відносинами людей у процесі виробництва.

Залежно від поводження з плином часу системи можна поділити на:

- статичні — не змінюють свій стан;
- динамічні —змінюють свій стан.

У свою чергу динамічні поділяються на:

- детерміновані — стан системи в будь-який момент часу можна визначити;

- стохастичні — конкретне передбачення стану системи неможливе.

Системи в сучасних умовах у більшості випадків є важкоспостережуваними та важкозрозумілими.

Наявність великої кількості факторів зовнішнього середовища та самої системи, мінливість їх властивостей визначають невизначеність поводження системи в цілому і окремих її частин.

З такої складної ситуації є вихід — це використання сучасної методології, яка одержала назву **системний підхід.**

Системний підхід — це методологія дослідження важкозрозумілих об'єктів, з урахуванням (не ігноруванням) великої кількості факторів, їх мінливістю та існуючою невизначеністю поводження системи з плином часу.

Можна виділити два варіанти використання системного підходу:

аналіз системи—визначення функцій системи при відомих елементах та організації системи;

синтез системи — визначення елементів та організації системи за заданими функціями.

1.5. Визначення та класифікація інформаційних систем

Традиційно інформацією називають відомості, які передаються людьми усно, письмово або будь-яким іншим способом.

ІНФОРМАЦІЙНА СИСТЕМА (ІС) —це програмно-апаратний комплекс, який складається з системного та прикладного програмного забезпечення, а також периферійного та мережевого обладнання.

Інформаційна система призначена для збирання і накопичення інформації, а також її ефективного використання в будь-яких цілях.

Автоматизація ІС відбувається за допомогою обчислювальної техніки, а інформація представляється у вигляді даних, які зберігаються в пам'яті ЕОМ.

Умовно можна виділити три покоління ІС.

В ІС першого покоління (1963— 1972 рр.) для кожної задачі окремо готувалися дані, створювалася математична модель і розроблялось програмне забезпечення. Такий підхід зумовлював інформаційну та математичну надмірність (записані на машинний носій дані не могли використовуватися для розв'язування іншої задачі). Був позначений тривалістю і трудомісткістю і процес розробки програмного забезпечення кожної задачі.

Для **ІС другого покоління** (1972— 1986 рр.) було характерним розширення технічної і програмної бази. А головна відмінність ІС другого покоління від першого полягає в тому, що ці системи вже мали спільне інформаційне забезпечення усіх задач — базу даних. Організація єдиної бази даних стала можливою лише завдяки створенню спеціальних програмних продуктів - систем керування базами даних (СКБД), основне призначення яких — у створенні та підтримці баз даних в актуальному стані.

Сучасний етап розробки і функціонування інформаційних систем характеризується створенням **ІС нового покоління,** до яких належать експертні системи, системи підтримки прийняття рішень, системи зі штучним інтелектом. Технічною передумовою створення таких систем є широке використання в усіх сферах людської діяльності персональних комп'ютерів.

Залежно від мети функціонування розрізняють такі типи **ІС:**

- інформаційно-пошукові системи (орієнтовані на розв'язування задач пошукуінформації);
- інформаційно-довідкові системи (в яких за результатами пошуку обчислюютьзначення арифметичних функцій);
- інформаційно-управляючі (що забезпечують вироблення рішення на основіавтоматизації інформаційних процесів у сфері управління);
- системи підтримки прийняття рішень;
- інтелектуальні системи.

Наочним прикладом **інформаційно-пошукової системи** є довідкова служба міста, яка містить в собі відомості про мешканців. Користувачі такої системи мають змогу дізнатися

про номер телефону будь-якої людини, якщо вони знають її адресу, а якщо відомі прізвище і дата народження, то можуть довідатися про адресу людини і т. д.

Прикладом **інформаційно-довідкової системи** може бути інформаційна система будь-якого банку. Вона містить в собі відомості про вклади мешканців міста, а обробка банківської інформації передбачає поновлення сум вкладів, розрахунок відсотків, підведення підсумків за певний період роботи і т. д.

Інформаційно-управляючі, або автоматизовані, системи (АС) — це системи, які складаються з персоналу і комплексу засобів автоматизації його діяльності. Залежно від виду діяльності розрізняють такі різновиди АС: автоматизовані системи управління (АСУ), системи автоматизованого проектування (САПР), автоматизовані системи наукових досліджень (АСНД) та інші.

У високорозвинутих країнах **системи підтримки прийняття рішень** дуже поширені. Прикладом такої системи може бути система, що застосовується в центрах зайнятості для надання допомоги у виборі можливого місця роботи на підставі особистих уявлень клієнтів про бажаний характер майбутньої діяльності. Робота з системою розпочинається з короткого опису альтернатив, між якими проводиться вибір. Користувачу засобами звичної йому мови пропонується дати багатокритеріальну оцінку кожного зрозглянутих варіантів. Далі система перевіряє узгодженість інформації, поданої людиною, виявляє суперечності і визначає цінність інформації, що надходить. Після цього інформація вводиться до системи і на основі концепції багатокритеріальної теорії корисності видаються пріоритети користувача, що дає змогу ранжирувати об'єкти вибору.

Для довідки:
Альтернатива *— необхідність вибору між двома можливостями, щовиключають одна одну.*
Критерій *— мірило для визначення, оцінки предмета, явища.*
Концепція *— система поглядів на певне явище; спосіб розуміння, тлумачення якихось явищ.*
Пріоритет *— переважне право, значення чогось.*
Ранжирування *— послідовне розміщення чогось.*

Інтелектуальна діяльність — це дії та розумові висновки людей у нестандартних ситуаціях. Системами штучного інтелекту називають системи, що здатні виконувати операції, імітуючі інтелектуальні здібності людей. Найбільш поширеним видом **інтелектуальних систем** є експертна система. Експертна система - це комп'ютерна система, яка втілює в собі досвід експерта, що грунтується на його знаннях в певній галузі. Експертна система на основі обробки цих знань може давати інтелектуальні поради, приймати рішення на рівні експерта-професіонала, а також за бажанням користувача пояснювати хід розв'язування в разі відшукання того чи іншого рішення.

Але незалежно від типу інформаційної системи до її складу обов'язково входить **база даних.**

Базаданих (БД) — цепойменована, структурованасукупністьвзаємопов'язанихданих, якіхарактеризуютьокремупредметнуобласть.

Прагнення виділити загальну частину інформаційних систем, відповідальну за керування складноструктурованими даними, було основною причиною створення **систем керуваннябазами даних** (СКБД), без яких успішне ведення бізнесу і керування підприємствами сьогодні практично неможливе.

У більшості випадків, коли говорять про бази даних, мають на увазі деяке автоматизоване сховище інформації різних типів (числового, символьного, логічного та ін.). Але таке визначення не цілком коректне.

У вузькому значенні слова, БД - це деякий набір даних, необхідних для повсякденної роботи. Однак дані — це теж абстракція.

Нехай, наприклад, потрібно зберегти відомості про автомобілі, що надійшли на станцію технічного обслуговування. Яким чином автомобіль як об'єкт реального світу представити в БД? Для того щоб відповісти на це запитання, необхідно знати, які властивості або характеристики автомобіля використовуватимуться. Серед них можуть бути марка автомобіля, номерний знак, колір, дата випуску та ін.

1.6.Користувачі інформаційної системи

Користувачів інформаційної системи умовно можна розділити на дві групи: **внутрішні і кінцеві.** Внутрішні користувачі розробляють ІС і підтримують її функціонування, кінцеві - ті, задля яких і створюється інформаційна система.

Групу внутрішніх користувачів складають:

- адміністратор БД;
- системні програмісти;
- прикладні програмісти.

Функції **адміністратора БД** на стадії розробки та експлуатації ІС різні і тому виконуються різними особами.

На стадії проектування адміністратор БД виступає як ідеолог і конструктор системи, керує роботами зі створення програмного оточення БД.

На стадії експлуатації адміністратор БД — особа, відповідальна за функціонування ІС; він керує режимом використання даних. Основні задачі адміністратора БД при експлуатації — захист даних від руйнування, забезпечення достовірності даних, аналіз ефективності використання ресурсів ІС.

Системні програмісти виконують генерацію СКБД, стежать за її функціонуванням у середовищі операційної системи, розробляють за завданням адміністратора БД програмні компоненти, що розширюють програмне забезпечення СКБД.

Завдання**прикладних програмістів** -у розробці прикладних програм Для цього їм необхідне знання алгоритмічних і мовних засобів СКБД.

Кінцеві користувачі або спілкуються з ІС в інтерактивному режимі, або формулюють свої запити службі адміністратора БД.

4. Завдання

Вкажіть (за допомогою різнокольорових стрілок) відповідність між користувачем ІС і роллю, яку він виконує.

Адміністратор БД	виступає як ідеолог і конструктор системи
Системний програміст	керує роботами зі створення програмного оточення БД
Прикладний програміст	захищає дані від руйнування
Кінцевий користувач	розробляє прикладні програми за допомогою мовних засобів СКБД
	виконує генерацію СКБД
	забезпечує достовірність даних

Самостійна робота №2

Тема: Архітектура інформаційних систем.

Архітектура інформаційної системи

Отже, для кінцевих користувачів ІС — інформаційна система являє собою зібрання відомостей про жителів і рахунки, постачання і договори, документи і цінники.А внутрішнім користувачам та ж інформаційна система представляється у вигляді елементів даних, записів, сторінок і файлів.

Сприйняття інформаційної системи кінцевими і внутрішніми користувачами розрізняється не менше, ніж сприйняття телевізора його рядовим власником і фахівцем в області радіоелектроніки.

Іншими словами, *в сучасній інформаційній системі підтримується декілька рівнів подання даних.*

Їхній різновид прийнято асоціювати з поняттям **архітектури інформаційної системи.**

Перший рівень подання даних:

-зовнішній.

Другий рівень подання даних:

-концептуальний (або даталогічний).

Третій рівень подання даних:

-внутрішній.

Іноді для зручності вводять допоміжний рівень (проміжний), який називають **інфологічним.** Він може бути самостійним або функціонувати як складова зовнішнього рівня.

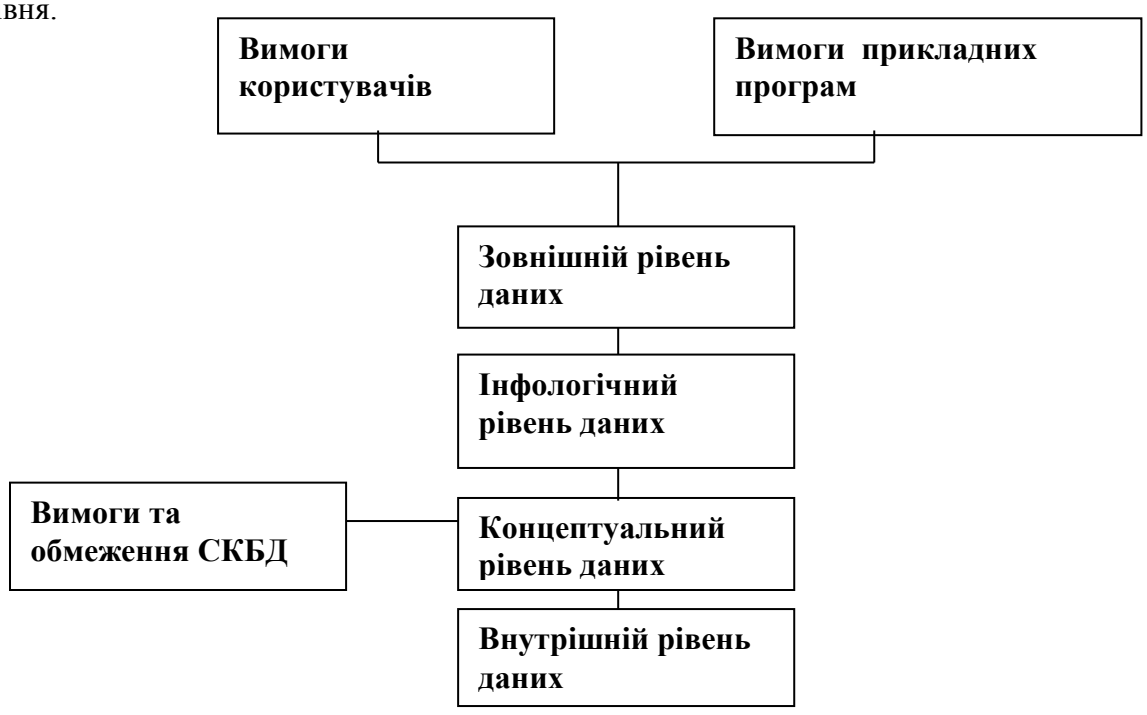


Рис. 1. Схема взаємозв'язку рівнів подання даних в БД

Зовнішній рівень відповідає погляду на предметну область кінцевих користувачів і відображає вимоги до даних з боку прикладних програм. Це, як правило, словесний опис даних та їх взаємозв'язків, який відбиває інформаційні потреби користувачів і прикладних програм.

Інфологічний рівень відповідає "погляду" на предметну область її адміністратора (директора, міністра, ректора вузу і т.д.) Він бачить всю множину інформаційних об'єктів, усі можливі асоціації між ними. Інфологічний рівень являє собою інформаційно-лопчну модель предметної області, в якій відображено інформаційні особливості об'єкта без урахування конкретної СКБД.

Сутність інфологічного моделювання полягає у виділенні інформаційних об'єктів, які підлягають зберіганню в БД, а також визначенні атрибутів об'єктів і зв'язків між ними.

Концептуальний (або даталогічний) рівень абстракції відповідає уявленню про логічну організацію даних адміністратора БД і формується з урахуванням специфіки і особливостей конкретної СКБД. Цей рівень абстракції дуже схожий з інфологічним, але його відмінність перебуває у прив'язці до засобів реалізації - СКБД. Опис БД на концептуальному рівні задається мовою опису даних використовуваної СКБД, у термінах і обмеженнях, прийнятих у цій системі.

Внутрішній рівень абстракції відповідає уявленню і збереженню даних в пам'яті ЕОМ. Параметри внутрішнього рівня уявлення БД впливають на ефективність роботи ІС, наприклад, на час реакції системи.

1. Завдання

Дайте відповіді на запитання:

1. Яким вимогам відповідає зовнішній рівень подання даних?
2. Чим відрізняється концептуальний рівень подання даних від інфологічного?
3. На що впливають параметри внутрішнього рівня подання даних?
4. На якому рівні подання даних відбувається визначення інформаційних об'єктів, їх атрибутів і зв'язків між ними?

Самостійна робота №3

Тема: Моделювання даних. Модель «об’єкт-атрибут-зв’язок».

Моделі даних

Опис об'єктів і зв'язків між ними спирається на **модель** даних.

Модель— цетакийматеріальночиобразноподанийоб'єкт, якийупроцесідослідженнязамінюєоб'єкт-оригінал івикористовуєтьсядлявивченняоб'єкта-оригіналу.

Модель як інструмент наукового пізнання має відтворити найхарактерніші ознаки досліджуваної системи. Відображатися можуть як самі об'єкти, так і зв'язки між ними. В інформаційних системах до інформації ставляться дві вимоги: упорядкованість та організованість. Відповідним засобом моделювання в комп'ютерних ІС є база даних — організована певним чином і підтримувана мовними та програмними засобами сукупність взаємопов'язаних даних, які зберігаються на машинних носіях системи і описують стан об'єкта управління.

В основу організації БД покладено модель даних. За її допомогою подаються множини даних і описуються взаємозв'язки між ними ("один-до-одного", "один-до-багатьох", "багато-до-багатьох" - ми вже розглядали особливості цих зв'язків).

У сучасних комп'ютерних ІС найчастіше застосовуються три типи моделей даних:

- ієрархічна;
- мережева;
- реляційна.

Ієрархічна модель

Ієрархічна модель будується на принципі субпідрядності між елементами даних і являє собою деревоподібну структуру, яка складається з вузлів (так званих сегментів) і дуг (гілок). Кожний вузол дерева - це *набір* логічно взаємопов'язаних елементів даних, які описують конкретні об'єкти предметної області.

Дерево в ієрархічній моделі даних упорядковане, тобто існують правила розміщення його вузлів і гілок. Правил вісім.

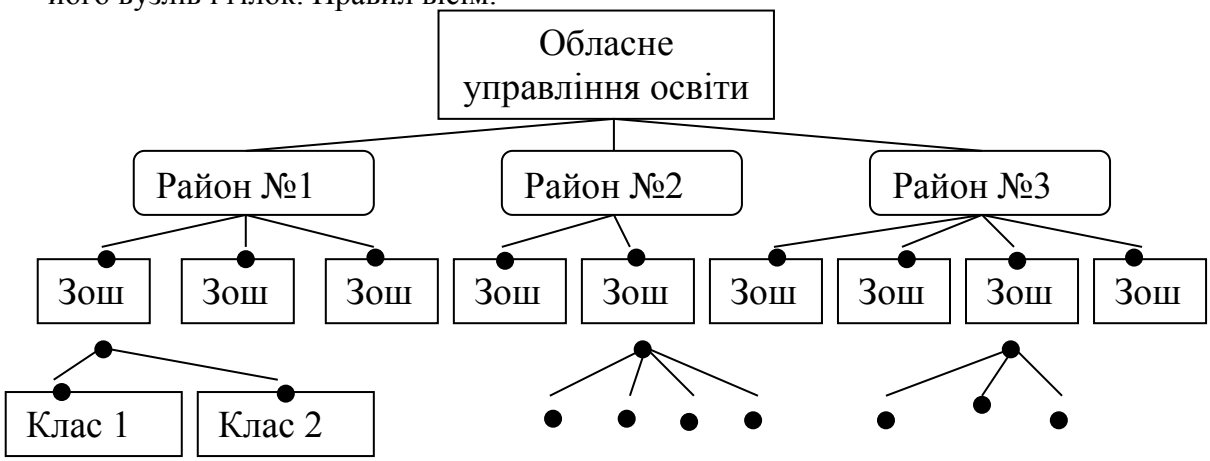


Рис. 2. Ієрархічна модель даних, яка інтерпретує структуру управління освіти

1. На найвищому рівні ієрархії міститься вузол, який називається корінний ("обласне управління освіти").
2. Взаємозв'язки в ієрархічній моделі даних будуються за принципом "корінний—породжений" ("батьківський—дочірній", або "-предок—нащадок"). зол другого рівня ієрархії залежить від першого і є породженим. У наведене му прикладі "управління освіти" первинне, а "район" - породжене.
3. Кожний первинний вузол може мати декілька породжених.
4. В ієрархічній моделі даних реалізовано два типи взаємозв'язків: "один-до одного" та "один-до-багатьох".

5. Доступ до кожного вузла відбувається через його первинний вузол.
6. Кожний примірник породженого вузла пов'язаний з вузлом первинного.
7. Примірник породженого вузла не може існувати за відсутності примірник первинного вузла.
8. При знищенні примірника первинного вузла знищуються також пов'язані з нимпримірники породжених вузлів.

Мережева модель

Мережева модель організації даних є розширенням ієрархічного дерева даних. В мережевій структурі даних нащадок може мати будь-яку кількість предків

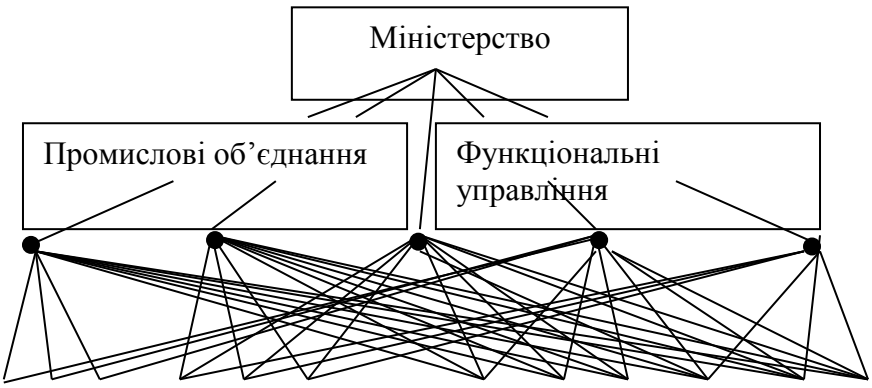


Рис. 3. Мережева модель структури даних

В мережевій моделі підтримуються всі три типи взаємозв'язків між даними: "один-до-одного", "один-до-багатьох", "багато-до-багатьох".

Реляційна модель

Реляційна модель даних подається набором двовимірних таблиць, які складаються із стовпців і рядків, а також має ім'я, унікальне в межах даної БД.

Ім’я таблиці

поле

Запчастини

Шрифт	Найменування	Кількість	Виробник
16-23	Шарова опора	11	ГАЗ
16-97	Стартер	45	ВАЗ
16-66	Генератор	56	Форд

——— Рядки (кортежі, записи)

Рис. 4. Основна перевага реляційної моделі даних — простота і наочність

Реляційну модель даних розробив Едгар Кодд у 1970 році.

Таблиця відображає тип об'єкта реального світу, а кожний рядок —конкретний екземпляр даного об'єкта.

Стовпець таблиці —це сукупність значень конкретного **атрибуту** об'єкта. Ці значення вибираються з множини всіх можливих значень атрибуту об'єкта, яка називається **доменом.**

Кожний стовпець має ім'я, яке записується у верхній частині таблиці і має бути у ній унікальним, але різні таблиці можуть мати стовпці з однаковими іменами. Будь-яка таблиця повинна мати хоча б один стовпець. Стовпці розташовані в таблиці відповідно до порядку наступності їх імен при створенні таблиці.

На відміну від стовпців, **рядки** імен не мають; порядок їх наступності в таблиці не визначений, а кількість логічно не обмежена. Оскільки рядки в таблиці не впорядковані, то неможливо вибрати рядок згідно з позицією -серед них не існує першої, другої або останньої.

Будь-яка таблиця має один або декілька стовпців, значення в яких однозначно ідентифікують кожний її рядок. Такий стовпець називається **первинним ключем. В** цьому стовпці значення не можуть дублюватися — в таблиці не повинно бути рядків, що мають одне й те ж значення для конкретного стовпця. Якщо таблиця відповідає цим вимогам, то вона називається **відношенням.**

Найважливішим елементом реляційної моделі даних є **взаємозв'язок таблиць,** який забезпечується зовнішніми ключами, відповідно до типу зв'язків, що встановлені між об'єктами.

1. Завдання

За темою ідивідуального завдання (додаток№1) виконати пункти а, б, в.

Самостійна робота №4
Тема:Теоретичні мови запитів.

Операції, які виконуються над відношеннями можна розділити на дві групи. Першу групу складають операції над множинами, до яких відносяться наступні операції: об’єднання, перетин, різниця, ділення і декартовий добуток. Другу групу складають спеціальні операції над відношеннями, до яких відносяться такі операції: проекція, з’єднання та вибору.

В різних СКБД реалізована деяка частина операцій над відношеннями, що визначає можливості даної СКБД, а також складність реалізації запитів до БД. В реляційних СКБД для виконання операцій над відношеннями використовують дві групи мов, які мають математичну основу та були запропоновані Едгаром Коддом: реляційна алгебра та реляційне числення. Ці мови представляють мінімальні можливості реальних мов маніпулювання даними, відповідно до реляційної моделі, і еквівалентні між собою за своїми можливостями.

1. Завдання

За темою ідивідуального завдання (додаток№1) виконати пункти 1, 2, 3, 4.

Самостійна робота №5

Тема:Додаткові операції реляційної алгебри запропоновані Дейтом.

Дейтом були запропоновані наступні додаткові операції реляційної алгебри: перейменування даних, розширення, підведення підсумків, присвоєння, вставки, оновлення даних, видалення даних, реляційного порівняння.

Операція перейменування даних.

RENAME<початкове відношення><старе ім'я атрибута>AS<нове ім'я атрибута>

Ця операція дає змогу перейменувати атрибути. Початкове відношення задається ім'ям відношення або виразом реляційної алгебри, який заключається в круглі дужки.

Операція множинного перейменування атрибутів.

RENAME<відношення><старе ім'я атрибута 1>AS<нове ім'я атрибута 1>, <старе ім'я атрибута 2>AS<нове ім'я атрибута 2>, …

Операція розширення.

EXTEND<початкове відношення>ADD<вираз>AS<новий атрибут>

Операція розширення породжує нове відношення схоже на початкове, яке відрізняється наявністю доданого атрибута, значення якого знаходять шляхом деяких скалярних обчислень. Початкове відношення може бути задано ім'ям відношення або за допомогою виразу реляційної алгебри, що заключається в круглі дужки. При цьому ім'я нового атрибута не повинно входити в заголовок початкового відношення і не може використовуватись у «виразі». Крім звичайних арифметичних операцій порівняння у виразі використовуються підсумкові функції, такі як: COUNT (кількість) SUM (сума) AVG (середнє арифметичне) MAX (максимальне значення) MIN (мінімальне значення).

Користуючись операцією розширення інколи виконують перейменування атрибутів. Для цього у виразі потрібно вказати ім'я атрибута, а в конструкції AS визначити нове ім'я цього атрибута, а потім виконати проекцію отриманого відношення на множину атрибутів, включаючи старий атрибут.

Операція множинного розширення.

EXTEND<відношення>ADD<вираз 1>AS<новий атрибут 1>, <вираз 2>AS<новий атрибут 2>, …

Ця операція є подібною до операції розширення, вона дозволяє в одній синтаксичній конструкції обчислювати кілька нових атрибутів.

Операція підведення підсумків.

SUMMARIZE<початкове відношення>ADD<вираз>AS<новий атрибут>

Дана операція виконує вертикальні або групові обчислення. Початкове відношення задається ім'ям відношення або виразом реляційної алгебри в круглих дужках. Результатом операції SUMMARIZE є відношення R з заголовком, що складається з атрибутів списку розширеного новим атрибутом. Для отримання тіла відношення R спочатку проводиться проектування початкового відношення на атрибути, а після цього кожний кортеж відношення розширюється новим N+1 атрибутом. Оскільки проектування приводить до скорочення кількості кортежів, бо видаляються однакові кортежі, то вважають, що відбувається своєрідне групування кортежів початкового відношення.

Значення N+1 атрибуту кожного кортежу відношення R формується шляхом обчислення виразу над відповідною цьому кортежу групою кортежів початкового відношення. Функція COUNT визначає кількість кортежів в кожній із груп початкового відношення.

Операція множинного підведення підсумків.

Операція множинного підведення підсумків подібна відповідним операціям перейменування і розширення. Вона виконує одночасно кілька вертикальних обчислень і записує результати в окремі нові атрибути.

Операція присвоєння.

<вираз-ціль> := <вираз-джерело>

Операція присвоєння являється основною операцією, що змінює тіло існуючого відношення, крім неї змінюють тіло існуючого відношення операції вставки, оновлення та

видалення. В операції присвоєння зліва вказане ім’я відношення, а праворуч – деякий вираз реляційної алгебри, при цьому ці вирази повинні бути сумісні за структурами і задавати сумісні по структурі відношення.

Виконання операції присвоєння зводиться до заміни попереднього значення на нове. За допомогою цієї операції можна здійснювати додавання та видалення кортежів.

Операція вставки.

INSERT<вираз-джерело>INTO<вираз-ціль>

Обидва вирази повинні бути сумісні по структурі. Виконання операції зводиться до обчислення виразу джерела і вставки отриманих кортежів в задане відношення вираз-ціль.

Операція оновлення.

UPDATE<вираз-ціль><список елементів>

Де список елементів є послідовністю розділених комами операцій присвоєння, що мають наступний вигляд:

<атрибут> := <скалярний вираз>

Результатом виконання цієї операції є відношення отримане після присвоєння відповідних значень атрибутам відношення, що задаються цільовим виразом.

Операція видалення.

DELETE<вираз-ціль>

Тут вираз-ціль є реляційним виразом, що описує кортежі, які видаляються.

Операція реляційного порівняння.

<вираз 1> 0 <вираз 2>

Тут обидва вирази задають сумісні за структурою відношення, а знак 0 – одну з наступних операцій: =, ≠, <, >.

1. Завдання

Відповісти на питання:

1. Які додаткові операції реляційної алгебри булизапропоновані Дейтом?
2. Яка операція дає змогу перейменувати атрибути?
3. Яка операція виконується: EXTEND <початкове відношення> ADD <вираз>AS<новий атрибут>?
4. Що відбувається під час операції підведення підсумків**?**
5. Яка операція є основною операцією, що змінює тіло існуючого відношення?
6. Яка виконується операція: INSERT<вираз-джерело>INTO<вираз-ціль>?
7. Як виконати операцію оновлення?
8. Яка виконується операція: DELETE<вираз-ціль>?

Самостійна робота №6

Тема:Мова реляційного числення за зразком QBE.

Припустимо, що ми працюємо з базою даних, яка має схемою СПІВРОБІТНИКИ (СОТР_НОМ, СОТР_ИМЯ, СОТР_ЗАРП, ОТД_НОМ) і ВІДДІЛИ (ОТД_НОМ, ОТД_КОЛ, ОТД_НАЧ), і хочемо дізнатися імена та номери співробітників, які є начальниками відділів з кількістю співробітників більше 50.

Якби для формулювання такого запиту використовувалася реляційна алгебра, то ми отримали б вираз, який читався б, наприклад, наступним чином:

• виконати з'єднання відносин СПІВРОБІТНИКИ і ВІДДІЛИ за умовою СОТР_НОМ = ОТД_НАЧ;

• обмежити отримане відношення за умовою ОТД_КОЛ> 50;

• спроектувати результат попередньої операції на атрибут СОТР_ИМЯ, СОТР_НОМ.

Ми чітко сформулювали послідовність кроків виконання запиту, кожен з яких відповідає одній реляційній операції. Якщо ж сформулювати той самий запит з використанням реляційного обчислення, якому присвячується цей розділ, то ми отримали б формулу, яку можна було б прочитати, наприклад, наступним чином: Видати СОТР_ИМЯ і СОТР_НОМ для співробітників таких, що існує відділ з таким же значенням ОТД_НАЧ і значенням ОТД_КОЛ великим 50.

У другому формулюванні ми вказали лише характеристики результуючого відношення, але нічого не сказали про спосіб його формування. В цьому випадку система повинна сама вирішити, які операції і в якому порядку потрібно виконати над відношеннями СПІВРОБІТНИКИ і ВІДДІЛИ. Зазвичай кажуть, що алгебраїчна формулювання є процедурної, тобто задає правила виконання запиту, а логічна - описової (або декларативною), оскільки вона всього лише описує властивості бажаного результату. Як ми вказували на початку лекції, насправді ці два механізми еквівалентні і існують не дуже складні правила перетворення одного формалізму в інший.

У численні доменів областю визначення змінних є не відносини, а домени. Стосовно до бази даних СПІВРОБІТНИКИ-ВІДДІЛИ можна говорити, наприклад, про доменні змінних ІМ'Я (значення - допустимі імена) або НОСОТР (значення - допустимі номери співробітників).

Основним формальним відзнакою обчислення доменів від обчислення кортежів є наявність додаткового набору предикатів, що дозволяють висловлювати так звані умови членства. Якщо R - це n-арное відношення з атрибутами a1, a2, ..., an, то умова членства має вигляд

R (ai1: vi1, ai2: vi2, ..., aim: vim) (m <= n),

де vij - це або літерально задається константа, або ім'я кортежних змінної. Умова членства приймає значення true в тому і тільки в тому випадку, якщо у відношенні R існує кортеж, що містить зазначені значення зазначених атрибутів. Якщо vij - константа, то на атрибут aij задається жорстку умову, що не залежить від поточних значень доменних змінних; якщо ж vij - ім'я доменної змінної, то умова членства може приймати різні значення при різних значеннях цієї змінної.

У всіх інших відносинах формули і вирази обчислення доменів виглядають схожими на формули і вирази обчислення кортежів. Зокрема, звичайно, розрізняються вільні і пов'язані входження доменних змінних.

Для прикладу сформулюємо з використанням обчислення доменів запит "Видати номери і імена співробітників, які не отримують мінімальну заробітну плату" (будемо вважати для простоти, що ми визначили доменні змінні, імена яких збігаються з іменами атрибутів відносини СПІВРОБІТНИКИ, а в разі, коли потрібно кілька доменних змінних, визначених на одному домені, ми будемо додавати в кінці імені цифри):

```
СОТР_НОМ, СОТР_ИМЯ WHERE EXISTS СОТР_ЗАРП1
(СПІВРОБІТНИКИ (СОТР_ЗАРП1) AND
СПІВРОБІТНИКИ (СОТР_НОМ, СОТР_ИМЯ, СОТР_ЗАРП) AND
СОТР_ЗАРП> СОТР_ЗАРП1)
```

Реляційне числення доменів є основою більшості мов запитів, заснованих на використанні форм. Зокрема, на цьому обчисленні базується відома мова Query-by-Example, яка була першою (і найбільш цікавою) мовою в сімействі мов, заснованих на табличних формах.

QBE (англ. Query by Example, запит за зразком) - спосіб створення запитів до бази даних з використанням зразків значень полів у вигляді текстового рядка. Реалізації QBE перетворять користувача введення в формальний запит до бази даних, що дозволяє користувачеві створювати складні запити без необхідності вивчати більш складні мови запитів, такі як SQL.

Даний метод відбору даних вперше запропонований Моше Злуфом (англ. Moshé M. Zloof), співробітником дослідного центру IBM в середині 1970-х років.

Експлуатаційною перевагою пошуку QBE є те, що для формування запиту не потрібно використовувати спеціалізовану мову запитів, синтаксис якої може бути складний і недоступний кінцевому користувачеві. Користувачеві виводиться вікно, в якому вказані всі поля даних, що зустрічаються в кожному запису даних; введення інформації в конкретне пошукове поле обмежить пошук збігом (повним або частковим, в залежності від домовленості реалізації) по даному полю. Перевірка умов здійснюється тільки по заповненим умовам на поля, а поля, умови на які вказані не будуть, можуть відповідати чому завгодно. Багатопрактичні реалізації QBE допускають також не тільки кон'юнктивне з'єднання умов в заповнених полях, а й інші варіанти з'єднання умов (наприклад, диз'юнкцію, заперечення, існування або неіснування пов'язаних записів та інші).

1. Завдання

За темою ідивідуального завдання (додаток№1) виконати пункти 5, 6, 7, 8, 9, 10, застосовуючи можливості QBE.

Самостійна робота №7

Тема: Побудова інтерфейсу. Звіти. Основні оператори мови SQL.

У SQL існує приблизно сорок інструкцій кожна з них «просить» СУБД виконати певну дію, наприклад, витягти дані, створити таблицю або додати в таблицю нові дані.

Інструкції SQL групуються по виконуваним діям:

обробка даних:

SELECT - отримує дані з таблиці;

INSERT - додає нові рядки в таблицю;

DELETE - видаляє рядки з таблиці;

UPDATE - оновлює дані існуючі в таблиці;

визначення даних:

CREATE TABLE - додає нову таблицю в БД;

DROP TABLE - видаляє таблицю;

ALTER TABLE - змінює структуру існуючої таблиці;

CREATE VIEW - додає нове подання до таблицю;

DROP VIEW - видаляє уявлення;

CREATE INDEX - створює індекс для стовпця;

DROP INDEX - видаляє індекс;

CREATE SCHEMA - додає нову схему в БД;

DROP SCHEMA - видаляє схему;

CREATE DOMAIN - додає новий домен в БД;

ALTER DOMAIN - змінює визначення домену;

DROP DOMAIN - видаляє домен;

управління доступом:

GRANT - представляє користувачеві певні привілеї доступу;

REVOKE - скасовує зазначені привілеї доступу;

управління транзакціями:

COMMIT - завершує поточну транзакцію;

ROLLBACK - скасовує поточну транзакцію;

SET TRANSACTION - визначає режим доступу до даних для поточної транзакції;

програмний SQL:

DECLARE - визначає набір записів в який будуть повернуті результати запиту;

OPEN - відкриває набір записів;

FETCH - витягує рядок з таблиці результатів запиту;

CLOSE - закриває набір записів;

PREPARE - готує інструкцію SQL до динамічного виконання;

EXECUTE - динамічно виконує інструкцію SQL;

DESCRIBE - повертає опис підготовленого запиту.

Всі інструкції SQL мають однакову структуру.

Кожна інструкція починається з команди, тобто з ключового слова, що описує дію, яку виконує інструкція. Після команди йде одна або кілька пропозицій. Пропозиція описує дані, з якими працює інструкція, або містить уточнюючу інформацію про дії, що виконуються інструкцією. Пропозиція може починатися і з ключового слова вказує на дію, яку потрібно зробити з даними. Одні пропозиції в інструкції є обов'язковими, а інші ні. Ключових слів в SQL приблизно 310.

Графічно допустимі форми інструкцій ілюструються за допомогою синтаксичних діаграм.

Імена в інструкціях SQL вказують, над яким об'єктом БД інструкція повинна виконати дію. Імена можуть містити від 1 до 128 символів, починатися з літери і не містять прогалини і спеціальних символів.

Звернення до таблиці може проводитися відповідно до імені або за схемою, в яку включена таблиця. У повному імені таблиці вказується ім'я користувача і ім'я таблиці (наприклад, Sam.orders). Звернення до таблиці, включеної в схему здійснюється із

зазначенням назви схеми і імені таблиці (наприклад, schema1.orders). Імена стовпців задаються із зазначенням імені таблиці (або без нього), імені користувача, якому належить таблиця (або без нього), назви схеми, в яку включена таблиця (або без нього). Наприклад, повне ім'я стовпця може виглядати наступним чином: orders.customer; schema1.orders.customer; Sam.orders.customer.

Мова SQL призначена для виконання запитів до БД. Найбільш потужною інструкцією мови є інструкція SELECT, за допомогою якої проводиться вибірка інформації з таблиць БД. наприклад,

SELECT city, target, sales

FROM offices

Вивести список офісів з їх плановими і фактичними обсягами продажів.

Синтаксична діаграма інструкції SELECT. Інструкція складається з шести пропозицій. Пропозиції SELECT і FROM є обов'язковими, а решта включаються в інструкцію при необхідності.

SELECT - вказується список стовпців, які повинні бути повернуті інструкцією.

FROM - вказується список таблиць, які містять елементи даних, які добуваються запитом.

WHERE - показує, що в результати запиту слід включати тільки деякі рядки (умови відбору).

GROUP BY - дозволяє створити підсумковий запит.

HAVING - показує, що в результати запиту слід включати тільки деякі з груп, створених за допомогою пропозиції GROUP BY.

ORDER BY - сортує результати запиту на підставі даних, що містяться в одному або декількох стовпцях.

Предикат Distinct дозволяє виключити повторювані рядки.

Результатом виконання SQL-запиту завжди є таблиця, яка містить дані і нічим не відрізняється від таблиці БД. Якщо користувач набирає інструкцію SQL в інтерактивному режимі, СУБД виводить результати на екран в табличній формі. Якщо програма надсилає запит СУБД за допомогою програмного SQL, то СУБД повертає таблицю результатів програмі.

1. Завдання:

Відповісти на питання:

1. За допомогою якої команди проводиться вибірка інформації з таблиць БД?
2. Які інструкції SQL призначені для обробки даних?
3. Які інструкції SQL призначені длявизначення даних?
4. Які інструкції SQL призначені дляуправління доступом?
5. З скількох пропозицій складається інструкція SELECT?
6. Що є результатом виконання SQL-запиту?

Самостійна робота №8

Тема:Історія мови SQL та огляд її можливостей.

На думку аналітиків CodingDojo, SQL - найважливіша і найпотрібніша мова запитів серед мов програмування.Рейтинг CodingDojo враховує статистику затребуваності мов програмування на ринку праці.

Адже СКБД - MySQL, PostgreSQL і MicrosoftSQLServer - поширені повсюдно: у великому і малому бізнесі, в лікарнях, банках, університетах і так далі. В принципі, SQL не обмежується тільки настільними девайсами: СУБД SQLite з успіхом зайняла своє місце на Android-смартфонах і мобільних пристроях компанії Apple. Відповідно, такі додатки, як Skype і Dropbox, постійно до неї звертаються.

Однак були часи, коли не було смартфонів, а ця мова вже існувала. Історія SQL - це не роки, але десятиліття. Повірили в нього не відразу.

Система R і IBM

Перші згадки про цю мову датуються 1974 роком. SQL створювалася в рамках проекту експериментальної реляційної СКБД SystemR. Займалася цим проектом компанія IBM.

Спочатку мова називалася SEQUEL (StructuredEnglishQueryLanguage), але потім слово «англійська» пропало з цього словосполучення, а абревіатура набула того вигляду, до якого ми давно вже звикли. З одного боку, SQL був орієнтований на зручне і зрозуміле користувачам формулювання запитів до реляційних БД. З іншого боку, практично з самого початку вона була так званою «повною мовою БД». Це означає, що SQL включала:

- засоби визначення та маніпулювання схемою БД;
- засоби визначення обмежень цілісності і тригерів;
- засоби визначення подань БД;
- засоби визначення структур фізичного рівня, що підтримують ефективне виконання запитів;
- засоби авторизації доступу до відносин і їхніх полів;
- засоби визначення точок збереження транзакції і виконання фіксації і відкатів транзакцій.

Правда, в ньому не були реалізовані засоби синхронізації доступу до об'єктів БД з боку паралельно виконуваних транзакцій. Справа в тому, що розробники спочатку розраховували, що необхідну синхронізацію неявно виконує СКБД.

Мова реалізована в переважній більшості СКБД - як в реляційних, так і нереляційних. Метою розробки було створення простої непроцедурної мови, якою міг скористатися будь-який користувач, який навіть не має навичок програмування.

Розробкою мови запитів займалися Дональд Чемберлін (Donald D. Chamberlin) і Рей Бойс (RayBoyce).

Шлях до комерційної реалізації SQL, який пройшла IBM, називають рухом «зверху вниз».

Oracle, Informix і Sybase пішли іншим шляхом - «від низу до верху»: в перших версіях цих систем, випущених на ринок, використовувалося істотно обмежена підмножина SQL System R. А далі вони почали поступово розширюватися. Однак у першій комерційній реалізації SQL в СУБД Oracle в операторах вибірки не допускалося використання вкладених підзапитів і була відсутня можливість формулювання запитів із з'єднаннями декількох відносин.

Зростаюча зацікавленість ринку в якнайшвидшому переході до реляційних систем управління базами даних дозволила розробникам перерахованих вище компаній домогтися комерційного успіху. Це сталося, скоріше, попри те, що СУБД були тоді дуже далекі від досконалості. Ну а тепер Oracle, Informix, Sybase і Microsoft SQL Server підтримують досить потужні діалекти SQL.

Поява численних діалектів SQL і їх розростання мало призвести до проблем сумісності та до інших суперечностей.

Однак діяльність по стандартизації мови SQL почалася дуже вчасно - практично одночасно з появою його перших комерційних реалізацій. У 1982 році комітету по

базамданих Американського національного інституту стандартів (ANSI) було доручено розробити специфікацію стандартної мови реляційних баз даних.

Після відхилення ряду невдалих версій стандарту в 1986 році експерти прийшли до єдиного знаменника. А в 1987 році стандарт SQL / 86 був схвалений Міжнародною організацією зі стандартизації (ISO).

За основу стандарту не можна було брати SQL System R. По-перше, цей варіант мови був недостатньо опрацьований технічно. По-друге, його занадто складно було б реалізувати. Тому за основу був узятий діалект мови SQL, що склався в IBM до початку 1980-х років. По суті, цей діалект являв собою підмножину SQL System R.

Стандарт SQL1

До 1989 року стандарт SQL / 86 був дещо розширено, після чого з'явився наступний стандарт, що одержав назву ANSI / ISOSQL / 89.

SQL / 89 став першим всесвітньо прийнятим стандартом мови SQL. У цієї мови є маса недоліків: багато важливих поняття не визначені, багато віддано на відкуп реалізацій. У цьому стандарті повністю відсутні такі важливі розділи, як маніпулювання схемою БД і динамічний SQL.

Але тим не менше він зіграв свою роль в становленні дійсно стандартизованих реляційних систем управління базами даних. Більш того, з появою стандарту SQL / 89 стало можливо проектувати, розробляти і супроводжувати інформаційні системи, не дуже прив'язані до конкретного виробника СКБД. У певному сенсі поява SQL / 89 стало просуванням технології баз даних в сторону відкритих систем.

Можливо, найбільш важливими досягненнями стандарту SQL / 89 є чітка стандартизація синтаксису, семантики операторів вибірки даних і маніпулювання даними, а також фіксація засобів обмеження цілісності БД.

У стандарті визначаються два рівні мови і окремо засіб підтримки цілісності. Рівень 2 - це повна мова баз даних SQL, що не включає засіб підтримки цілісності. Рівень 1 - це специфікована підмножина рівня 2.

Засіб фіксації результатів включає можливості визначення:

• необхідних обмежень на посилання між таблицями;

• перевірочних обмежень на рядки таблиці;

• значень стовпця за замовчуванням при занесенні рядка в таблицю.

Засоби визначення зовнішніх ключів дозволяють легко формулювати вимоги так званої посилальної цілісності БД. Це поширене в реляційних БД вимогу можна було сформулювати і на основі загального механізму обмежень цілісності SQL System R, але формулювання на основі поняття зовнішнього ключа більш проста і зрозуміла.

Стандарт SQL2 тайого додатки

Усвідомлюючи неповноту стандарту SQL, фахівці різних компаній почали роботу над черговим стандартом, який отримав назву SQL2. Ця робота також тривала кілька років, було випущено безліч проектів стандарту, поки нарешті в березні 1992 році не був прийнятий остаточний проект стандарту (SQL / 92). Цей стандарт істотно повніше стандарту SQL / 89 і охоплює практично всі аспекти, необхідні для реалізації програм: маніпулювання схемою БД, управління транзакціями (з'явилися точки збереження) і сесіями (сесія - це послідовність транзакцій, в межах якої зберігаються тимчасові відносини), підключення до БД , SQL динамічний. Нарешті, були стандартизовані відносини-каталоги БД, що взагалі-то не пов'язане безпосередньо з мовою, але дуже сильно впливає на реалізацію.

У 1995 році стандарт був доповнений специфікацією інтерфейсу рівня виклику (Call-Level Interface - SQL / CLI). SQL / CLI являє собою набір специфікацій інтерфейсів процедур, виклики яких дозволяють виконувати динамічно задаються оператори SQL. По суті справи, SQL / CLI являє собою альтернативу динамічному SQL.

Стандарт SQL / CLI послужив основою для створення повсюдно поширених сьогодні інтерфейсів ODBC (Open Database Connectivity) і JDBC (Java Database Connectivity).

У 1996 році до стандарту SQL / 92 було додано ще один компонент - SQL / PSM (Persistent Stored модулі). Основна мета цієї специфікації - стандартизувати способи визначення і використання збережених процедур, тобто спеціальним чином оформлених програм, що

включають оператори SQL, які зберігаються в базі даних, можуть викликатися додатками і виконуються всередині СКБД.

Oracle є однією з найбільш популярних СКБД. Більш того, саме там вперше була реалізована сумісність зі стандартом SQL / 92.

Oracle підтримує ряд різних платформ, включаючи Windows, Linux, MaxOSX і SunSolaris.

Процедурне розширення SQL, розроблене Oracle, називається PL / SQL (процедурна мова / StructuredQueryLanguage) і засноване на синтаксисі мов Ада і Паскаль. Третьою ключовою мовою, що використовується в СКБД Oracle нарівні з SQL і PL / SQL, є Java.

PL / SQL підтримує програмні блоки, а також різноманітні типи даних для зберігання чисел, рядків і дат, оператори управління потоком обчислень (в тому числі умовні переходи і цикли) і три типи контейнерів (колекцій) - масиви змінної довжини, асоціативні масиви і вкладені таблиці.

Стандарт SQL3

Спочатку планувалося закінчити роботу над новим стандартом в 1995 році. Реально роботу над новим стандартом вдалося частково завершити тільки в 1999 році, і тому стандарт отримав назву SQL: 1999.

Кожен новий варіант стандарту мови SQL був істотно об'ємніше попередніх версій. Так, якщо стандарт SQL / 89 займав близько 600 сторінок, то обсяг SQL / 92 становив на 300 з гаком сторінок більше.

Найперші проекти SQL3 займали близько 1500 сторінок.

Однак розробники SQL3 прийшли до висновку, що при таких обсягах стандарту ймовірність його прийняття і подальшої успішної підтримки помітно зменшується. Тому вони вирішили розбити стандарт на відносно незалежні частини, які можна було б розробляти і підтримувати окремо.

У 1999 році були прийняті п'ять частин стандарту SQL: 1999.

Перша частина (SQL / Framework) присвячена опису концептуальної структури стандарту. У цій частині наводиться розгорнута анотація наступних чотирьох частин і формулюються вимоги до реалізацій, що претендують на відповідність стандарту.

Друга частина SQL: 1999 (SQL / Foundation) утворює базис стандарту. Вводиться система типів мови, формулюються правила визначення функціональних залежностей і можливих ключів, визначаються синтаксис і семантика основних операторів SQL:

- операторів визначення і маніпулювання схемою бази даних;
- операторів маніпулювання даними;
- операторів управління транзакціями;
- операторів управління підключеннями до бази даних і т. д.

Третю частину займає уточнена в порівнянні з SQL / 92 специфікація SQL / CLI. У четвертій частині специфікується SQL / PSM - синтаксис і семантика мови визначення збережених процедур. Нарешті, в п'ятій частині - SQL / Bindings - визначаються правила зв'язування SQL для стандартних версій мов програмування.

У SQL стандарт: 1999 повинні були увійти ще кілька частин. Серед них специфікації наступних засобів:

- управління розподіленими транзакціями (SQL / Transaction);
- підтримка темпоральних властивостей даних (SQL / Temporal);
- управління зовнішніми даними (SQL / MED);
- зв'язування з об'єктно-орієнтованими мовами програмування (SQL / OLB);
- підтримка оперативної аналітичної обробки (SQL / OLAP).

SQL в XXI веке

В кінці 2003 року було прийнято та опубліковано і новий варіант міжнародного стандарту SQL : 2003. Багато фахівців вважали, що у варіанті стандарту, наступного за SQL: 1999, будуть всього лише виправлені неточності SQL: 1999. Але насправді, в SQL: 2003 специфікований ряд нових і важливих властивостей, з невеликими модифікаціями, внесеними пізніше в 2008 році.

Найбільш серйозні зміни мови SQL, специфіковані в частині 2 стандарту SQL: 2003, стосуються наступних аспектів:

• типи даних;

• підпрограми, що викликаються з SQL;

• розширені можливості оператора CREATE TABLE;

• новий об'єкт схеми - генератор послідовностей;

• нові види стовпців - що ідентифікують стовпці (identity column) і що генерують стовпці (generated column);

• новий оператор MERGE;

Зазнала деяких змін загальна організація стандарту. SQL Стандарт: 2003 складається з наступних частин:

• 9075-1, SQL / Framework;

• 9075-2, SQL / Foundation;

• 9075-3, SQL / CLI;

• 9075-4, SQL / PSM;

• 9075-9, SQL / MED;

• 9075-10, SQL / OLB;

• 9075-11, SQL / Schemata;

• 9075-13, SQL / JRT;

• 9075-14, SQL / XML.

Частини 1-4 і 9-10 з необхідними змінами залишилися такими ж, як і в SQL: 1999. Частина 5 (SQL / Bindings) перестала існувати; відповідні специфікації включені в частину 2.

Розділ 2 частини SQL: 1999, присвячений інформаційній схемі, виділений в окрему частину 11. З'явилися дві нові частини - 13 і 14.

Частина 13 повністю називається «SQL Routines and Types Using the Java Programming Language» («Використання підпрограм і типів SQL в мові програмування Java»). Поява такої частини стандарту виправдано підвищеною увагою до мови Java з боку провідних виробників SQL-орієнтованих СКБД.

Нарешті, остання частина SQL: 2003 присвячена специфікаціям мовних засобів, що дозволяють працювати з XML-документами в середовищі SQL.

Незважаючи на намагання розробників, процес стандартизації явно не встигає за змінами, що відбуваються.

Таблиця 1 - Основні моменти в історіїSQL

Рік	Назва	Інша назва	Зміни
1986	SQL-86	SQL-87	Перший варіант стандарта, прийнятий інститутом ANSIта одобрений ISO в 1987 р.
1989	SQL-89	FIPS 127-1	Трохи допрацьований варіант попереднього стандарта.
1992	SQL-92	SQL2, FIPS 127-2	Значні зміни (ISO 9075); рівень Entry Level стандарту SQL-92 ,боло прийнято як стандарт FIPS-127-2.
1999	SQL:1999	SQL3	Додано підтримку регулярних виразів, рекурсивних запитів, підтримку тригерів, базові процедурні розширення, не скалярні типи даних та деякі об’єктно-орієнтовані можливості.
2003	SQL:2003		Введені розширення для роботи с XML-даними, віконні функції, які застосовують для роботи з OLAPбазами даних, генератори послідовностей та основані на них типи даних.
2006	SQL:2006		Функціональність роботи з XMLданими значно розширена. Появилась возможно

			совместно использовать в за просах SQLiXQuery.
2008	SQL:2008		Покращені можливості віконних функцій, знищені деякі неоднозначності стандатра SQL:2003.

Тим не менш, можна стверджувати, що базовий набір операторів SQL, що включає операторивизначення схеми БД, выбірки та маніпулювання даними, авторизації доступа до даних, підтримки вбудови SQL в мови програмування та оператори динамічного SQL, в комерційних реалізаціях склавсятабільш-менш відповідає стандарту.

1. Завдання:

Відповісти на питання:

1. Якими роками датуються перші згадки про SQL?
2. Яка компанія займалася проектом СКБД SystemR?
3. Який стандарт став першим всесвітньо прийнятим стандартом мови SQL?
4. Скільки частин стандарту SQL: 1999 було прийнято у 1999 році?
5. Яких аспектів стосуються найбільш серйозні зміни мови SQL, специфіковані в частині 2 стандарту SQL: 2003?
6. Що нового у SQL:2008?

Самостійна робота №9

Тема:SQL-оператори групування, теоретико-множинні оператори

Команди SQL поділяються на такі групи:

- Командимовивизначенняданих - DDL (Data Definition Language).Ці SQL команди можна використовувати для створення, зміни та видалення різних об'єктів бази даних.
- Команди мови управління даними - DCL (Data Control Language). За допомогою цих SQL команд можна управляти доступом користувачів до бази даних і використовувати конкретні дані (таблиці, представлення і т.д.).
- Команди мови управління транзакціями - TCL (Transaction Control Language). Ці SQL команди дозволяють визначити результат транзакції.
- Команди мови маніпулювання даними - DML (Data Manipulation Language). Ці SQL команди дозволяють користувачеві переміщати дані в базу даних і з неї.

Саме завдяки цим якостям мова запитів SQL набула такої популярності і досить активно використовується на сучасних веб-ресурсах та СУБД.

Основні відомості про сервер БД MySQLServer та панель керування phpMyAdmin

MySQL — вільна система керування реляційними базами даних.

Ця СУБД з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і 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 — компактний багатонитковий сервер баз даних. Характеризується великою швидкістю, стійкістю і простотою використання.

MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних.

MySQL вважається гарним рішенням для малих і середніх застосувань. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитковості, що підвищує продуктивність системи в цілому.

Активне використання MySQL в веб-програмуванні зумовило його актуальність, а інтуїтивно зрозумілий інтерфейс в сукупності з широкою функціональністю і підтримкою більш 60 мов (в т.ч. і українською) забезпечило йому популярність серед веб-розробників.

Для некомерційного використання MySQL є безкоштовним.

Можливості сервера MySQL:

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

Доступ до БД може здійснюватися:

- Через скрипти сайту;
- За допомогою програми phpMyAadmin.

Для виконання завдання достатньо ознайомитись з панеллю керування phpMyAdmin.

Основні відомості про PHPMyAdmin

PhpMyAdmin - це програмне забезпечення, написане на PHP, яке забезпечує повноцінну, у тому числі віддалену, роботу з базами даних MySQL через браузер. Так як phpMyAdmin дозволяє у багатьох випадках обійтися без безпосереднього введення команд SQL, то робота з базами даних стає цілком посильною завданням навіть для людини досить поверхнево знайомої з MySQL.

Для виконання завдання рекомендується використовувати систему Denwer, яку можна завантажити з веб-сайту: http://www.denwer.ru

Детальне встановлення та налаштування цієї системи можна подивитись в навчальному відео на вищезазначеному сайті.

Про основні функціональні можливості та деталі використання PhpMyAdmin можна дізнатися в статті «PhpMyAdmin инструкция».

Треба зазначити, що основні дії та запити будуть виконуватись у вкладці SQL, але паралельно буде продемонстровано як виконувати ці ж дії за допомогою PhpMyAdmin, для того щоб наочно продемонструвати корисність цієї системи.

Тепер можемо приступити до безпосереднього проектування та створення БД у середовищі MySQL.

Проектування та створення БД у середовищі MySQL

CREATE DATABASE

Створення бази даних виконується за допомогою оператора **CREATE DATABASE**.
Синтаксис оператора **CREATE DATABASE**:

```
CREATE DATABASE [IF NOT EXISTS] db_name [CHARACTER SET charset] [COLLATE collation];
```

db_name - ім'я, яке буде присвоєно створюваній базі даних;
IF NOT EXISTS - якщо не вказати цей параметр, то при спробі створення бази даних з вже існуючим ім'ям, виникне помилка виконання команди;
CHARACTER SET, COLLATE - Використовується для завдання стандартного кодування таблиці і порядку сортування.

Якщо при створенні таблиці ці параметри не вказуються, то кодування і порядок сортування новостворюваної таблиці беруться зі значень, зазначених для всієї бази даних. Якщо заданий параметр CHARACTER SET, але не заданий параметр COLLATE, то використовується стандартний порядок сортування. Якщо заданий параметр COLLATE, але не заданий CHARACTER SET, то кодування визначає перша частина імені порядку сортування в COLLATE.

Кодування, задане в CHARACTER SET, повинне підтримуватися сервером (latin1 або sjis), а порядок сортування повинен бути допустимим для поточного кодування.

Приклади використання CREATE DATABASE:

Наступний приклад створює базу даних "my_db":

```
CREATE DATABASE `my_db`
```

або

```
CREATE DATABASE `my_db` CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Для того, щоб подивитися налаштування вже існуючої бази даних необхідно виконати оператор SHOW CREATE DATABASE:

```
mysql> SHOW CREATE DATABASE `test`;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| test     | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.02 sec)
```

При створенні нової бази даних в MySQL **слід дотримуватися деяких правил** щодо імені бази даних:

- Максимальна довжина імені не повинна перевищувати 64 символи;
- Дозволені будь-які символи, які допускаються в імені каталогу, за винятком / (слеш) і. (точка). Але слід взяти до уваги, що не можна використовувати символи ASCII (0) і ASCII (255).

Створення БД за допомогою PhpMyAdmin:

Для цього необхідно перейти на вкладку Databases і в формі Createnewdatabase вказати назву БД та її кодування.На рис. 1 та рис.2 зображено створення БД з назвою “my_db”:

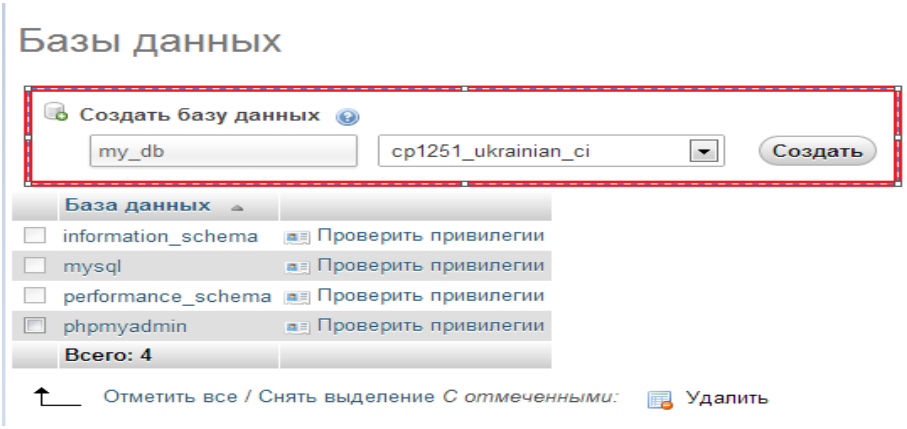


Рисунок 1 – створення нової БД

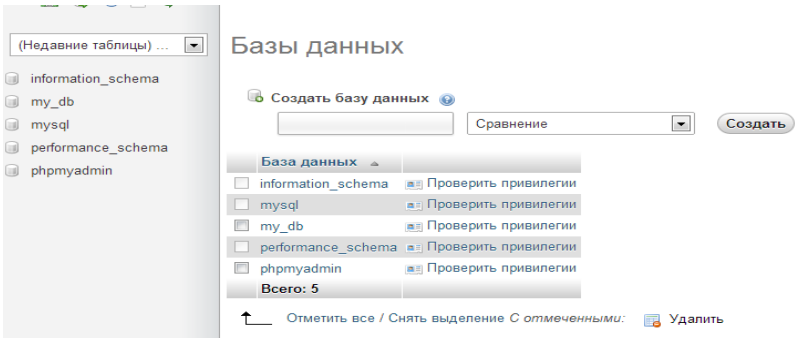


Рисунок 2 - успішне створення нової БД.

CREATE TABLE

Створення таблиці в базі даних проводиться командою **CREATE TABLE.**

Синтаксис:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name [(create_definition,...)]
  [table_options] [select_statement]
```

tbl_name - Визначаєім'ятаблиці, якабудестворенавпоточнійбазіданих. Якщо ніяка база даних на момент виклику команди CREATE TABLE не була прийнята поточною, то виникне помилка виконання команди. Починаючи з MySQL 3.22 введена можливість явно вказати базу даних, в якій буде створена нова таблиця, за допомогою синтаксису db_name.tbl_name.

TEMPORARY - Цей параметр використовується для створення тимчасової таблиці з ім'ям tbl_name протягом лише поточного сценарію. По закінченню виконання сценарію створена таблиця видаляється. Дана можливість з'явилася в MySQL 3.23. В MySQL 4.0.2 для створення тимчасових таблиць потрібні привілеї створення тимчасових таблиць.

IF NOT EXISTS - Якщо зазначений цей параметр і проводиться спроба створити таблицю з дублюючим ім'ям (тобто таблиця з таким ім'ям у поточній БД вже є), то таблиця створена не буде і повідомлення про помилку не з'явитися. В іншому випадку таблиця також створена не буде, але команда викличе помилку. Слід зазначити, що при створенні порівнюються тільки імена таблиць. Внутрішні структури не порівнюються.

create_definition - Визначає внутрішню структуру створюваної таблиці (назви і типи полів, ключі, індекси і т.д.)

Можливісинтаксиси**create_definition:**

```
col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT]
[PRIMARY KEY] [reference_definition]
```

PRIMARY KEY (index_col_name,...)

KEY [index_name] (index_col_name,...)

INDEX [index_name] (index_col_name,...)

UNIQUE [INDEX] [index_name] (index_col_name,...)

FULLTEXT [INDEX] [index_name] (index_col_name,...)

[CONSTRAINT symbol] FOREIGN KEY [index_name] (index_col_name,...)
[reference_definition]
CHECK (expr)

col_name - Визначає ім'я стовпця в створюваній таблиці;
type - Задає тип даних для стовпця ім'я_стовпця
Можливі значення параметра type:

- **TINYINT[(length)] [UNSIGNED] [ZEROFILL]**
  Дуже мале ціле число. Діапазон зі знаком від -128 до 127. Діапазон без знаку від 0 до 255.
- **SMALLINT[(length)] [UNSIGNED] [ZEROFILL]**
  Мале ціле число. Діапазон зі знаком від -32768 до 32767. Діапазон без знаку від 0 до 65535.
- **MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]**
  Ціле число середнього розміру. Діапазон зі знаком від -8388608 до 8388607. Діапазон без знаку від 0 до 16777215.
- **INT[(length)] [UNSIGNED] [ZEROFILL]**
  Ціле число нормального розміру. Діапазон зі знаком від -2147483648 до 2147483647. Діапазон без знаку від 0 до 4294967295.
- **INTEGER[(length)] [UNSIGNED] [ZEROFILL]**
  Синонім для INT
- **BIGINT[(length)] [UNSIGNED] [ZEROFILL]**
  Велике ціле число. Діапазон зі знаком від -9223372036854775808 до +9223372036854775807. Діапазон без знаку від 0 до 18446744073709551615.
- **DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]**
  Число з плаваючою крапкою подвоєної точності нормального розміру. Допустимі значення: від -1,7976931348623157 E +308 до -2,2250738585072014 E-308, 0, і від 2, 2250738585072014E-308 до +1,7976931348623157 E +308. Якщо вказаний атрибут UNSIGNED, негативні значення неприпустимі.
- **REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]**
  Синонім для DOUBLE
- **FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]**

Мале число з плаваючою крапкою звичайної точності. Допустимі значення: від -3,402823466 E +38 до -1,175494351 E-38, 0, і від 1,175494351 E-38 до 3,402823466 E +38.

- **DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]**
  «Неупаковане» число з плаваючою крапкою. Веде себе подібно колонці CHAR, яка містить цифрове значення. Термін «неупаковане» означає, що число зберігається у вигляді рядка і при цьому для кожного десяткового знака використовується один символ.
- **CHAR(length) [BINARY]**
  Рядок фіксованої довжини, при зберіганні завжди доповнюється пробілами в кінці рядка до заданого розміру. Діапазон аргументу length становить від 0 до 255 символів (від 1 до 255 у версіях, що передують MySQL 3.23). Кінцеві прогалини видаляються при виведенні значення. Якщо не заданий атрибут чутливості до регістру BINARY, то величини CHAR сортуються і порівнюються як незалежні від регістра відповідно до встановленого за замовчуванням алфавіту.
- **VARCHAR(length) [BINARY]**
  Рядок змінної довжини
- **DATE**
  Дата. Підтримується інтервал від '1000-01-01 'до '9999-12-31'. MySQL виводить значення DATE у форматі 'YYYY-MM-DD', але можна встановити значення в стовпець DATE, використовуючи як рядки, так і числа.
- **TIME**
  Час. Інтервал від '-838:59:59' до '838: 59:59 '. MySQL виводить значення TIME у форматі 'HH: MM: SS', але можна встановлювати значення в стовпці TIME, використовуючи як рядки, так і числа.
- **TIMESTAMP**
  Часова мітка. Інтервал від '1970-01-01 00:00:00 'до деякого значення часу в 2037. MySQL виводить значення TIMESTAMP у форматах YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD або YYMMDD.
- **DATETIME**
  Комбінація дати і часу. Підтримується інтервал від '1000-01-01 00:00:00 'до '9999-12-31 23:59:59'. MySQL виводить значення DATETIME у форматі 'YYYY-MM-DD HH: MM: SS', але можна встановлювати значення в стовпці DATETIME, використовуючи як рядки, так і числа.
- **TINYBLOB**
  Стовпець типу BLOB з максимальною довжиною 255 (2 ^ 8 - 1) символів.
- **BLOB**
  Стовпець типу BLOB з максимальною довжиною 65535 (2^16 - 1) символів.
- **MEDIUMBLOB**
  Стовпець типу BLOB з максимальною довжиною 16777215 (2^24 - 1) символів.
- **LONGBLOB**
  Стовпець типу BLOB з максимальною довжиною 4294967295 (2 ^ 32 - 1) символів. Слід враховувати, що в даний час протокол передачі даних сервер / клієнт і таблиці MyISAM мають обмеження 16 Мб на переданий пакет / рядок таблиці, тому поки не можна використовувати цей тип даних у його повному діапазоні.
- **TINYTEXT**
  Стовпець типу TEXT з максимальною довжиною 255 (2 ^ 8 - 1) символів.
- **TEXT**
  Стовпець типу TEXT з максимальною довжиною 65535 (2^16 - 1) символів.
- **MEDIUMTEXT**
  Стовпець типу TEXT з максимальною довжиною 16777215 (2^24 - 1) символів.
- **LONGTEXT**
  Стовпець типу TEXT з максимальною довжиною 4294967295 (2 ^ 32 - 1) символів. Слід враховувати, що в даний час протокол передачі даних сервер / клієнт і таблиці

MyISAM мають обмеження 16 Мб на переданий пакет / рядок таблиці, тому поки не можна використовувати цей тип даних у його повному діапазоні.

- **ENUM(value1,value2,value3,...)**
  Перерахування. Перераховується тип даних. Об'єкт рядка може мати тільки одне значення, вибране із заданого списку величин 'value 1', 'value 2', ..., NULL або спеціальна величина помилки. Список ENUM може містити максимум 65535 різних величин.
- **SET**(value1,value2,value3,...)
  Набір. Об'єкт рядка може мати нуль або більше значень, кожне з яких має бути вибрано із заданого списку величин 'value 1', 'value 2', ... Список SETможет містити максимум 64 елементи.

[NOT NULL | NULL] - вказує, чи може даний стовпець містити значення NULL чи ні. Якщо не вказано, то за замовчуванням приймається NULL (тобто може міститиNULL);
[DEFAULT default_value] - Задає значення за замовчуванням для даного стовпця. При вставці нового запису в таблицю командою INSERT якщо значення для поля col_name явно вказано не було, то встановлюється значення default_value;
[AUTO_INCREMENT] - При вставці нового запису в таблицю поле з цим атрибутом автоматично отримає числове значення, більше самого великого значення для цього поля в поточний момент часу на 1. Дана можливість зазвичай використовується для генерування унікальних ідентифікаторів рядків. Стовпець, для якого застосовується атрибут AUTO_INCREMENT, повинен мати цілочисельний тип. У таблиці може бути тільки один стовпчик з атрибутом AUTO_INCREMENT. Так само цей стовпець повинен бути проіндексований. Відлік послідовності чисел для AUTO_INCREMENT починається з 1. Це можуть бути лише додатні числа.

Наступний приклад створює таблицю користувачів з 3 полями, де перше поле - унікальний ідентифікатор запису, друге поле - ім'я користувача, а третє поле - його вік:

```
`users` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` CHAR(30) NOT NULL,
  `age` SMALLINT(6) NOT NULL,
  PRIMARY KEY(`id`))
```

На рисунку 5 можна побачити результат виконання запиту, який свідчить про те, що таблиця успішно створена. Нижче вікна запитів можна побачити створені поля, які вже можна редагувати.
Отже, в даній роботі було детально розглянуто базові поняття в проектування БД засобами MySQL та мови запитів SQL.

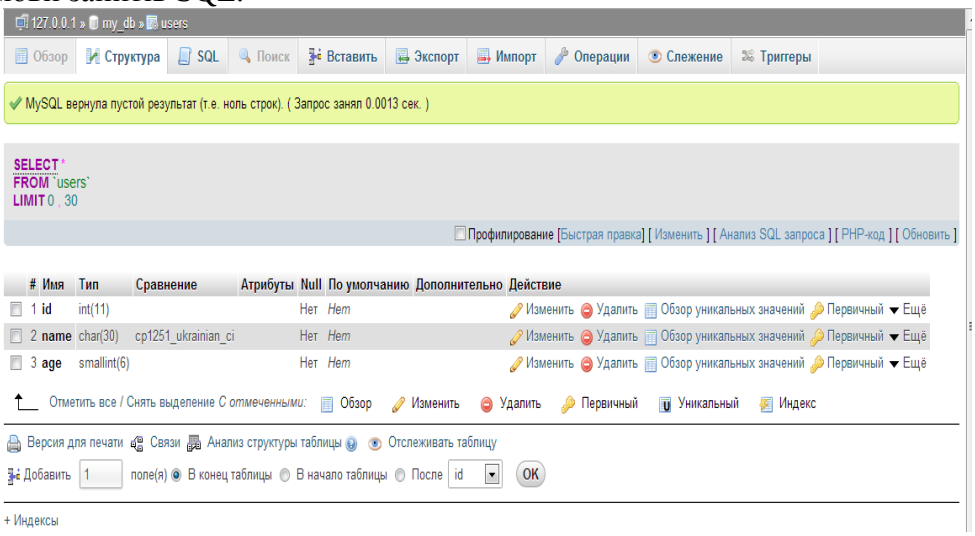


Рисунок 5 - результат успішного створення таблиці в БД

1. Завдання:

Відповісти на питання:

1. На які групи розділяють команди SQL?

2. Що таке MySQL?
3. Для чого призначена програма PhpMyAdmin?
4. Як створити БД засобами SQL?
5. Як створити БД засобами PhpMyAdmin?
6. Як додати до БД нову таблицю за допомогою засобів SQL?
7. Як додати до БД нову таблицю за допомогою засобів PhpMyAdmin?
8. За що відповідає параметр AUTO_INCREMENT?
9. Який параметр відповідає за тип кодування БД та таблиць?
10. Який параметр відповідає за сортування БД та таблиць?

2. Завдання:

На основі теоретичних знань створити БД з таблицями відповідно індивідуального завдання(додаток 1).

Самостійна робота №10
Тема:Додаткові можливості мови SQL. Віртуальні таблиці.

Для видалення рядків існуючої таблиці необхідно їх додати в таблицю. Для додавання записів використовується оператор INSERT:

```
INSERTINTO Ім’я_таблиці [(Список полів)]
VALUES (Список констант);
```

Після виконання оператора **INSERT** буде створено новий запис, як значення полів будуть використані відповідні константи, зазначені в списку **VALUES**.
Нехайіснуєтаблиця:

```
CREATE TABLE CLIENTS
(
        C_NO    int      NOT NULL,
        FIO     char(40)  NOT NULL,
        ADDR    char(30)  NOT NULL,
        CITY    char(15)  NOT NULL,
        PHONE   char(11)  NOT NULL
);
```

Тепер необхідно до неї додати дані. Додати дані можна за допомогою оператора INSERT. Розглянемо приклад використання оператора INSERT:

```
INSERT INTO CLIENTS VALUES (1, 'Іванов І. І.', 'вул. Велика Перспективна 3', 'Кіровоград',
'09599911100');
```

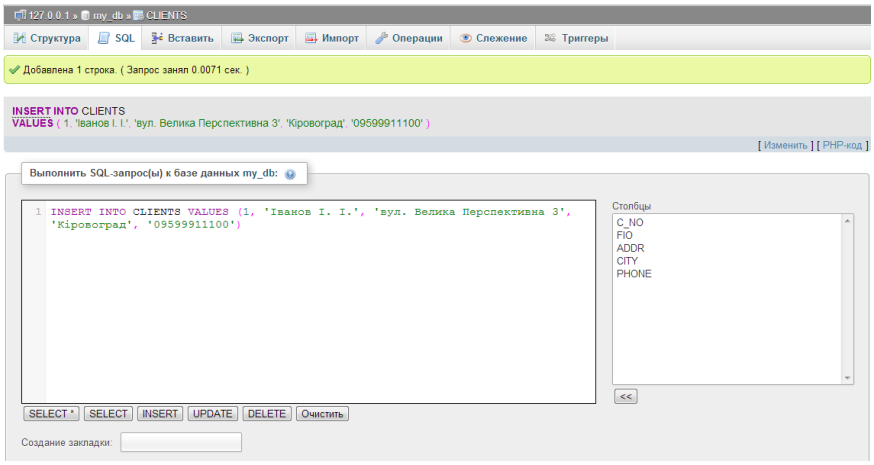


Рисунок 1 - Результат виконання.

Додаватися значення повинні в тому порядку, в якому поля перераховані в операторі CREATE. Якщо необхідно додавати інформацію в іншому порядку, то потрібно вказати цей порядок в операторі INSERT, наприклад:

```
INSERT INTO CLIENTS VALUES (FIO, C_NO, ADDR, PHONE, CITY) ('Іванов І. І.', 1, 'вул.
Велика Перспективна 3', '09599911100', 'Кіровоград');
```

За допомогою INSERT можливо додавати дані і в певні поля, наприклад, C_NO та FIO:

```
INSERT INTO CLIENTS (C_NO, FIO) VALUES (1, 'Іванов');
```

Проте, в даному випадку сервер MySQL не виконає такий запит, оскільки всі інші поля дорівнюють NULL (порожнє значення), а таблиця не приймає порожні значення (NOTNULL).

Оновлення записів. Оператор UPDATE

Синтаксис оператора **UPDATE**, який використовується для оновлення записів, виглядає так:

UPDATE Ім’я_таблиці SET Поле1 = Значення1, … , ПолеN = ЗначенняN [WHERE Умова];

Якщо не задана умова WHERE, буде модифікована вся таблиця, а це може спричинити непередбачувані наслідки, оскільки для всіх записів будуть встановлені однакові значення полів, тому завжди бажано вказувати умову WHERE.

Припустимо, необхідно оновити запис, якщо, наприклад, клієнт Іванов переїхав в інше місто і потрібно відзначити цю подію в базі даних. В такому випадку потрібно виконати наступний запит:

UPDATECLIENTSSETCITY = 'Київ'WHEREC_NO = 1;

Результат виконання зображено на рисунку 2.

Даний запит потрібно розуміти так: знайти запис, поле C_NO якого дорівнює 1 (це код клієнта Іванова), і встановити значення CITY рівним "Київ".

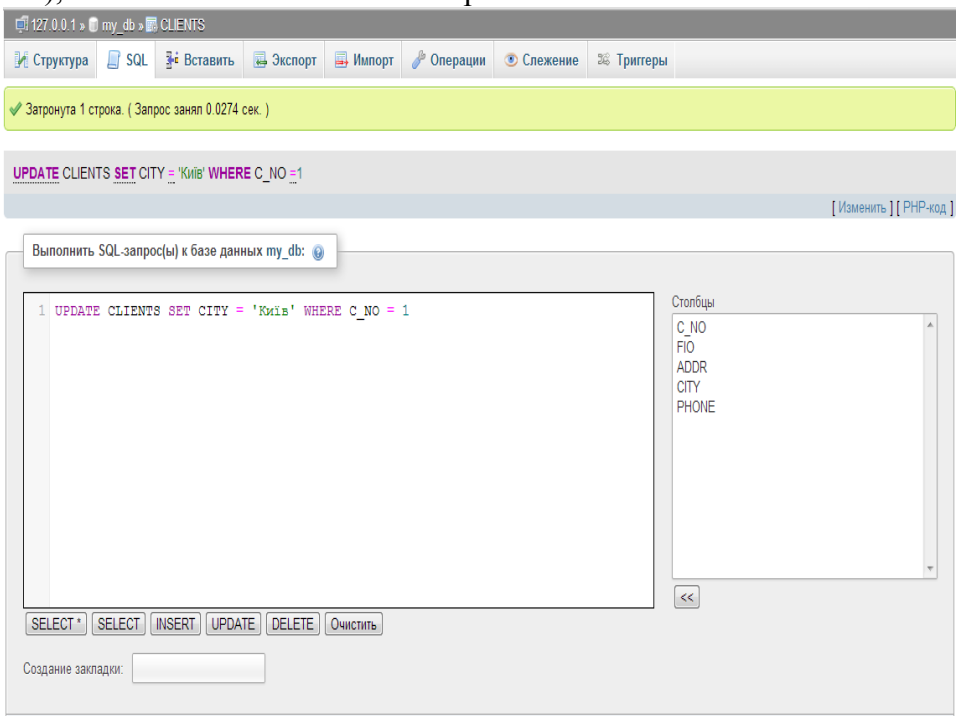


Рисунок 2 – виконання оператору UPDATE

Видалення записів. Оператор DELETE

За допомогою оператора **DELETE** можна видалити всі записи таблиці, вказавши умову, яка підійде для всіх записів, наприклад:

DELETEFROMCLIENTS;

Якщо необхідно видалити всіх клієнтів, номери яких перевищують 5, то потрібно застосувати наступний запит:

DELETE FROM CLIENTS WHERE C_NO > 5;

Результат виконання зображено на рисунку 3.

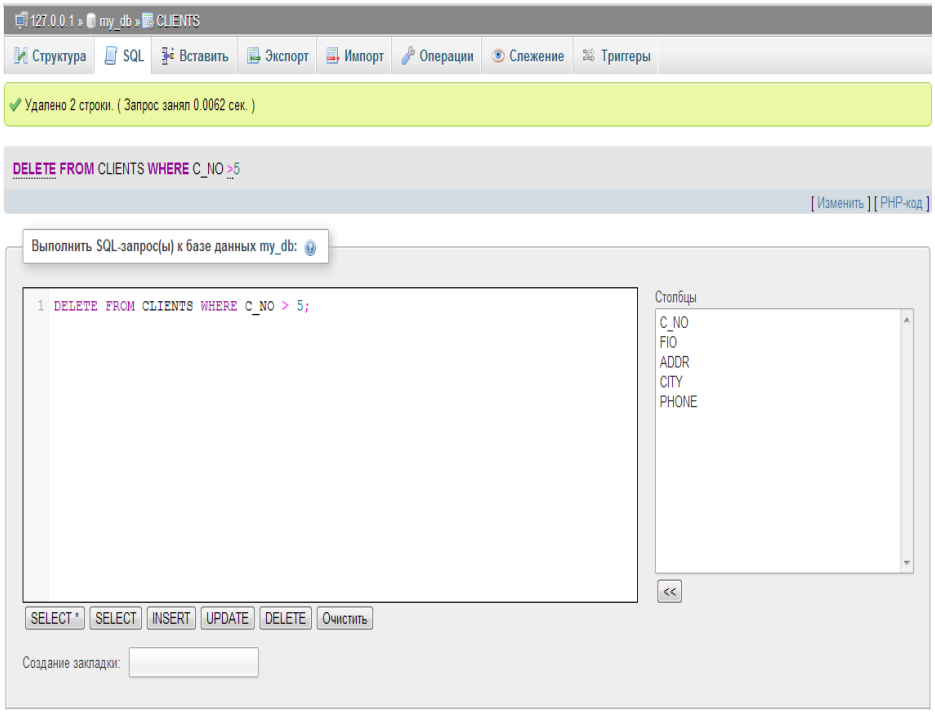


Рисунок 3 – виконання оператору DELETE

Якщо друга частина оператора DELETE-WHERE не вказана, це означає, що дія оператора поширюється на всі записи одразу.

Видалення полів та таблиць. Оператор DROP

Стандартом SQL не передбачено видалення стовпців, проте в MySQL є можливість це зробити:

ALTER TABLE CLIENTS DROP ADDR;

Результат виконання зображено на рисунку 4: в полі «Стодбцы», де містяться всі стовпці таблиці, відсутній стовпець ADDR.

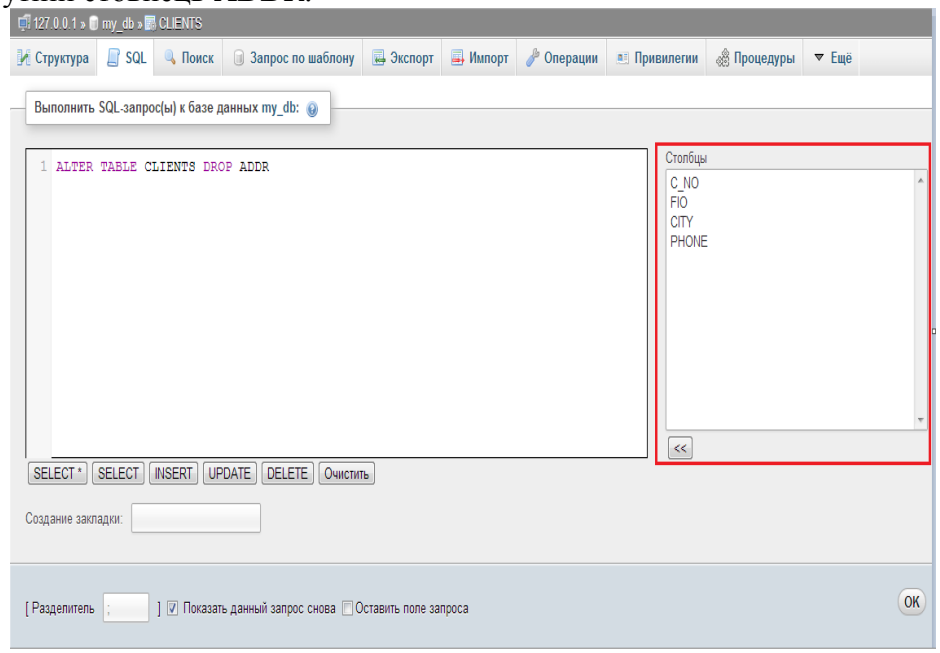


Рисунок 4 – виконання оператору DROP

А видалити таблицю ще простіше:

DROP ORDERS;

1. Завдання:

Відповісти на питання:

1. Для чого використовується оператор INSERT.
2. В яких комбінаціях можна використовувати оператор INSERT? Наведіть приклади.

3. Як змінити порядок додавання записів до таблиці, якщо це необхідно?
4. Оператор UPDATE. Його призначення.
5. Як видалити запис з таблиці?
6. Чи можна під час видалення перевіряти деяку умову? Як саме?
7. Як видалити таблицю із БД?

2. Завдання:

1. Вставити по 30 записів до кожної таблиці БД.

2.Видалити в кожній таблиці кожен 5 запис.

3.В вашій таблиці виконати (взаємозаміну) Наприклад із співробітниками продемонструвати ситуацію, коли перші 3 співробітники переїхали за адресами останніх 3-х і навпаки.

4.Створити нову пусту таблицю з назвою BUFFER та скопіювати до неї з кожної таблиці по 5 перших записів.

5.За допомогою оператора UPDATE оновити поле 2 (адреси) кожного 10-го запису (співробітника) та встановити, наприклад, ім’я кожного 7-го співробітника в значення «Default».

Самостійна робота №11

Тема: Проектування і використання баз даних.

Етапи проектування та побудови бази даних.

Перед створенням бази даних необхідно володіти описом вибраної предметної області, що повинно охоплювати реальні об'єкти і процеси, мати всю необхідну інформацію для задоволення передбачуваних запитів користувача і визначати потреби в обробці даних. На основі такого опису на етапі проектування бази даних здійснюється визначення складу і структури даних предметної області, що повинні знаходитися в базі даних і забезпечувати виконання необхідних запитів і задач користувача. Структура даних предметної області може відображатися інформаційно-логічною моделлю. На основі цієї моделі легко створюється реляційна база даних.

При розробці моделі даних можуть використовуватися два підходи. В першому підході спочатку визначаються основні задачі, для рішення яких будується база, і виявляються потреби задач в даних. При другому підході відразу встановлюються типові об'єкти предметної області. Найбільш раціонально сполучення обох підходів. Це зв'язане з тим, що на початковому етапі, як правило, немає вичерпних відомостей про всі задачі. Використання такої технології тим більш виправдане, що гнучкі засоби створення реляційної бази даних в Access дозволяють на будь-якому етапі розробки внести зміни в базу даних і модифікувати її структуру без збитків для введених раніше даних.

В процесі розробки моделі даних необхідно виділити інформаційні об'єкти, відповідні вимогам нормалізації даних і визначити зв'язки між ними. Ця модель дозволяє створити реляційну базу даних без дублювання, в якій забезпечується одноразове введення даних при початковому завантаженні і корегуванні. В такій базі даних Access забезпечує цілісність даних при внесенні змін. При визначенні логічної структури реляційної бази даних на основі моделі кожний інформаційний об'єкт адекватно відображається реляційною таблицею, а зв'язки між таблицями відповідають зв'язкам між інформаційними об'єктами.

В процесі створення бази даних спочатку здійснюється конструювання таблиць, відповідних інформаційним об'єктам побудованої моделі даних. Далі може створюватися схема даних, в якій фіксуються існуючі логічні зв'язки між таблицями. Ці зв'язки відповідають зв'язкам інформаційних об'єктів. В схемі даних можуть бути задані параметри забезпечення цілісності бази даних**,** якщо модель даних була розроблена в відповідності з вимогами нормалізації. Цілісність даних означає, що в базі даних встановлені і коректно підтримуються взаємозв'язки між записами різних таблиць при завантаженні, додаванні і вилученні записів в зв'язаних таблицях, а також при зміні значень ключових полів.

Після формування схеми даних здійснюється введення даних, що містяться в документах предметної області.

Етапи проектування і створення бази даних ілюструє схема, наведена на рисунку 1.

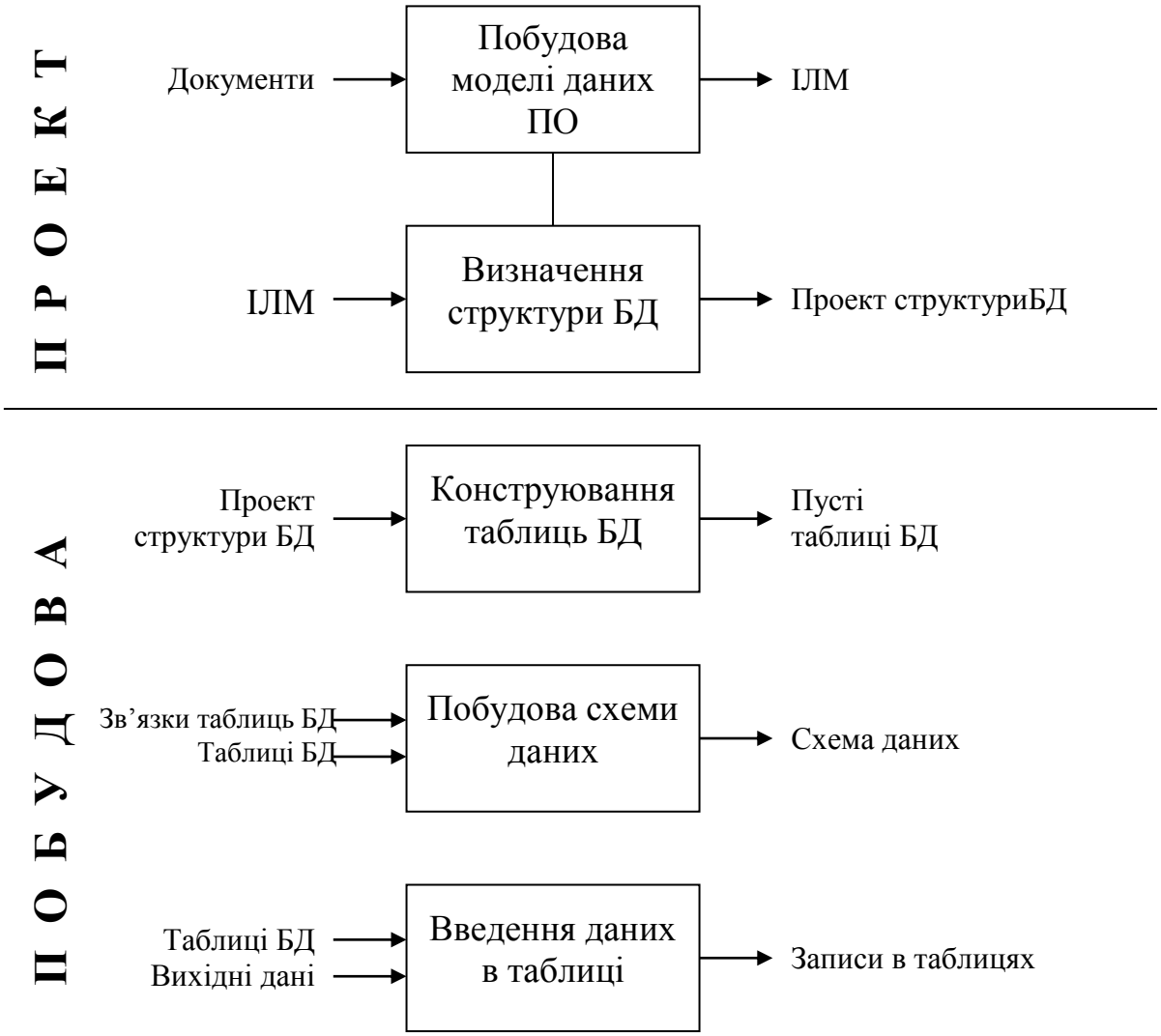


Рисунок 1

Побудова інформаційно-логічної моделі даних

Інформаційно-логічна модель (ІЛМ) відображає дані предметної області в вигляді сукупності інформаційних об'єктів і зв'язків між ними. Ця модель представляє дані, що підлягають зберіганню в базі даних.

Інформаційні об'єкти

Інформаційний об'єкт — це інформаційний опис деякої суттєвості реального об'єкту, процесу, явища або події. Інформаційний об'єкт утворюється сукупністю логічно взаємозв'язаних реквізитів, що встановлюють якісні і кількісні характеристики деякої суттєвості предметної області. Прикладами інформаційних об'єктів можуть бути — ТОВАР, ПОСТАЧАЛЬНИК, ЗАМОВНИК, ПОСТАВКА, ВІДВАНТАЖЕННЯ СПІВРОБІТНИК, ВІДДІЛ, СТУДЕНТ, ВИКЛАДАЧ, КАФЕДРА і т.п.

Інформаційні об'єкти виділяються на основі опису предметної області шляхом визначення функціональних залежностей між реквізитами. Сукупність реквізитів інформаційного об'єкту повинна відповідати вимогам нормалізації.

Кожному інформаційному об'єкту потрібно присвоїти унікальне ім'я, наприклад, СТУДЕНТ, ПРЕДМЕТ, ВИКЛАДАЧ, КАФЕДРА.

Інформаційний об'єкт має безліч реалізацій — примірників. Наприклад, кожний примірник об'єкту СТУДЕНТ представляє конкретного студента. Примірник утворюється сукупністю конкретних значень реквізитів і повинен однозначно визначатися (ідентифікуватися) значенням ключа інформаційного об'єкту, що складається з одного або декількох ключових реквізитів. Таким чином реквізити поділяються на ключові і описові. Останні є функціонально залежними від ключа.

Функціональна залежність реквізитів має місце в тому випадку, якщо одному значенню ключа відповідає тільки одне значення описового (залежного) реквізиту.

Зауваження. При виявленні функціональних залежностей реквізитів не розглядаються арифметичні залежності (наприклад, вартість від кількості), оскільки встановлюється тільки функціональна залежність, визначаюча зв'язки описових і ключових реквізитів, і на основі якої виявляється реквізитний склад кожного інформаційного об'єкту.

При графічному зображенні моделі даних кожний інформаційний об'єкт представляється прямокутником з позначенням його імені і ідентифікатора- ключа. Приклад такого зображення для інформаційних об'єктів ТОВАР і ПОСТАЧАННЯ показаний на рисунку 2. Тут KODT (код товару) — простий ключ об'єкту ТОВАР, а KODT+KPOST (код постачальника) — складний ключ об'єкту ПОСТАЧАННЯ.

Товар
KODT

Поставка
KODT+KPOST

Рисунок 2 - Приклад графічного зображення інформаційних об'єктів з простим і складеним ключем.

Реквізити кожного інформаційного об'єкту повинні відповідати вимогам нормалізації:

1. інформаційний об'єкт повинен містити унікальний ідентифікатор (ключ); Ключ є *простим,* якщо він складається з одного реквізиту або *складеним,* якщо з декількох.
2. всі описові реквізити повинні бути взаємозалежними, тобто між ними не може бути функціональних залежностей;
3. всі реквізити, що входять в складений ключ, повинні бути також взаємозалежні;
4. кожний описовий реквізит повинен функціонально повно залежати від ключа, тобто кожному значенню ключа відповідає тільки одне значення описового реквізиту;
5. при складеному ключі описові реквізити повинні залежати цілком від всієї сукупності реквізитів, що утворюють ключ;
6. кожний описовий реквізит не може залежати від ключа транзитивне,тобто через інший проміжний реквізит.

Зауваження. В випадку транзитивної залежності між реквізитами можна виконати розщеплення сукупності реквізитів з утворенням двох інформаційних об'єктів замість одного.

Завдання:

Згідно індивідуального завдання (додаток 1) необхідно виконати:

1. Побудувати моделі даних ПО.
2. Визначити структури БД.
3. Виконати конструювання таблиць БД.
4. Побудувати схему даних.
5. Ввести дані в таблиці.

Самостійна робота №12
Тема: Нормальні форми.

Нормальна форма - властивість зв'язку в реляційній моделі даних, що характеризує його з точки зору надмірності, потенційно призводить до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовільняти відношення.

Процес перетворення відносин бази даних до виду, який відповідає нормальним формам, називається нормалізацією. Нормалізація призначена для приведення структури БД до виду, що забезпечує мінімальну логічну надмірність, і не має на меті зменшення або збільшення продуктивності роботи або ж зменшення або збільшення фізичного обсягу бази даних. Кінцевою метою нормалізації є зменшення потенційної суперечливості збереженої в базі даних інформації. Як зазначає К. Дейт, загальне призначення процесу нормалізації полягає в наступному:

-виключення деяких типів надмірності;

-усунення деяких аномалій оновлення;

-розробка проекту бази даних, який є досить «якісним» поданням реального світу, інтуїтивно зрозумілий і може служити хорошою основою для подальшого розширення;

-спрощення процедури застосування необхідних обмежень цілісності.

Усунення надмірності проводиться, як правило, за рахунок декомпозиції відносин таким чином, щоб в кожному відношенні зберігалися тільки первинні факти (тобто факти, що не виводяться з інших збережених фактів).

При тому, що ідеї нормалізації вельми корисні для проектування баз даних, вони аж ніяк не є універсальним або вичерпним засобом підвищення якості проекту БД. Це пов'язано з тим, що існує занадто велика різноманітність можливих помилок і недоліків в структурі БД, які нормалізацією усуваються. Незважаючи на ці міркування, теорія нормалізації є дуже цінним здобутком реляційної теорії і практики, оскільки вона дає науково строгі і обґрунтовані критерії якості проекту БД та формальні методи для удосконалення цієї якості.

Змінна відношення знаходиться в першій нормальній формі (1НФ) тоді і тільки тоді, коли в будь-якому допустимому значенні відношення кожен його кортеж містить тільки одне значення для кожного з атрибутів.

Перша нормальна форма

У реляційній моделі відношення завжди знаходиться в першій нормальній формі за визначенням поняття відношення. Що ж стосується різних таблиць, то вони можуть не бути правильними представленнями відношень і, відповідно, можуть не перебувати в 1НФ.

Друга нормальна форма

Змінна відношення знаходиться в другій нормальній формі тоді і тільки тоді, коли вона знаходиться в першій нормальній формі і кожен неключових атрибут функціонально повно залежить від її потенційного ключа.

Третя нормальна форма

Змінна відношення знаходиться в третій нормальній формі тоді і тільки тоді, коли вона знаходиться в другій нормальній формі, і відсутні транзитивні функціональні залежності неключових атрибутів від ключових.

Нормальная форма **Бойса— Кодда** (BCNF)

Змінна відношення знаходиться в нормальній формі Бойса - Кодда (інакше - в посиленій третій нормальній формі) тоді і тільки тоді, коли кожна її нетривіальна і не приводима зліва функціональна залежність має в якості свого детермінанта деякий потенційний ключ.

Четверта нормальна форма

Змінна відношення знаходиться в четвертій нормальній формі, якщо вона знаходиться в нормальній формі Бойса - Кодда і не містить нетривіальних багатозначних залежностей.

П’ята нормальна форма

Змінна відношення знаходиться в п'ятій нормальній формі (інакше - в проекційно-сполучній нормальній формі) тоді і тільки тоді, коли кожна нетривіальна залежність з'єднання в ній визначається потенційним ключем (ключами) цього відношення.

Доменно-ключева нормальна форма

Змінна відношення знаходиться в ДКНФ тоді і тільки тоді, коли кожне накладене на неї обмеження є логічним наслідком обмежень доменів і обмежень ключів, накладених на дану змінну відношення.

Шоста нормальна форма

Змінна відношення знаходиться в шостий нормальній формі тоді і тільки тоді, коли вона задовільняє всім нетривіальним залежностям з'єднання. З визначення випливає, що змінна знаходиться в 6НФ тоді і тільки тоді, коли вона не приводима, тобто не може бути піддана подальшій декомпозиції без втрат. Кожна змінна відношення, яка знаходиться в 6НФ, також знаходиться і в 5НФ.

Введена К. Дейтом в його книзі «Введение в системы баз данных», як узагальнення п'ятої нормальної форми для хронологічної бази даних.

Основні властивості нормальних форм полягають у такому: кожна наступна нормальна форма у деякому змісті краще попередньої нормальної форми; при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

Завдання:

Згідно індивідуального завдання (додаток 1) необхідноперевірити можливі помилки і недоліки в структурі БД за допомогою нормалізації і усунути їх.

Самостійна робота №13

Тема: Рекомендації по розробці структур.

Реляційна модель нічого не говорить про фізичне проектування. Проте є ряд корисних речей, які можна сказати на цю тему в реляционному контексті. Як мінімум, вони випливають з духу моделі, хоча явно не сформульовані (і не дивлячись на те, що деталі фізичної реалізації за потребою залежать від СКБД і в різних системах різні).

Перше: фізичне проектування повинно слідувати за логічним.

Тобто «правильний» підхід полягає в тому, щоб спочатку створити без вад логічний проект, а вже потім відобразити його на ті фізичні структури, які підтримує цільова СКБД. По-іншому цю думку можна висловити, сказавши, що фізичний проект повинен витікати з логічного, а не навпаки. В ідеалі система повинна самостійно вивести оптимальний фізичний проект без втручання людини. (Це завдання не таке вже амбітне, як може здатися).

Друге: одна з причин виключення будь-яких фізичних аспектів з реляційній моделі полягає в тому, щоб дати розробникам можливість реалізувати її, як вони вважають за потрібне, - і тут недостатнє розуміння моделі широкими масами розробників нам гукнулося. Без сумніву, більшість продуктів на основі SQL не зуміли розкрити весь закладений в моделі потенціал; в цих продуктах видиме користувачеві і що фізично зберігається - по суті, одне і те саме. Простими словами, те, що фізично зберігається, - пряме відображення того, що користувач логічно бачить (рис. 1).

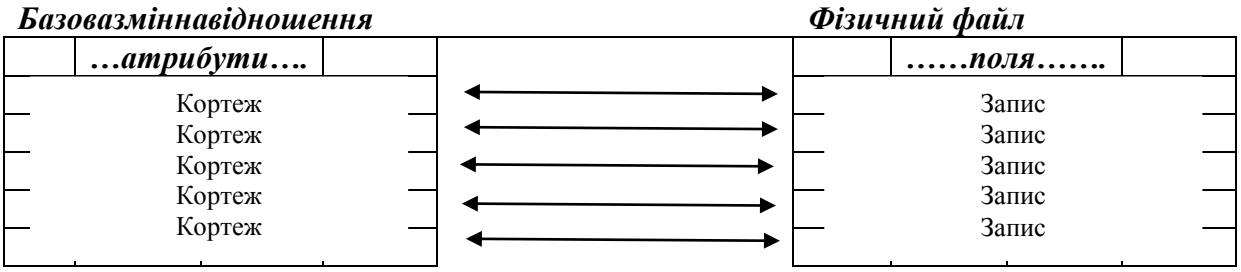


Рис. 1*Реалізація з прямим відображенням (гідна осуду)*

У реалізації з прямим відображенням багато чого не так. Але найголовніше майже повна відсутність незалежності від даних: якщо нам знадобиться змінити фізичну структуру (як правило, з міркувань продуктивності), то доведеться міняти і логічну структуру теж. Зокрема, мова йде про пропозицію "денормалізувати заради підвищення продуктивності". В принципі, логічна структура не має нічого спільного з продуктивністю, але якщо вона один до одного відображається на фізичну .... Очевидно, можна знайти більш вдале рішення. Захисники реляційної теорії багато років переконували, що реляційну модель необов'язково реалізовувати таким способом.

Ось кілька бажаних наслідків реалізації, в якій фізичний і логічний рівні строго розділені.

По-перше, нам ніколи не доведеться «денормалізовувати заради підвищення продуктивності» (на логічному рівні); всі змінні відносини можуть перебувати в 5НФ, а то і в 6НФ, і це не зашкодить продуктивності. Логічна структура взагалі ніяк не відображатиметься на продуктивності.

По-друге, 6НФ створює основу для по-справжньому реляційного підходу до проблеми відсутньої інформації (без використання null-значень і тризначної логіки). Використовуючи null-значення, ви по суті змушуєте базу даних явно повідомляти про те, що є щось, чого ви не знаєте. Але якщо ви чогось не знаєте, то набагато краще про це взагалі не говорити.

Наприклад, припустимо для простоти,

що є всього два постачальники S1 і S2, і статус S1 нам відомий, а статус S2 - немає. У цій ситуації 6НФ могла б виглядати, як показано на рис. 2

SN

SNO	SNAME

SS

SNO	STATUS

SC

SNO	CITY

S1	Smith
S2	Jones

S1	20

S1	London
S2	Paris

Рис. 2. *Статус постачальника S2 невідомий*

При такій структурі відсутній «кортеж», в якому статус постачальника S2 містив би null, - у нас взагалі немає кортежу, що показує статус S2.

Зауваження. Основний упор в даному додатку був зроблений на теорії проектування баз даних, під якою розуміють, перш за все, нормалізацію і ортогональность (наукові складові дисципліни проектування). Ідея в тому, що логічна структура, на відміну від фізичної, є - або повинна бути - незалежної від конкретної СКБД, і, як ми бачили, існують солідні теоретичні принципи, які можна з користю застосувати до цієї проблеми.

Логічна структура, взагалі кажучи, повинна в ідеалі не залежати і від програми, а не тільки від СКБД. Завдання полягає в тому, щоб скласти проект, який описує, що означають дані, а не як вони будуть використовуватися, і в зв'язку з цим підкреслюється важливість обмежень і предикатів («бізнес-правил»). База даних повинна бути вірним поданням семантики ситуації, а семантику представляють саме обмеження. Таким чином, абстрактно процес логічного проектування виглядає наступним чином:

1. Максимально ретельно виписати предикати переменних-

відносин.

2. Показати результати кроку 1 на змінні-відношення і обмеження. (В ролі деяких обмежень виступатимуть ФЗ, МЗЗ і ЗС).

Значна частина теорії проектування пов'язана зі зменшенням надмірності. Нормалізація зменшує надмірність всередині змінних-відносин, ортогональность - між декількома змінними-відносинами.

Під час обговорення нормалізації основна увага приділяється НФБК і 5НФ, які представляють собою нормальні форми щодо ФЗ і ЗС відповідно. Підкреслюється, що нормалізація дозволяє простіше формулювати (і, можливо, перевіряти) деякі обмеження; еквівалентно можна сказати, що її наявність допускає більшу кількість оновлень єдиного кортежу, ніж було б можливо в іншому випад випадку. Нормалізація - це, по суті, формалізований здоровий глузд. Логічний і, можливо, не надто широко відомий аргумент на користь відмови від денормалізації, розглядаючи проблему з точки зору не стільки поновлення, скільки вибірки. Хоча найчастіше до денормалізації закликають в ім'я підвищення продуктивності, насправді це може привести до зниження продуктивності (і в плані оновлення, і в плані вибірки). Фактично «денормализация заради продуктивності» зазвичай означає підвищення продуктивності однієї програми за рахунок інших.

Ортогональное проектування це ще один приклад формалізації здорового глузду а, на рахунок фізичного проектування: по-перше фізична структура повинна витікати з логічної, а не навпаки.

По-друге, нам гостро необхідно піти від повсюдно поширився способу реалізації шляхом прямого відображення. По-третє, було б добре добитися повної автоматизації фізичного проектування.

І останнє: принципи нормалізації та ортогональность в деякому сенсі є обов'язковими. Їх не можна назвати незаперечними правилами, які слід завчити і ніколи не порушувати. Як ми знаємо, бувають поважні причини не доводити нормалізацію «до кінця» (маються на увазі ґрунтовні логічні причини, а не «денормализация заради продуктивності»). Те ж саме справедливо по відношенню до ортогональности. Правда, треба розуміти, що як неповна нормалізація, так і порушення ортогональності можуть стати причиною надмірності і деяких аномалій при оновленні. Навіть при наявності теорії проектування, проектування баз даних зазвичай має на увазі різні компроміси.

1 Завдання:

Спроектуйте базу даних для наступного завдання.Сутностями є торгові представники, регіони продажіві вироби. Кожен представник відповідає за один або кількарегіонів; в кожному регіоні є один або кілька представників.Кожен представник відповідає за продаж

одного або кількохвиробів; кожен виріб продає один або кілька представників. Кожен виріб продається в кожному регіоні, проте ніякідва представника не продають один і той же виріб в одному і тому жрегіоні. Кожен представник продає один і той же набір виробів в кожному регіоні, за який відповідає.

2 Завдання:

Спроектуйте базу даних для наступного завдання. Сутностями є торгові представники, регіони продажів і вироби. Кожен представник відповідає за один або кілька регіонів; в кожному регіоні є один або кілька представників. Кожен представник відповідає за продаж одного або декількох виробів; кожен виріб продає один або кілька представників. Кожен виріб продається в одному або декількох регіонах, і в кожному регіоні продається одне або кілька виробів. Нарешті, якщо представник r відповідає за регіон a, і виріб p продається в регіоні a, і представник r продає виріб p, то r продає p в a.

Самостійна робота №14

Тема: Семантичне моделювання даних. ER – діаграми.

Семантичне моделювання стало предметом інтенсивних досліджень з кінця 1970-х років Основним спонукальним мотивом подібних досліджень (тобто проблемою, яку намагалися вирішити дослідники) був наступний факт. Найчастіше системи баз даних дозволяють лише маніпулювати даними певних простих типів і визначають деякі найпростіші обмеження цілісності, накладені на ці дані. Будь більш складна інтерпретація покладається на користувача. Однак було б чудово, якби системи могли володіти трохи більш широким обсягом відомостей і дещо інтеллектуальніше відповідали на запити користувача, а також підтримували складніші (тобто більш високорівневі) інтерфейси користувача. Наприклад, було б чудово, якби система могла визначати, що:

Слід зазначити, що з нашої точки зору система з підтримкою**предикатів**повинна була б розуміти трохи більше. Інакше кажучи, можна було б заперечити, що описана підтримка предикатів може сприйматися як зручна та ефективна підстава для семантичногомоделювання. Однак, як це не сумно, більшість схем семантичного моделювання не має ніякого суворого обгрунтування, оскільки всі вони в тій чи іншій мірі є довільними. Можливо, такий стан справ у майбутньому зміниться, особливо завдяки зростаючому розумінню в світі комерції важливості**бізнес-правил. З**овнішні предикати по суті якраз і є такими бізнес-правилами.Вага деталі та кількість, що поставляється є хоча і числовими, але все ж*семантично*різними величинами Це дозволило б системі поставити під сумнів або навіть зовсім відкинути запит, в якому зєднання інформації про деталі і постачаннях виконується шляхом порівняння ваги деталі з кількістю, що поставляється.

Безумовно, до наведеного прикладу пряме відношення має поняття типів (або *доменів)* Це служить прекрасною ілюстрацією того факту, що існуючі моделі даних все ж не позбавлені семантичних аспектів Зокрема, домени, потенційні і зовнішні ключі є прикладами семантичних аспектів існуючої реляційної моделі, що безпосередньо випливає з її визначення. В якості іншого способу реалізації семантики були розроблені різні розширені моделі даних, які, тим не менш, несуть лише ненабагато більше смислове навантаження, ніж моделі, запропоновані раніше.

Термін**семантична модель,**який часто використовується по відношенню до тієї чи іншої розширеної моделі, в даному випадку не зовсім підходить, оскільки передбачає, що модель побудована таким чином, щоб виявити *всю* семантику ситуації, що розглядається. З іншого боку, поняття **семантичне моделювання** *дійсно* є досить вдалою назвою цілої галузі досліджень, присвячених способам представлення смислового значення.

Слід зазначити, що семантичнемоделюваннямає також кілька інших назв, наприклад, *моделювання, ER-моделювання, моделювання сутностей* і *об'єктне моделювання. П*еревага віддається терміну *семантичне моделювання* (яке іноді називають також *концептуальним моделюванням).*

■ Термін *моделювання* не підходить, оскільки він конфліктує зі сформованим раніше визначенням терміна *модель даних,* який позначає формальну систему на зразок реляційної моделі. До того ж термін *моделювання* сприяє зміцненню широко поширеної помилки, що модель даних (у наведеному вище сенсі) охоплює *тільки* структуру даних.

■ Також невдалим є термін *ER-моделювання,*оскільки в ньому неявно мається на увазі існування тільки одного підходу до даної проблеми, тоді як на практиці, безумовно, існує багато різних підходів Тим не менш, термін *ER-моделювання* добре відомий, вельми популярний **і** прийнятий в широкому колу фахівців.

■ Щодо терміна *моделювання сутностей* немає ніяких значних заперечень за винятком того, що він здається більш вузько спеціалізованим в порівнянні з терміном *семантичне моделювання* і носить певний смисловий акцент, який не зовсім підходить в цьому випадку.

■ Що стосується терміну *обєктне моделювання,*то проблема тут полягає в тому, що термін*об'єкт* в даному контексті, очевидно, є синонімом терміна *сутність,*тоді як він використовується в зовсім іншому сенсі і іншому контексті. На думку Дейт К. Дж.

(«Введення в системи баз даних»), саме цей факт (наявність у даного терміна двох різних значень) зявився причиною такого явища, як **Перша Велика Омана** (FirstGreatBlunder).

Найбільш часто на практиці семантичне моделювання використовується на першій стадіїпроектування бази даних. При цьому в термінах семантичної моделі проводиться концептуальна схема бази даних, яка потім вручну концептуальна схема перетвориться до реляційної (або який-небудь інший) схемою. Цей процес виконується під управлінням методик, в яких досить чітко обумовлено всі етапи такого перетворення.Менш часто реалізується автоматизована компіляція концептуальної схеми в реляційну. При цьому відомі два підходи: на основі явного подання концептуальної схеми як вихідної інформації для компілятора і побудови інтегрованих систем проектування з автоматизованим створенням концептуальної схеми на основі інтерв'ю з експертами предметної області. І в тому, і в іншому випадку в результаті виробляється реляційна схема бази даних в третій нормальній формі (більш точно слід було б сказати, що авторові невідомі системи, що забезпечують більш високий рівень нормалізації). Нарешті, третя можливість, яка ще не вийшла (або тільки виходить) за межі дослідницьких та експериментальних проектів, - це робота з базою даних у семантичній моделі, тобто СКБД, засновані на семантичних моделях даних. При цьому знову розглядаються два варіанти: забезпечення для користувача інтерфейсу на основі семантичної моделі даних завтоматичним відображенням конструкцій в реляційну модель даних (це завдання приблизно такого ж рівня складності, як автоматична компіляція концептуальної схеми бази даних в реляційну схему) і пряма реалізація СУБД, заснована на будь-якої семантичної моделі даних. Найбільш близько до другого підходу знаходяться сучасні об'єктно-орієнтовані СУБД, моделі даних яких за багатьма параметрами близькі до семантичних моделей (хоча в деяких аспектах вони більш потужні, а в деяких - слабші).

Процес моделювання полягає в виділенні об’єктів (сутностей предметної області), встановленні властивостей виділених об’єктів і виявленні існуючих між ними зв’язків. Одна з найбільш важливих і розповсюджених семантичних моделей є модель “сутність-зв’язок”, ER-модель *(EntityRelationship).* Основними поняттями ER-моделі є сутність, зв’язок і атрибут. Як і в реляційних схемах, в ER-моделях вводиться поняття нормальних форм, їх зміст дуже близький до змісту реляційних нормальних форм. Цей підхід дозволяє на початковій стадії правильно спроектувати логічну структуру БД.

ER-модель являє собою формальну конструкцію, яка сама по собі не наказує ніяких графічних засобів її візуалізації. Як стандартна графічна нотація, за допомогою якої можна візуалізувати ER-модель, була запропонована діаграма сутність-зв'язок (ER-діаграма).

Один з єтапів побудови ER-моделі є побудова діаграм потоків, при якій будують ієрархію діаграм потоків даних. Розглянемо на прикладі.

Відповідно до методології модель системи визначається як ієрархія діаграм потоків даних (ДПД або DateFlowDiagrams - DFD), що описують асинхронний процес перетворення інформації від її введення в систему до видачі користувачу. Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси або підсистеми ІС з зовнішніми входами і виходами. Вони деталізуються за допомогою діаграм нижнього рівня. Така декомпозиція триває, створюючи багаторівневу ієрархію діаграм, до тих пір, поки не буде досягнутий такий рівень декомпозиції, на якому процеси стають елементарними і деталізувати їх далі недоцільно. У термінах діалогового інтерфейсу - це рівень окремих діалогових вікон. Наприклад процес (вікно) «Ідентифікація та коригування даних про клієнта» або «Введення нового договору» тощо.

Основними компонентами графічної мови діаграм потоків даних є:

- зовнішні сутності (джерела даних);
- системи / підсистеми, процеси;
- накопичувачі даних;
- потоки даних.

Джерела інформації (зовнішні сутності) породжують інформаційні потоки (потоки даних), які переносять інформацію до підсистем або процесам. Ті в свою чергу перетворюють інформацію і породжують нові потоки, які переносять інформацію до інших

процесів або підсистемам, накопичувачам даних або зовнішнім сутностям - споживачам інформації.

Зовнішні сутності

Зовнішня сутність являє собою матеріальний об'єкт, організацію або фізичну особу, яка представляє собою джерело або приймач інформації. Наприклад:

- замовники;
- персонал;
- постачальники;
- клієнти;
- склад.

Першим кроком при побудові ієрархії ДПД є побудова контекстних діаграм 0-го рівня (рис 1). Зазвичай при проектуванні відносно простих ІС будується єдина контекстна діаграма із зіркоподібною топологією, в центрі якої знаходиться так званий головний процес (сама ІС), з'єднаний з приймачами і джерелами інформації, за допомогою яких з системою взаємодіють користувачі та інші зовнішні системи. Потоки - зовсім конкретні речі, які спочатку виражаються у вигляді документів та регламентованих повідомленнь. Слід згадати, що ІС - комплекс засобів для автоматизації або отримання інформації про що завгодно, відрізняється від інформаціних технологій - це деяка переробка інформації, та включає в себе ІС і людей.

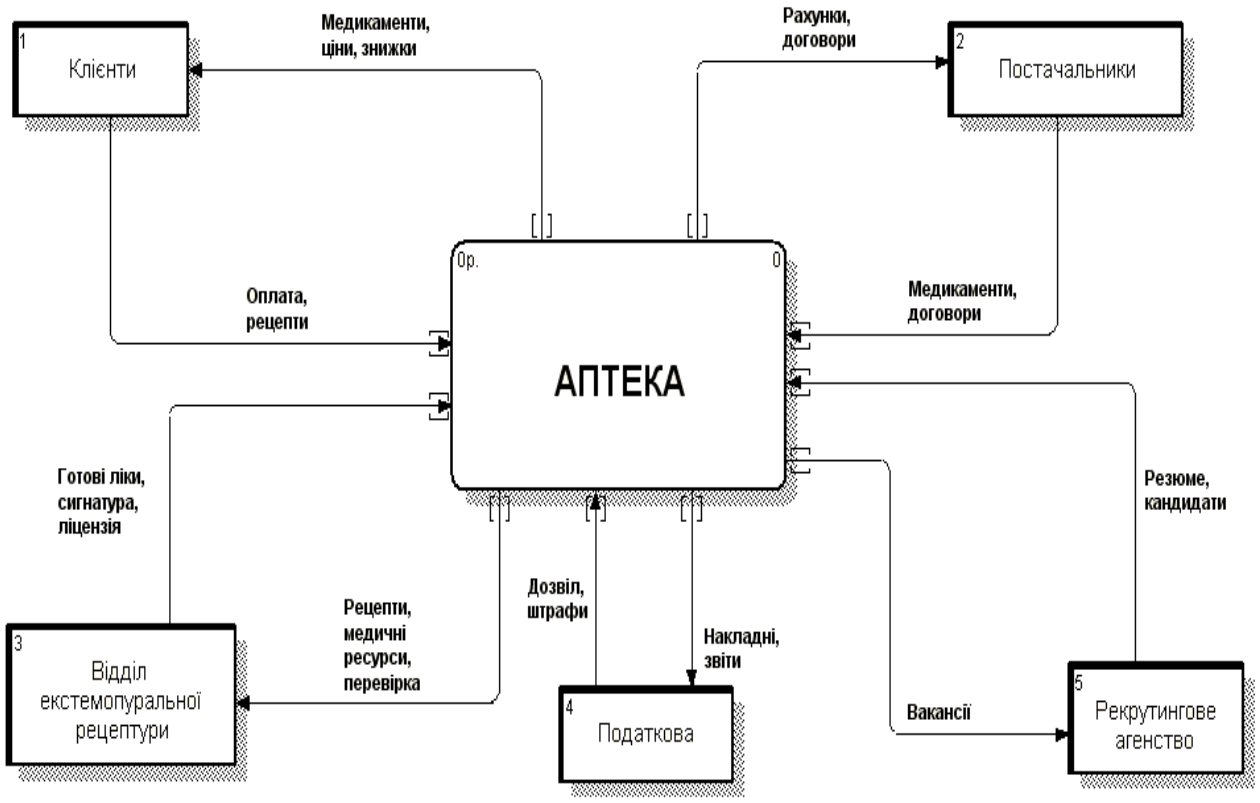


Рисунок 1 - Діаграма потоків даних нульового рівня (DFD0) для аптеки

Якщо ж для складної системи обмежитися єдиною контекстною діаграмою, то вона буде містити дуже велику кількість джерел і приймачів інформації, які важко розташувати на аркуші паперу нормального формату, і крім того, єдиний головний процес не розкриває структури розподіленої системи.

Діаграма потоків даних DFD1

Для складних ІС будується ієрархія контекстних діаграм. При цьому контекстна діаграма 1-го рівня містить набір підсистем, з'єднаних потоками даних (рис. 2.).

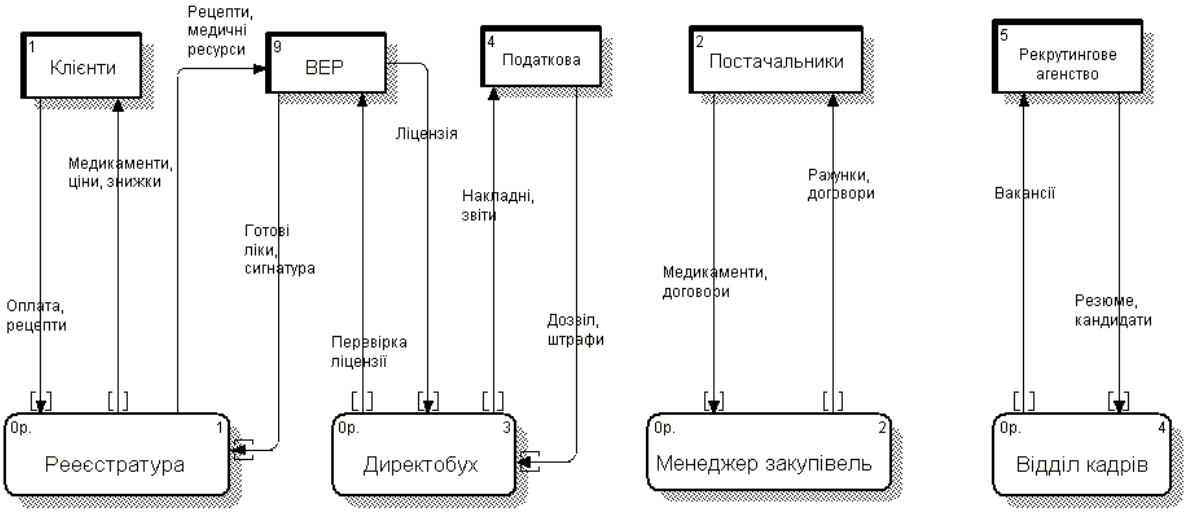


Рис. 2 - Діаграма потоків даних першого рівня (DFD1) для аптеки

Сенс DFD1, що кожну підсистему «Реєстратура», «Директобух», «Менеджер закупівель» та «Відділ кадрів» будуть робити різні люди. Така схема каже про те, що у цих чотирьох підсистем є загальні структури даних (але може бути і різниця). «Хвороба» - зайва структура. Але на першому етапі правильно буде, якщо цю структуру поки що сховати. На DFD1 виносяться лише *загальні* структури.

Якщо потрібно додати відділ «Гуртових продаж», але четвертого місця немає. Тоді є два виходи з даної ситуації:

- у випадку наявності зайвого робочого місця, зробити щє один процес. Можна зробити структуру даних «Ціни» всередині відділу «Менеджер закупівель»;
- зробити щє одне джерело інформації. Тимчасово в підсистемі «Менеджер закупівель» зробити введення інформації про необхідні продажи ліків. В цьому випадку сутність «Гуртові продажі» є зовнішнею.

Для кожної структури даних, існує робоче місце, на якому вносяться дані.

Діаграма потоків даних DFD2

Контекстні діаграми наступного рівня (DFD2) деталізують контекст і структуру підсистем до рівня окремих завдань. Ця діаграмма відображає наше особисте робоче місце. На рис. 3 відображено приклад діаграми DFD2 для підсистеми «Реєстратура». Необхідно розділяти структури даних і процеси з однаковими назвами. І якщо є можливість, тоді переіменувати для унікальності назв. До процесів відносяться дії.

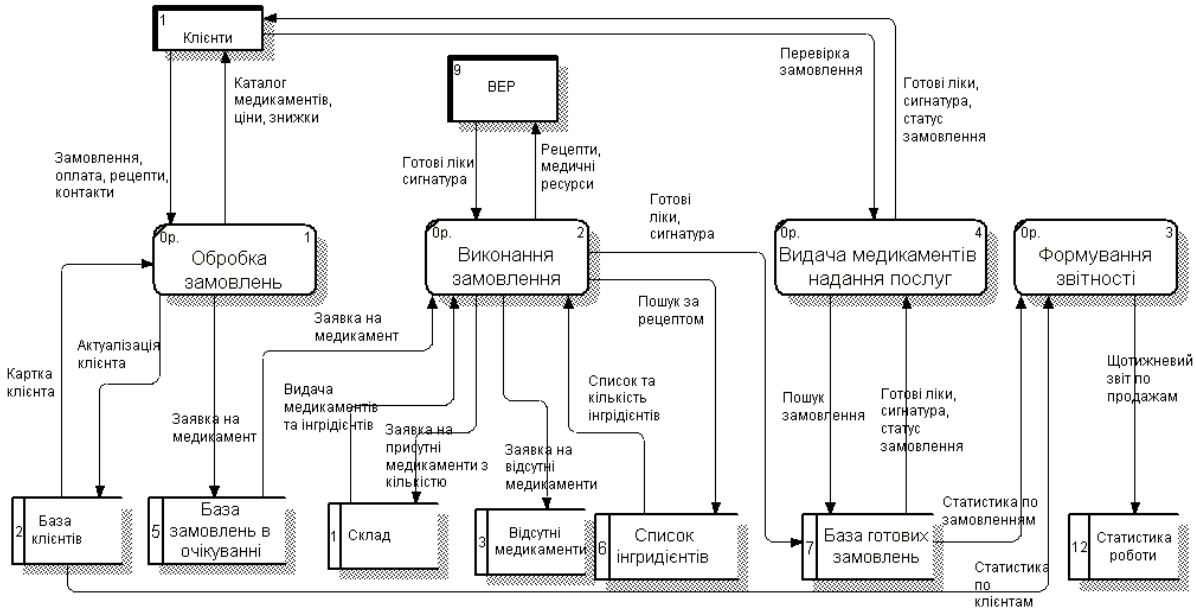


Рис. 3 Діаграма потоків даних другого рівня (DFD2) для відділу рєстратури

Ієрархія контекстних діаграм визначає взаємодію основних функціональних підсистем проектованої ІС як між собою, так і з зовнішніми вхідними та вихідними потоками даних та зовнішніми об'єктами (джерелами і приймачами інформації), з якими взаємодіє ІС. Діаграма містить накопичувачі, які є прообразами майбутньої БД.

Розробка контекстних діаграм вирішує проблему суворого визначення функціональної структури ІС на самій ранній стадії її проектування, що особливо важливо для складних багатофункціональних систем, у розробці яких беруть участь різні організації та колективи розробників.

Після побудови тематичних діаграм отриману модель слід перевірити на повноту вихідних даних про об'єкти системи та ізольованість об'єктів (відсутність інформаційних зв'язків з іншими об'єктами).

Для кожної підсистеми, що присутня на контекстних діаграмах, виконується її деталізація за допомогою ДПД. Кожен процес на ДПД, в свою чергу, може бути деталізований за допомогою ДПД або мініспецифікацій.

Завдання:

Згідно індивідуального завдання (додакок 1) побудувати ієрархію діаграм потоків даних.

Самостійна робота №15

Тема: Проектування концептуальної схеми бази даних. ER – моделювання даних.

Під час проектування баз даних відбувається перетворення ER-моделі в конкретну схему бази даних на основі обраної моделі даних (реляційної, об'єктної, мережевої або ін.).

Модель «сутність-зв'язок» була запропонована в 1976 році Пітером Пін-Шен Ченом. Згідно Ченовій нотації безлічі сутностей зображуються (рис. 1) у вигляді прямокутників, безлічі відносин зображуються у вигляді ромбів. Якщо сутність бере участь в відношенні, вони пов'язані лінією. Якщо відношення не є обов'язковим, то лінія пунктирна. Атрибути зображуються у вигляді овалів і зв'язуються лінією з одним ставленням або з однією сутністю.

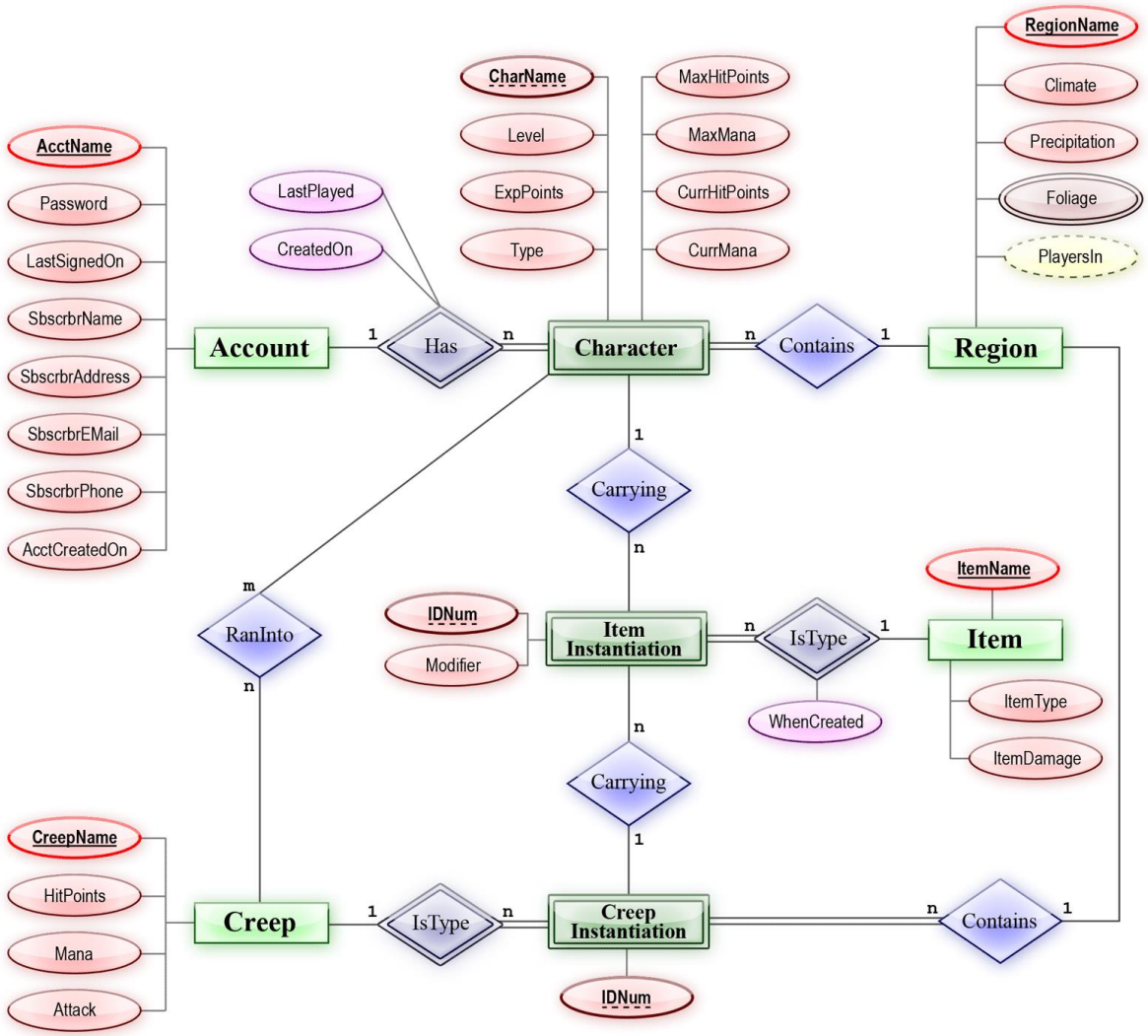


Рисунок 1 - Проста ER-модель з використанням нотаціїПітера Чена

Дана нотація була запропонована Гордоном Еверестом під назвою Inverted Arrow («перевернута стрілка»), однак зараз частіше звана Crow's Foot («вороння лапка») або Fork («вилка»).

Відповідно до даної нотації, сутність зображується (рис. 2) у вигляді прямокутника, що містить її ім'я, яке виражається іменником. Ім'я сутності повинно бути унікальним в рамках однієї моделі. При цьому, ім'я сутності - це ім'я типу, а не конкретного екземпляра даного типу. Екземпляром сутності називається конкретний представник даної суті.

Зв'язок зображується лінією, яка зв'язує дві сутності, що беруть участь в відношенні. Ступінь кінця зв'язку вказується графічно, множинність зв'язку зображується у вигляді «вилки» на кінці зв'язку. Модальність зв'язку так само зображується графічно -

необов'язковість зв'язку позначається кружком на кінці зв'язку. Іменування зазвичай виражається одним дієсловом в дійсного способу теперішнього часу: «Має», «Належить» і т. д.; або дієсловом з пояснювальними словами: «Включає в себе», і т.п. Найменування може бути одне для всієї зв'язку або два для кожного з кінців зв'язку. У другому випадку, назва лівого кінця зв'язку вказується над лінією зв'язку, а правого - під лінією. Кожне з назв розташовуються поруч з сутністю, до якої воно відноситься.

Атрибути сутності записуються всередині прямокутника, що зображує сутність і виражаються іменником в однині (можливо, з уточнюючими словами). Серед атрибутів виділяється ключ сутності - ненадлишкових набір атрибутів, значення яких в сукупності є унікальними для кожного екземпляра сутності.

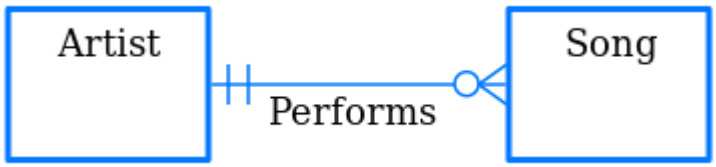


Рисунок 2 - Приклад відношення між сутностями згідно нотації Crow'sFoot

Завдання:

Побудувати ER-модель ІС згідно індивідуального завдання (додаток 1).

Самостійна робота №16

Тема:Етапи проектування баз даних.

Проектування бази даних починається з вивчення технічного завдання на проектування бази даних, яке повинен надати замовник. Отже, бажано, щоб замовник володів відповідною термінологією і знав, принаймні в загальних рисах, технічні можливості основних СУБД. На жаль, на практиці ці побажання виконуються не завжди. Тому зазвичай розробники використовують такі підходи: демонструють замовникові роботу аналогічної бази даних, після чого узгоджують специфікацію відмінностей; якщо аналога немає, з'ясовують коло задач і вимог замовника, після чого допомагають йому підготувати технічне завдання. Під час підготовки технічного завдання складають: перелік вхідних даних, з якими працює замовник; перелік вихідних даних, потрібних замовникові для управління структурою свого підприємства; перелік вихідних даних, які не є необхідними для замовника, але які він повинен надати іншим організаціям (у вищестоящі структури, в органи статистики, інші адміністративні і контрольні організації).

Процес проектування бази даних поділяється на етапи, кожний з яких передбачає виконання певних дій.

Перший етап-розробка інформаційно-логічної моделі даних предметної області, який базується на описі предметної області, отриманому в результаті її обстеження. На цьому етапі спочатку визначають склад і структуру даних предметної області, які мають міститись у базі даних та забезпечувати виконання запитів, задач і застосувань користувача. Ці дані мають форму реквізитів, що містяться в різних документах - джерелах завантаження бази даних. Аналіз виявлених даних дозволить визначити функціональні залежності реквізитів, які використовують для виділення інформаційних об'єктів, що відповідають вимогам нормалізації даних. Подальше визначення структурних зв'язків між об'єктами дозволяє побудувати інформаційно-логічну модель.

Другий етап - визначення логічної структури бази даних. Для реляційної бази даних цей етап є значною мірою формальним, оскільки інформаційно-логічна модель відображається в структуру реляційної бази даних адекватно.

Наступний етап - конструювання таблиць бази даних, який здійснюється засобами СКБД, та узгодження їх із замовником. Структура таблиць бази даних задається за допомогою засобів опису (конструювання) таблиць у СКБД із цілковитою відповідністю інформаційним об'єктам. Крім таблиць, проектувальники розробляють й інші об'єкти бази даних, які призначені, з одного боку, для автоматизації работи з базою, а з іншого - для обмеження функціональних можливостей работи з базою (безпека бази даних). Після формування структури таблиць база даних може наповнюватись даними з документів-джерел. Проектувальники, як правило, не наповнюють базу конкретними даними (оскільки замовник може вважати дані конфіденційними і не надавати стороннім особам). Винятком є експериментальне наповнення модельними даними на етапі відлагодження об'єктів бази.

Завдання:

Дайте відповіді на запитання:

1. З чого починається проектування БД?
2. Які дані повинні міститись у технічному завданні?
3. Які етапи проектування БД ви знаєте?
4. Що виконують на першому етапі проектування БД?
5. Що виконують на другоу етапі проектування БД?
6. Що виконують на наступному етапі після другого етапа проектування БД?
7. На якому етапі БД наповнюють конкретними даними з документів-джерел?

Самостійна робота №17
Тема: Приклад побудови ER-моделі.

Завдання:

Згідно індивідуального завдання (додаток 1) необхідно виконати:

- опис ПО
- побудувати початкову контекстну діаграму
- побудувати таблицю – список подій
- побудувати таблицю – розподілу потоків
- побудувати повну контекстну діаграму
- побудувати діаграму структур даних
- побудувати модульну модель

Самостійна робота №18

Тема: Зберігання інформації у базах даних.

Не завжди користувачі замислюються про те, наскільки оптимально працює їх база, як можна оптимізувати процеси, що відбуваються при роботі з MYSQL і яке буде функціонування віртуального сервера при навантаженні, що збільшилась кількість користувачів в результаті, наприклад, "розкручування" сайту.

Як оптимізувати роботу з СУБД MYSQL. Які найчастіші помилки зроблені користувачами і, як їх уникнути**. Які дані потрібно зберігати в MYSQL.**

Не намагайтеся помістити в бази даних всю інформацію, яка у є. Наприклад, не потрібно зберігати там малюнки, хоч MYSQL це і дозволяє. Поміщаючи в базу даних двійкові образи графічних файлів, можна тільки уповільните роботу свого сервера. Прочитати файл з картинкою з диска набагато простіше і, з погляду споживаних ресурсів, економічніше, ніж з'єднатися з скрипта до SQL, зробити запит, отримати образ, обробити його і, видавши потрібні http-заголовки, показати відвідувачеві веб-сервера. У другому випадку операція видачі картинки зажадає у декілька разів більше ресурсів процесора, пам'яті і диска. Також варто пам'ятати про те, що існують механізми кешування веб-документів, які дозволяють користувачеві економити на трафіку, а при динамічній генерації контента Ви фактично позбавляєте своїх відвідувачів цієї зручної можливості.

Замість малюнків краще зберігати в MYSQL інформацію, на основі якої можна генерувати посилання на статичні картинки в динамічно створюваних скриптами документах.

Оптимізація запитів

У ситуаціях, коли реально потрібно отримати тільки певну порцію даних з MYSQL, можна використовувати ключ LIMIT для функції SELECT. Це корисно, коли, наприклад, потрібно показати результати пошуку чого-небудь в базі даних. Допустимо, в базі є список товарів, які пропонує Ваш інтернет-магазин. Видавати ваш список товарів в потрібній категорії дещо негуманно по відношенню до користувача - канали зв'язку з інтернет не у всіх швидкі і видача зайвих ста кілобайт інформації часто примушує користувачів провести не одну хвилину в очікуванні результатів завантаження сторінки. У таких ситуаціях інформацію видають порціями, наприклад по 10 позицій.

Неправильно робити вибірку з бази всієї інформації і фільтрацію висновку скриптом. Набагато оптимальніше буде зробити запит вигляду:

```
select good, price from books limit 20,10
```

В результаті, MYSQL "віддасть" Вам 10 записів з бази починаючи з 20-ої позиції. Видавши результат користувачеві, зробіть посилання "Наступні 10 товарів", як параметр передавши скрипту наступну позицію, з якою робитиметься виведення списку товарів, і використовуйте це число при генерації запиту до MYSQL.

Також слід пам'ятати, що при складанні запитів до бази даних (SQL queries) слід запрошувати тільки ту інформацію, яка Вам реально потрібна. Наприклад, якщо в базі 10 полів, а в даний момент реально потрібно отримати тільки два з них, замість запиту:

```
select * from table_name
```

використовуйтеконструкціювиду:

```
select field1, field2 from table_name
```

Такимчином, Виненавантажуватимете MYSQL непотрібноюроботою, займатизайвупам'ятьіздійснюватидодатковідисковіоперації.

Також слід використовувати ключ WHERE там, де потрібно отримувати інформацію, що потрапляє під певний шаблон. Наприклад, якщо потрібно отримати з бази поля з назвами книг, автором яких є Іванов, слід використовувати конструкцію вигляду:

```
select title from books where author='Іванов'
```

Також є ключ LIKE, який дозволяє шукати поля, значення яких "схожі" на заданий шаблон:

```
select title from books where author like 'Іванов%'
```

В даному випадку MYSQL видасть назви книг, значення поля author у яких починаються з 'Іванов'.

Ресурсоємні операції

Разом з тим слід пам'ятати, що існують операції, виконання яких саме по собі вимагає великих ресурсів, чим для звичайних запитів. Наприклад, використання операції DISTINCT до функції SELECT вимагає набагато більшої кількості процесорного часу, чим звичайний SELECT. DISTINCT намагається шукати унікальні значення, часто проводячи безліч порівнянь, підстановок і розрахунків. Причому, чим більше стає об'єм даних, до якого застосовується DISTINCT (адже Ваша база з часом росте), тим повільніше виконуватиметься такий запит і зростання ресурсів, потрібних для виконання такої функції, відбуватиметься не прямопропорційно об'єму даних, що зберігаються, а набагато швидше.

Індекси

Індекси використовують для швидшого пошуку по значенню одного з полів. Якщо індекс не створюється, то MYSQL здійснює послідовний перегляд всіх полів з найпершого запису до найостаннішого, здійснюючи зіставлення вибраного значення з результатним. Чим більше таблиця і чим більше в ній полів, тим довше здійснюється вибірка. Якщо ж у даної таблиці існує індекс для даного стовпця, то MYSQL зможе зробити швидке позиціонування до фізичного розташування даних без необхідності здійснювати повний перегляд таблиці. Наприклад, якщо таблиця складається з 1000 рядків, то швидкість пошуку буде як мінімум в 100 разів швидше. Ця швидкість буде ще вища, якщо є необхідність звернутися відразу до всіх 1000 стовпців, оскільки в цьому випадку не відбувається витрат часу на позиціонування жорсткого диску.

У яких ситуаціях створення індексу доцільне:

Швидкий пошук рядків при використанні конструкції WHERE;

Пошук рядків з інших таблиць при виконанні об'єднання;

Пошук значення MIN() або MAX() для проіндексованого поля;

Сортування або угрупування таблиці у випадку, якщо використовується проіндексоване поле.

Якщо виконуються запити вигляду

```
SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;
```

і існує змішаний індекс для полів col1 і col2, то дані будуть повернені безпосередньо. Якщо ж створені окремі індекси для col1 і для col2, то оптимізатор спробує знайти найбільш обмежений індекс шляхом визначення того, який з індексів може знайти менше рядків, і використовуватиме цей індекс для отримання даних.

Якщо у таблиці є змішаний індекс, то використовуватиметься будь-який лівобічний збіг з існуючим індексом. Наприклад, якщо є змішаний індекс 3-х полів (col1, col2, col3), то індексний пошук можна здійснювати по полях (col1), (col1, col2) і (col1, col2, col3).

Однак збільшення числа індексів уповільнює операції додавання, оновлення, видалення рядків таблиці, оскільки при цьому доводиться оновлювати самі індекси. Крім того, індекси займають додатковий обсяг пам'яті, тому перед створенням індексу слід переконатися, що планований виграш в продуктивності запитів перевищить додаткову витрату ресурсів комп'ютера на супровід індексу.

Підтримказ'єднання

Для роботи з MySQL-сервером необхідно заздалегідь встановити з ним з'єднання, пред'явивши логін і пароль. Процес установки з'єднання може продовжуватися набагато більший час, ніж безпосередня обробка запиту до бази після установки з'єднання. Слідуючи логіці, треба уникати зайвих з'єднань до бази, не від'єднуючись від неї там, де це можна зробити, якщо надалі планується продовжити роботу з SQL-сервером. Наприклад, якщо скрипт встановив з'єднання до бази, зробив вибірку даних для аналізу, не потрібно закривати з'єднання до бази, якщо в процесі роботи цього ж скрипта планується результати аналізу помістити в базу.

Також можна підтримувати так зване persistent (постійне) з'єднання до бази, але це можливо в повному об'ємі при використанні складніших середовищ програмування, чим php

або perl в звичайному CGI-режимі, коли інтерпретатор відповідної мови разовий запускається веб-сервером для виконання запиту, що прийшов.

Завдання:

Створіть таблицю table_name з полями field1, field2, field3…

Заповніть інформацією згідно темі індивідуального завдання (додаток 1)

Виконайте наступні запити. Який з 2-х запитів оптимальніший? Чому?

1. select * from table_name
2. select field1, field2 from table_name

Виконайте наступні запити. Яка з операцій вимагає більшої кількості процедурного часу? Чому?

1. SELECT*FROM*table_name*;
2. SELECT DISTINCT field3 FROM *table_name*;

Самостійна робота №19
Тема:Індексація даних.

Індекс (англ. Index) - об'єкт бази даних, який створюється з метою підвищення продуктивності пошуку даних. Таблиці в базі даних можуть мати велику кількість рядків, які зберігаються в довільному порядку, і їх пошук по заданому критерію шляхом послідовного перегляду таблиці рядок за рядком може займати багато часу. Індекс формується із значень одного або декількох стовпців таблиці і покажчиків на відповідні рядки таблиці і, таким чином, дозволяє шукати рядки, що задовольняють критерію пошуку. Прискорення роботи з використанням індексів досягається в першу чергу за рахунок того, що індекс має структуру, оптимізовану під пошук - наприклад, збалансованого дерева.

Деякі СУБД розширюють можливості індексів введенням можливості створення індексів по стовпчиках уявлень або індексів за виразами. Наприклад, індекс може бути створений за висловом upper (last_name) і відповідно буде зберігати посилання, ключем до яких буде значення поля last_name в верхньому регістрі. Крім того, індекси можуть бути оголошені як унікальні і як неунікальні. Унікальний індекс реалізує обмеження цілісності на таблиці, виключаючи можливість вставки повторюваних значень.

Існує два типи індексів: кластерні і некластерні. При наявності кластерного індексу рядка таблиці впорядковані за значенням ключа цього індексу. Якщо в таблиці немає кластерного індексу, таблиця називається купою. Некластерние індекс, створений для такої таблиці, містить тільки покажчики на записи таблиці. Кластерний індекс може бути тільки одним для кожної таблиці, але кожна таблиця може мати декілька різних некластерних індексів, кожен з яких визначає свій власний порядок проходження записів.

Індекси можуть бути реалізовані різними структурами. Найбільш часто вживані B * -дерева, B + -дерева, B-дерева і хеш-кодування.

Індекси корисні для багатьох додатків, однак на їх використання накладаються обмеження. Візьмемо такий запит SQL:

SELECT first_name FROM people WHERE last_name = 'Франкенштейн' ;.

Для виконання такого запиту без індексу СКБД повинна перевірити поле last_name в кожному рядку таблиці (цей механізм відомий як «повний перебір» або «повне сканування таблиці», в плані може відображатися словом NATURAL). При використанні індексу СУБД просто проходить по B-дереву, поки не знайде запис «Франкенштейн». Такий прохід вимагає набагато менше ресурсів, ніж повний перебір таблиці.

Тепер візьмемо такий запит:

SELECT email_address FROM customers WHERE email_address LIKE '%@yahoo.com' ;.

Цей запит повинен нам знайти всіх клієнтів, у яких е-мейл закінчується на @ yahoo.com, проте навіть якщо по стовпчику email_address є індекс, СУБД все одно буде використовувати повний перебір таблиці. Це пов'язано з тим, що індекси будуються в припущенні, що слова / символи йдуть зліва направо. Використання символу підстановки на початку умови пошуку виключає для СУБД можливість використання пошуку по B-дереву. Ця проблема може бути вирішена створенням додаткового індексу за висловом reverse (email_address) і формуванням запиту виду:

SELECT email_address FROM customers WHERE reverse (email_address) LIKE reverse ('% @ yahoo.com') ;.

В даному випадку символ підстановки виявиться в самій правій позиції (moc.oohay@%), що не виключає використання індексу по reverse (email_address).

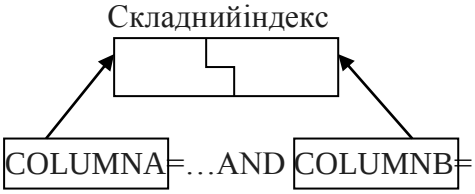
Щоб додати індекс в таблицю необхідно виконати такий код:

CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name [USING = index_type] ON table_name (index_columns)

Тут UNIQUE, FULLTEXT, SPATIAL додаються для специфічних індексів, наприклад для повнотекстового пошуку або для створення унікального індексу, просторового індексу. Якщо параметр відсутній, буде створено звичайний індекс. Вводимо назву індексу, таблицю і список полів через кому.

Складні індекси

MySQL може застосовувати тільки один індекс для запиту. Тому для запитів, в яких застосовується декількаполів, необхіднозастосовувать складні індекси.



Є такий запит:

SELECT * FROM users WHERE **age = 29 AND gender = 'male'**

Неохідно створити складнийіндекс на обидва поля:

CREATE INDEX age_gender ON users(**age, gender**);

Щоб правильно використовувати складені індекси, необхідно зрозуміти структуру їх зберігання. Все працює точно так само, як і для звичайного індексу. Але для значень використовуються значень всіх вхідних колонок відразу. Для таблиці з такими даними:

id	**name**	**age**	**gender**
1	Den	**29**	**male**
2	Alyona	**15**	**female**
3	Putin	**89**	**tsar**
4	Petro	**12**	**male**

значення складного індекса будуть такими:

age_gender
12male
15female
29male
89tsar

Це означає, що черговість колонок в індексі буде відігравати велику роль. Зазвичай колонки, які використовуються в умовах WHERE, слід ставити в початок індексу. Колонки з ORDER BY - в кінець.

Пошук по діапазону

Уявімо, що наш запит буде використовувати не порівняння, а пошук за діапазоном:

SELECT * FROM users WHERE **age <= 29** AND gender = 'male'

Тоді MySQL не зможе використовувати повний індекс, тому що значення gender будуть відрізнятися для різних значень колонки age. У цьому випадку база даних спробує використати частину індексу (тільки age), щоб виконати цей запит:

age_gender
12male
15female
29male
89tsar

Спочаткубудутьвідфільтрованівсідані, якіпідходятьпідумову age <= 29. Потім, пошук за значенням "male" буде проведений без використання індексу.

Сортування

Складові індекси також можна використовувати, якщо виконується сортування:

SELECT * FROM users WHERE gender = 'male' **ORDER BY age**

Уцьомувипадкунампотрібнобудестворитиіндексвіншомупорядку, томущосортування (ORDER) відбуваєтьсяпісляфільтрації (WHERE):

CREATE INDEX gender_age ON users(**gender, age**);

Такий порядок колонок в індексі дозволить виконати фільтрацію по першій частині індексу, а потім відсортувати результат по другій.

Колоноквіндексіможебутибільше, якщопотрібно:

SELECT * FROM users WHERE **gender = 'male' AND country = 'UA' ORDER BY age, register_time**

В цьому випадку слід створити такий індекс:
CREATE INDEX gender_country_age_register ON users(**gender, country, age, register_time**);
Використання EXPLAIN для аналізу індексів
Інструкція EXPLAIN покаже дані про використання індексів для конкретного запиту.
наприклад:
mysql> EXPLAIN SELECT * FROM users WHERE email = 'golotyuk@gmail.com';
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | **key** | key_len | ref | **rows** | Extra |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| 1 | SIMPLE | users | ALL | NULL | **NULL**| NULL| NULL| **336** | Using where |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+

Колонка key показує використовуваний індекс. Колонка possible_keys показує всі індекси, які можуть бути використані для цього запиту. Колонка rows показує число записів, які довелося прочитати базі даних для виконання цього запиту (в таблиці всього 336 записів).
Як бачимо, в прикладі не використовується жоден індекс. Післястворенняіндексу:
mysql> EXPLAIN SELECT * FROM users WHERE email = 'golotyuk@gmail.com';

+----+-------------+-------+-------+---------------+-------+---------+-------+------+-------+
| id | select_type | table | type | possible_keys | **key** | key_len | ref | **rows** | Extra |
+----+-------------+-------+-------+---------------+-------+---------+-------+------+-------+
| 1 | SIMPLE | users | const | email | **email** | 386 | const | **1** | |
+----+-------------+-------+-------+---------------+-------+---------+-------+------+-------+

Прочитана всього один запис, тому що був використаний індекс.
Перевірка довжини складових індексів
Explain також допоможе визначити правильність використання складеного індексу.
Перевіримо запит з прикладу (з індексом на колонки age і gender):
mysql> EXPLAIN SELECT * FROM users WHERE age = 29 AND gender = 'male';
+----+-------------+--------+------+---------------+------------+---------+-------------+------+-------------+
| id | select_type | table | type | possible_keys | key | **key_len** | ref | rows | Extra |
+----+-------------+--------+------+---------------+------------+---------+-------------+------+-------------+
| 1 | SIMPLE | users | ref | age_gender |**age_gender**| **24** | const,const | 1 | Using where |
+----+-------------+--------+------+---------------+------------+---------+-------------+------+-------------+

Значення key_len показує використовувану довжину індексу. У нашому випадку 24 байта - довжина всього індексу (5 байт age + 19 байт gender).
Якщо ми виконаємо змінимо точне порівняння на пошук по діапазону, побачимо що MySQL використовує тільки частину індексу:
mysql> EXPLAIN SELECT * FROM users WHERE **age <= 29** AND gender = 'male';
+----+-------------+--------+------+---------------+------------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key | **key_len** | ref | rows | Extra |
+----+-------------+--------+------+---------------+------------+---------+------+------+-------------+
| 1 | SIMPLE | users | ref | age_gender |**age_gender**| **5** | | 82 | Using where |
+----+-------------+--------+------+---------------+------------+---------+------+------+-------------+

Це сигнал про те, що створений індекс не підходить для цього запиту.
Якщожмистворимоправильнийіндекс:
mysql> Create index gender_age on users(gender, age);
mysql> EXPLAIN SELECT * FROM users WHERE age < 29 and gender = 'male';
+----+-------------+--------+-------+---------------------+------------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key | **key_len** | ref | rows | Extra |
+----+-------------+--------+-------+---------------------+------------+---------+------+------+-------------+

| 1 | SIMPLE | users | range | age_gender,gender_age | **gender_age** | **24** | NULL | 47 | Using where |
+----+-------------+--------+-------+----------------------+------------+---------+------+------+-------------+

В цьому випадку MySQL використовує весь індекс gender_age, тому що порядок колонок в ньому дозволяє зробити цю вибірку.

селективність індексів

Повернемося до запиту:

SELECT * FROM users WHERE age = 29 AND gender = 'male'

Для такого запиту необхідно створити складовий індекс. Але як правильно вибрати послідовність колонок в індексі? Варіанта два:

• age, gender

• gender, age

Підійдуть обидва. Але працювати вони будуть з різною ефективністю.

Коли створювати індекси?

• Індекси слід створювати в міру виявлення повільних запитів. У цьому допоможе slow log в MySQL. Запити, які виконуються більш 1 секунди є першими кандидатами на оптимізацію.

• Починайте створення індексів з найбільш частих запитів. Запит, що виконується секунду, але 1000 разів в день завдає більше шкоди, ніж 10-секундний запит, який виконується кілька разів на день.

• Не створюйте індекси на таблицях, число записів в яких менше кількох тисяч. Для таких розмірів виграш від використання індексу буде майже непомітний.

• Не створюйте індекси заздалегідь, наприклад, в середовищі розробки. Індекси повинні встановлюватися виключно під форму і тип навантаження працюючої системи.

• Видаляйте невживані індекси.

Для роботи з MySQL ви можете застосувати PHPMyAdmin або інші безкоштовні програми. Щоб створити індекс за допомогою PHPMyAdmin, необхідно зайти в структуру таблиці і напроти потрібного стовпчика натиснути на список, що випадає. Там вам буде запропоновано додати PRIMARY KEY, INDEX, UNIQUE, FULLTEXT. Тут я перерахував основні індекси, які слід використовувати. Зазвичай первинний ключ я призначаю для порядкового номера запису в таблиці, унікальний індекс для ідентифікатора запису, а повнотекстовий для самого запису, щоб організувати пошук. Для використання FULLTEXT індексу таблиці повинні бути MyISAM.

Завдання:

Згідно теми індивідуального завдання (додаток 1) в таблиці БДвиконайте:

1. Створіть простий індекс.
2. Створіть складний індекс.
3. Створіть унікальний індекс.
4. Створіть індекс для повнотекстового пошуку.

Самостійна робота №20

Тема: Методологія функціонального моделювання.

Функціональна модель системи будується на основі функціона- льної діаграми. Крім чисто функціональних діаграм IDEF0 ця модель може включати діаграми, орієнтовані на дані, а саме DFD та IDEF3. От же, до складу функціональної моделі можуть входити такі діаграми:

- функціональні діаграми IDEF0;
- діаграми потоків даних потоків даних DFD (DataFlowDiagramming);
- діаграми опису послідовності процесів IDEF3 (WorkFlowDiagramming);
- діаграма дерева вузлів функціональної моделі (NodeTreeDiagramming).

У функціональній моделі діаграми IDEF0 відіграють головну роль. Діаграми DFD (потоків даних) і IDEF3 (опису послідовності про- цесів), як правило, доповнюють модель на нижніх рівнях декомпозиції, хоча вони можуть мати самостійне значення і будуватись як самостійні діаграми, починаючи з верхнього рівня. Діаграма Node Tree (дерева вуз- лів) просто демонстраційна, вона показує модель у загальному вигляді. Серед названих діаграм у першу чергу розглянемо функціональну діаграму IDEF0. Сукупність таких діаграм складає функціональну модель. Остання призначена для аналізу функціонування технічних та ор- ганізаційних систем. Вона відображає процеси роботи системи, взаємо- дію її частин у процесі функціонування.

Область використання даної моделі є надзвичайно широкою. Од- ним з напрямків її використання є аналіз та удосконалення бізнес- процесів на підприємствах і розробка методів покращання ефективності їх роботи. Функціональна модель дозволяє виконати де- тальний аналіз роботи системи, її функціонально-вартісний аналіз, розглянути і проаналізувати напрямки удосконалення роботи. Якщо розглядати тільки технічні системи, то дана модель дозволяє вирішити проблеми раціонального проектування складних технічних систем, зробити їх найбільш дешевими, простими, функціонально спрямованими.

Для покращання ефективності роботи підприємства необхідно зібрати знання великої кількості людей в одному місці, в одній моделі. Це дозволяє зробити функціональна мо- дель IDEF0 за методикою, затвердженою стандартами.

Модель являє собою деревовидну топологічну структуру і створюється на основі функціональної декомпозиції цілей та задач системи. Пояснимо, що це означає. Спочатку функції системи описуються в цілому без деталей. Цьому опису відповідає так звана контекстна діаграма. Вона має багато спільного з моделлю системи типу “Чорний ящик”, тільки в ній вказується не назва системи, а головна її функція (ціль, завдання). У подальшому контекстна діаграма піддається функціональній декомпозиції. У результаті останньої головна функція 123 системи, описана в контекстній діаграмі, розбивається на підфункції (ціль - на підцілі, завдання - на підзавдання). Потім кожна підфункція розбивається на більш дрібні підфункції і т. д. до досягнення найбільш простого ступеня деталізації. Результатом функціональної декомпозиції є розбиття функцій системи на елементарні процеси з точки зору аналі- зу, які можуть бути описані простими специфікаціями. Результат функ- ціональної декомпозиції може бути зображений у вигляді ієрархічної моделі, яка має назву дерева вузлів функціональної моделі (Node Tree Diagramming).. Дерево вузлів є каркасом функціональної моделі, але не самою функціональною моделлю. Ця діаграма показує загальний склад моделі. Вигляд її зображений на рис.1

Контекстна діаграма зображається прямокутником з вхідними й вихідними величинами. На відміну від моделі типу “Чорний ящик” у прямокутнику контекстної діаграми вказується не назва системи, а її основна функція (ціль системи). Входи й виходи контекстної діаграми розподілені не по двох, а по чотирьох сторонах прямокутника.

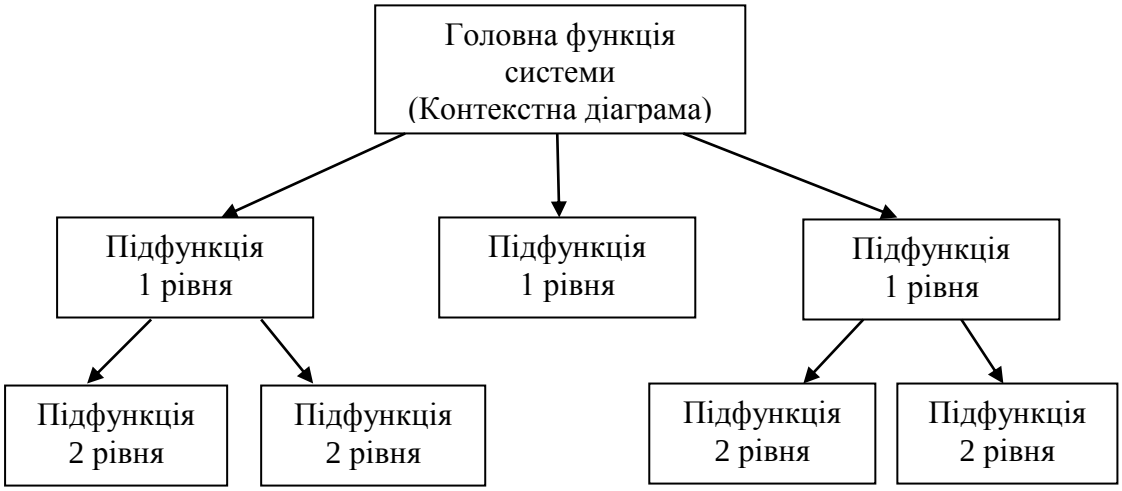


Рисунок 1 - Приклад дерева вузлів функціональної моделі

Призначення цих сторін такі:

• Ліва сторона відповідає входам системи (input), величинам, які поступають у систему і переробляються нею у вихідні величини.

• Верхня сторона відповідає входам по керуванню (control), тобто різним керуючим діям, командам, стратегіям поведінки, процедурам, документам, що регламентують виконання роботи тощо. Ці величини не змінюються, а служать тільки для керування.

• Права сторона відповідає виходам системи (output), продуктам її діяльності, результатам перетворення вхідних величин, шкідливим виділенням, відходам тощо.

• Нижня сторона відповідає механізмам (mechanism), а саме засобам, ресурсам, за допомогою яких виконуються вказані в прямокутнику функції.

Функціональна модель є подальшою деталізацією (декомпозицією) контекстної діаграми. Спочатку функція системи в цілому (ціль, призначення, головне завдання) розбивається на декілька окремих функцій (завдань, робіт, цілей). Таких функцій рекомендується вибирати від 2 до 6. Ці функції (їх деколи називають роботами) (activity) зобража- ються на окремому аркуші декомпозиції у вигляді функціональних блоків. Кожен функціональний блок (робота), яким виступає окрема функція (робота, ціль чи завдання), зображується прямокутником. Сторони прямокутників робіт (функціональних блоків) мають таке ж призначення, що й розглянуті вище сторони контекстної діаграми. Між окремими функціональними блоками встановлюють зв’язки, що відповідають логіці функціонування системи. Зв’язки між функціональними блоками зображуються стрілками (Arrow) (часто їх називають дугами). Кожна дуга (стрілка) відповідає передачі від блоку до блоку якогось конкрет ного об’єкта (предмета, речовини, документа, а інколи і усного розпо- рядження) чи їх сукупності.

Перед визначенням і описом порядку побудови моделі слід зробити декілька зауважень: 1. Всяка SADT модель вимагає точного визначення границь сис- теми, цілей моделювання, точки зору, контексту розгляду системи. 128 2. SADT моделі вимагають, щоб система розглядалась весь час з однієї точки зору, оскільки зміна точки зору може зробити модель неа- декватною. Точка зору диктує підбір потрібної інформації та спосіб її подачі. 3. SADT моделі будуються з верхнього рівня “з голови”. У них діаграми нижчого рівня є деталізованими діаграмами верхнього рівня. Кінцевий результат це ієрархічна структура діаграм. На початку побудови функціональної моделі необхідно чітко ви- значити: - границі системи, - ціль моделювання, - контекст розгляду системи - точку зору, тобто відповісти на запитання, які є вихідними усякого системного аналізу. Ціль є напрямком діяльності і критерієм закінченості моделі. Ви- значаючи ціль, слід розрізняти: - ціль системи, для якої будують функціональну модель і - ціль моделювання. Ціль системи визначає зміст моделі її конкретне наповнення. Всяку модель ми будуємо відповідно до її цілі, до завдань, які виконує система. Ціллю моделювання може бути виконання певного завдання, ви- вчення

системи, розробка та удосконалення системи і т.п. Ця ціль ви- значає обсяг роботи, потрібний ступінь деталізації системи, розміри моделі в глибину і ширину. Точка зору визначається тим, з погляду якої особи ми розглядаємо модель, точка зору якої особи найбільш повно відповідає цілі моделювання. Усяка модель повинна бути цілісною і розглядатися з однієї точки зору. Різні точки зору приводять до різних моделей, а зміна точки зору в ході моделювання не допускається, оскільки може призвести до невірного опису системи, до неадекватної моделі. Наприклад, система "Тролейбусне депо". Модель системи виглядатиме по–різному, якщо розглядати з різних точок зору. Для директора головними елементами є робочий колектив, підрозділи депо, виконання ними своїх функцій, оде- ржання прибутку та оплата праці. Для головного інженера в основі знаходяться технологічні процеси роботи депо, їх технічне забезпечення, технічний рівень виконання операцій, використання наукових досягнень, технічний стан обладнання. Для економіста найбільш важливою є економічна ефективність роботи підрозділів, наявність фінансових ресурсів, одержання прибутку від діяльності. Для начальника служби охорони праці найбільш важливими є умови праці кожного робітника, обладнання робочих місць засобами захисту, організація профілактичної роботи по дотриманню правил охорони праці і т.п. Отже, всяка система виглядає по-іншому з різних точок зору. Модель системи повинна виконуватися з однієї точки зору. Зміна точки зору в ході моделювання приводить до змішування і створення еклектичної моделі, яка не може дати відповіді на жодне запитання. Для повного опису системи інколи виникає потреба розглядати її з різних точок зору, але в такому разі необхідно будувати декілька моделей. У кожній моделі система розглядається тільки з однієї точки зору.

Завдання:

За темою індивідуального завдання (додаток 1) побудувати функціональну схему системи.

Самостійна робота №21

Тема:Інформаційні системи в мережах.

Розподілена база даних (англ. *distributeddatabase, DDB*) — сукупність логічно взаємопов'язаних баз даних, розподілених у комп'ютерній мережі. Логічний зв'язок баз даних в розподіленій базі даних забезпечує система управління розподіленою базою даних, яка дозволяє управляти розподіленою базою даних таким чином, щоб створювати у користувачів ілюзію цілісної бази даних.

Система управління розподіленою базою даних складається з (можливо, порожнього) набору вузлів прийому запитів і набору вузлів збереження даних. Вузли прийому запитів реалізують прозорий інтерфейс доступу до даних, що зберігаються в вузлах збереження даних, та приховують фрагментацію даних між вузлами. Кожен з вузлів може бути представлений незалежним комп'ютером в комп'ютерній мережі з власною (можливо відмінною від інших вузлів) операційною системою.

Система управління розподіленою базою даних є однорідною (гомогенною), якщо на кожному з вузлів збереження даних застосовуються однакові СКБД, в іншому випадку система управління розподіленою базою даних є неоднорідною (гетерогенною).

Розподіл даних між вузлами збереження даних забезпечується на основі механізмів фрагментації та реплікації, і досягається шляхом вертикального (на окремі поля записів бази даних) або горизонтального (на окремі записи бази даних) поділу даних. Наприклад, при вертикальному поділі даних інформація про покупців може зберігатись на одному вузлі збереження даних, про їх купівлі — на другому, про товари — на третьому тощо; при горизонтальному поділі даних інформація про покупців, купівлі та товари зберігається на одному вузлі, а поділ між вузлами здійснюється за країнами покупців (для кожної країни — окремий вузол збереження даних), або за типами товарів тощо. Фрагментація здійснюється за принципами, що дозволяють наблизити дані до місць їх найбільш інтенсивного використання для зменшення витрат на пересилання даних.

Сутьфрагментаціїполягає в тому, щоб поділити логічну базу даних на фрагменти з метою зберігання кожного фрагмента на певному вузлі мережі. Одиницями фрагментації можуть бути відношення та складені відношення. У випадку, коли одиницею фрагментації є відношення, вирішується проблема, яке відношення в якій базі даних має зберігатися. За іншого підходу допускається, що будь-яке відношення може бути зображене у вигляді сукупності фрагментів, що розподіляються за різними базами даних.

Реплікаціяє механізмом розподілу даних за вузлами, що в свою чергу дозволяє зберігати копії тих самих даних на різних вузлах мережі для прискорення пошуку і підвищення стійкості до відмов. Відношення або фрагмент є реплікованим, у випадку якщо його копії(або репліки) зберігаються на двох або більше вузлах . За повної реплікації відношення його копії зберігаються на всіх вузлах мережі. Допускається ситуація, коли вся база даних зберігається на всіх вузлах мережі — це називається повною реплікацією бази даних.

Доступ до даних в розподіленій базі даних звичайно забезпечується в трирівневій моделі: клієнт — сервер додатків — вузли збереження даних. При інтернет-доступі до даних роль сервера додатків відіграє веб-сервер або спеціальний додаток на боці клієнта.

Для вирішення спеціальних задач обслуговування даних можливий локальний доступ до окремих вузлів збереження даних, при цьому недоступні можливості доступу, що базуються на прозорості інтерфейсів доступу до даних.

12 властивостей розподілених баз даних, були сформульовані Крістофером Дейтом, одним з найбільших діячів в галузі баз даних.В них виділяють:

- **Локальна автономія** - управління даними на кожному з вузлів розподіленої системи виконується локально.
- **Незалежність вузлів** - всі вузли рівноправні і незалежні, а розташовані на них БД є рівноправними постачальниками даних в загальний простір даних.

- **Безперервні операції** - можливість безперервного доступу до даних в рамках розподіленої БД незалежно від їх розташування і незалежно від операцій, що виконуються на локальних вузлах.
- **Прозорість розташування** - користувач, що звертається до БД, нічого не повинен знати про реальне, фізичне розміщення даних у вузлах інформаційної системи.
- **Прозора фрагментація** - можливість розподіленого (тобто на різних вузлах) розміщення даних, логічно поєднаних в єдине ціле. Існує фрагментація двох типів: горизонтальна і вертикальна.
- **Прозоре тиражування** - тиражування даних - це асинхронний процес перенесення змін об'єктів вихідної бази даних в бази, розташовані на інших вузлах розподіленої системи
- **Обробка розподілених запитів** - можливість виконання операцій вибірки даних з розподіленої БД, за допомогою запитів, сформульованих на мові SQL
- **Обробка розподілених транзакцій** - можливість виконання операцій оновлення розподіленої бази даних, які не порушують цілісність і узгодженість даних.
- **Незалежність від устаткування** - як вузли розподіленої системи можуть виступати ПК будь-яких моделей і виробників
- **Незалежність від операційних систем** - різноманіття операційних систем, керуючих вузлами розподіленої системи
- **Прозорість мережі** - спектр підтримуваних конкретною СУБД мережевих протоколів не має бути обмеженням системи, заснованої на розподіленій БД
- **Незалежність від баз даних** - в розподіленій системі можуть працювати СУБД різних виробників, і можливі операції пошуку і оновлення в базах даних різних моделей і форматів.

Завдання:

Дайте відповіді на запитання:

Що означає поняття розподіленої бази даних?

Що забезпечує система управління розподіленою базою даних?

З чого складається система управління розподіленою базою даних?

Яка система називається однорідною, а яка гетерогенною?

Яким чином забезпечується розподіл даних між вузлами збереження даних?

Що означає реплікація?

В чому полягає суть фрагментації?

Які властивості розподілених БД?

Додаток №1

Теми індивідуальної роботи

№	Назва, зміст та обсяг завдань для індивідуальної роботи	
1	Склад бази даних «Інститут»: таблиця «НДІ», таблиця «Кадри», таблиця «Довідник відділів», таблиця «Довідник тем». Структура таблиці «НДІ»: номер відділу; табельний номер; номер теми, над якою працює працівник; тривалість роботи над темою (у місяцях).**Примітка:** Одну тему можуть розробляти працівники кількох відділів. Структура таблиці «Кадри»: номер відділу; табельний номер; прізвище та ініціали; код посади; розмір зарплати. Структура таблиці «Довідник відділів»: номер відділу; назва відділу; табельний номер керівника. Структура таблиці «Довідник тем»: номер теми; назва теми; дата початку розробки; дата завершення роботи; табельний номер керівника. Використовуючи інформаційно-логічну модель: а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. Використовуючи зазначену базу даних тазасоби мови SQL: 1.Визначити теми (у порядку зростання їх номерів), що розробляються в НДІ, кількість зайнятих працівників та суми коштів, які витрачені на зарплату. 2. Отримати інформацію про працівників, зайнятих у розробці тем різних відділів, та визначити загальну суму зарплати, яка їм сплачується. 3.Отримати інформацію про чисельність працівників у відділах та працюючих над окремими темами у розрізі посад. 4. Отримати інформацію про середній заробіток працівників різних відділів. 5.Визначити обсяг робіт (кількість людино-місяців) за темами та відділами. 6. Отримати інформацію про тривалість роботи працівників за темами. 7.Визначити в цілому по НДІ й окремо по кожній темі чисельність зайнятих працівників та суму коштів, витрачених на зарплату. 8.Визначити чисельність працівників кожного відділу у розрізі посад, які вони займають. 9.Визначити загальну тривалість роботи над кожною темою (у людино-місяцях) з виділенням даних по відділах та в цілому по НДІ. 10.Скласти список працівників по відділах, тривалість роботи яких над темою не перевищує заданої користувачем величини, та підрахувати їх кількість по відділах та НДІ в цілому. Примітка: Тут і далі «*» означає спрощений варіант задачі.	

2	Склад бази даних «Матеріали»: таблиця «Замовлення», таблиця «Ліміт», таблиця «Довідник підприємств», таблиця «Прейскурант». Структура таблиці «Замовлення»: дата замовлення; код підприємства; номенклатурний номер матеріалу, який замовляється. Примітка: У таблиці містяться відомості за поточний місяць. Структура таблиці «Ліміт»: номенклатурний номер матеріалу; розмір ліміту на місяць. Структура таблиці «Довідник підприємств»: код підприємства; назва. Структура таблиці «Прейскурант»: номенклатурний номер матеріалу; назва матеріалу; код одиниці вимірювання; ціна за одиницю. ***Використовуючи інформаційно-логічну модель:*** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL*** 1. Отримати інформацію про потреби підприємства в матеріалах. 2. Визначити сумарну потребу в матеріалах по кожному номенклатурному номеру та співвідношення потреби і ліміту. 3.Отримати інформацію по підприємствах, які надіслали замовлення на визначені види матеріалу в конкретний період часу. 4. Отримати інформацію про задоволення замовлень підприємств. 5. Визначити кількість замовлень, що надійшли від підприємств на визначені види матеріалів в різні періоди часу. 6. Визначити сумарну потребу в матеріалах кожного підприємства та загальну потребу всіх підприємств по кожному номенклатурному номеру. 7. Визначити перелік підприємств, які надіслали замовлення не пізніше дати, що цікавить користувача. 8. Визначити кількість замовлень, які надійшли від кожного підприємства на визначений матеріал за звітний період та загальну кількість замовлень на нього.	
3	Склад бази даних «Торгівля»: таблиця «Товар», таблиця «Магази6н», таблиця «Прейскурант». Структура таблиці «Товар»: номер магазину, номер секції; номер чека; дата продажу; артикул товару; кількість одиниць товару; ціна за одиницю; загальна вартість. Примітка: У таблиці містяться відомості за звітний місяць, реквізити, ціна за одиницю та загальна вартість. Вони використовуються для контролю за введенням даних. Структура таблиці «Магазин»: номер магазину; назва магазину. Структуратаблиці «Прейскурант»: артикул товару; назва товару; одиниця вимірювання; ціна за одиницю.	

	Використовуючи інформаційно-логічну модель: а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** Визначити обсяг товарообігу магазинів за відповідний період часу. Визначити обсяг товарообігу секцій магазинів за відповідний період часу. 3.Визначити виручку від реалізації відповідного виду товару в різних магазинах. 4.Отримати інформацію про обсяг денної виручки магазинів, 5.Отримати інформацію про динаміку денної виручки магазинів протягом відповідного періоду (відрізку) часу. 6.Визначити магазини з найкращими та найгіршими показника ми обсягу товарообігу. 7.Отримати інформацію про секції магазинів, які забезпечать найбільший та найменший обсяг реалізації відповідного виду товару. 8.Отримати інформацію про виконання плану товарообігу магазинами. 9.Отримати інформацію про магазини, які перевиконали та не виконали план товарообігу. 10.Визначити обсяг товарообігу по управлінню в цілому та по кожному магазину за період, що минув. 11.Визначити обсяг товарообігу магазинів в цілому та по кожній секції за відповідний день. 12.Визначити обсяг продажу відповідних товарів по кожному магазину та по всіх магазинах в цілому. 13.Визначити магазини, які забезпечують найбільший і найменший обсяг денної виручки та обсяг виручки по кожній секції в цих магазинах. 14. Визначити виконання плану (у відсотках) по кожному магазину та в цілому по управлінню.	
4	Склад бази даних «Фабрика»: таблиця «Замовлення», таблиця «Фабрика», таблиця «Потужність». Структура таблиці «Замовлення»: номер фабрики-пральні; номер квитанції; дата прийому; дата виконання; вага; вартість. **Примітка:**У таблиці містяться відомості за поточний місяць. Структура таблиці «Фабрика»: номер фабрики-пральні; район міста; адреса. Структура таблиці «Потужність»: номер фабрики-пральні; пропускна спроможність (кг за добу). ***Використовуючи інформаційно-логічну модель:*** а) Побудувати та заповнити даними таблиці бази даних, що	

	відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** 1. Отриматиінформацію про порушення термінів виконання замовлень. 2. Визначити обсяг виконаних робіт. 3.Визначити дні, у які фабрика має найбільшу і найменшу завантаженість. 4. Отримати інформацію про фабрики з максимальною та мінімальною кількістю порушень термінів виконання замовлень. 5. інформацію про використання потужності фабрик. 6.Визначити кількість порушень термінів виконання замовлень по кожній фабриці та загальну кількість по всіх фабриках міста. 7.Визначити перелік фабрик, на яких кількість порушень термінів виконання замовлення не перевищує заданої величини. 8.Визначити обсяг денного приймання кожної фабрики та кількість фабрик, на яких цей обсяг не досягає визначеної величини.	
5	Склад бази даних «Виробництво»: таблиця «Наряд», таблиця «Довідник кадрового складу», таблиця «Довідник операцій», таблиця «Довідник цехів». Структура таблиці «Наряд»: номер наряду; номер цеху; номер ланки; табельний номер робітника; код операції; розряд роботи; кількість виготовлених деталей; кількість прийнятих деталей. **Примітка:** У таблиці містяться відомості за поточний місяць. Структура таблиці «Довідник кадрового складу»: номер цеху; табельний номер робітника, прізвище та ініціали. Структура таблиці «Довідник операцій»: код операції; назва операції; норма часу на виготовлення деталі (хв.). Структура таблиці «Довідник цехів»: номер цеху; назва цеху. ***Використовуючи інформаційно-логічну модель:*** а) Побудувати та заповнити даними таблиці бази даних, щовідповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** 1. Отримати інформацію про обсяг браку на виробництві. 2. Отримати інформацію про рівень браку на різних операціях. 3. Отримати інформацію про результати роботи конкретного робітника. 4. Отримати інформацію про втрати часу, пов'язані з виробництвом бракованих деталей. 5. Отримати інформацію щодо робітників, які допустили випуск бракованих деталей на визначених операціях.	

	6. Визначити перелік операцій в кожному цеху, на яких допущено випуск найбільшої чи найменшої кількості бракованих деталей. 7. Визначити кількість бракованих деталей в цілому по заводу та по кожному цеху. 8. Визначити кількість бракованих деталей, випущених робітниками, табельні номери яких знаходяться в заданому інтервалі. 9. Скласти список робітників, які допустили випуск бракованих деталей на операції із зазначеним кодом по всіх цехах, де ця операція виконується.	
6	Склад бази даних «Простої»: таблиця «Простій», таблиця «Робітник», таблиця «Довідник цехів», таблиця «Довідник причин простою». Структура таблиці «Простій»: номер документа; дата; номер цеху; номер ланки; табельний номер робітника; код причини і винуватця простою; % оплати; тривалість простою (хв.). Примітка: У таблиці містяться відомості за поточний місяць. Структура таблиці «Робітник»: код цеху; табельний номер; прізвище та ініціали; середня заробітна плата за місяць. Структура таблиці «Довідник цехів»: номер цеху; назва цеху. Структура таблиці «Довідник причин простою»: код причини чи винуватця простою; назва. Використовуючи інформаційно-логічну модель: а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквізита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. Використовуючи зазначену базу даних та засоби мови SQL: 1. Отримати інформацію про тривалість простоїв у цехах. 2.Отримати інформацію про тривалість простоїв за кодами причин та винуватців. 3.Отримати інформацію про простої, що перевищили відповідну величину. 4.Отримати інформацію про найбільшу та найменшу тривалість простоїв. 5.Отримати інформацію про тривалість простоїв окремих робітників. 6.Визначити величину втрат, викликаних оплатою простоїв. 7.Отримати інформацію про динаміку простоїв протягом відповідного періоду часу. 8.Визначити питому вагу простоїв у загальному фонді робочого часу. 9.Визначити тривалість простоїв в цілому по заводу та по кожному цеху. 10.Визначити тривалість простоїв по кожному цеху за кодами причин та винуватців.	

	11.Скласти список цехів, у яких тривалість простоїв за день не перевищує задану величину. 12.Визначити ділянки кожного цеху, на яких тривалість простоїв була найбільшою чи найменшою. 13.Визначити суму втрат, викликаних оплатою простоїв по кожному цеху та заводу в цілому.	
7	Склад бази даних «Студент»: таблиця «Студент», таблиця «Довідник факультетів». Структура таблиці «Студент»: код факультету; курс; вік; код статі; код родинного стану; середній дохід сім'ї; час на дорогу до інституту (хв.). Структура таблиці «Довідник факультетів»: код факультету; назва факультету. Використовуючи інформаційно-логічну модель: а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквізита та побудувати схему даних,яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. Використовуючи зазначену базу даних та засоби мови SQL: 1. Отримати інформацію про час (в середньому), якийстуденти витрачають на дорогу до університету. 2. Отримати інформацію про віковий склад студентів. 3. Отримати інформацію про чисельний склад студентів. 4. Визначити середній дохід в сім'ях студентів. 5. Отримати інформацію про студентів, середнійприбуток в сім'ях яких не досягає заданої користувачемвеличини. 6.Отримати інформацію про студентів, вік яких перевищує заданий користувачем рівень. 7.Отримати інформацію про сімейний стан студентів. 8.Визначити середній дохід студентів, які перебувають у шлюбі. 9.Визначити чисельний склад студентів, які перебувають ушлюбі. 10.Визначити віковий склад студентів, які перебувають у шлюбі. 11.Визначити питому вагу студенток у загальній кількості студентів. 12.Отримати інформацію про середній дохід в сім'ях студентів, які належать до різних вікових груп. 13.Отримати інформацію про співвідношення студентів та студенток, які перебувають у шлюбі. 14.Визначити середній час, який витрачають студенти кожного факультету на дорогу до університету. 15.Визначити віковий склад студентів у цілому по університету та по кожному факультету за віковими групами: до 25 років; 25-30 років; після 30 років. 16.Визначити кількість студентів та студенток на кожному факультеті та по університету в цілому.	

	17.Визначити середній дохід в сім'ях студентів на кожному факультеті з виділенням даних по курсах. 18.Визначити кількість студентів на кожному курсі факультету, які мають мінімальний та максимальний вік. 19.Визначити кількість студентів, що перебувають у шлюбі, по університету в цілому та по кожному факультету. 20.Визначити віковий склад студентів, які перебувають у шлюбі, по кожному факультету та курсу. 21.Визначити середній дохід студентів, які перебувають у шлюбі, по кожному курсу університету. 22.Визначити чисельність студентів та студенток на кожному курсі та по університету в цілому (за віковими групами: до 20 років; 20—25 років; після 25 років). 23.Визначити середній дохід студентів різних вікових груп по університету (за віковими групами: до 20 років; 20-25 років; після 25 років).	
8	Склад бази даних «Обстеження»: таблиця «Анкета», таблиця «Довідник рівнів освіти». Структура таблиці «Анкета»: рік обстеження; код групи обстеження; код рівня освіти; код родинного стану; заробітна плата; кількість дітей. Структура таблиці «Довідник рівнів освіти»: код рівня освіти; повна назва. ***Використовуючи інформаційно-логічну модель:*** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б)визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі нарахованої і виданої кожному працівникові цеху, який цікавить користувача. 3.Отримати інформацію про суму прибуткового податку, який утримано із заробітної плати працюючих. 4. Отримати інформацію про питому вагу прибуткового податку в загальній сумі утримань. 5. Отримати інформацію про динаміку заробітної плати. 6 Отримати інформацію про питому вагу суми утримань в нарахованій заробітній платі. 7 Визначити межові значення нарахованої заробітної плати. 8 Отримати інформацію про працюючих, що мають заробітну плату (суму до виплати) нижче відповідної величини. 9 иОтримати інформацію про питому вагу робітників, що перераховують заробітну плату до ощадних кас, та про суму перерахувань. 10 Визначити суму зарплати, виданої працюючим кожного цеху за місяць. 11 Визначити заробітну плату, нараховану і виплачену кожному	

	робітникові цеху, що цікавить користувача, та загальну суму по цеху. 12 Визначити суму прибуткового податку, утриманого із зарплати працюючих в цілому по підприємству та по кожному цеху. 13 Визначити питому вагу суми утримань в нарахованій зарплаті кожного робітника цеху, що цікавить користувача, та в цілому по цеху. 14. Визначити найбільшу і найменшу суму зарплати, нарахованої по кожному цеху та по підприємству в цілому. 15. Скласти список робітників по кожному цеху, що мають зарплату менше вказаної величини, та визначити їх кількість по кожному цеху.	
9	Склад бази даних «Успішність»: таблиця «Сесія», таблиця «Довідник факультетів», таблиця «Довідник спеціальностей», таблиця «Довідник дисциплін». Структура таблиці «Сесія»: код факультету; код спеціальності; курс; група; номер залікової книжки; код дисципліни; оцінка. Таблиці «Довідник факультетів», «Довідник спеціальностей», «Довідник дисциплін» мають однакову структуру: код реквізиту; назва реквізиту. ***Використовуючи інформаційно-логічну модель:*** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** 1. Отримати інформацію про кількість відмінних та незадовільних оцінок, виставлених в період сесії. 2. Отримати інформацію про студентів-відмінників та про їх кількість. 3. Отримати інформацію про студентів, які мають дві і більше оцінок «незадовільно» за результатами сесії. 4.Визначити питому вагу студентів, що мають незадовільні оцінки, в загальній кількості студентів. 5. Отримати інформацію про дисципліни, з яких на іспитах було виставлено найбільшу і найменшу кількість оцінок «відмінно» та «добре». 6. Отримати інформацію за результатами сесії конкретних студентів. 7. Визначити кількість оцінок «відмінно» і «незадовільно», отриманих в період сесії в цілому по університету та з кожної спеціальності. 8. Скласти список студентів-відмінників з кожної спеціальності та визначити кількість студентів-відмінників в цілому по університету. 9. Визначити кількість студентів, що мають дві і більше оцінок «незадовільно» за результатами сесії, з кожної спеціальності з переліком прізвищ студентів та дисциплін, з яких отримані	

	незадовільні оцінки, а також визначити загальну кількість таких студентів в цілому по університету. 10. Визначити питому вагу студентів, які мають оцінки «незадовільно» в загальній кількості студентів з кожної спеціальності та в цілому по університету. 11. Визначити дисципліни, з яких на іспитах було виставлено найбільшу і найменшу кількість оцінок «відмінно» та «добре».	
10	Склад бази даних «Виробництво»: таблиця «Напої», таблиця «Довідник заводів», таблиця «План». Структура таблиці «Напої»: дата звітного періоду; код заводу; обсяг виробництва: соків; газових напоїв; мінеральної води; квасу. Структура таблиці «Довідник заводів»: код заводу; назва заводу. Структура таблиці «План»: код заводу; плановий обсяг виробництва всього, у тому числі: соків; газових напоїв; мінеральної води; квасу. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквізита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Отримати інформацію про обсяг виробництва безалкогольних напоїв. 2. Отримати інформацію про питому вагу виробництва тих напоїв, що цікавлять користувача, в загальному обсязі виробництва. 3.Отримати інформацію про виконання плану щодо виробництва безалкогольних напоїв. 4. Отримати інформацію про заводи, на яких обсяг виробництва того напою, що цікавить користувача, не перевищує заданої величини. 5. Отримати інформацію про динаміку виробництва безалкогольних напоїв. 6. Отримати інформацію про максимальний та мінімальний рівень виробництва безалкогольних напоїв. 7. Визначити обсяг виробництва кожного виду напою заводами міста та загальний обсяг виробництва безалкогольних напоїв. 8. Визначити питому вагу виробництва кожного виду напою в загальному обсязі виробництва по місту. 9. Визначити заводи, які досягли максимального виробництва кожного виду напою. 10. Визначити перелік заводів, на яких обсяг виробництва мінеральної води не перевищує заданої величини.	
11	Склад бази даних «Кадри»: таблиця «Кадрові відомості», таблиця «Довідник підрозділів», таблиця «Довідник категорій», таблиця	**11**

	«Довідник посад», таблиця «Довідник рівнів освіти». Структура таблиці «Кадрові відомості»: код підрозділу: табельний номер робітника; прізвище та ініціали; код категорії; код посади; розряд; дата прийому на завод; загальний стаж роботи; стаж роботи на заводі; рік народження; код статі; код рівня освіти; код родинного стану; домашня адреса; середня заробітна плата. Таблиці «Довідник підрозділів», «Довідник категорій», «Довідник посад», «Довідник рівня освіти» мають однакову структуру: код реквізиту; назва реквізиту. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Отримати інформацію про прийняття робітників на завод. 2.Отримати інформацію про освітній рівень робітників. 3.Отримати інформацію про стаж працюючих. 4.Отримати інформацію про сімейний стан працюючих. 5.Отримати інформацію про склад посад працюючих. 6.Отримати інформацію про віковий склад працюючих. 7.Отримати інформацію про місячний фонд заробітної плати. 8.Отримати інформацію про співвідношення працюючих різних категорій. 9.Отримати інформацію про межові значення розмірів середньої заробітної плати. 10.Отримати інформацію про чисельність працюючих в структурних підрозділах. 11.Отримати інформацію про відповідність освітнього рівня працюючих займаним посадам. 12.Отримати інформацію про працюючих пенсіонерів. 13.Отримати інформацію про "середню заробітну плату різних вікових груп працюючих. 14.Отримати інформацію про чисельність працюючих чоловіків та жінок по різних категоріях.	
12	Склад бази даних «Навчальний процес»: таблиця «Практика», таблиця «Довідник факультетів», таблиця «Довідник дисциплін», таблиця «Довідник рівня завершеності». Структура таблиці «Практика»: код факультету; курс; код залікової книжки студента; код дисципліни; номер лабораторної роботи з дисципліни; дата видачі завдання; термін завершення роботи; код рівня завершеності; кількість виходів на машину. Таблиці «Довідник факультетів», «Довідник дисциплін», «Довідник рівня завершеності» мають однакову структуру: код реквізиту; назва реквізиту. **Використовуючи інформаційно-логічну модель:**	**11**

	а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) визначити ключові реквизита та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Отримати інформацію про рівень завершеності визначеної лабораторної роботи. 2. Отримати інформацію про середню кількість виходів на машину під час виконання визначеної лабораторної роботи. 3.Отримати інформацію про питому вагу налагоджених програмна певну дату. 4. Отримати інформацію про завантаженість студентів лабораторними роботами. 5. Отримати інформацію про хід виконання графіка лабораторних робіт. 6. Визначити кількість студентів у кожній групі по курсу, заданому користувачем, які виконали чергову лабораторну роботу й підготували текст програми до наступної роботи. 7. Визначити середню кількість виходів на машину студентів певної групи під час виконання лабораторної роботи з дисципліни, що цікавить користувача. 8. Визначити кількість лабораторних робіт, які виконуються студентами факультету, що цікавить користувача, за певний період. 9. Визначити кількість лабораторних робіт, які виконуються на ЕОМ студентами певного факультету на кожному курсі.	
13	Склад бази даних «Моя колекція музики»: таблиця «Групи», таблиця «Альбоми», таблиця «Треки». Структура таблиці «Групи»: група (ключове поле); кількість альбомів; країна; стиль. Структура таблиці «Альбоми»: альбом; група; число записів; час звучання; рік випуску; примітки; ідентифікатор (ключове поле). Структура таблиці «Треки»: номер запису; трек; альбом; час звучання; Kbps (Кб/с); частота дискретизації (Гц); якість; розмір; формат файлу; замітки; ідентифікатор; ідентифікатор альбому. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:**	

	1. Вивести інформацію по всіх групах. 2. У визначеному альбомі визначити найкоротший трек. 3. Вивести інформацію про те, якій групі належить який альбом. 4. Вивести інформацію про групи. 5. Вивести інформацію про «наймолодший» альбом. 6. Вивести інформацію про «найстарший» альбом. 7. Визначити загальну довжину звучання вказаного альбому. 8. Вивести назви груп у порядку від найстаршої до наймолодшої. 9. Керуючись інформацією про треки, виведіть 10 найякісніших робіт. 10.Провівши аналіз по базі даних дайте висновок про найпопулярнішу групу і чому.	
14	Склад бази даних «Відділ кадрів»: таблиця «Кадри», таблиця «Посади», таблиця «Штатний розклад», таблиця «Стан штату», таблиця «Планові робочі дні». Структура таблиці «Кадри»;табельний номер; ПІБ; дата народження; стать; ідентифікаційний код; паспортні дані; адреса; освіта; форма працевлаштування; підрозділ. Структура таблиці «Посади»: код посади; посада; тариф оплати за 1 год.; надбавка. Структура таблиці «Штатний розклад»: код посади; код підрозділу, кількість штатних одиниць, верхня межа окладу, нижня межа окладу. Структура таблиці «Стан штату»: табельний номер; код посади; код підрозділу, рік, місяць, код форми працевлаштування, оклад, кількість відпрацьованих днів. Структура таблиці «Планові робочі дні»: рік, місяць, кількість робочих днів. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** 1. Отримати інформацію по наймолодшому працівнику. 2. Вивести інформацію по працівникам з вищою освітою. 3. Отримати інформацію по всім працюючим на підприємстві. 4. Отримати інформацію про чисельність працюючих чоловіків та жінок окремо. 5. Вивести інформацію по одній вказаній посаді. 6. Користуючись інформацією по вказаній базі даних скласти вікову діаграму працівників підприємства. 7. Вказати кількість хворих за поточний місяць. 8. Скласти список робітників по кожній посаді, що мають зарплату	

	менше вказаної величини. 9. Визначити суму прибуткового податку, утриманого з зарплати працюючих в цілому по підприємству. 10. Отримати інформацію про працюючих, що мають заробітну плату нижче відповідної величини.	
15	Склад бази даних «Абітурієнт»: таблиця «Абітурієнти», таблиця «Паспортні дані», таблиця «Відомості про освіту», таблиця «Результати вступних іспитів». Структура таблиці «Абітурієнти»: порядковий номер (ключове поле), ШБ, дата народження, дата подачі документів, спеціальність, основа навчання, форма навчання. Структура таблиці «Паспортні дані»: порядковий номер (ключове поле), серія, номер, ким виданий, дата видачі, сімейний стан. Структура таблиці «Результати вступних іспитів»: фізика, математика, диктант, порядковий номер (ключове поле***).*** Використовуючи інформаційно-логічну модель: а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** вказаної літери. 7. Визначити, який товар замовив вказаний клієнт. 8. Видалити усіх клієнтів, що не зробили жодного замовлення, але перебувають у списку. 9. Додати до переліку товарів на складі будь-який новий товар. 10. Після видалення прізвищ клієнтів, поновити ці дані.	
16	Склад бази даних «Бібліотека»: таблиця «Бібліотекарі», таблиця «Каталог», таблиця «Формуляр читача», таблиця «Читальний зал», таблиця «Читач». Структура таблиці «Бібліотекарі»: особистий номер (ключове поле), ШБ, дата народження, освіта, адреса, телефон, паспортні дані, коли і ким виданий. Структура таблиці «Каталог»: номер відділу, назва відділу, інвентарний номер (ключове поле), автор, заголовок. Структура таблиці «Формуляр читача»: № формуляра (ключове поле), дата видачі, інвентарний номер, дата здачі, видав, прийняв. Структура таблиці «Читальний зал»: № п/п (ключове поле), № формуляру, дата видачі, час видачі, інвентарний номер, час здачі, число здачі, сума, сплачено, видав, прийняв. Структура таблиці «Читач»: № формуляру (ключове поле), ПІБ, дата народження, освіта, професія, місце роботи, навчальний заклад,	

	домашня адреса, телефон, паспортні дані, ким і коли виданий, читач бібліотеки з (число). **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Вивести дані по всіх працівниках бібліотеки. 2. Вивести дані по усіх боржниках. 3. Вивести дані по книжці, вказавши її інвентарний номер. 4. Вивести дані про читача по прізвищу. 5. Вивести інформацію по читачу, що взяв вказану книжку. 6. Вивести список читачів, що записались до бібліотеки у липні минулого року. 7. Вивести список усіх книжок, що знаходяться «на руках». 8. Дати інформацію, які книжки видавав вказаний бібліотекар. 9. Вивести список книжок, назви яких починаються з вказаної літери. 10. Вивести список літератури по вказаному відділу.	
17	Склад бази даних «Автоматизована система торгівлі»: таблиця «Товари», таблиця «Постачальники», таблиця «Клієнти», таблиця «Замовлення», таблиця «Замовлено товару», таблиця «Типи доставки», таблиця «Типи оплати». Структура таблиці «Товари»: код, найменування, кількість на складі, ціна за одиницю товару, опис, код постачальника. Структура таблиці «Постачальники»: код, ПШ, контактний телефон, адреса, факс, e-mail. Структура таблиці «Клієнти»: код, ПІБ, адреса, контактний телефон, e-mail. Структура таблиці «Типи оплати»: код, назва типу, опис. Структура таблиці «Типи доставки»: код, тип доставки, опис. Структура таблиці «Замовлення»: код замовлення, замовлено товару на (грн.), вартість доставки, коментар, тип доставки, тип оплати. Структура таблиці «Замовлено товару»: код, код замовлення, код товару, куплено за (ціна), кількість. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Вивести інформацію по замовленню кожного клієнта. 2. Вивести повну інформацію по клієнту.	

	3. Вивести інформацію по вказаному товару. 4. Вивести інформацію лише по типу доставки. 5. Вивести динаміку по замовленнях. 6. Вивести інформацію про те, який клієнт яким способом оплачує замовлення та суму, на яку було здійснене замовлення. 7. Вивести усіх клієнтів, що зробили замовлення вище вказаної суми. 8. За вказаним найменуванням товару вивести інформацію про його постачальника. Вивести інформацію по усіх постачальниках та товарах, що вони постачають. 9. Дізнатись кількість замовленого товару одного найменування.	
18	Склад бази даних «Туристична агенція»: таблиця «Замовлення», таблиця «Клієнти», таблиця «Путівки». Структура таблиці «Замовлення»: № замовлення, дата замовлення, дата відправлення, код клієнта, код путівки, кількість білетів. Структура таблиці «Клієнти»: код клієнта, ПІБ клієнта, № страховки, адреса, телефон, паспортні дані. Структура таблиці «Путівки»: код путівки, країна, вид відпочинку, проїзд, дата відправлення, термін, ціна. **Використовуючи інформаційно-логічну модель:** а)Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б)Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в)побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Вивести інформацію по клієнтах. 2. Вивести ціну по вказаній путівці. 3. Вивести інформацію по вказаному клієнту. 4. Вивести номери страховок всіх клієнтів. 5. Вказати тривалість зазначеної путівки. 6. Визначити найпопулярнішу країну відпочинку. 7. Ввівши прізвище клієнта, отримати інформацію про його путівку. 8. Показати усі замовлення за вказаний місяць. 9. Визначити, путівка до якої країни є найтривалішою. 10. Вивести усіх клієнтів та дані по ним у алфавітному порядку.	
19	Склад бази даних «Факультет»: таблиця «Список студентів», таблиця «Оцінки», таблиця «Іспити», таблиця «Викладачі». Структура таблиці «Список студентів»: прізвище, рік народження, адреса, державна основа, контрактна основа, спеціальність. Структура таблиці «Оцінки»: прізвище, № з/к, група, код предмету, оцінка. Структура таблиці «Іспити»: код іспиту, назва предмету, викладач,	**10**

	табельний №. Структура таблиці «Викладачі»: табельний №, ПІБ, дата народження, адреса, телефон, вчений ступінь. **Використовуючи інформаційно-логічну модель:** а)Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б)Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в)побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. По табельному номеру вивести дані по викладачу. 2. Вивести список усіх студентів. 3. Вказати студентів, що навчаються на державній і контрактній основі окремо. 4. При введенні групи, вивести список студентів цієї групи. 5. Дізнатись який викладач читає який предмет. 6. Вивести інформацію про успіхи студентів вказаної групи з кожного предмету за даними іспитів. 7. По прізвищу студента вивести його оцінки за іспит з кожного предмету. 8. Отримати інформацію по кожному викладачу окремо. 9. Ввівши прізвище викладача, отримати інформацію про предмет та групи, в яких цей предмет читався. 10.Вказавши предмет і групу дізнатись за списком оцінки за іспит з цього предмету студента кожної групи.	
20	Склад бази даних «Аеропорт»: таблиця «Квитки», таблиця «Пасажири», таблиця «Замовлення квитка», таблиця «Розклад авіарейсів». Структура таблиці «Квитки»: код квитка, напрям, дата вильоту, тип літака, термінал, номер рейсу, кад напрямку, вартість квитка. Структура таблиці «Пасажири»: код пасажира, ПІБ, квиток клас, номер рейсу, код напряму. Структура таблиці «Замовлення квитка»: номер замовлення, дата замовлення, код квитка, код пасажира, термінал. Структура таблиці «Розклад авіарейсів»: дні вильоту, код напряму, час зльоту, напрям, час посадки, тип літака. Використовуючи інформаційно-логічну модель:а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. По коду пасажира вивести дані по замовленому квитку. 2. Вивести розклад усіх авіарейсів. 3. Визначити куди летить вказаний пасажир.	**10**

	4. Визначити дату вильоту та час рейсу. 5. Вивести інформацію про усіх пасажирів. 6. Вказати з якого терміналу і коли здійснюється виліт. 7. Вказавши країну і час вильоту отримати список пасажирів. 8.Дізнатись, які квитки були замовлені вказаного числа. 9. Використовуючи дані по пасажирам отримати інформацію, хто летить першим класом. 10. Вказавши клас польоту вивести список усіх пасажирів, що летять даним класом.	
21	Склад бази даних «Успішність у школі»: таблиця «Список учнів», таблиця «Список вчителів», таблиця «Список кабінетів». Структура таблиці «Список учнів»: № п/п, ПІБ, дата народження, клас. Структура таблиці «Список вчителів»: табельний номер, ПІБ, дата народження, освіта. Структура таблиці «Список кабінетів»: № кабінету, найменування кабінету, телефон, відповідальний. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. **Використовуючи зазначену базу даних та засоби мови SQL:** 1. Вказати який викладач відповідає за який кабінет. 2. Вивести дані про класне керівництво. 3. По вказано прізвищу вивести дані про учня. 4. Вивести дані по усіх викладачах. 5. При введенні класу вивести список учнів цього класу. 6. Вивести дані по усіх учнях школи. 7. По табельному номеру отримати всю можливу інформацію по викладачу. 8. Вказавши клас отримати список учнів, та прізвище класного керівника. 9. Отримати вписок учнів, що народились вказаного року. 10. Підрахувати кількість учнів у школі.	
22	Склад бази даних «Магазин»; таблиця «Товари», таблиця «Постачальники», таблиця «Клієнти», таблиця «Склад». Структура таблиці «Товари»: код товару, ціна, код постачальника, найменування. Структура таблиці «Постачальники»: код постачальника, місто, телефон, ПІБ. Структура таблиці «Клієнти»: код клієнта, місто, ПІБ, телефон.	**10**

	Структура таблиці «Склад»: код товару, код клієнту, наявність на складі. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** 1. Вивести на екран замовлення вказаного клієнту. 2. Визначити наявність та кількість вказаного товару на складі. 3. По номеру замовлення вивести дані по цьому замовленню. 4. Вивести дані про клієнтів. 5. Вивести ціну на вказаний товар. 6. По коду постачальника визначити який товар він постачає. 7. Визначити найдорожчий та найдешевший товар у магазині. 8. Вивести список постачальників з вказаного міста. 9. Вивести список клієнтів з вказаного міста. 10. Вивести дані по усіх постачальниках.	
23	Склад бази даних «Читальний зал»: таблиця «Бібліотекар», таблиця «Каталог», таблиця «Читач». Структура таблиці «Бібліотекар»: табельний №, ПІБ, дата народження, освіта, адреса, телефон, паспортні дані, ким і коли виданий. Структура таблиці «Каталог»: код напрямку, назва напрямку, інвентарний номер, автор, назва. Структура таблиці «Читач»: № формуляру, ПІБ, дата народження, освіта, професія, місце роботи, учбовий заклад, адреса, телефон, паспортні дані, ким і коли виданий. **Використовуючи інформаційно-логічну модель:** а) Побудувати та заповнити даними таблиці бази даних, що відповідають об'єктам моделі; б) Визначити ключові реквізити та побудувати схему даних, яка відповідає інформаційно-логічній моделі; в) побудувати форми для перегляду та заповнення таблиць даними. ***Використовуючи зазначену базу даних та засоби мови SQL:*** 1. Вивести інформацію по усіх читачах. 2. Вивести інформацію по вказаному бібліотекарю. 3. По номеру формуляра отримати дані по читачу. 4. Отримати список читачів, що працюють і де. 5. Отримати список читачів, що навчаються і де. 6. Отримати список літератури по вказаному напряму. 7. Отримати дані про книжки, написані вказаним автором. 8. Вивести особистий номер вказаного бібліотекаря. 9. Вивести список читачів у алфавітному порядку. 10. Отримати список бібліотекарів, що не мають вищої освіти.	**10**