МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

РОБОЧА ПРОГРАМА,
методичні вказівки та індивідуальні завдання до вивчення дисципліни «Організація баз даних та знань» для студентів спеціальності 122 - комп'ютерні науки

Дніпро НМетАУ 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

РОБОЧА ПРОГРАМА,
методичні вказівки та індивідуальні завдання
до вивчення дисципліни «Організація баз даних та знань»
для студентів спеціальності 122 -ком'пютерні науки
заочної форми навчання

Затверджено
на засіданні кафедри ІТС
Протокол № 9 від
06.03.2019

Дніпро НМетАУ 2019

УДК 681.3.06

Робоча програма, методичні вказівки та індивідуальні завдання до вивчення дисципліни «Організація баз даних і знань» для студентів спеціальності 122 - комп'ютерні науки заочної форми навчання/Укл. Н.Л. Дорош, Г. Л. Євтушенко. - Дніпро: НМетАУ, 2019 – 51 с.

Містяться робоча програма дисципліни «Організація баз даних і знань» з коротким викладом змісту її розділів і методичними вказівками до вивчення навчального матеріалу, а також завдання до контрольної роботи, на основі якої студент освоює принципи проектування і розробки баз даних, застосування реляційної моделі даних, роботу у системі Microsoft Access, мову запитів SQL.

Призначені для студентів спеціальності 122 - комп'ютерні науки заочної форми навчання.

Друкується за авторською редакцією.

Укладачі: Н. Л. Дорош, канд. техн. наук, доц.

Г. Л. Євтушенко, канд. техн. наук, доц.

Відповідальний за випуск О.І. Михальов, д-р техн. наук, проф.

Рецензент : В.Є. Бєлозьоров, д-р фіз.-мат. наук, проф. (ДНУ ім. О. Гончара)

Підписано до друку 10.03.2019. Формат 60х84 1/16. Папір типогр. Друк різограф. Обл.-вид. арк. 3,0. Умов. друк. арк. 2,95. Тираж 100 прим. Замовл. № 6/19.

Національна металургійна академія України.

49600, Дніпро, пр. Гагаріна, 4

ЗМІСТ

ВСТУП

Навчальна дисципліна «Організація баз даних та знань» присвячена питанням проектування і розробки баз даних.

Сьогодні відповідно до чинного державного освітнього стандарту «Організація баз даних та знань» вивчається як самостійна дисципліна студентами спеціальності 122 «Комп'ютерні науки». Дисципліна входить до циклу професійно-орієнтованих дисциплін за фахом.

Дисципліна «Організація баз даних та знань» складається з таких основних тем:

- огляд сучасних баз даних і моделей даних;
- характеристика реляційної моделі даних;
- реляційна алгебра і реляційне числення;
- проектування і розробка баз даних;
- життєвий цикл баз даних, архітектура бази даних;
- нормалізація баз даних, надмірність даних в базах даних, аномалії відновлювання в базах даних, функціональні залежності та ключі, аксиоми Армстронга;
- сучасні мови розробки баз даних.

Основні вимоги до виконання контрольної роботи є такі, що до моменту приїзду студента на екзаменаційну сесію він повинен мати конспект дисципліни, виконану і зараховану контрольну роботу. Якщо контрольну роботу не зараховано, то студент не допускається до іспиту.

1 ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ

Навчальна дисципліна «Організація баз даних та знань» є обов'язковою і входить до циклу професійно-орієнтованих дисциплін за фахом.

Мета вивчення дисципліни – вивчення студентами теоретичних основ, придбання практичних навичок й освоєння інструментальних засобів рішення задач обробки даних за допомогою систем управління базами даних (СУБД).

У результаті вивчення дисципліни студент повинен:

знати теоретичні основи побудови інформаційних систем і баз даних; інструментальні засоби аналізу й проектування моделей даних; теоретичні основи реляційної моделі даних, побудову моделей даних; реалізацію бази даних в одній-двох СУБД;

вміти провести аналіз предметної області для побудови бази даних; проектувати логічні моделі даних; користуватися інструментальними засобами для аналізу й проектування; реалізувати розробку бази даних в середовищі одной-двох СУБД.

Для успішного складання іспиту з дисципліни необхідно: опрацювати питання контрольної роботи, підготуватись до іспиту згідно питань дисципліни.

Зв'язок з іншими дисциплінами: навички та вміння цієї дисципліни базуються на дисциплінах "Технологія програмування та створення програмних продуктів", "Основи дискретної математики", "Основи програмування та алгоритмічні мови", "Системний аналіз та проектування систем обробки інформації", "Основи інформації". Навички та вміння цієї дисципліни застосовуються у таких дисциплінах, як "Проектування організаційних і технологічних інформаційних управляючих систем", "Супровід і експлуатація програмного забезпечення інформаційних управляючих систем".

Набуті знання і вміння у подальшому можуть бути використані при виконанні курсових робіт та дипломному проектуванні.

2 РОБОЧА ПРОГРАМА ТА МЕТОДИЧНІ ВКАЗІВКИ

Технологічна карта до робочої програми вивчення дисципліни «Організація баз даних та знань» для студентів заочної форми навчання спеціальності 122 «Комп'ютерні науки» представлена у таблиці 2.1.

Таблиця 2.1- Розподіл навчальних годин та форми контролю

Усього годин за навчальним планом	180
у тому числі: Аудиторні заняття	72
з них: - лекції	32
- лабораторні заняття	40
- практичні заняття	
- семінари	
Самостійна робота	108
- виконанні курсовоїроботи	30
- підготовці до занять та іспиту	51
- вивчення окремих розділів програми, які не увійшли до лекційного курсу	27
Підсумковий контроль	іспит, курсова робота

2.1 Перші етапи життєвого циклу бази даних

Розглянутий нижче матеріал присвячено проектуванню і розробці складних баз даних (БД), тобто баз даних з великою кількістю даних, для розробки якої потрібна й велика кількість розробників. При проектуванні складних БД виникає потреба виконання всіх етапів життєвого циклу бази даних (ЖЦБД). Для БД з невеликою кількістю розробників і в деяких інших окремих випадках, навпаки, життєвий цикл бази даних може бути значно спрощений у результаті корегування змісту окремих етапів.

ЖЦБД містить наступні основні етапи (рис. 2.1):

- Планування розробки бази даних.
- Визначення вимог до системи.
- Збір та аналіз вимог користувачів.
- Проектування бази даних (концептуальне, логічне, фізичне).

- Розробка застосунку (проектування транзакцій; проектування користувальницького інтерфейсу).
- Реалізація.
- Завантаження даних.
- Тестування.
- Експлуатація і супровід (а: аналіз функціонування та підтримка початкового коду БД, б: адаптація, модернізація і підтримка перероблених варіантів).

На рисунку 2.1 представлено перелік основних етапів ЖЦБД [1]. Звісно, конкретне наповнення кожного етапу в значній мірі залежить від складності розроблюваного продукту.

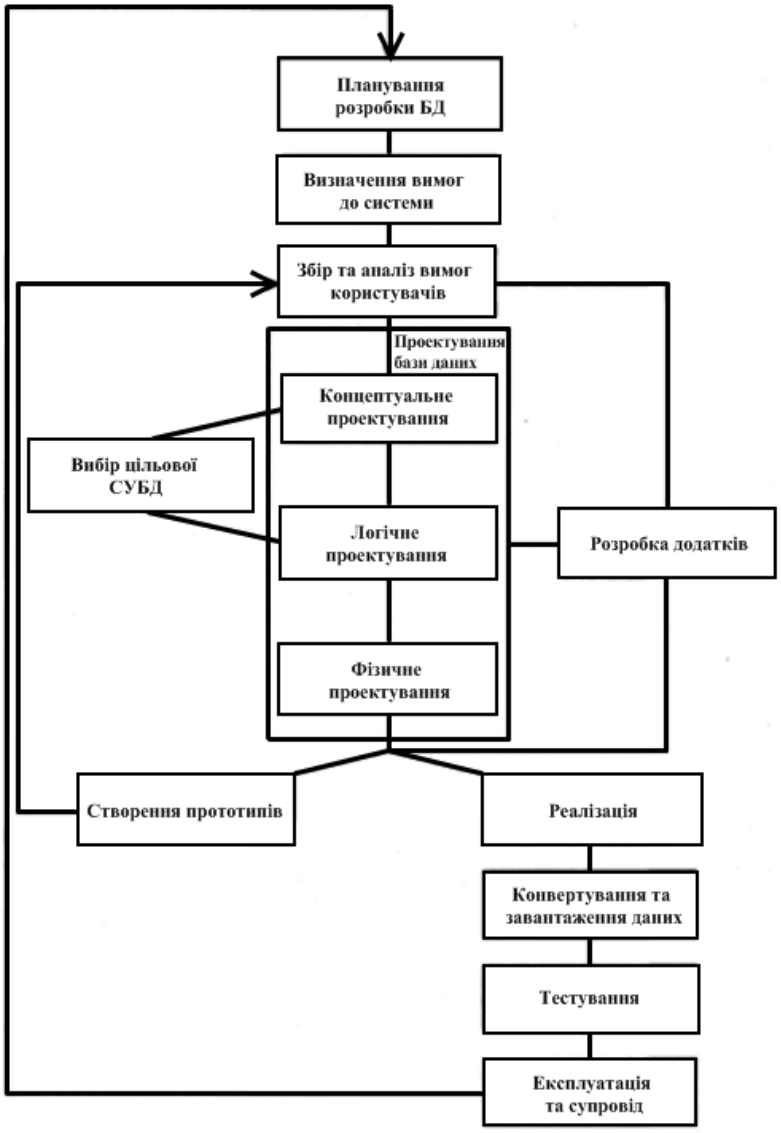

Рисунок 2.1. – Життєвий цикл баз даних

1 Планування розробки бази даних

Суть даного етапу – розробка стратегічного плану, в процесі якого здійснюється попереднє планування конкретної системи управління базами даних. Загальна інформаційна модель, створена на цьому кроці, повинна бути знову проаналізована і, якщо потрібно, змінена на наступному етапі, етапі розробки проекту реалізації.

Планування розробки бази даних полягає у визначенні трьох основних компонентів: обсягу робіт, ресурсів і вартості проекту. Планування розробки бази даних повинне бути пов'язане із загальною стратегією побудови інформаційної системи організації.

Важливою частиною розробки стратегічного плану є перевірка здійсненності проекту, що складається з декількох частин.

Перша частина – перевірка технологічної здійсненності. Вона полягає у з'ясуванні питання, чи існує обладнання та програмне забезпечення, яке задовольняє інформаційним потребам фірми.

Друга частина – перевірка операційної здійсненності – з'ясування наявності експертів та персоналу, необхідних для роботи БД.

Третя частина – перевірка економічної доцільності здійснення проекту. При дослідженні цієї проблеми важливо дати оцінку ряду факторів, у тому числі і таким:

- доцільність спільного використання даних різними відділами;
- величина ризику, пов'язаного з реалізацією системи бази даних;
- очікувана вигода від впровадження підлягає створенню застосунків;
- час окупності впровадженої БД;
- вплив системи управління БД на реалізацію довготривалих планів організації.

Планування розробки баз даних також має включати розробку стандартів, які визначають, як буде здійснюватися збір даних, яким буде їхній формат, яка буде потрібна документація, як буде виконуватися проектування та реалізація застосунків.

Для підтримки планування розробки бази даних може бути створена корпоративна модель даних, що має вигляд спрощеної ER-діаграми.

Якщо результат перевірки здійсненності проекту виявився позитивним, можна перейти до визначення вимог до проекту.

2 Визначення вимог до системи

На даному етапі необхідно визначити діапазон дії програми бази даних, складу його користувачів і області застосування.

Визначення вимог включає вибір цілей БД, з'ясування інформаційних потреб різних відділів і керівників фірми, вимог до обладнання та програмного забезпечення. При цьому також потрібно розглянути питання, чи слід створювати розподілену базу даних або ж централізовану, і які в розглянутій ситуації знадобляться комунікаційні засоби. Написати короткий коментар, що описує цілі системи.

Перш ніж приступати до проектування програми бази даних, важливо встановити межі досліджуваної області та способи взаємодії застосунку з іншими частинами інформаційної системи організації. Ці межі повинні охоплювати не тільки поточних користувачів і області застосування розроблюваної системи, але і майбутніх користувачів і можлива галузь застосування.

3 Збір та аналіз вимог користувачів

Цей етап є попереднім етапом концептуального проектування бази даних. Проектування бази даних засноване на інформації про ту частину організації, яка буде обслуговуватися базою даних.

Інформаційні потреби з'ясовуються за допомогою анкет, опитувань менеджерів і працівників фірми, за допомогою спостережень за діяльністю підприємства, а також звітів і форм, якими фірма користується у даний момент.

На даному етапі необхідно створити для себе модель руху важливих матеріальних об'єктів і з'ясувати процес документообігу. По кожному документу необхідно встановити періодичність використання, визначити дані, необхідні для виконання виділених функцій (аналізуючи існуючу і заплановану документацію, з'ясовують, як виходить кожен елемент даних, ким виходить, де в подальшому використовується, ким контролюється).

Найпильнішу увагу має бути приділено дублюванню інформації, можливості появи помилкової інформації і причин, які ведуть до їх появи. Також на цьому етапі бажано представити загальні параметри створюваної бази.

У підсумку зібрана інформація щодо кожної важливої сфери застосування застосунку та користувацької групи повинна включати наступні

компоненти: вихідну та генеровану документацію, докладні відомості про виконувані транзакції, а також список вимог із зазначенням їх пріоритетів. На підставі всієї цієї інформації будуть складені специфікації вимог користувачів у вигляді набору документів, що описують діяльність підприємства з різних точок зору.

Формалізація зібраної на цьому етапі інформації може бути підвищена за допомогою методів складання специфікацій вимог, до числа яких відносяться, наприклад, технологія структурного аналізу і проектування, діаграми потоків даних і графіки "вхід - процес - вихід".

Оскільки системи з неадекватною чи неповною функціональністю будуть лише дратувати користувачів, а надмірно збільшений набір функціональних можливостей викличе істотне ускладнення системи, важливість цього етапу в процесі розробки БД важко переоцінити.

4 Проектування бази даних

Повний цикл розробки бази даних включає концептуальне, логічне і фізичне її проектування.

Основними цілями проектування бази даних є:

- подання даних і зв'язків між ними, необхідних для всіх основних областей застосування даного застосунку і будь-яких існуючих груп його користувачів;
- створення моделі даних, здатної підтримувати виконання будь-яких необхідних транзакцій обробки даних;
- розробка попереднього варіанту проекту, структура якого дозволяє задовольнити вимоги, що пред'являються до продуктивності системи.

У створенні БД як моделі предметної області виділяють:

- об'єктну (предметну) систему, що представляє фрагмент реального світу;
- інформаційну систему, що описує деяку об'єктну систему;
- датологічну систему, що представляє інформаційну систему за допомогою даних.

Оптимальна модель даних повинна задовольняти таким критеріям, як: структурна достовірність, простота, виразність, відсутність надмірності, розширюваність, цілісність, здатність до спільного використання.

5 Концептуальне проектування бази даних

Перша фаза процесу проектування бази даних полягає у створенні для аналізованої частини підприємства концептуальної моделі даних. Побудова її здійснюється в певному порядку: на початку створюються докладні моделі користувальницьких представлень даних; потім вони інтегруються в концептуальну модель даних. Концептуальне проектування призводить до створення концептуальної схеми бази даних.

Існує два основних підходи до проектування систем баз даних: «спадний» та «висхідний».

При висхідному підході, який застосовується для проектування простих баз даних з відносно невеликою кількістю атрибутів, робота починається з самого нижнього рівня – рівня визначення атрибутів, які на основі аналізу існуючих між ними зв'язків групуються у відносини. Отримані відносини надалі піддаються процесу нормалізації, який призводить до створення нормалізованих взаємопов'язаних таблиць, заснованих на функціональних залежностях між атрибутами.

Проектування складних баз даних з великою кількістю атрибутів, оскільки встановити серед атрибутів всі існуючі функціональні залежності досить важко, здійснюється використанням спадного підходу. Починається цей підхід з розробки моделей даних, які містять кілька високорівневих сутностей і зв'язків, потім робота продовжується у вигляді серії спадних уточнень низькорівневих сутностей, зв'язків і належних до них атрибутів.

Спадний підхід демонструється в концепції моделі «сутність-зв'язок» (Entity – Relationship model – ER-модель) – найпопулярнішої технології високорівневого моделювання даних, запропонованої Ченом.

Модель «сутність-зв'язок» відноситься до семантичних моделей. Методи семантичного моделювання виявилися застосовні до багатьох користувальницьких проблем і легко перетворені в мережеві, ієрархічні та реляційні моделі.

Крім «спадного» і «висхідного» підходів, для проектування баз даних можуть застосовуватися інші підходи, які є деякими комбінаціями зазначених.

У побудові загальної концептуальної моделі даних виділяють ряд етапів:

1) Виділення локальних уявлень, відповідних зазвичай відносно незалежним даними. Кожне уявлення проектується як підзадача.

2) Виділення об’єктів, що описують локальну предметну область проектованої БД, і опис атрибутів, що становлять структуру кожного об’єкта.
3) Виділення ключових атрибутів.
4) Специфікація зв’язків між об’єктами. Видалення надлишкових зв’язків.
5) Аналіз і додавання не ключових атрибутів.
6) Об’єднання локальних уявлень.

Побудова концептуальної моделі даних здійснюється на основі аналізу опису предметної області природною мовою, яке надає, як правило, замовник. У процесі розробки концептуальна модель даних постійно піддається тестуванню та перевірці на відповідність вимогам користувачів. Створена концептуальна модель даних є джерелом інформації для фази логічного проектування бази даних.

Контрольні запитання для самоперевірки

Охарактеризуйте життєвий цикл бази даних. Які основні етапи він включає?
Назвіть основні цілі процесу проектування бази даних та наведіть характеристику його фаз.
Пояснити функції та загальне призначення концептуального моделювання.
Дати визначення первинного ключа об’єкта (відношення, таблиці).
Пояснити призначення зовнішніх ключів відносин (таблиць).
Визначити два основних правила цілісності реляційної моделі.
Описати проблеми, пов’язані з наявністю надмірності даних.

2.2 Процес нормалізації (1НФ, 2НФ, 3НФ) на основі аналізу функціональних залежностей

Процес нормалізації був вперше запропонований Коддом в 1972 році. Цей процес заснований на понятті функціональної залежності. За визначенням *«функціональна залежність – це такий зв’язок між атрибутами B і A одного і того ж відношення (таблиці), коли кожному значенню A відповідає тільки одне значення B». Атрибут A називають детермінантою*. Детермінанти можуть бути складовими, тобто являти собою не поодинокі атрибути, а групи, що складаються з двох і більше атрибутів.

Нормалізація звичайно призводить до поділу однієї таблиці на дві або більше таблиці, що відповідають вимогам нормальних форм. Загальноприйнятими вважаються п’ять нормальних форм [2]. Спочатку було

запропоновано тільки три види нормальних форм: перша (1НФ), друга (2НФ) і третя (3НФ). Потім Бойсом і Коддом в 1974 році було сформульовано більш суворе визначення третьої нормальної форми, яке отримало назву нормальної форми Бойса-Кодда (НФБК).

Слідом за НФБК з'явилися визначення четвертої (4НФ) і п'ятої (5НФ) нормальних форм в 1977 і в 1979 роках. Однак на практиці ці нормальні форми більш високих порядків використовуються рідко.

Кожна наступна форма задовольняє вимогам попередньої. Якщо слідувати тільки першим правилам нормалізації, то дані будуть представлені в 1НФ. Якщо дані задовольняють третьому правилу нормалізації, вони перебуватимуть в 3НФ (а також у 1НФ і 2НФ) і т.д.

Таким чином, кожна наступна форма пред'являє більше вимог до даних, ніж попередня. Перша нормальна форма вимагає, щоб на будь-якому перетині рядка і стовпця знаходилося єдине значення, яке має бути атомарним (неподільним). У таблиці, що задовольняє 1НФ, не повинно бути повторюваних груп. Розглянемо таблицю представлену на рис. 2.2.

Повторяющиеся группы значений

ACCOUNTCD	STREETCD	STREETNM	HOUSENO	FLATNO	FIO	PHONE
005488	3	ВОЙКОВ ПЕРЕУЛОК	4	1	АКСЕНОВ С.А.	556893
015527	3	ВОЙКОВ ПЕРЕУЛОК	1	65	КОНОХОВ В.С.	761699
080047	8	МОСКОВСКОЕ ШОССЕ УЛИЦА	39	36	СЕРОВА Т.П.	257842
080270	6	МОСКОВСКАЯ УЛИЦА	35	6	ТИМОШКИНА Н.Г.	321002
080613	8	МОСКОВСКОЕ ШОССЕ УЛИЦА	35	11	ЛУКАШИНА Р.М.	254417
115705	3	ВОЙКОВ ПЕРЕУЛОК	1	82	МИЩЕНКО Е.В.	769975
126112	4	ТАТАРСКАЯ УЛИЦА	7	11	МАРКОВА В.П.	683301
136159	7	КУТУЗОВА УЛИЦА	39	1	СВИРИНА З.А.	350003
136160	4	ТАТАРСКАЯ УЛИЦА	9	15	ШМАКОВ С.В.	982222
136169	4	ТАТАРСКАЯ УЛИЦА	7	13	ДЕНИСОВА Е.К.	680305
443069	4	ТАТАРСКАЯ УЛИЦА	51	55	СТАРОДУБЦЕВ Е.В.	683014
443690	7	КУТУЗОВА УЛИЦА	5	1	ТУЛУПОВА М.И.	214833

Рисунок 2.2 – Ненормалізована таблиця

Шляхом розбиття цієї таблиці на дві можна отримати таблиці Abonent (рис. 2.4) і Street (рис. 2.3), що задовольняють вимогам 1НФ.

STREETCD	STREETNM
1	ЦИОЛКОВСКОГО УЛИЦА
2	НОВАЯ УЛИЦА
3	ВОЙКОВ ПЕРЕУЛОК
4	ТАТАРСКАЯ УЛИЦА
5	ГАГАРИНА УЛИЦА
6	МОСКОВСКАЯ УЛИЦА
7	КУТУЗОВА УЛИЦА
8	МОСКОВСКОЕ ШОССЕ УЛИЦА

Рисунок 2.3 – Таблиця «Street»

ACCOUNTCD	STREETCD	HOUSENO	FLATNO	FIO	PHONE
005488	3	4	1	АКСЕНОВ С.А.	556893
015527	3	1	65	КОНЮХОВ В.С.	761699
080047	8	39	36	ШУБИНА Т.П.	257842
080270	6	35	6	ТИМОШКИНА Н.Г.	321002
080613	8	35	11	ЛУКАШИНА Р.М.	254417
115705	3	1	82	МИЩЕНКО Е.В.	769975
126112	4	7	11	МАРКОВА В.П.	683301
136159	7	39	1	СВИРИНА З.А.	350003
136160	4	9	15	ШМАКОВ С.В.	982222
136169	4	7	13	ДЕНИСОВА Е.К.	680305
443069	4	51	55	СТАРОДУБЦЕВ Е.В.	683014
443690	7	5	1	ТУЛУПОВА М.И.	214833

Рисунок 2.4 – Таблиця «Abonent»

Друга нормальна форма заснована на понятті повної функціональної залежності. *Атрибут B називається повністю функціонально залежним від атрибуту A, якщо атрибут B функціонально залежить від повного значення атрибуту A і не залежить від будь-якої підмножини атрибута A*.

Відношення знаходиться у 2НФ, якщо воно знаходиться в 1НФ і кожен його атрибут, який не входить до складу первинного ключа, функціонально повно залежить від первинного ключа. Іншими словами, друге правило нормалізації вимагає, щоб будь-який з стовпців, який не є ключовим, залежав від усього первинного ключа, а не від його окремих компонентів. Це правило відноситься до випадку, коли первинний ключ утворений з декількох стовпців. Первинні ключі таблиць Abonent (рис. 2.4) і Street (рис. 2.3) є простими (складаються з одного стовпця), а тому знаходяться не тільки в 1НФ, але і однозначно у 2НФ.

Третя нормальна форма заснована на понятті транзитивної залежності. *Якщо для атрибутів A, B і C деякого відносини існують залежності C від B і*

В від А, то говорять, що атрибут С транзитивній залежить від атрибуту А через атрибут В.

Відношення знаходиться в 3НФ, якщо воно знаходиться в 1НФ і 2НФ, і в ньому не існує транзитивних залежностей стовпців, які не є ключовими, від первинного ключа. Іншими словами, третя нормальна форма вимагає, щоб жоден з стовпців, який не є ключовим, не залежав би від іншого стовпця, який також не є ключовим. Будь-який з стовпців, який не є ключовим, повинен залежати тільки від стовпця первинного ключа.

Розглянемо, наприклад, залежності між стовпцями в таблиці PaySumma (рис. 2.5). Наприклад, стовпець PaySum в цій таблиці не залежить від стовпця AccountCD, так як одному абоненту відповідає безліч сплачених сум. Також стовпець PaySum не залежить від стовпця PayDate, так як на одну дату може припадати кілька сплачених сум і т.д. Таким чином, між стовпцями, які не є ключовими, немає функціональних залежностей і, отже, немає транзитивних залежностей цих стовпців від первинного ключа. Таблиця PaySumma (рис. 2.5) задовольняє вимогам 3НФ.

PAYFACTCD	ACCOUNTCD	GAZSERVICECD	PAYSUM	PAYDATE	PAYMONTH	PAYYEAR
1	5488	2	58,70	08.01.2002	12	2 001
2	5488	2	46,00	06.01.2001	12	2 000
3	5488	2	56,00	06.05.1999	4	1 999
4	115705	2	40,00	10.02.2000	1	2 000
5	115705	2	250,00	03.10.2001	9	2 001
6	136160	2	20,00	13.06.2001	5	2 001
7	136160	2	56,00	12.02.1999	1	1 999
8	136169	2	20,00	22.06.2001	5	2 001
9	80047	2	80,00	26.11.1998	10	1 998
10	80047	2	80,00	21.11.2001	10	2 001
11	80270	2	46,00	03.01.2002	12	2 001
12	80613	2	56,00	19.07.2001	6	2 001
13	115705	2	250,00	06.10.2000	9	2 000
14	115705	2	58,70	04.09.2001	8	2 001

Рисунок 2.5 – Таблиця «PaySumma»

Нормальна форма Бойса-Кодда враховує функціональні залежності, в яких беруть участь усі потенційні ключі відношення, а не тільки його первинний ключ. Для відношення з єдиним потенційним ключем 3НФ і НФБК еквівалентні.

Відношення знаходиться в НФБК тоді і тільки тоді, коли кожен його детермінант є потенційним ключем.

Четверта нормальна форма пов’язана з поняттям багатозначної залежності. *У разі багатозначної залежності, що існує між атрибутами А,*

В і С деякого відношення, для кожного значення А є набір значень атрибуту В і набір значень атрибуту С. Однак значення атрибутів В і С, що входять в ці набори, не залежать один від одного.

Відношення знаходиться в 4НФ, якщо воно знаходиться в НФБК і не містить багатозначних залежностей.

П'ятою нормальною формою називається відношення, яке не містить залежностей з'єднання. Залежність з'єднання – це така ситуація, при якій декомпозиція відносини може супроводжуватися генерацією помилкових рядків при зворотному з'єднанні декомпозованих відношень за допомогою операції з'єднання.

Контрольні запитання для самоперевірки

1) Дати визначення концепції функціональної залежності.
2) Як концепція функціональної залежності пов'язана з процесом нормалізації?
3) Привести відношення до першої нормальної формі.
4) Привести відношення до другої нормальної форми.
5) Привести відношення до третьої нормальної форми.

2.3 Основи реляційної алгебри та реляційного числення щодо застосування у реляційній моделі даних

У 1970-му році Е. Кодд запропонував використовувати в базах даних реляційну алгебру. Перші БД створювалися за допомогою процедурних мов. Кодд також запропонував ввести реляційне числення [1-4].

Реляційна алгебра (РА) і реляційне числення (РЧ) використовувалися в якості основи для створення реляційних мов баз даних. Ці мови – недружні, формальні, вони використовувалися для розробки мов управління даними більш високого рівня. Тому, вивчення реляційної алгебри дає можливість розібратися у функціонуванні високорівневої СУБД.

Основна мета використання РА – забезпечити запис реляційних виразів, тобто написати математичні формули запитів, транслювати і працювати з ними.

Основні поняття реляційної алгебри

Відношення – це плоска таблиця, що складається із стовпців і рядків (таблиця 2.2).

Таблиця 2.2 – Відношення (таблиця) «Студент»

Номер залікової книжки	Прізвище	Рік народження
123	Легеза	1990
345	Третяк	1991

Реляційна БД може складатися з довільної кількості відносин, представлених реляційними схемами.

Реляційна алгебра – це теоретична мова операцій, які на основі одного або декількох відношень дозволяють створити інше відношення, без зміни вихідних відношень.

В результаті застосування операцій реляційної алгебри можна створювати нові відношення (без застосування інших математичних апаратів). Ця властивість називається *замкнутістю*.

Реляційна БД – це набір нормалізованих відносин.

Атрибут – це пойменований стовпець-відношення. Наприклад, у таблиці «Студент» (табл. 2.2) атрибутами є: Номер залікової книжки, Прізвище, Рік народження.

Кортеж – це рядок відношення (таблиці).

Ступінь відношення визначається кількістю атрибутів, яку воно містить. Для таблиці «Студент» (табл. 2.2) вона дорівнює 3.

Кардинальність – це кількість кортежів, яку містить відношення. Для таблиці «Студент» (табл..2.2) вона дорівнює 2.

Реляційна схема – ім’я відносини, за яким слідує безліч пар імен атрибутів і доменів.

Щоб зрозуміти, що таке **домен**, розглянемо приклади запису реляційної схеми:

«Студент» (Номер залікової книжки, Прізвище, Рік народження)

Інший приклад – у вигляді пар, в які входить атрибут і домен:

{Номер залікової книжки: D1, Прізвище: D2, Рік народження: D3 },

де D1, D2, … – це домени, які можуть бути обмеженням типу:

D1:

1) Тип даних – рядок.
2) Кількість символів – 5.

….

D3:

1) Тип даних – дата.
2) Обмеження в діапазоні 1989 – 1992.

Властивості відношень:

1) Відношення має ім’я, яке відрізняється від імен всіх інших відносин в БД.
2) Кожна комірка відносини містить тільки атомарне (або «неподільне») значення. Приклад: у відношенні «Студент» атрибут рік народження містить 4-значне число – значення року, наприклад 1991, а можна б було написати дату народження: 1991 рік, 3 червня. Якщо використовувати це значення як строкове, то на таблицю це ніяк не вплине. Але якщо нам стане цікаво, скільки студентів народилося у червні, в реляційній базі це буде неможливо через неподільності даних (з рядка «1991 рік, 3 червня» неможливо витягнути тільки місяць народження, тому що цей атрибут «місяць» ніде не визначений.)
3) Кожен атрибут має унікальне ім’я.
4) Значення атрибуту беруться з одного і того ж домену.
5) Порядок слідування атрибутів і кортежів не має ніякого значення. Тобто для перегляду даних можна сортувати кортежі в таблиці різними способами (за алфавітом, датою і т.п.) і це ніяк не вплине на структуру відношення.
6) Кожен кортеж є унікальним, тобто дублікатів кортежів бути не може.
7) Теоретично, порядок проходження кортежів у відношенні не має значення. Практично цей порядок може істотно вплинути на ефективність доступу до них. Йдеться про спосіб зберігання великих масивів.

Реляційні ключі.

Суперключ – це атрибут або безліч атрибутів, який (які) єдиним чином ідентифікує кортеж даного відношення.

Вибір ключа – дуже відповідальне завдання, оскільки це оптимізує БД, робить її зручною у використанні, модернізації і т.п.

Потенційний ключ – це суперключ, який не містить підмножини, що також є суперключем даного відношення.

Властивості потенційного ключа:

1) Унікальність – це означає, що цей ключ єдиним чином ідентифікує кортеж.
2) Неприводимість – тобто ніяка допустима підмножина потенційного ключа не має властивість унікальності.

Потенційних ключів може бути скільки завгодно. Питання – «який ключ вибрати?». Це питання вирішується на стадії проектування. Наприклад,

можна вибрати ключем прізвище, але це помилка: можуть зустрітися кортежі з однаковими значеннями атрибуту «прізвище».

Первинний ключ – це потенційний ключ, який обраний для унікальної ідентифікації кортежів усередині відношення. Потенційні ключі, які не вибрані в якості первинного ключа, називаються **альтернативними ключами**.

Наприклад, адреса електронної пошти студента.

Зовнішній ключ – це атрибут або безліч атрибутів всередині відносини, яке відповідає потенційному ключу деякого іншого відношення. Зовнішній ключ показує зв'язок між відношеннями БД.

Відношення може бути тільки «один до багатьох» або «однини до одного» в реляційній моделі даних.

Приклад.

Візьмемо відношення «Студент» і доповнимо його атрибутом «Код групи».

Таблиця 2.3 – Відношення «Студент_2»

Номер залікової книжки	Прізвище	Рік народження	Код групи
123	Легеза	1990	2
345	Третяк	1991	2

Таблиця 2.4 – Відношення «Група»

Код групи	Найменування
1	ІТС-08-1
2	ІТС-08-2

Тепер «Код групи» для відносини «Студент_2» – зовнішній ключ для зв'язку з таблицею «Група», тому що «Код групи» є потенційним ключем відношення «Група».

Реляційна цілісність.

Сенс цього поняття – що можна і чого не можна робити в БД, тобто – чи можна залишати у відношенні деякі поля порожніми, чи припустимо це?

Визначник NULL – значення атрибута на даний момент є невідомим чи неприйнятним для цього кортежу. Дозволяє вирішити питання – чи дозволити створення кортежу з порожнім значенням деякого атрибуту.

Реляційна цілісність пов’язана з заповненням кортежів і гарантує коректність даних. Існує 2 правила цілісності:

Цілісність сутностей – заповнення даних одного відношення. Наприклад, у студента відсутній рік народження.

Цілісність посилальна – виникає при роботі з двома і більше відношеннями. Наприклад, у відношенні «Група» видалимо групу «ІТС-08-2», виникає питання – видаляти студентів цієї групи у відношенні «Студент_2»? Якщо видалити – це буде каскадне видалення, тобто видаляються ВСІ студенти, що складалися раніше в групі ІТС-08-2, а також усі дані, пов’язані з цими студентами в інших таблицях. Ще можна просто поставити в цьому атрибуті NULL:

Таблиця 2.5 – Відношення «Студент_2»

Номер залікової книжки	Прізвище	Рік народження	Код групи
123	Легеза	1990	NULL
345	Третяк	1991	NULL

Операції реляційної алгебри.

Одне з призначень реляційної алгебри – це вибірка даних. Для вибірки даних використовуються наступні операції:

- декартовий добуток;
- вибірка;
- проекція.

А також набір загальних операцій, які виводяться з перерахованих.

1. Декартовий добуток (cartesian product)

Нехай:

R – множина 1, має ступінь n, кардинальність j;

S – множина 2, має ступінь m, кардинальність i;

Декартовий добуток – $\boldsymbol{M = R \times S}$ – визначає нове відношення (M), яке є результатом конкатенації кожного кортежу з відносини R з кожним кортежем з відносини S. M – множина 3, має ступінь (m + n) і кардинальність (i * j).

Приклад: Таблиця 2.6 – Відношення «Коти»

Кличка	Вага
Боня	7
Ярик	6
Мура	5
Ося	8

Таблиця 2.7 – Відношення «Господарі»

Ім’я	Вік
Наумова І.Ю.	53
Кострова М.М.	20

Таблиця 2.8 – Відношення М = Господарі × Коти

Ім’я	Вік	Кличка	Вага
Наумова І.Ю.	53	Боня	7
Наумова І.Ю.	53	Ярик	6
Наумова І.Ю.	53	Мура	5
Наумова І.Ю.	53	Ося	8
Кострова М.М.	20	Боня	7
Кострова М.М.	20	Ярик	6
Кострова М.М.	20	Мура	5
Кострова М.М.	20	Ося	8

2. Вибірка (selection) – $\sigma_{предикат}(R)$

Операція вибірки працює з одним відношенням R і визначає результуюче відношення, яке містить тільки ті кортежі (рядки) відносини R, які задовольняють заданій умові (предикату).

Наприклад:

Таблиця 2.9 – Відношення «Студент»

Номер залікової книжки	Прізвище	Рік народження	Код групи
123	Легеза	1990	2
789	Шкурко	1991	1
345	Третяк	1991	2

Таблиця 2.10 – Відношення: $\sigma_{рік\_народження>1991}(Студент)$.

Номер залікової книжки	Прізвище	Рік народження	Код групи
789	Шкурко	1991	1
345	Третяк	1991	2

3. Проекція (projection) – $\prod_{атрибут1,атрибут2...}(R)$

Визначає нове відношення, що містить вертикальну підмножина відношення R, створювану за допомогою витягання значень зазначених атрибутів і виключення з результату рядків – дублікатів.

Наприклад:

Записати засобами РА результат: відношення, яке є списком прізвищ студентів, рік народження яких більше, ніж 1991.

$\prod_{прізвище}(\sigma_{рік\_народження>1991}(Студент))$

Поетапно це виглядає так:

1) $S = \sigma_{рік\_народження>1991}(Студент)$

Таблиця 2.11 – Відношення S

Номер залікової книжки	Прізвище	Рік народження	Код групи
345	Іванов	1992	2
789	Петров	1992	1
657	Лобанов	1989	1

2) $\text{Result} = \prod_{прізвище}(S)$

Таблиця 2.12 – Відношення Result

Прізвище
Іванов
Петров

4. Об'єднання (union) – $R \cup S$

Беруть участь лише відношення однакової структури.

Об'єднання відношень R і S з кортежами j та і виходить в результаті їх конкатенації з утворенням одного кортежу кардинальністю (i + j), якщо кортежі-дублікати виключені, при цьому R і S мають бути сумісні за об'єднанням – тобто містити однакові атрибути і однієї кількості.

Приклад. Таблиця 2.13 – Відношення «Відомість_1»

Студент	Оцінка	Група
А. Анощенков	3	ІТС-08-1
Г. Бондарь	6	ІТС-08-1

Таблиця 2.14 – Відношення «Відомість_2»

Студент	Оцінка	Група
В. Бахтіаров	2	ІТС-08-2
Д. Костирко	7	ІТП-08
Е. Суліган	8	ІТС-08-1
Г. Бондарь	6	ІТС-08-1

Відомість3 = Відомість1 U Відомість2

або *Відомість3 = Відомість1 union Відомість2*

Таблиця 2.15 – Відношення «Відомість_3»

Студент	Оцінка	Група
А. Анощенков	3	ІТС-08-1
Г. Бондарь	6	ІТС-08-1
В. Бахтіаров	2	ІТС-08-2
Д. Костирко	7	ІТП-08
Е. Суліган	8	ІТС-08-1

5. Перетин (intersection) – $R_1 \cap R_2$

Перетином двох сумісних по типу відношень R_1 і R_2 називається відношення з тим же заголовком, що і у відношень R_1 і R_2 та тілом, що складається з кортежів, що належать одночасно обом відношенням R_1 і R_2. Таким чином, операція перетину двох відношень дає відношення, що включає всі кортежі, що входять в обидва відносини-предиката. Синтаксис такий:

$$R_1 \textit{ intersect } R_2 \quad \textit{або} \quad R_1 \cap R_2$$

Приклад.

Відомість4 = Відомість1 intersect Відомість2

Таблиця 2.16 – Відношення «Відомість_4»

Студент	Оцінка	Група
Г. Бондарь	6	ІТС-08-1

6. Різниця (set difference або minus) – R_1 minus R_2, $R_1 - R_2$

Різницею двох сумісних по типу відношень R_1 і R_2 називається відношення з тим же заголовком, що і у відношеннях R_1 і R_2 і тілом, що складається з кортежів, що належать відношенню R_1 і не належать відношенню R_2.

Відомість5 = Відомість2 – Відомість1

Відомість6 = Відомість1 – Відомість2

Таблиця 2.17 – Відношення «Відомість_5»

Студент	Оцінка	Група
В. Бахтіаров	2	ІТС-08-2
Д. Костирко	7	ІТП-08
Е. Суліган	8	ІТС-08-1

Таблиця 2.18 – Відношення «Відомість_6»

Студент	Оцінка	Група
А. Анощенков	3	ІТС-08-1

7. З'єднання (Join) – (R_1 times R_2) where c

З'єднанням відношень R_1 і R_2 за умови **c** називається відношення:

(R_1 times R_2) where c

де ***c*** являє собою логічне вираження, до якого можуть входити атрибути відношень R_1 і R_2 і/або скалярні вирази.

Таким чином, операція з'єднання є результатом послідовного застосування операції декартового добутку та вибірки, є логічним продовженням операції декартового добутку, тому користувачів, як правило, цікавить лише деяка частина кортежів декартового добутку, яка задовольняє заданій умові, тобто вибірка.

Якщо в R_1 і R_2 є атрибути з однаковими найменуваннями, то перед виконанням такі атрибути треба перейменувати.

Реляційне числення

Реляційне числення – це приклад загальної мови програмування.

Існує:

– Реляційне числення кортежів.

– Реляційне обчислення доменів.

Реляційне числення кортежів – це мова суджень, тут важливо показати чи є рядок істинний або помилковий? Маючи такий інструмент можна «пробігти» всі кортежі, визначивши їх належність до нової множини.

Основні поняття реляційного числення

Числення засноване на змінних кортежу – це такі змінні, областю визначення яких є вказане відношення, для яких допустимими значеннями можуть бути тільки кортежі даного відношення.

Приклад.
Відношення R – дані про деталі.

Таблиця 2.19 – Відношення R

Код деталі	Назва	Вага
01	A	1
02	D	2
03	B	2
04	C	3

Які деталі мають вагу «два»?
Введемо змінну t, яка визначає рядок (кортеж):
t.вага == 2
Рядок 1 – false, 2 – true, 3 – true, 4 – false.
1) В реляційній алгебрі:
Вибірка – $R_1 = \sigma_{вага=2}(R)$, проекція – $I = \prod_{назва}(R_1)$.

Таблиця 2.20 – Відношення I

Назва
D
B

2) Засобами реляційного числення:
{t.*назва деталі* | t in R and t.*вага* = 2}, де t.*назва деталі* – цільовий список; t in R and t.*вага* = 2 – визначальні вирази.
t.вага = 2 – значення атрибута «вага» в рядку t дорівнює 2.

У загальному вигляді, вираз РЧ має вигляд:
{S | P (S)}, де P – предикат, умова, називається **формулою.**
Тобто у реляційному численні будуються вирази з загальними формулами. Їх всього декілька. Допускаючими формулами можуть бути тільки недвозначні і небеззмістовні послідовності. Визначаються 4 правила:

1) Якщо P є n-арною формулою, а t_1, t_2, t_n – це константи або змінні, то $P(t_1, t_2, ..., t_n)$ є вірним.
2) Якщо t_1, t_2 – це константи або змінні з одного домену, а Θ – це один з операторів порівняння, то вираз $t_1 \Theta t_2$ – правильно побудована формула.
3) Якщо вираження F_1, F_2 є формулами, то їх кон'юнкція $F_1 \cap F_2$ і диз'юнкція $F_1 \cup F_2$ і заперечення $-F_1$ також формули.
4) Використання квантора $\exists$ (існування) і $\forall$ (загальності).

Якщо вираз $F_1(x)$ є формулою з вільною змінною, то вирази $\exists F_1(x)$ і $\forall F_1(x)$ також формули.

У формулах використовуються поняття: вільні і пов'язані входження змінних до формули. Їх можна також називати локальними і пов'язаними.

Квантори існування і загальності відповідають деклараціям. Вони пов'язують входження змінних, знаходяться в сфері їхньої дії.

Квантор існування ∃.

Означає, що існує хоча б один екземпляр певного типу. Квантор ∃ - це аналог операції «з'єднання» (join).

Приклад.

Перерахувати назви деталей, потреби в яких покриваються деталями, що зберігаються на складі.

Таблиця 2.21 – Відношення «Потреби»

Код деталі	Назва	Кількість деталей
01	A	1
02	D	2
03	B	2
04	C	3

Таблиця 2.22 – Відношення «Склад»

Код деталі	Стелаж	Кількість деталей
01	05	100
03	10	250
04	02	2

Таблиця 2.23 – Відношення «Відповідь»

Назва
A
B

Рішення:

Складемо цільової список:

– t.назва_деталі – цільовий список;
– t – рядок відношення «Потреби»;
– s – рядок відношення «Склад».

Запит містить визначальний вираз:

∃*s in* Склад(s.кільк_дет = t.кільк_дет and s.кільк_дет ≥ t.кільк_дет).

Вираз РЧ має вигляд:

{t.назва_деталі | t *in* Потреби *and* ∃*s in*
Склад(*s in* Склад(s.кільк_дет = t.кільк_дет and s.кільк_дет ≥ t.кільк_дет))}

Квантор загальності ∀.

Квантор загальності ∀ (будь-який) означає, що деяка умова застосовується до всіх рядків або до кожного рядка деякого типу. Він відповідає операції ділення реляційної алгебри.

Приклад.

Нехай дано 2 відношення: «Відомість» і «Розклад_іспитів». Потрібно визначити прізвища тих студентів, кожен з яких здавав всі іспити, які перераховані у відношенні «Розклад_іспитів».

Таблиця 2.24 – Відношення «Відомість»

№ Залікової книжки	Прізвище	Дисципліна	Дата
02-Е-1	Іванов	Фізика	10.01.19
02-Е-1	Іванов	Хімія	14.01.19
02-Е-2	Сидоров	Фізика	10.01.19
02-Е-2	Сидоров	Хімія	14.01.19
02-Е-5	Коровін	Хімія	14.01.19

Таблиця 2.25 – Відношення «Розклад_іспитів»

Дисципліна	Дата
Хімія	14.01.19
Фізика	10.01.19

Таблиця 2.26 – Відношення «Відповідь»

Прізвище
Іванов
Сидоров

∀ – For ALL

Вираз РЧ має вигляд:

{*t*.Прізвище | *t in* Відомість *and s in* Розклад_іспитів *and* ∀*S* (*t*.дисц = *t*.дисц)}

Контрольні запитання для самоперевірки

6) Визначити операцію «Вибірка» реляційної алгебри.
7) Визначити операцію «Проекція» реляційної алгебри.
8) Визначити операцію «Декартів добуток» реляційної алгебри.
9) Поясніть застосування реляційного числення в базах даних.
10) Складіть вираз реляційного числення.

2.4 Розробка бази даних у реляційній СУБД MS ACCESS

СУБД Microsoft Access [5] є системою управління базами даних реляційного типу. Дані зберігаються в такій базі у вигляді таблиць, рядки (записи) яких складаються з наборів полий певних типів. З кожною таблицею можуть бути зв’язані індекси (ключі), задаючи потрібні користувачеві порядки на безліч рядків. Таблиці можуть мати однотипні поля (стовпці), і це дозволяє встановлювати між ними зв’язки, виконувати операції реляційної алгебри. Типовими операціями над базами даних є визначення, створення і видалення таблиць, модифікація визначень (структур, схем) існуючих таблиць, пошук даних в таблицях по певних критеріях (виконання запитів), створення звітів про вміст бази даних.

СУБД MS Access дозволяє задавати типи даних і способи їх зберігання. Можна також задати критерії (умови), які СУБД надалі використовуватиме для забезпечення правильності введення даних.

Microsoft Access надає максимальну свободу в завданні типу даних (текст, числові дані, дати, час, грошові значення, малюнки, звук, електронні таблиці). Можна задавати також формати зберігання представлення цих даних при висновку на екран або друк.

Оскільки Microsoft Access є сучасним програмним засобом Windows, можна використовувати в роботі всі можливості DDE (динамічний обмін даними) і OLE (зв'язок і впровадження об'єктів). DDE дозволяє здійснювати обмін даними між MS Access і будь-яким іншим підтримуючим DDE системам для Windows. У Microsoft Access можна за допомогою макросів або Access Basic здійснювати динамічний обмін даними з іншими програмними засобами.

У Microsoft Access для обробки даних базових таблиць використовується мова SQL (структурована мова запитів). Використовуючи SQL можна виділити з однієї або декількох таблиць необхідну для вирішення конкретного завдання інформацію.

Властивості полів бази даних

Бази даних можуть містити різні об'єкти. Основними об'єктами будь-якої бази даних є її таблиці. Проста база даних має хоч би одну таблицю. Відповідно, структура простої бази даних тотожно рівна структурі її таблиці.

Структуру двовимірної таблиці утворюють стовпці і рядки. Їх аналогами в простій базі даних є поля і записи.

Поля бази даних не просто визначають структуру бази – вони ще визначають групові властивості даних, записуваних в осередки, що належать кожному з полів. Нижче перераховані основні властивості полів таблиць баз даних на прикладі СУБД Microsoft Access.

1) Ім'я поля – визначає, як слід звертатися до даних цього поля при автоматичних операціях з базою.
2) Тип поля – визначає тип даних, які можуть міститися в даному полі.
3) Розмір поля – визначає граничну довжину (у символах) даних, які можуть розміщуватися в даному полі.
4) Формат поля – визначає спосіб форматування даних в осередках, що належать полю.
5) Маска введення – визначає форму, в якій вводяться дані в поле.
6) Підпис – визначає заголовок стовпця таблиці для даного.
7) Значення за умовчанням – те значення, яке вводиться в осередки поля автоматично.).
8) Умова на значення – обмеження, використовуване для перевірки правильності введення даних.
9) Повідомлення про помилку – текстове повідомлення, яке видається автоматично при спробі введення в полі помилкових даних.

10) Обов’язкове поле – властивість, що визначає обов’язковість заповнення даного поля при наповненні бази.
11) Порожні рядки – властивість, що вирішує введення порожніх строкових даних
12) Індексоване поле – якщо поле володіє цією властивістю, всі операції, пов’язані з пошуком або сортуванням записів по значенню, що зберігається в даному полі, істотно прискорюються.

Оскільки в різних полях можуть міститися дані різного типа, то і властивості у полів можуть розрізнятися залежно від типу даних. Так, наприклад, список вищезгаданих властивостей полів відноситься в основному до полів текстового типа. Поля інших типів можуть мати або не мати ці властивості, але можуть додавати до них і свої. Наприклад, для даних, що представляють дійсні числа, важливою властивістю є кількість знаків після десяткової коми. З іншого боку, для полів, використовуваних для зберігання малюнків, звукозаписів, відео кліпів і інших об’єктів OLE, більшість вищезгаданих властивостей не мають сенсу.

Типи даних

Бази даних Microsoft Access працюють з наступними типами даних.

1) Текстовий – тип даних, використовуваний для зберігання звичайного тексту обмеженого розміру (до 255 символів).
2) Числовий – тип даних для зберігання дійсних чисел.
3) Поле Мемо – спеціальний тип даних для зберігання великих об’ємів тексту (до 65 535 символів).
4) Дата/час – тип даних для зберігання календарних дат і поточного часу.
5) Грошовий – тип даних для зберігання грошових сум.
6) Лічильник – спеціальний тип даних для унікальних натуральних чисел з автоматичним нарощуванням. Природне використання – для порядкової нумерації записів.
7) Логічний – тип для зберігання логічних даних.
8) Гіперпосилання – спеціальне поле для зберігання адрес URL Web-об’єктів Інтернету.
9) Майстер підстановок – це не спеціальний тип даних. Це об’єкт, настройкою якого можна автоматизувати введення даних в полі так, щоб не вводити їх вручну, а вибирати їх із списку, що розкривається.

Об’єкти бази даних

База даних в MS Access вміщує об’єкти різних категорій, кожній з яких відповідає своя вкладка вікна бази даних: таблиці, запити, форми, звіти, сторінки, макроси і модулі.

1) Таблиці – це основні об’єкти будь-якої бази даних. По-перше, в таблицях зберігаються всі дані, наявні в базі, а по-друге, таблиці зберігають і структуру бази (поля, їх типи і властивості).
2) Запити. Ці об’єкти служать для витягання даних з таблиць і надання їх користувачу в зручному вигляді. За допомогою запитів виконують такі операції як відбір даних, їх сортування і фільтрацію. За допомогою запитів можна виконувати перетворення даних по заданому алгоритму, створювати нові таблиці, виконувати автоматичне наповнення таблиць даними, імпортованими з інших джерел, виконувати прості обчислення в таблицях і багато що інше.
3) Форми. Це засоби для введення даних. Сенс – надати користувачу засоби для заповнення тільки тих полів, які йому заповнювати покладено. Одночасно з цим у формі можна розмістити спеціальні елементи управління (лічильники, списки, що розкриваються, перемикачі, прапорці і інше) для автоматизації введення.
4) Звіти. Призначені для виведення даних.
5) Сторінки. Це спеціальні об’єкти баз даних, реалізованих в останній версії СУБД Microsoft Access. Правда, коректніше їх називати сторінками доступу до даних. Цей об’єкт містить компоненти, через які здійснюється зв’язок переданої Web-сторінки з базою даних, що залишається на сервері. Користуючись цими компонентами, відвідувач Web-вузла може проглядати записи бази в полях сторінки доступу. Таким чином, сторінки доступу до даних здійснюють інтерфейс між клієнтом, сервером і базою даних, розміщеною на сервері. Ця база даних не обов’язково повинна бути базою даних Microsoft Access.
6) Макроси і модулі. Ці категорії об’єктів призначені як для автоматизації операцій, що повторюються, при роботі з СУБД, так і для створення нових функцій шляхом програмування. У Microsoft Access макроси складаються з послідовності внутрішніх команд СУБД і є одним із засобів автоматизації роботи з базою. Модулі створюються засобами зовнішньої мови програмування, в даному випадку мови Visual Basic for Applications. Це один із засобів, за допомогою яких розробник бази може закласти в неї нестандартні функціональні можливості,

задовольнити специфічну вимогу замовника, підвищити швидкодію системи управління, а також рівень її захищеності.

Розробка структури бази даних

З'ясувавши основну частину даних, які замовник споживає або поставляє, можна приступати до створення структури бази, тобто структури її основних таблиць [5].

1) Робота починається зі складання генерального списку полів – він може налічувати десятки і навіть сотні позицій.
2) Відповідно до типу даних, що розміщуються в кожному полі, визначають найбільш відповідний тип для кожного поля.
3) Далі розподіляють поля генерального списку по базових таблицях. На першому етапі розподіл проводять за функціональною ознакою. Мета – забезпечити, щоб введення даних в одну таблицю проводилося, по можливості, в рамках одного підрозділу, а ще краще – на одному робочому місці.
4) У кожній з таблиць намічають ключове поле. Як такого вибирають поле, дані в якому повторюватися не можуть. Якщо в таблиці взагалі немає ніяких полів, які можна було б використовувати, як ключові, завжди можна ввести додаткове поле типу Лічильник – воно не може містити даних, що повторюються, за визначенням.
5) За допомогою олівця і паперу розкреслюють зв'язки між таблицями. Таке креслення називається схемою даних. Існує декілька типів можливих зв'язків між таблицями. Найбільш поширеними є зв'язки «один до багатьох» і «один до одного». Зв'язок між таблицями організовується на основі загального поля, причому в одній з таблиць воно обов'язково повинне бути ключовим, тобто на стороні «один» повинне виступати ключове поле, що містить унікальні значення, що не повторюються. Значення на стороні «багато» може повторюватися.
6) Розробкою схеми даних закінчується «паперовий» етап роботи над технічною пропозицією. Цю схему можна погоджувати із замовником, після чого приступати до безпосереднього створення бази даних.

Слід пам'ятати, що по ходу розробки проекту замовнику неодмінно приходитимуть в голову нові ідеї. На всіх етапах проектування він прагне охопити єдиною системою все нові і нові підрозділи і служби підприємства. Можливість гнучкого використання його побажань багато в чому визначається кваліфікацією розробника бази даних.

Якщо схема даних складена правильно, підключати до бази нові таблиці неважко. Якщо структура бази нераціональна, розробник може випробувати серйозні труднощі і увійти до суперечності із замовником. Суперечності виконавця із замовником завжди свідчать про недостатню кваліфікацію виконавця. Саме по цьому етап попереднього проектування бази даних слід вважати основним.

На цьому етапі завершується попереднє проектування бази даних, і на наступному етапі починається її безпосередня розробка. З цієї миті слід почати роботу з СУБД.

Розробка бази даних в СУБД MS Access

Засобами Microsoft Access розробимо автоматизовану систему керування даними «Автоматизована система обліку навчальних досягнень учнів».

I. Створення таблиць.

Першим кроком при створенні бази даних є проектування таблиць.

Кожна таблиця MS Access має жорстку структуру, тому вся інформація в ній повинна бути чітко систематизована. Для кожної теми в базі даних відводиться окрема таблиця. Це дозволяє уникнути повторень при збереженні інформації та зменшує ймовірність виникнення помилок при введенні інформації. Для створення структури таблиць в режимі *Конструктора* необхідно скористатись наступним алгоритмом:

- активізувати вкладку *Таблиця (Таблица)*;
- обрати засіб створення (в роботі використовується режим *Конструктор*);
- задати імена всіх полів;
- задати властивості полів.

Наведена таблиця “Класи” містить поля: «Назва класу», «Кількість учнів» (рис. 2.6).

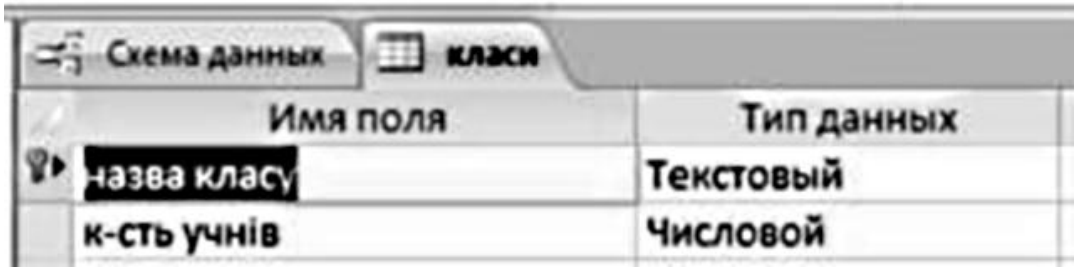

Рисунок 2.6 – Таблиця «Класи» в режимі конструктора

В наступній таблиці «Предмети» містилися такі поля: «Предмет», «Кількість годин» (рис. 2.7).

Схема данных | класи | **Предмети**

Имя поля	Тип данных
Предмет	Текстовый
к-сть годин	Числовой

Рисунок 2.7 – Таблиця «Предмети» в режимі конструктора

В наступній таблиці «Оцінки 1 семестр» містилися такі поля: «№», «клас», «Назва предмету», «ПІП учня», «Оцінка» (рис. 2.8).

Схема данных | класи | Предмети | **Оцінки**

Имя поля	Тип данных
№	Текстовый
клас	Текстовый
Назва предмету	Текстовый
ПІП учня	Текстовый
оцінка1	Числовой
оцінка2	Числовой
оцінка3	Числовой
оцінка4	Числовой
оцінка5	Числовой
оцінка6	Числовой
оцінка7	Числовой
оцінка8	Числовой
оцінка9	Числовой
оцінка10	Числовой
оцінка11	Числовой
оцінка12	Числовой

Рисунок 2.8 – Таблиця «Оцінки 1 семестр» в режимі конструктора

Аналогічну структуру мають таблиці «Оцінки 2 семестр». У цій таблиці поля «Клас» і «Назва предмету» мають тип «Мастер подстановок» і використовують дані із раніше розглянутих таблиць.

II. Побудова зв’язків між таблицями.

Для створення зв’язків між таблицями вручну в *Microsoft Access* є спеціальне діалогове вікно, яке називається *Схема даних* (*Схема данных*).

Для створення зв’язків між таблицями потрібно скористатися алгоритмом зв’язування таблиць у схемі даних:

- вибрати в меню *Сервіс* (*Сервис*) команду *Схема даних* (*Схема данных*);
- додати у вікно схеми даних таблиці, які необхідно зв’язати, шляхом подвійного натиснення лівої кнопки мишки на відповідних назвах таблиць, після чого натиснути кнопку *Закрити* (*Закрыть*) (рис. 2.9);

Рисунок 2.9 – Додавання таблиць

- перетягнути ключове поле основної таблиці до відповідного поля підлеглої;
- для забезпечення цілісності даних ввімкнути прапорець у полі Забезпечення цілісності даних (Обеспечение целостности данных) (рис. 2.10);
- для каскадного поновлення та вилучення даних ввімкнути прапорці в полях Каскадне поновлення зв’язаних полів (Каскадное обновление связанных полей) та Каскадне вилучення зв’язаних записів (Каскадное удаление связанных записей);
- натиснути кнопку OK.

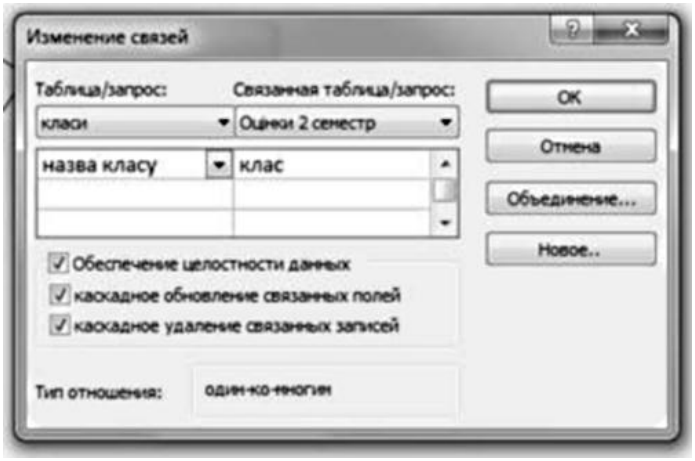

Рисунок 2.10 – Вікно «Изменение связей»

Між даними різних таблиць можна встановити зв’язок, використовуючи однакові значення їхніх полів. Функція зв’язку дає змогу користуватися даними кількох таблиць одночасно.

Поля, що застосовуються для встановлення зв’язку, повинні бути однакового типу і мати однакові значення.

Міжтабличні зв’язки можуть бути таких типів:

– відношення *один до одного.* При такому зв’язку кожному запису першої таблиці відповідає не більш як один запис другої. І, навпаки, один запис другої таблиці відповідає одному запису першої. Відношення між записами встановлюється при збігу значень ключових полів обох таблиць;
– відношення *один до багатьох* – тип зв’язку, що використовується найчастіше. При такому зв’язку кожному запису першої таблиці можуть відповідати кілька записів другої, але один запис другої таблиці не може мати зв’язок із більш як одним записом першої;
– відношення *багато до багатьох* — тип зв’язку, що дає змогу встановити відношення між кількома записами однієї таблиці та кількома записами другої.

III. Створення запитів

Запити є потужним засобом обробки даних, які зберігаються у таблицях СУБД MS Access. За їх допомогою можна переглядати, аналізувати та змінювати дані з кількох таблиць. Вони також можуть використовуватись як джерело даних для форм та звітів. Запити дозволяють обраховувати підсумкові значення і виводити їх у компактному форматі, а також виконувати обчислення над групами записів.

В СУБД MS Access можна створювати такі види запитів:

1) Запит на вибірку – використовується найчастіше. При його виконанні дані, які задовольняють вказаним умовам відбору, вибираються з однієї або кількох таблиць і виводяться у певному порядку. Запит на вибірку також використовується для групових операцій – для обрахування сум, середніх значень, перерахунків та інших дій.
2) Запит з параметрами – критерій відбору задає користувач, вводячи потрібний параметр при виклику запита.
3) Перехресний запит – дозволяє створювати результуючі таблиці на основі розрахунків, отриманих при аналізі групи таблиць. У перехресному запиті відображаються результати статистичних розрахунків (сума, кількість записів, середні значення), які виконуються за даними з одного поля таблиці. Ці результати групуються за двома наборами даних, один з яких розміщений у лівому стовпці таблиці, в другий – у верхній стрічці.
4) Запит на зміну – це запит, який за одну операцію вносить зміни у кілька записів. Існує 4 види запитів на зміну: на видалення, оновлення та додавання записів, а також на створення таблиці:
    a) Запит на видалення видаляє групу записів, які задовольняють вказаним умовам, з однієї або кількох таблиць, при чому видаляти можна лише весь запис, а не лише окремі поля з нього.
    b) Запит на оновлення вносить загальні зміни в групу записів однієї або кількох таблиць.
    c) Запит на додавання додає групу записів з однієї або кількох таблиць в кінець однієї або кількох таблиць.
    d) Запит на створення таблиці створює нову таблицю на основі всіх або частини даних з однієї або кількох таблиць.
5) Запит SQL – це запит, який створюється за допомогою інструкцій SQL. Цей тип запитів є досить складним для користувачів-початківців і використовується зазвичай досвідченими користувачами, які мають навики програмування та роботи із серверами баз даних.

Запити можна створювати самостійно і за допомогою майстра.

Майстри запитів автоматично виконують основні дії залежно від відповідей користувача на поставлені питання. Самостійно створювати запити можна за допомогою конструктора.

Для створення нового запиту потрібно у вікні бази даних вибрати вкладку ЗАПИТИ, клацнути на кнопці СТВОРИТИ і у вікні, що з'явилося, вибрати один з п'яти пунктів: Конструктор, Простий запит, Перехресний запит (перекрёстный запрос), Записи, що повторюються (повторяющиеся записи), Записи без підрядних (записи без подчинённых).

IV. Створення форм.

Форми в Microsoft Access виконують дві основні функції. По-перше, вони дозволяють користувачу оперувати даними, що зберігаються в конкретної СУБД, а по-друге, вони дозволяють зв'язувати різноманітні дискретні модулі СУБД, перетворюючи їх у цілісний закінчений інструмент для визначеної роботи.

Форми дозволяють відображати дані з таблиць і запитів у зручнішому для сприйняття вигляді. Існує можливість створення форм динамічно при виконання програми, проте природним режимом їх створення є режим візуального конструювання. Вибір команди *Форма* в меню *Вставка* виводить на екран вікно *Нова Форма*, що дозволяє задати таблицю або запит, для яких створюється нова форма, і вказати режим її створення. Окрім створення форми «уручну», створення форми можна автоматизувати, використовуючи *Майстер форм* (Мастер форм). Крім того, можна створити спеціальні форми, зокрема з листами даних (Автоформа), діаграмами (Диаграмма) і зведеними таблицями (Сводная таблица) у форматі Excel.

Мова SQL

Мова SQL (Structured Query Language) в даний час отримала дуже широке поширення і фактично перетворився на стандартний мову реляційних баз даних. Стандарт на мову SQL був випущений Американським національним інститутом стандартів (ANSI) в 1986 році, а в 1987 році Міжнародна організація стандартів (ISO) прийняла цей стандарт в якості міжнародного. В даний час мова SQL підтримується багатьма десятками СУБД різних типів.

В ідеалі, будь-яка мова роботи з базами даних повинна надавати користувачу наступні можливості:

- створювати бази даних і таблиці з повним описом їх структури;
- виконувати основні операції маніпулювання даними, такі як вставка, модифікація і видалення даних з таблиць;

– виконувати прості і складні запити, здійснюють перетворення необроблених даних в необхідну інформацію.

SQL є прикладом мови, призначеної для роботи з таблицями з метою перетворення вхідних даних до необхідного вихідного виду. Мова SQL має два основних компоненти:

– мова DDL (Data Definition Language), призначена для визначення структур бази даних;
– мова DML (Data Manipulation Language), призначена для вибірки і поновлення даних.

Мова SQL включає тільки команди визначення та маніпулювання даними – в ньому відсутні будь-які команди управління ходом обчислень. Іншими словами, в цій мові немає команд IF ... THEN ... ELSE, GO TO, DO ... WHILE і будь-яких інших, призначених для управління ходом обчислювального процесу. Подібні завдання повинні вирішуватися за допомогою мов програмування або керування завданнями або інтерактивно, в результаті дій, виконуваних самим користувачем. З причини подібної незавершеності в плані організації обчислювального процесу мова SQL може використовуватися двома способами. Перший передбачає інтерактивну роботу, яка полягає в введенні користувачем з терміналу окремих SQL-операторів. Другий полягає у запровадженні SQL-операторів в програми на процедурних мовах.

Деякі особливості:

– Це не процедурний мова, тому в ньому необхідно вказувати, яка інформація повинна бути отримана, а не як її можна отримати. Інакше кажучи, мова SQL не вимагає вказівки методів доступу до даних.
– Як і більшість сучасних мов, SQL підтримує вільний формат запису операторів. Це означає, що при введенні окремі елементи операторів не пов’язані з фіксованими позиціями екрану.
– Структура команд задається набором ключових слів, що представляють собою звичайні слова англійської мови – такі, як CREATE TABLE (Створити таблицю), INSERT (Вставити), SELECT (Вибрати).
– Мова SQL може використовуватися широким колом користувачів, включаючи адміністраторів баз даних (АБД), керівний персонал

компанії, прикладних програмістів і безліч інших типів кінцевих користувачів.

В даний час для мови SQL існує міжнародний стандарт (ISO, 1992 р.), формально перетворює цю мову в стандартну мову визначення й маніпулювання реляційними базами даних – якою вона фактично і є.

Запис SQL-операторів

SQL-оператор складається із зарезервованих слів, а також зі слів, що визначаються користувачем. Зарезервовані слова є постійною частиною мови SQL і мають фіксоване значення. Їх слід записувати в точності так, як це встановлено, і не можна розбивати на частини для переносу з одного рядка в іншу. Слова, визначені користувачем, задаються самим користувачем (у відповідності з певними синтаксичними правилами) і являють собою імена різних об’єктів бази даних – таблиць, стовпців, подань, індексів і т.д. Слова в операторі розміщуються відповідно установлених синтаксичних правил.

Більшість компонентів SQL-операторів не чутливе до регістру. Одним важливим винятком з цього правила є символьні літерали – дані, які повинні вводитися точно так само, як були введені відповідні їм значення, що зберігаються в базі даних. Наприклад, якщо в базі даних зберігається значення прізвища ‘SMITH’, а в умові пошуку вказаний символьний літерал ‘Smith’, то цей запис не буде знайдений.

Оскільки мова SQL має вільний формат, окремі SQL-оператори і їх послідовності будуть мати більш читабельний вигляд при використанні відступів і вирівнювання. Рекомендується дотримуватися наступних правил:

- Кожна фраза в операторі повинна починатися з нового рядка.
- Початок кожної фрази має бути вирівняні з початком решти фраз оператора.
- Якщо фраза має кілька частин, кожна з них повинна починатися з нового рядка з деяким відступом відносно початку фрази, що буде вказувати на їх підпорядкованість.

Для визначення формату SQL-операторів ми будемо застосовувати наступну розширену форму BNF-нотації (Backus Naur Form).

- Прописні букви будуть використовуватися для запису зарезервованих слів і повинні вказуватися в операторах точно так само, як це буде показано.
- Рядкові букви будуть використовуватися для запису слів, що визначаються користувачем.

- Вертикальна риска (|) вказує на необхідність вибору одного з декількох наведених значень, наприклад, a | b | c.
- Фігурні дужки визначають обов'язковий елемент, наприклад, {a}.
- Квадратні дужки визначають необов'язковий елемент, наприклад, [a].
- Три крапки (...) використовуються для вказівки необов'язкової можливості повторення конструкції, від нуля до декількох разів, наприклад, {a | b} [, c ...]. Цей запис означає, що після a чи b може слідувати від нуля до декількох повторень c, розділених комами.

На практиці для визначення структури бази даних (тобто її таблиць) використовуються DDL-оператори, а для заповнення цих таблиць даними і вибірки з них інформації за допомогою запитів – DML-оператори.

Маніпулювання даними

У цьому розділі обговорюються наступні оператори мови SQL DML:

- SELECT – вибірка даних з бази;
- INSERT – вставка даних в таблицю;
- UPDATE – оновлення (зміна) даних у таблиці;
- DELETE – видалення даних з таблиці.

Зважаючи на складність оператора SELECT і відносної простоти решти DML-операторів, більш детально зупинимось на обговоренні можливостей оператора SELECT і його різних форматів.

Літерали

Літерали являють собою константи, які використовуються в SQL-операторах. Існують різні форми літералів для кожного типу даних, які підтримуються SQL. Вкажемо відмінності між літералами, які слід укладати в одинарні лапки, і тими, які не слід. Всі нецифрові значення даних завжди повинні укладатися в одинарні лапки.

Прості запити

Призначення оператора SELECT складається у вибірці та відображенні даних однієї або більше таблиць бази даних. Це виключно потужний оператор, здатний виконувати дії, еквівалентні операторам реляційної алгебри selection, projection та join, причому в межах єдиної виконуваної команди. Оператор SELECT є найчастіше використовуваною командою мови SQL. Загальний формат оператора SELECT має наступний вигляд:

SELECT [DISTINCT | ALL] {* | [colum_expression [AS new_name]] [,…]}

FROM table_name [alias] [,…]

[WHERE condition]

[GROUP BY column_list] [HAVING condition]

[ORDER BY column_list]

Тут параметр column_expression являє собою ім'я стовпця або вираз з декількох імен. Параметр table_name є ім'ям існуючої в базі даних таблиці або подання, до яких необхідно отримати доступ. Необов'язковий параметр alias – це скорочення, яке встановлюється для імені таблиці table_name. Обробка елементів оператора SELECT виконується в наступній послідовності.

FROM – визначаються імена використовуваної таблиці або декількох таблиць.

WHERE – виконується фільтрація рядків об'єкта відповідно заданим умовам.

GROUP BY – утворюються групи рядків, які мають одне і те ж значення у вказаному стовпці.

HAVING – фільтруються групи рядків об'єкта відповідно зазначеної умови.

SELECT – встановлюється, які стовпці повинні бути присутніми у вихідних даних.

ORDER BY – визначається впорядкованість результатів виконання оператора.

Порядок пропозицій і фраз в операторі SELECT не може бути змінений. Тільки дві пропозиції оператора – SELECT і FROM – є обов'язковими, решта може бути опущена. Операція SELECT є закритою: результат запиту до таблиці являє собою іншу таблицю. Існує безліч варіантів запису даного оператора.

Приклади запитів.

1) Вибірка всіх стовпців і всіх рядків

   SELECT * from Знайомі;

   SELECT Знайомі.* FROM Знайомі; (Access)

2) Вибірка КодЗнайомого, Ім'я, По батькові, Прізвище, Зарплата для всіх рядків

SELECT КодЗнайомого, Ім’я, ПоБатькові, Прізвище, Зарплата FROM Знайомі;

SELECT Знайомі.КодЗнайомого, Знайомі.Ім’я, Знайомі.ПоБатькові, Знайомі.Прізвище, Знайомі.Зарплата FROM Знайомі; (Access)

3) Вибірка всіх імен знайомих (і вибірка без повторень)

SELECT Ім’я FROM Знайомі;

результат виконання запиту містить значення, що дублюються, оскільки, на відміну від операції реляційної алгебри projection, оператор SELECT не виключає значення, що дублюються, при виконанні проекції одного або декількох стовпців. Для видалення з результуючої таблиці рядків, що дублюються, використовується ключове слово DISTINCT.

Відкоригований запит виглядає наступним чином:

SELECT DISTINCT Ім’я FROM Знайомі;

4) Обчислювані поля

Створіть звіт про щомісячної зарплати всіх знайомих зарплату вивести в національній валюті та в у. е. (курс 5:1). Тобто необхідно розділити ЗП на 5.

SELECT DISTINCT Ім’я, По батькові, Прізвище, Зарплата, Зарплата / 5 FROM Знайомі;

Введемо власне позначення поля – Зарплата_уе.

SELECT DISTINCT Ім’я, По батькові, Прізвище, Зарплата, Зарплата / 5 AS ‘Зарплата уе’ FROM Знайомі;

5) Вибір рядків (пропозиція WHERE)

a) Перерахувати знайомих з розміром заробітної плати більше 100 грн.

SELECT DISTINCT Ім’я, По батькові, Прізвище, Зарплата FROM Знайомі

WHERE Зарплата> 100;

У мові SQL можна використовувати наступні оператори порівняння:

= *Рівність;*

< *Менше;*

> *Більше;*

<= *Менше або дорівнює;*

> = *Більше або дорівнює;*

<> *Не дорівнює (стандарт ISO).*

b) Перерахуємо знайомих чоловічої статі

SELECT DISTINCT Ім’я, По батькові, Прізвище, Пол FROM Знайомі WHERE Пол = «м»;

c) Виберемо серед знайомих водіїв і менеджерів

SELECT Ім’я, По батькові, Прізвище, Посада FROM Знайомі

WHERE Посада = «водій» OR Посада = «менеджер»;

6) Використання діапазонів (BETWEEN / NOT BETWEEN) в умовах пошуку.

Знайомі з 82 по 85 роки народження.

SELECT Ім’я, По батькові, Прізвище, ДатаНародження FROM Знайомі

WHERE ДатаНародження Between # 1.1.1982 # And # 31.12.1985 #;

Еквівалентний запис:

SELECT Ім’я, По батькові, Прізвище, ДатаНародження FROM Знайомі

WHERE ДатаНародження> = # 1.1.1982 # And

ДатаНародження <= # 31.12.85 #;

7) Умови пошуку із зазначенням шаблонів (LIKE / NOT LIKE)

Виберемо серед знайомих тих, чий телефон зареєстрований на 24 АТС

SELECT Ім’я, По батькові, Прізвище, ДомашнійТелефон FROM Знайомі WHERE ДомашнійТелефон LIKE ‘24% ‘; (MySQL)

SELECT Ім’я, По батькові, Прізвище, ДомашнійТелефон FROM Знайомі WHERE ДомашнійТелефон LIKE «24 *»; (Access)

8) Використання узагальнюючих функцій мови SQL

Стандарт ISO містить визначення наступних п’яти узагальнюючих функцій.

COUNT – повертає кількість значень у вказаному стовпці.

SUM – повертає суму значень у вказаному стовпці.

AVG – повертає усереднене значення у вказаному стовпці.

MIN – повертає мінімальне значення у вказаному стовпці.

MAX – повертає максимальне значення у вказаному стовпці.

a) Число повнолітніх знайомих

SELECT Count (*) AS Повнолітні FROM Знайомі WHERE ДатаНародження <# 1/1/1984 #;

b) Мінімальна, максимальна, середня, загальна зарплати знайомих.

SELECT MIN (Зарплата) as min, MAX (Зарплата) as max, AVG (Зарплата) as avg,

SUM (Зарплата) as sum FROM Знайомі;

9) Групування результатів (фраза Group By)

Визначте кількість знайомих у різних містах і їх сумарну зарплату, результати розташувати по зростанню ЗП.

SELECT Місто, COUNT (Прізвище) AS ЧИСЛО, SUM (ЗАРПЛАТА) AS ЗП FROM Знайомі
GROUP BY Місто ORDER BY 3 ASC;

10) Обмеження на виконання групування (фраза HAVING)

Фраза Having призначена для використання спільно з фразою GROUP BY для завдання обмежень, що вказуються з метою відбору тих груп, які будуть поміщені в результуючу таблицю запиту.

a) Для кожного міста з числом знайомих більше 1, визначте кількість знайомих і їх сумарну зарплату, результати розташувати по зростанню ЗП

SELECT Місто, COUNT (Прізвище) AS ЧИСЛО, SUM (ЗАРПЛАТА) AS ЗПFROM Знайомі
GROUP BY Місто
HAVING Count (Прізвище)> 1
ORDER BY 3 ASC;

b) Виведення списку міст, які вживаються тільки один раз

SELECT [Місто] FROM [Знайомі] GROUP BY [Місто] HAVING Count (*) <2;

c) Виведення списку міст, які вживаються більше одного разу

SELECT [Місто] FROM [Знайомі] GROUP BY [Місто] HAVING Count (*)> 1;

d) Виведення списку міст, які вживаються більше одного разу і ПІБ проживають в них

SELECT Знайомі.Місто, Знайомі.Ім’я, Знайомі.ПоБатькові, Знайомі.Прізвище
FROM Знайомі
WHERE (((Знайомі.Місто) In (SELECT [Місто] FROM [Знайомі] GROUP BY [Місто] HAVING Count (*)> 1)))
ORDER BY Знайомі.Місто;

11) Багатотабличні запити

a) Звичайне (закрите) з’єднання таблиць виконується за допомогою наступного SQL-оператора

SELECT Прізвище, Ім’я, По батькові, b.Назва_группи
FROM Знайомі a, Група b
WHERE a.Код_группи = b.Код_группи;

b) Ліве відкрите з’єднання

Перерахуйте всіх знайомих по групах, а також всіх інших знайомих які не належать жодній групі

```
SELECT Прізвище, Ім’я, По батькові, b.Назва_группи
FROM Знайомі a LEFT JOIN Група b
ON a.Код_группи = b.Код_группи;
```

c) Праве відкрите з’єднання

Перерахуйте всіх знайомих по групах, а також назви всіх інших груп. Використовуємо праве відкрите з’єднання цих двох таблиць, яке виглядає наступним чином:

```
SELECT Прізвище, Ім’я, По батькові, b.Названіе_группи
FROM Знайомі a RIGHT JOIN Група b
ON a.Код_группи = b.Код_группи;
```

d) Повне відкрите з’єднання

```
SELECT Прізвище, Ім’я, По батькові, b.Названіе_группи
FROM Знайомі a FULL JOIN Група b
ON a.Код_группи = b.Код_группи;
```

Контрольні запитання для самоперевірки

1) Опишіть послідовність дій при розробці бази даних в MS Access.
2) Які типи даних визначені в MS Access?
3) Як створити таблицю в MS Access?
4) Як створити схему даних із зазначенням потужності зв’язку між таблицями в MS Access?
5) Як створити форму в MS Access?
6) Які типи запитів існують в MS Access?

3 КУРСОВА РОБОТА

Тема роботи: Проектування й розробка бази даних з використанням реляційної моделі даних.

3.1 Завдання

Спроектувати й розробити базу даних з використанням реляційної моделі даних згідно варіанту, якій обирається за номером в журналі.

1. Виконати наступні етапи життєвого циклу бази даних:
   1.1. Планування розробки БД.
   1.2. Визначення вимог до ІС.

1.3. Проектування БД.

1.4. Концептуальне проектування. Створити ER- модель (модель сутність-зв’язок) за нотацією Чена. Проаналізувати типи зв’язків між об’єктами. Ступінь відношень – не менш п’яти.

Визначити первинні ключі відношень і типи зв’язків між відношеннями.

2. Виконати процес нормалізації (1НФ, 2НФ, 3НФ) на основі аналізу функціональних залежностей.
3. Розробити базу даних у будь-який реляційній СУБД. Створити таблиці. Встановити зв’язки між таблицями. Заповнити кортежі (не менш десяти записів). Створити запити (не менш п’яти).
4. Скласти 4 запити: з використанням теорії реляційної алгебри (2 запити) та реляційного числення (2 запити) згідно варіанту.
5. Скласти звіт:
   5.1. Титульний аркуш (стандартний).
   5.2. Мета роботи.
   5.3. Завдання.
   5.4. Короткі теоретичні відомості.
   5.5. Опис виконаної роботи (отриманих результатів) по пунктах.
   5.6. Висновки.
   5.7. Література

3.2 Варіанти

№ вар.	Предметні області для проектування бази даних
1	Автовокзал
2	Автосалон (продаж автомобілів)
3	Автосервіс
4	Авіакомпанія
5	Аеропорт
6	Аптека
7	Аптекоуправління
8	Бібліотека (книги)
9	Бібліотека (періодика)
10	Бібліотека (персонал)
11	Вантажні перевезення
12	Ветлікарня

№ вар.	Предметні області для проектування бази даних
13	Ветпритулок
14	Видавництво
15	Вирощування та продаж квітів
16	Гастроном
17	Гороскоп
18	Готель
19	Екскурсії по місті
20	Залізничний вокзал
21	Зимовий туризм
22	Картинна галерея
23	Кінні перегони (дербі)
24	Кінопрокат у місті
25	Книжковий магазин
26	Колектив театру
27	Косметичний салон
28	Маршрутні таксі
29	Масажний кабінет
30	Меблевий магазин
31	Морський круїз
32	Морські вантажні перевезення
33	Музеї міста
34	Надання кредитів
35	Олімпіада
36	Пекарня
37	Перукарня
38	Поліклініка
39	Попередній продаж квитків
40	Порт (морський торгівельний)
41	Працевлаштування
42	Продуктова гартівня
43	Прокат CD–дисків
44	Прокат автомобілів
45	Прокат відеофільмів
46	Прокат турспорядження
47	Ралі
48	Реабілітаційне відділення

№ вар.	Предметні області для проектування бази даних
49	Регата
50	Ремонт автомобілів
51	Санаторій
52	Театри міста
53	Туристична агенція
54	Туристична база
55	Туристичні фірми міста
56	Універмаг
57	Фотосалон
58	Футбольна команда
59	Чемпіонат по футболу
60	Облік успішності навчальної групи.
61	Розклад занять навчальної групи.
62	Відділ кадрів підприємства.
63	Дилерська мережа з продажу майна
64	Облік продукції компанії–виробника
65	Прогноз погоди
66	Дилерська мережа з косметики
67	Дилерська мережа з продажу вин
68	Студмістечко
69	Каталог музичних дисків
70	Каталог товарів (OTTO)
71	Танцювальний клуб
72	Вітрильний клуб
73	Вітрильна регата
74	Ліга чемпіонів з футболу
75	Кіностудія (виробництво фільмів)

ВИСНОВКИ

Методичні вказівки призначені для студентів спеціальності 122 «Комп'ютерні науки» заочної форми навчання.

Приведена технологічна карта для дисципліни «Організація баз даних та знань».

Розглянуті основні питання дисципліни «Організація баз даних та знань»: проектування баз даних, реляційна модель даних, питання реляційної алгебри і реляційного числення стосовно до баз даних.

Наведено короткі відомості з розробки бази даних в системі MS Access.

Матеріал, який викладено, призначений для самостійної роботи студентів заочної форми навчання і може бути використаний для виконання курсової роботи з дисципліни «Організація баз даних та знань».

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Коннолли Т., Бегг К., Страчан Ф. Бази данных. Проектирование, реализация и сопровождение. Теория и практика, 2-е изд. :Пер. С англ.: Уч. пос. – М. : Издательский дом «Вильямс», 2000. – 1120 с.
2. Мейер Д. Теория реляционных баз данных. – М. Наука. 1987. 608 с.
3. Малыхина М. П. Базы данных: основы, проектирование, использование. – СПб.: БХВ-Петербург, 2004. – 512 с.
4. Рудикова Л.В. Базы данных. Разработка приложений. – СПб.: БХВ-Петербург, 2006. – 496 с.
5. Проектирование баз данных. СУБД Microsoft Access: учебное пособие для вузов/ Н.Н. Гринченко, Е.В. Гусев, Н.П. Макаров и др. – М.: Горячая линия – Телеком, 2004. – 240с.