ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ВІННИЦЬКИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ ІНСТИТУТ

СИСТЕМА УПРАВЛІННЯ ЯКІСТЮ
Сертифікована на відповідність ДСТУ ISO 9001:2015 (ISO 9001:2015, IDT)

Кафедра економічної кібернетики та інформаційних систем

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ

ORGANIZATION OF DATABASE AND KNOWLEDGE

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО САМОСТІЙНОЇ РОБОТИ

Ступінь вищої освіти	«бакалавр» / «bachelor»
Галузь знань	12 «Інформаційні технології» / «Information technologies»
Спеціальність	126 «Інформаційні системи та технології» / «Information systems and technologies»
Освітня програма	«Інформаційні технології у бізнесі» / «Information technologies in business»

Вінниця 2022

Розробник: Романюк Вадим, доктор технічних наук, професор

Обговорено та схвалено на засіданні кафедри економічної кібернетики та інформаційних систем 31 жовтня 2022 р., протокол № 11; на засіданні методичної комісії факультету економіки, менеджменту та права 25 листопада 2022 р., протокол № 08.

Рецензент: Мерінова Світлана, кандидат економічних наук, доцент

Редактор: Фатєєва Т.
Комп'ютерна верстка: Тимощук М.

Підп. до друку 20.01.2023. Формат 60x84/16. Папір офсетний
Друк ксероксний. Ум. друк. арк. 2,03.
Обл.-вид. арк. 1,76. Тираж 2. Зам. № 05.

Редакційно-видавничий відділ ВТЕІ ДТЕУ
21000, м. Вінниця, вул. Хмельницьке шосе, 25

1. ЗАГАЛЬНІ МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО САМОСТІЙНОЇ РОБОТИ

Самостійна робота здобувачів є однією із форм організації навчання та засобом засвоєння навчального матеріалу у вільний від обов'язкових занять час. Організація та проведення самостійної роботи здобувачів вищої освіти інституту регламентуються Положенням про самостійну роботу студентів ВТЕІ ДТЕУ.

Методичні рекомендації для самостійної роботи з дисципліни «Алгоритми і структури даних» призначені для здобувачів освітнього ступеня «бакалавр» спеціальності «Інформаційні системи та технології» освітньої програми «Інформаційні технології у бізнесі». Метою вивчення дисципліни є формування у майбутніх бакалаврів з інформаційних систем та технологій необхідного рівня інформаційної та комп'ютерної культури, здобуття студентами теоретичних знань і набуття практичних навичок з основ створення та функціонування програмних систем, реляційних та логічних баз даних, сховищ даних; набуття навичок маніпулювання даними та опанування принципами створення SQL запитів; формування уміння приймати управлінські рішення на основі аналізу інформації в базах даних і сховищах даних, прогнозувати ситуацію, підтримувати безпеку та цілісність даних.

У запропонованих методичних рекомендаціях продемонстровано використання технологій та методології роботи з базами даних, модифікації бази даних; представлена система генерації запитів та мова запитів (стандарт) SQL; продемонстровано можливості маніпулювання даними та принципи створення SQL запитів, сортування результатів, групування результатів; розглянуто технологію та методологію роботи з базами даних, модифікації бази даних: створення, видалення, редагування; встановлено порядок організації самостійної роботи студентів.

Опанувавши дану дисципліну, студенти набувають навички актуалізації запитів і управління ними; формують здатність використання запитів для аналізу даних; підтримки заходів безпеки засобами мови SQL.

Методичні рекомендації до самостійної роботи мають теоретичний та прикладний виміри, спрямовані на поглиблене самостійне оволодіння теоретичних відомостей та можуть бути використані для виконання практичних завдань як під час аудиторних занять, так і при самостійній роботі.

Для зручності користування методичними рекомендаціями наведені методичні рекомендації по вивченню конкретних тем, допоміжні матеріали для опанування питань, які виносяться на самостійне опрацювання, рекомендовані джерела, перелік індивідуальних завдань та ключові слова. Для закріплення самостійно опрацьованого матеріалу здобувач вищої освіти може використати питання для самоконтролю, які запропоновані після кожної теми.

Практична реалізація завдань з теми сприятиме розвитку ініціативи та самостійності, аналітичного мислення майбутніх фахівців, формуванню вмінь аналізувати дані у базах даних та сховищах даних засобами мови SQL та на основі аналізу приймати правильні рішення, прогнозувати ситуацію, підтримувати безпеку та цілісність даних.

2. ТЕМАТИЧНИЙ ПЛАН ДИСЦИПЛІНИ «ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ» для здобувачів освітнього ступеня «бакалавр» галузі знань 12 «Інформаційні технології» спеціальності 126 «Інформаційні системи та технології» освітньої програми «Інформаційні технології у бізнесі»

Назва теми	Кількість годин				Форми контролю
	Усього годин / кредитів	з них			
		лекції	лабор. заняття	самост. робота студ.	
Тема 1. Інтелектуальні системи, засновані на знаннях	13	2	4	7	УО, СУН
Тема 2. Напрями та технології створення інтелектуальних систем	13	2	4	7	Т, Р
Тема 3. Характеристика сучасних баз даних та систем управління базами даних	13	2	4	7	ПО, СУН
Тема 4. Сутність реляційного підходу до проектування баз даних	13	2	4	7	ІЗ, ДК, УО
Тема 5. Адміністрування даних та адміністрування баз даних	13	2	4	7	ІЗ, ДК, Т, СУН
Тема 6. Система генерації запитів та мова запитів (стандарт) SQL	13	2	4	7	ІЗ, ДК, СУН
Тема 7. Запити мови SQL для вибірки даних	13	2	4	7	ІЗ, ДК, СУН
Тема 8. Запити мови SQL для визначення та обробки даних	13	2	4	7	ІЗ, ДК, ПО, Т
Тема 9. Підтримка цілісності даних у СУБД	13	2	4	7	ІЗ, ДК, ПО
Тема 10. Принципи створення клієнт-серверних баз даних з веб-інтерфейсом	13	2	4	7	Р, СУН
Тема 11. Концепція побудови сховищ даних	13	2	4	7	Р, ПО
Тема 12. Засоби створення сховищ даних	13	2	4	7	Т, ПО, СУН
Тема 13. Сучасні СУБД та сховища даних, інтелектуальний аналіз даних	13	2	4	7	Р, Т, СУН
Тема 14. Інформаційні системи ділового адміністрування	11	2	4	5	ІЗ, Т, ПО
Разом за навчальний рік	**180/6**	**28**	**56**	**96**	
Підсумковий контроль	**Екзамен**				

Умовні позначення:
УО/ПО – усне/письмове опитування; Т – тестування; Р – реферат;
ІЗ – індивідуальне завдання; ДК – розрахунки та дослідження з використанням комп'ютера,
СУН – використання системи управління навчанням Moodle

3. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИВЧЕННЯ КОНКРЕТНИХ ТЕМ

ТЕМА 1. ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, ЗАСНОВАНІ НА ЗНАННЯХ

Процес пошуку в Інтернеті

Ефективний пошук інформації в Інтернеті через збільшення обсягу і розосередження джерел стає складнішим. При цьому критичним є не стільки час пошуку, скільки відбір інформації, що релевантна запиту користувача. Пошук в Інтернеті ускладнює різноманіття форматів подання інформації, стихійність організації ІР і слабку виразність запитів. Динамічність інформаційного середовища Інтернету, що вимагає постійного відновлення відомостей про наявність і місце розташування інформації, також ускладнює задачу.

Процес пошуку в інформаційному середовищі будемо розглядати як послідовну взаємодію моделі інформації, яку прагне отримати користувач, з моделями інформаційних ресурсів. Модель інформаційного середовища, в якому здійснюється процес інформаційного пошуку, включає модель ІР, що описує властивості і правила роботи з ІР, та модель засобів отримання інформації про середовище та впливу на нього.

Інформаційний пошук – процес співставлення запиту користувача з даними про ІР, відомими інформаційно-пошуковій системі (ІПС), до якої надійшов цей запит.

Відомості про інформаційне середовище можна отримати в результаті виконання процесу пошуку. Моделлю інформації, яку намагається одержати користувач, є запит.

Поширені типи пошуку:

- пошук документа, інформація про який вже міститься в БД агента (приміром, ІПА має перевірити, чи змінилася html-сторінка, яка відповідає певній URL-адресі);
- пошук документа у рамках певної підобласті Інтернету (приміром, ІПА має здійснити перевірку усіх гіперпосилань, що містяться у певному html-документі або групі документів, пошук в електронній бібліотеці, пошук на сайті, що має локальну ПМ);
- пошук документів в Інтернеті (за допомогою ПМ, каталогів тощо);
- пошук нових сайтів та інших ІР.

Модель інформаційного середовища ІПА містить засоби впливу на навколишнє середовище, але ці засоби не є принциповими для вирішення проблеми пошуку і мають тільки допоміжне значення. Приміром, відвідання користувачем якогось сайту, посилання на який містилось у результатах пошуку, фіксується лічильником й впливає на статистичні дані, що використовуються для подальших модифікацій цього сайту. Інший приклад – реєстрація користувача на знайденому сайті та заповнення різноманітних анкет. Ця інформація теж може змінювати інформаційне середовище, але досить слабо.

Приклади застосування агентних технологій для пошуку інформації

Зараз багато розробників ПЗ звертаються до використання агентно-орієнтованих технологій для створення інформаційнопошукових механізмів. На сьогодні розроблено багато ІПА, більшість з яких – безкоштовні. Вони різняться за рівнем інтелектуальності, ІПМ, інтерфейсом тощо.

Розглянемо кілька відомих мультиагентних ІПС.

Copernic (www.copernic.com), працює з багатьма ІПС, має зручний інтерфейс користувача і дозволяє перевіряти доступність знайдених посилань. Посилання, що дублюються, відкидаються, а інші упорядковуються за рівнем релевантності. Недоліки програми – неможливість відключення реклами і те, що інформація про нові ІПС може додаватися тільки розробником.

Більш вдалою в цьому відношенні є програма ***SSSpider (Subject Search Spider)***, яку її розробники відносять до ІПА другого покоління: результати, отримані від пошукових машин, відразу перевіряють на доступність. Некоректні і дубльовані посилання знищуються, після цього програма перевіряє наявність на знайдених сторінках ключових слів і готує звіт на основі актуальної інформації. Програма дозволяє користувачу самостійно підключати пошукові служби. Ця програма, також як і Copernic, може обробляти тільки прості запити (можна шукати всі ключові слова, будь-яке з них або фразу цілком). Проте у багатьох випадках зручніше використовувати формальні запити, створені за допомогою логічних операторів). Проблема полягає в тому, що синтаксис таких запитів не стандартизований і у різних ПМ використовуються різні позначення (приміром, логічне “і“ позначають як “&”, “^”, “+” або “AND”).

Mata Hari (www.thewebtools. com) забезпечує роботу з формальними запитами. Запит, розмічений за допомогою логічних операторів, дужок і лапок, може бути переданий до багатьох ПМ. Якщо якась з них не розуміє формальних запитів, їй направляється звичайний запит, а логічна обробка здійснюється локально. Mata Hari працює аналогічно SSSpider: автоматично завантажує знайдені сторінки і перевіряє їх актуальність та релевантність. Головний недолік програми – нерозуміння української та російської мов. Пошук з використанням усіх припустимих ПМ займає багато часу, тому можна вказувати максимальну кількість посилань, що повертаються кожною ПМ, мінімальний і максимальний розмір документів, дату останньої зміни тощо. Можна використовувати певну підмножину ПМ. Для цього виділені спеціальні групи ПМ, у які ІПС об'єднані за тематикою (Computers, Business-Finances, Games, Graphics тощо) або за деякими іншими принципами (існує, наприклад, група Starting Point, до якої входять найбільш популярні ПМ, та група Metasearchers, яка поєднує метапошукові системи). Користувачам дозволяється створювати власні групи. Ще одна корисна властивість цієї системи – якщо на знайденій сторінці є гіперпосилання, що можуть зацікавити користувача, то вони також будуть перевірені на відповідність запиту.

Kenjin, на відміну від інших інтелектуальних ІПА, працює не з ключовими словами, а з цілим текстом.

Tryllian (www.tryllian.com) – компанія, яка спеціалізується на виробництві Інтернет-застосунків, розробила систему для створення ***MAC Tryllian Agent Developer Kit (http://www.tryllian. com/development/index.html).***

Cybion (www.cybion.com) – французька IT-компанія, яка займається розробкою і впровадженням інтелектуальних програмних продуктів для пошуку, аналізу і менеджменту інформації. Її продукція – адаптовані під потреби конкретного користувача інтелектуальні сервіси Cybion Eye, Cybion Report тощо.

ТОВ "НейрОК" (www.neurok.ru) – розроблювач інтелектуальних систем обробки інформації на основі нейронних мереж, у тому числі ПА, які шукають інформацію в Інтернеті й аналізують результати пошуку. Розробки базуються на платформі NeurOK Semantic Suite – наборі серверів застосувань і програмних модулів, що вирішують найпоширеніші задачі по доставці, сортуванню, збереженню і пошуку текстової інформації. Для організації діалогу з користувачем використовуються семантичні технології НейрОК. Однією з функцій серверного компонента NeurOK Semantic Engine є побудова за будьяким фрагментом тексту набору асоціативно пов'язаних з ним слів.

BinNet (www.binnetcorp.com) спеціалізується на виробництві Java- і Prolog-застосувань, а також інструментарію для виробництва інтелектуальних мобільних програмних продуктів для Інтернету (Intelligent Mobile Agent Programming ToolKit).

Мультиагентна ІПС ***Autonomy*** забезпечує користувачів інтелектуальними засобами пошуку інформації, релевантної його інтересам. Система дозволяє впорядковувати знайдені в Інтернеті документи за темами і автоматизувати сам процес пошуку.

MAC Autonomy – набір ПА для інтелектуального пошуку й обробки інформації, що організовані в рамках спеціалізованої програмної оболонки. Система більше орієнтована на кінцевих користувачів, а не на спеціалістів у певній галузі. Інтерфейс системи Autonomy досить простий та інтуїтивно зрозумілий. Запит на пошук інформації в системі Autonomy задається у вигляді природномовного тексту. Система автоматично аналізує цей текст і здобуває з нього зміст. У цій ІПС використовується технологія нейронних мереж, а також метод подання ПрО із застосуванням алгоритмів розпізнавання образів і обробки сигналів. При цьому системою формується образ документа, що є релевантним запиту, який використовується на наступних етапах пошуку. Пошук здійснюється за допомогою методів нечіткої логіки. В основі пошукового алгоритму лежить механізм динамічних міркувань. Важливою особливістю Autonomy є наявність режиму навчання пошукових агентів.

Intelliseek (www.intelliseek.com) – провідний виробник інтелектуальних on-line пошукових систем (Intelligent Applications™), орієнтованих на конкретного користувача, які забезпечують фільтрацію інформації, текстовий аналіз, навчання ПА, тематичну автоматичну категоризацію. Найвідоміша розробка цієї компанії – ***пошукова систма BullsEye®,*** яка працює на стороні клієнта.

Інтеграція агентних технологій до набору технологій пошуку та трекінгу iAsk™, що забезпечують знаходження відповідей на питання природною мовою, призвело до створення потужного пошукового інструментарію, який забезпечує кінцевому користувачу можливості задавати спеціалізовані питання та отримувати спеціалізовані відповіді замість набору можливих відповідей. Це дозволяє обробляти не тільки ліцензований фінансовий контент та ринкові повідомлення, а також релевантні зв'язки з відповідними Web-сайтами.

Інтелектуальний пошуковий агент ***Personal Finder*** здійснює пошук потрібної користувачу інформації як у Інтернет/Інтранет, так і на локальному комп'ютері користувача. Під час пошуку агент здійснює опитування спеціалізованих ПМ, що працюють у відповідній ПрО.

Слід вказати ще кілька відомих досить простих метапошукових ПА. Кожний з них працює з кількома ІПС і видає загальний покажчик знайдених посилань. ***WebCompass*** і ***WebSeeker*** дозволяють складати розклад для періодичного виконання та відновлення пошуку. ***EchoSearch*** корпорації Iconovex працює швидше за рахунок малої кількості ПМ, до яких він звертається. ***WebSeeke***r фірми ForeFront Group дозволяє виконувати ітеративний пошук.

Розробка ***TransCom Software – пошуковий агент BeeLine (www.beeline4oz.com)*** відрізняється простотою експлуатації і зручною системою настроювань. BeeLine підтримує інтерфейс п'ятьма мовами, у програму убудовані автоматична перевірка орфографії і словник синонімів. Після обробки запиту знайдені посилання автоматично перевіряються на доступність і ті, що не існують або дублюються, видаляються. BeeLine дозволяє виконувати тільки прості запити, однак функція Exclude дозволяє вказувати слова, які повинні бути відсутніми у результатах пошуку.

Інтелектуальний ***ІПА DigOut4U (www.arisem.com/en/index.html)*** французької компанії Arisem підтримує природну мову (англійську і французьку). Характерна риса технології пошуку запитуваних документів – семантичний аналіз.

Інтелектуальний ***ІПА DirectSeek (www.directseek.net)*** одночасно опитує сотні ПМ і надає користувачеві добірку матеріалів з високим ступенем релевантності. Тематично орієнтовані ПА, реалізовані в DirectSeek, забезпечують глибокий пошук інформації з тематики запиту користувача, досліджуючи сайти, конференції, форуми, архіви новин, on-line аукціони і БД.

CyberAlert (www.cyberalert.com/overview.html) – цілком автоматичний Інтернет-сервіс, що знаходить потрібну інформацію та регулярно відслідковує більш 3500 web-публікацій, 63000 конференцій і форумів. Текстовий аналіз і потрійна фільтрація дозволяють досягти високої релевантності результатів пошуку.

MySimon (www.mysimon.com) – інтелектуальний ПА, побудований за технологією "The Virtual Learning Agent", призначений для пошуку в on-line магазинах. MySimon здійснює інтелектуальний пошук, порівнюючи ціни товарів у різних on-line магазинах. Агент має простий інтерфейс, здійснює швидкий і релевантний пошук.

MP3-Wolf (www.trellian.com/mwolf/index.html) – пошуковий робот, що сканує Інтернет у пошуках потрібних користувачу мультимедійних ресурсів (mp3, midi, wave і інших музичних файлів). Робот працює в режимі реального часу і здатний знаходити, сортувати й аналізувати десятки тисяч музичних файлів і посилань у годину, відкидаючи при цьому застарілі посилання. Він може досліджувати кілька сайтів одночасно.

ПА Clickthebutton (www.clickthebutton.com) призначений для полегшення покупок в режимі on-line. Коли користувач знаходить товар, який він бажає купити, Clickthebutton пропонує список інших сайтів, що продають даний продукт, указуючи також і ціну. Агент здатний порівнювати ціни на продукти у різних Інтернет-магазинах.

Система ***MARRI*** призначена для пошуку Web-сторінок, релевантних запитам у певній ПрО. У MARRI використовується онтологічне подання знань. Онтологія розуміється як множина концептів і зв'язків між ними. MARRI містить кілька типів ПА: агенти-брокери, агенти мережі, інтерфейсні і спеціалізовані агенти. Агенти мережі використовують для попереднього добору інформації стандартні ПМ. Потім спеціалізовані агенти аналізують отримані ІР, а після цього порівнюють їх з онтологією ПрО. Користувачу надаються тільки ті ІР, що успішно пройшли цю онтологічну перевірку.

Кожний з ПА має наступні властивості:

- є автономною Java-програмою з власною мережною URL-адресою;
- взаємодіє з іншими агентами за допомогою мови ACL (Agent Communication Language);
- є споживачем і постачальником інформації залежно від того, з якими агентами системи він спілкується;
- може взаємодіяти з такими автономними програмними компонентами, як браузери, аналізатори природної мови та онтологічні БД.

Система ***OntoSeek*** розроблена для здобуття інформації з “жовтих сторінок” і каталогів у режимі on-line. Для точного опису ІР використовується обмежена кількість термінів природної мови. Для подання запитів і опису ресурсів замість списків типу “атрибут значення” використовуються прості концептуальні графи, для яких проблема контекстного ототожнення зводиться до керованого онтологією пошуку в графі. Якщо онтологія показує, що між вузлами і дугами існує задане відношення, то вони порівняні. Система базується на лінгвістичній отології, тому вузли графа мають бути прив'язані до відповідних лексичних одиниць.

Проект ***SAIRE*** – це заснований на агентах механізм інформаційного пошуку. Він забезпечує інтегрований доступ користувачів до розподілених джерел даних, у тому числі – до електронних бібліотек NASA і NOAA. SAIRE дозволяє задавати запити природною мовою (мова системи – англійська) з різних розділів науки, використовуючи спеціальні терміни.

Amalthaea – MAC для фільтрації, виявлення і спостереження ІР. Взаємодія агентів здійснюється відповідно до ринкової економічної моделі. Amalthaea може вивчати інтереси конкретних користувачів і пристосовуватися до них.

Racing (Rational Agent Coalitions for INtelliGent Mediation of Information Retrieval on the Net – http://www.zsu.zp.ua/racing) – набір Web-сервісів, який здійснює раціональний інформаційний пошук. Система базується на агентних технологіях. В проекті Racing використовується методологія семантичного перетворення початкового пошукового запиту користувача у формі ключових слів або фраз до результуючого запиту, складеного з відповідних термінів онтології ПрО. Проект застосовує парадигму раціонального агентства (Rational Agency Paradigm – RAP) для розподілених, автономно створених, довільно структурованих та формалізованих ІР, що є частиною наступного покоління Semantic Web. У проекті для виконання пошуку використовуються онтології, подані у форматі DAML+OIL. В рамках проекту розроблена методологія раціонального посередництва.

Розглянуті вище приклади показують, що для пошуку інформації в Інтернеті ефективним є використання МАС, у яких спеціалізовані ПА взаємодіють один з одним для спільного рішення поставленої користувачем цілі. Інформаційний пошук полягає у взаємодії між користувачами та постачальниками інформації, в процесі якої у відповідь на інформаційний запит користувача постачальник інформації надсилає йому певні відомості, що релевантні цьому запиту.

.

І. Питання до самостійного вивчення

1. Метод потоків даних та особливості його використання.
2. Інтелектуальні програмні агенти.
3. Інформаційна економіка як сфера застосування агентних технологій.

Рекомендовані джерела:

Основні: 2, 5.
Додаткові: 9, 10.
Інтернет-ресурси: 12.

ІІ. Перелік індивідуальних завдань

Знайти приклади в Інтернеті застосування агентних технологій для пошуку інформації в е-бізнесі.

ІІІ. Питання для самоконтролю

1. Навести приклади програмних агентів електронної медицини.
2. Навести приклади програмних агентів електронного урядування.
3. Навести приклади програмних агентів дистанційної освіти.
4. Навести приклади програмних агентів електронних ринків, е-комерції та програмних агентів логістики.

ТЕМА 2. НАПРЯМИ ТА ТЕХНОЛОГІЇ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

Проектування БД

Мета завдання - придбання навичок аналізу предметної області і побудови концептуальної, логічної та фізичної моделі.

Короткі теоретичні відомості

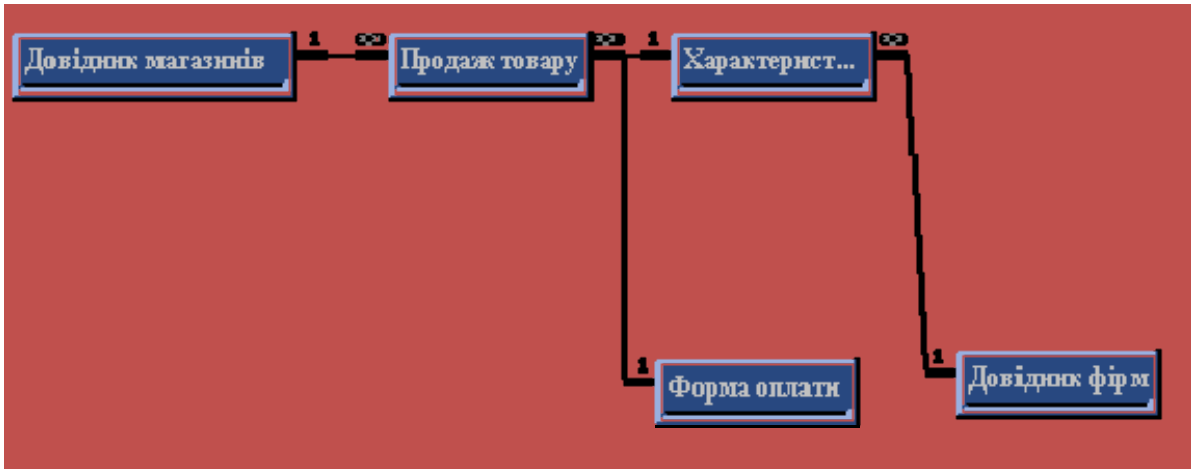
Загальна постановка задачі проектування схеми реляційної БД.

Для проектування реляційної бази даних потрібно:

1. Визначити об’єкти, які містяться в базі даних і зв’язки між об’єктами.
3. Визначити основні властивості об’єктів.
4. Визначити зв’язки між властивостями об’єктів.
5. Створити робочий словник даних для визначення таблиць, що входять до бази даних.
6. Визначити відношення між таблицями баз даних, і включити цю інформацію до словника даних.
7. Продумати операції, що виконуються при створенні та зміні інформації таблиць, включаючи забезпечення цілісності даних.
8. Визначити, як використовувати індекси для прискорення виконання запитів, щоб уникнути сильного уповільнення роботи і надмірного збільшення об’єму дискового простору, що займається базою.
9. Визначити користувачів, яким дозволений доступ до даних.
10. Описати структуру бази даних в цілому, завершити створення словників даних, розробити процедури для операцій з базою даних, включаючи створення резервних копій і відновлення вихідних файлів.

Концептуальне, (інфологічне) проектування – модель створюється на основі аналізу інформації, записаної в специфікації користувача, і сформованої в ході підготовки до проектування користувальницької моделі БД. У результаті цього визначаються об'єкти, властивості і зв'язки об'єктів, які мають відбиватися при проектуванні БД.

Концептуальна модель - *модель об`єктів з елементами даних та взаємозв`язками між ними.*



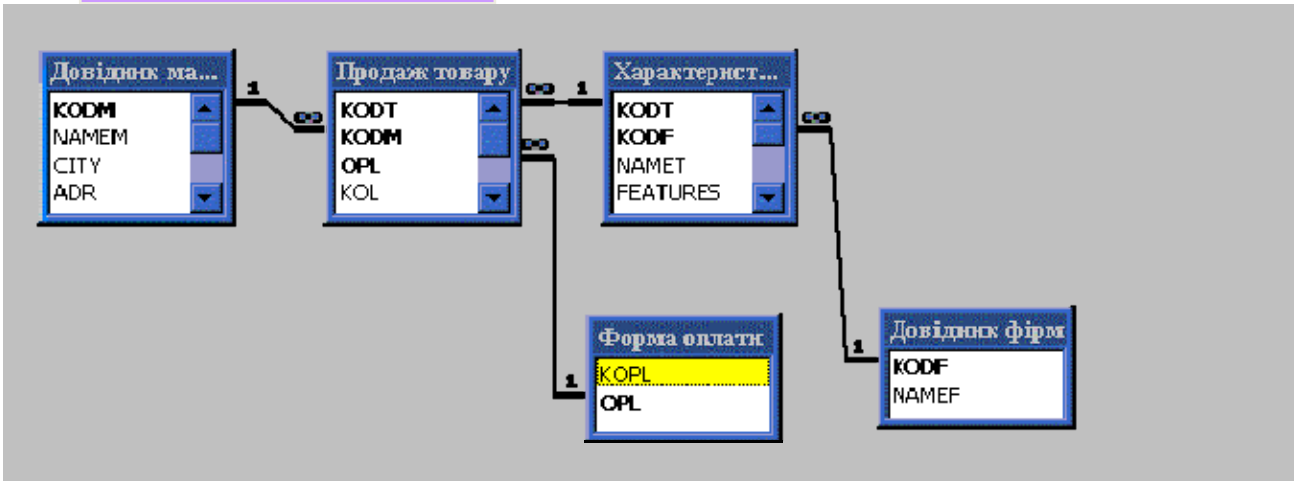
Концептуальне проектування передбачає виконання наступних кроків:

1. аналіз вимог до БД із боку користувачів і побудову так називаної користувацької моделі (вивчення і систематизація інформації про користувачів, їхні інформаційні потреби, джерела інформації, реальні інформаційні потоки і бізнес-правила);
2. виявлення основних об'єктів (сутностей), що підлягають відображенню в моделі, виходячи зі специфіки предметної області, цілей замовника і наявного ділового регламенту;
3. визначення атрибутів сутностей і їхнє приведення у відповідність із вимогами майбутньої об'єктної моделі;
4. визначення зв'язків між сутностями і властивостей сутностей;
5. вибір типу інфологічної (об'єктної) моделі і її побудова. Найчастіше це модель «сутність – зв'язок» (ER-діаграма);
6. перевірку моделі даних;
7. побудову розподіленої моделі даних;
8. вибір моделі організації даних у концептуальній схемі.

Логічне проектування –Для реляційних моделей на цьому етапі фіксується набір і структура таблиць, що представляють сутності, визначаються ключі і проводиться нормалізація таблиць.

Логічна модель - *це версія концептуальної моделі для конкретної моделі даних.*

Зв'язок ключових полів таблиць з іншими полями



Фізичне проектування – це процес створення опису реалізації БД на вторинних запам'ятовуючих пристроях із указівкою структур збереження і методів доступу, використовуваних для організації ефективної обробки даних. *Фізичний етап проектування* забезпечує вибір раціональної структури збереження даних і методів доступу до них, виходячи з арсеналу методів і засобів, що надаються розроблювачу конкретної СУБД.

Характеристика товарів		
Код товару	Код товару	Счетчик
Код фірми	Код фірми	Числовой
	Назва товару	Текстовый
	Характеристики	Текстовый
	Екранне меню	Логический
	Ціна, грн.	Денежный
	Знижка, %	Числовой
Довідник фірм		
Код фірми	Код фірми	Числовой
	Назва фірми	Текстовый
Довідник магазинів		
Код магазину	Код магазину	Числовой
	Назва магазину	Текстовый
	Місто	Текстовый
	Адреса	Текстовый
	Телефон	Текстовый

Фізична модель *- є відображенням логічної на об`єкти конкретної СУБД.*

Основною метою фізичного проектування БД є опис способу фізичної реалізації логічного проекту БД. У випадку реляційної моделі даних під цим мається на увазі наступне:

- створення набору реляційних таблиць і обмежень на них на основі інформації, представленої в глобальній логічній моделі даних;
- визначення конкретних структур збереження даних і методів доступу до них, що забезпечують оптимальну продуктивність системи з БД;
- розробка засобів захисту створюваної системи.

І. Питання до самостійного вивчення

1. Надати характеристику генетичним алгоритмам, програмним агентам та мультиагентним системам.
2. Визначити переваги та проблеми, які виникають під час моделювання колективної поведінки.

Рекомендовані джерела:

Основні: 3, 5.
Додаткові: 6, 10.
Інтернет-ресурси: 12, 13.

ІІ. Перелік індивідуальних завдань

1. Постановка задачі. Спроектувати БД певної предметної галузі.
2. Опис предметної області
3. Виділити основні абстракції (суть, атрибут, зв'язок) в предметній області і визначити їх параметри.
4. Сформувати максимально повний перелік можливих запитів до бази даних на основі аналізу предметної області.
5. Побудувати концептуальну модель у вигляді ER -діаграми.
6. Побудувати логічну модель. Описати домени (допустима безліч значень, які можуть приймати атрибути), вказуючи типи відповідних даних і їх характеристики.

III. Питання для самоконтролю

1. Класифікація інтелектуальних систем.
2. У чому суть концептуального проектування?
3. Що відбувається на етапі логчного проектування?
4. Що відбувається на етапі фізичного проектування?

ТЕМА 3. ХАРАКТЕРИСТИКА СУЧАСНИХ БАЗ ДАНИХ ТА СИСТЕМ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Проектування реляційних БД

РБД – набір нормалізованих реляційних відношень.

Об'єктом є реляційні відношення – двовимірна таблиця в якій зберігається інформація з предметної області і на структуру якої накладається певні вимоги.

Характеристики НФ 1. Відношення називається нормалізованим або зведеним до першої форми, якщо всі його атрибути прості (значення яких атомарні, неподільні)

2НФ-Відношення знаходиться у 2 НФ, якщо воно знаходиться у 1НФ і кожний не ключовий атрибут функціонально повно залежить від первинного ключа.

3НФ-реляц.відношення,якщо воно:1)знаходиться у 2НФ.2)кожний не ключовий атрибут нетранзитивно залежить від первинного ключа.

Базові типи даних для реляційних БД (на прикладі MS Access).

У MS Access можна використовувати дані наступних типів:

• *Текстовий:* алфавітно-цифрові символи (До 255 байт)

• *Поле МЕМО (Довгий текст):* алфавітно-цифрові символи (до 65 535 символів)

• *Числовий:* будь-які числові дані (1,2,4,8 байт)

• *Дата/час:* дата і час (до 8 байтів)

• *Грошовий:* округлені числові дані для грошових значень. (8 байт)

• *Лічильник:* унікальні, послідовно зростаючі (на 1) чи випадкові числа, використовувані Access для автоматичної нумерації кожного запису, що вводиться (4 байта)

• *Логічний:* логічні значення, що можуть містити одне з двох можливих значень, такі як True/False і On/Off (1 байт)

• *Поле об'єкта OLE:* Об'єкти OLE, графічні чи зображення інші дані в вийковому форматі (До 1 Гбайт)

• *Гіперпосилання:* рядок з букв і цифр, що визначає шлях доступу до документа, сторінці Web чи конкретному місцеві в документі

• *Майстер підстановок:* майстер, що створює поле зі списком, що відображає список припустимих значень, що розкривається, з іншої таблиці

По замовчуванню Access автоматично призначає всім новоствореним полям тип даних текстовий.

І. Питання до самостійного вивчення

1. Забезпечення секретності, захист цілісності даних, синхронізація, захист від відмов і відновлення.

2. Порівняльна характеристика локальних та розподілених базам даних, а також файл-серверної та клієнт-серверної технологій управління даними

Рекомендовані джерела:

Основні: 2, 5.
Інтернет-ресурси: 13.

ІІ. Перелік індивідуальних завдань

1. Визначити ключі і зовнішні ключі (якщо вони є).
2. Побудувати фізичну модель. За допомогою середовища MS Access створити структури таблиць для представлення предметної області у рамках реляційної моделі з вказівкою типів даних і їх характеристик.
3. Визначити схему бази даних, зв'язки між таблицями і накласти умови цілісності на таблиці, пов'язані відношенням «один-до-багатьох»
4. Отримання нормальних форм відношення для різноманітних економічних документів для реляційних БД.

ІІІ. Питання для самоконтролю

1. Нормалізація відношень.
2. Поняття ключового поля.
3. Основні типи даних у MS Access.

ТЕМА 4. СУТНІСТЬ РЕЛЯЦІЙНОГО ПІДХОДУ ДО ПРОЕКТУВАННЯ БАЗ ДАНИХ

Під цілісністю даних розуміють повні, несуперечливі дані, що адекватно відображають певну предметну галузь.
Цілісність даних забезпечується на фізичному та логічному рівнях.

На фізичному рівні здійснюється надійне зберігання даних і доступ до них користувачів відповідно до наданих прав.

На *логічному рівні* передбачається відсутність логічних помилок у БД. На етапі проектування цілісність даних забезпечується на основі обмежень цілісності.

Обмеження цілісності реляційної БД.

1. Цілісність сутностей (реляційних відношень): в базовому відношенні ні один атрибут первісного ключа не може містити відсутніх даних, тобто не може приймати значень NULL

Кваліфікатор NULL означає, що значення атрибута в даний момент часу невідоме або неприйнятне для кортежа відношення

Увага: NULL [1] не відповідає нульовому значенню!

2. посилкова цілісність: Якщо у відношенні існує зовнішній ключ, то значення зовнішнього ключа повинно або відповідати значенню потенційного ключа деякого кортежа у базовому відношенні, або задаватися кваліфікатором NULL.

3. Корпоративні обмеження цілісності – додаткові правила підтримки цілісності даних в базі, що визначаються користувачем чи адміністратором БД.

В багатьох СУБД використовують ***Механізм зовнішніх ключів*** для підтримки посилальної **цілісності даних**. Суть цього механізму полягає в тому, що деякому атрибуту (або групі атрибутів) одного відношення призначається посилання на первинний ключ іншого відношення; тим самим закріпляються зв'язки підлеглості між цими відношеннями. При цьому відношення, на первинний ключ якого посилається зовнішній ключ іншого відношення, називається **master-відношенням**, або головним відношенням; а відношення, від якого виходить посилання, називається **detail-відношенням** (частиною головного, підлеглим). Після призначення такого посилання СУБД має змогу автоматично відслідковувати питання "непорушності" зв'язків між відношеннями, а саме:

- якщо ви спробуєте вставити в підлеглу таблицю запис, для зовнішнього ключа якої не існує відповідності в головній таблиці (наприклад, там немає ще запису з таким первинним ключем), СУБД згенерує помилку;
- якщо ви спробуєте знищити з головної таблиці запис, на первинний ключ якого є хоча б одне посилання з підлеглої таблиці, СУБД також згенерує помилку;
- якщо ви спробуєте змінити первинний ключ запису головної таблиці, на який є хоча б одне посилання з підлеглої таблиці, СУБД також згенерує помилку.

Традиційні механізми:

- Засоби контролю вхідної інформації (шаблони, правила вводу);
- Тригери, тобто процедури;
- Збережені процедури, тобто програмний код, який включається в БД і виконується при настанні деяких умов чи подій.

У багатьох сучасних СУБД є засоби автоматичного забезпечення цілісності у вигляді обмежень значень полів таблиць і структурних обмежень на записи таблиць.

Реалізація обмежень цілісності в середовищі реляційних СКБД.

Реалізація обмежень цілісності в середовищі реляційних СКБД.Обмеження на прикладі Access у формі шаблонів, умови на значення полів та цілісності на таблицю.

Структура шаблону; маска_вводу[;01[;символ_вказати]]

Маска вводу – це набір текстових констант і знаків маски, які визначають яким даним і які образи можна вводити. Наприклад маска вводу для телефонного номеру США (999)000-0000;

0;-.*Символ вказівник* – це символ який вказує на екрані позицію вводу (зазвичай "_").

І. Питання до самостійного вивчення

Методи забезпечення цілісності даних.

Рекомендовані джерела:

Основні: 2, 3, 4, 5.
Інтернет-ресурси: 13.

ІІ. Перелік індивідуальних завдань

1. Визначити та надати характеристику таким поняттям: сутність, атрибут, зв'язки, типи сутностей, ієрархія наслідування спадкоємності, ключі, таблиці та представлення, тригери та процедури зберігання.
2. Описати технологію (послідовність кроків та засоби) зв'язування моделі процесів та моделі даних.
3. Накласти умови цілісності на таблиці.

ІІІ. Питання для самоконтролю

1. Які умови мають задовольняти поля, по яким утворюються зв'язки між таблицями, для забезпечення цілісності даних?
2. Які методи забезпечення цілісності даних використовуються у MS Access?
3. У чому полягає сутність механізму зовнішніх ключів?

ТЕМА 5. АДМІНІСТРУВАННЯ ДАНИХ ТА АДМІНІСТРУВАННЯ БАЗ ДАНИХ

Адміністрування базами даних передбачає виконання функцій, спрямованих на забезпечення надійного та ефективного функціонування системи баз даних, адекватності змісту бази даних інформаційним потребам користувачів, відображення в базі даних актуального стану предметної області.

Управління даними в базах даних включає:

- Безпосереднє управління даними у зовнішній пам'яті.
- Управління буферами оперативної пам'яті.
- Управління транзакціями.
- Серіалізацію.
- Журналізацію.

Основні програмно-технічні заходи, застосування яких дозволить вирішити проблеми управління безпекою в СУБД:

- аутентифікація та ідентичність;
- управління доступом;
- підтримка цілісності;
- протоколювання і аудит;
- захист комунікацій між клієнтом і сервером;
- відображення загроз, специфічних для СУБД.

І. Питання до самостійного вивчення

1. Управління даними в базах даних.
2. Управління безпекою в СУБД.

Рекомендовані джерела:

Основні: 1, 2, 4.
Додаткові: 10.

ІІ. Перелік індивідуальних завдань

1. Встановити пароль на вхід в БД.
2. Надати системні та об’єктні повноваження, встановити право доступу для різних користувачів.
3. Створити резервне копіювання БД.

ІІІ. Питання для самоконтролю

1. Способами здійснюється захист даних?
2. Яким чином можна обмежити права на використання можливостей Access?
3. Що входить в обов’язки адміністратора БД?

ТЕМА 6. СИСТЕМА ГЕНЕРАЦІЇ ЗАПИТІВ ТА МОВА ЗАПИТІВ (СТАНДАРТ) SQL

SQL (Structured query language — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних.

Перетворення QBE-запиту в SQL-інструкцію.

QBE-запити Access автоматично перетворює у SQL-запити. До тих пір, доки користувач працює з даними, створеними Access, йому непотрібно напряму звертатися до SQL-інструкції. SQL представляє інтерес тільки в тому випадку, якщо із Access користувач робить запит до SQL-баз даних.

У вікні проектування QBE-запиту спроектуйте QBE-запит, який належить перетворенню в SQL-запит в режимі конструктора. В області *Работа с запросами* виберіть режим SQL. У діалоговому вікні SQL відредагуйте спроектований запит. Внесені зміни після закриття вікна автоматично актуалізуються і в QBE-проекті. Можна і безпосередньо вводити у SQL-вікно команди, що складають SQL-запит.

Процедура створення SQL-запиту починається із створення QBE-запиту. Після відкриття вікна проектування QBE-запиту слід зразу ж активізувати режим SQL. Access відобразить на екрані порожнє вікно SQL. Введіть SQL-інструкції, що складають SQL-запит. При вводі тексту по мірі необхідності надто довгі рядки розриваються і переносяться на наступний рядок. Для підвищення наочності інструкцій кожну командну рядка SQL можна розпочинати з нового рядка, використовуючи комбінацію клавіш [Ctrl+Enter].

Для виконання набраної у SQL-вікні послідовності інструкцій слід натиснути кнопку Выполнить (активізувати піктограму із зображенням знаку оклику на панелі інструментів). Якщо у введеному SQL-запиті немає помилок, Access в SQL-вікні автоматично сформує еквівалентний QBE-запит, заповнюючи поля QBE-області проектування.

Структура запиту на вибірку

Запит до бази даних виконує оператор вибірки, який задається ключовим словом **SELECT**. Операція SELECT щонайкраще демонструє витонченість і міць засобів вибірки даних у реляційних СКБД. Ця команда використовується для вибірки атрибутів одного або декількох таблиць відповідно до зазначеного критерію відбору.

Фрагменти пропозиції SELECT повинні бути записані у строго визначеному порядку. Найпростіша структура має вигляд

SELECT<список імен полів> FROM<ім'я таблиці>

Імена відокремлюються один від одного комою. Порядок розміщення полів у фразі SELECT може бути довільним і не збігатися з порядком їх розміщення в початковій таблиці. Інколи доцільно вказувати не тільки ім'я поля, але й ім'я таблиці.

Тоді використовується форма<ім'я таблиці.ім'я поля>.

Оператор WHERE визначає логічну умову вибору даних. Якщо ключове слово WHERE відсутнє, то здійснюється вибір усіх даних із таблиць, зазначених у переліку значень ключового слова FROM.

В запитах з умовою WHERE найпростіші умови створюються з використанням операторів AND, OR, NOT, IN, LIKE, BETWEEN.

Для відбору даних можна використовувати **шаблони** (тобто текст, який містить підстановочні символи:

- «**%**» або *- будь-який набір символів,
- «_» або ? - будь-який, але один символ.

Наприклад в результаті застосування шаблону '_етро%' будуть видані прізвища Петров, Петровський, Ветров.

I. Питання до самостійного вивчення

SQL- символи підстановки та регулярні вирази (LIKE)

Рекомендовані джерела:

Основні: 1, 4, 5.

Інтернет-ресурси: 14.

II. Перелік індивідуальних завдань

Створити обчислювальні поля та сформувати умови відбору записів з використанням символів підстановки та логічних операторів.

III. Питання для самоконтролю

1. Чим відрізняється послідовність виконання запитів, створених в режимі SQL, від послідовності виконання запитів, створених в режимі конструктора?

2. Яка загальна структура запитів на вибірку в режимі SQL?
3. Чим відрізняється синтаксис опису обчислювальних полів в режимах конструктора та SQL?
4. В яких випадках використовується SELECT запит?
5. Як встановити порядок виведення записів у запиті?
6. Які правила побудови арифметичних виразів?

ТЕМА 7. ЗАПИТИ МОВИ SQL ДЛЯ ВИБІРКИ ДАНИХ

Мова маніпулювання даними (ММД) (або мова запитів до БД) представляється зазвичай системою команд маніпулювання даними.

Приклад типових команд:

- провести вибірку з бази даного по його найменуванню;
- провести вибірку з бази всіх даних певного типу, значення яких задовольняють певними ознаками;
- знайти в базі позицію даного і помістити туди його нове значення.

Параметр ORDER BY

Для сортування результатів запиту по зростанню або спаданню використовують ключове слово **ORDER BY.** Сортування задається використанням наступної синтаксичної конструкції:

ORDER BY {вираз | положення | альтернативне ім’я стовпця} {ASC|DESC }

Ключові слова **ASC** або **DESC** визначають зростаючий або спадаючий відповідно порядок сортування. При цьому значення “NULL” розглядається як «найважче» і розміщується наприкінці списку при сортуванні в порядку зростання ASC і на початку списку - при сортуванні в порядку убування (Я-А, 9-0) – DESC.

Параметр GROUP BY

Для організації групування відібраних даних із метою їхнього спільного опрацювання використовується параметр **GROUP BY**.

Параметр GROUP BY являє собою уточнюючий (необов’язковий) параметр при використанні параметра FROM і WHERE.

Параметр HAVING

Ключове слово **HAVING** використовується для формування додаткових умов включення в множину даних, що відбираються. Це додаткова можливість “фільтрування” вихідного набору. Використовуючи параметр HAVING слід мати на увазі наступне:

- HAVING – необов’язковий параметр, якщо ж він заданий, то повинен йти слідом за параметром GROUP BY.
- параметр HAVING виконує ті ж функції, що і параметр WHERE, але вже в рамках вихідного набору. WHERE визначає, які записи повинні бути вибрані. HAVING встановлює, які записи, згруповані GROUP BY, повинні відображатися на екрані.

Використання ключового слова GROUP BY приводить до того, що оператор SELECT видає один результуючий рядок для кожної групи рядків, сформованих на основі однакових значень для стовпців або виразів.

Спільне опрацювання даних звичайно зводиться до обчислення деякої функції: суми, середнього значення, числа елементів множини відібраних значень і т.п.

Параметр IN

При роботі з базами даних іншого (не Access) формату доводиться використовувати параметр **IN**.

Параметр **IN** слід застосовувати для запиту даних, збережених в базі даних іншого формату, з яким Access може працювати (наприклад dBASE або Paradox), або для запиту даних, що знаходяться в неактивній в даний час базі даних Access.

При використанні параметра IN слід мати на увазі, що:

- Користувач все може звертатися тільки до однієї зовнішньої бази даних (під зовнішньою мається на увазі будь-яка інша, окрім активної, бази даних).
- При заданні типу бази даних, створеної не з допомогою Access, до його позначення слід додати крапку з комою (;), а ім'я взяти в лапки або апострофи, наприклад 'dBASE;' або "dBASE;".

Параметр OLL

Параметр WHERE визначає критерій відбору записів із вхідного набору. Але в таблиці можуть бути присутні дублікати. Наприклад, помилково в таблицю клієнтів двічі був введений запис про одного і того ж клієнта. По замовчуванню у вихідному наборі, що генерується при виконанні SQL-запиту, будуть присутні всі дублікати. Управляти включенням дублікатів у вихідний набір можна з допомогою спеціальних параметрів – предикатів. По замовчуванню команді SELECT відповідає предикат ALL. Його можна навіть явно не вказувати. Він задає включення у вихідний набір всіх дублікатів, відібраних по критерію WHERE. Предикат ALL включається в команду SELECT відразу за ключовим словом SELECT (перед іменами полів, що вибираються).

Для боротьби з дублікатами записів в SELECT команді, замість предикату ALL, можна вказати інші предикати: DISTINCT чи DISTINCT-ROW, який не являється обов'язковим. Якщо не введений ні один з цих предикатів, приймається предикат ALL, і Access вибирає всі записи, які задовольняють WHERE-умовам в SQL-інструкції.

Параметри DISTING та DISTINCTROW

На відміну від параметра **OLL**, який дозволяє виводити всі рядки, що повторюються, для виведення значень полів таблиці, що не повторюються у фразі **SELECT** використовується параметр **DISTING** з такою структурою

SELECT DISTING <список імен полів>

Отже, в SELECT команді, для боротьби з дублікатами записів замість предикату ALL, використовують предикати: DISTINCT чи DISTINCT-ROW, який не являється обов'язковим. Якщо не введений ні один з цих предикатів, приймається предикат ALL, і Access вибирає всі записи, які задовольняють WHERE-умовам в SQL-інструкції.

Предикат DESTINCT слід застосовувати у тих випадках, коли необхідно вилучити записи, які містять дані, що дублюються у вибраних полях. При використанні цього предикату значення для кожного із наведених в інструкції SELECT полів повинні бути унікальними, щоб запис в якому вони містяться міг ввійти у вихідний набір.

Результат пошуку в якому застосовується предикат DISTINCT не може бути актуалізованим. Дія команди з цим предикатом аналогічна її дії при встановленому значенні Yes опції Unique Values (унікальні значення) в діалоговому вікні Query Properties (Властивості запиту).

Інколи необхідно виключити дублікати не тільки з вибраного, але й з усіх решти полів запису, тоді застосовують предикат **DISTINCTROW**

Предикат DISTINCTROW слід застосовувати, коли потрібно відкинути записи, що дублюють одне одного, не тільки у вибраних полях. Якщо предикат DISTINCTROW не буде заданий, наведений вище запит згенерує по декілька стрічок для кожної фірми, що розмістила більше одного замовлення.

Предикат DISTINCTROW ефективний тільки тоді, коли поля вибираються не з однієї, а з усіх таблиць, що використовуються в запиті. Предикат DISTINCTROW ігнорується, якщо запит включає в себе тільки одну таблицю.

Операція INNER JOIN

З допомогою операції INNER JOIN користувач може створити спеціальне об’єднання таблиць. Об’єднання виконується по умові рівності. Це найбільш часто вживаний спосіб об’єднання, при якому умовою об’єднання є рівність вмістимого полів, які наведені після ключового слова ON в записах таблиць, вказаних в операції INNER JOIN. Записи із двох таблиць об’єднуються як тільки у вказаних полях будуть знайдені значення, що співпадають.

Операція INNER JOIN не є обов’язковою частиною інструкції SELECT. Якщо ж вона все-таки застосовується, то її слід оформляти як частину параметра FROM. В загальному вигляді операція INNER JOIN виглядає наступним чином:

Таблиця1 INNER JOIN Таблиця2 ON Таблиця1. Поле А = Таблиця2. Поле Б

В даному випадку об’єднаються (зв’яжуться) таблиці Таблиця1 і Таблиця2. у вихідний набір (при відсутності інших умов) будуть включені записи таблиць Таблиця1 і Таблиця2, для яких виконується умова рівності вмістимого: Поле А = Таблиця2. Поле Б. В умові об’єднання можуть брати участь два числових поля будь-яких (в тому числі і різних типів). Для інших типів полів необхідно дотримуватися наступного правила: поля повинні відноситися до одного і того ж типу даних і містити один і той самий вид даних, але вони можуть мати різні імена.

І. Питання до самостійного вивчення

Можливості оператора Select, в якому значення його параметрів визначаються в діалоговому та автоматичних режимах.

Рекомендовані джерела:

Основні: 5.

Інтернет-ресурси: 14.

II. Перелік індивідуальних завдань

1. Створіть в режимі SQL запити *SQL Кількості Зареєстрованих Клієнтів По Днях Тижня*, та *SQL Кількості Зареєстрованих Клієнтів По Місяцях.*
2. Створіть запит *SQL ФізичніОсоби* для відображення алфавітного списку унікальних прізвищ та ініціалів співробітників, постачальників і клієнтів.

III. Питання для самоконтролю

1. Як в режимі SQL описуються зв'язки та параметри поєднання між таблицями?
2. Де в режимі SQL описуються поля групування і де – поля з груповими операціями?
3. Де в режимі SQL описуються умови відбору записів джерела даних і як – умови відбору груп?
4. Яка загальна структура запитів на поєднання в режимі SQL?
5. Які основні вимоги висуваються до запитів на вибірку у загальному запиті на поєднання?

ТЕМА 8. ЗАПИТИ МОВИ SQL ДЛЯ ВИЗНАЧЕННЯ ТА ОБРОБКИ ДАНИХ

Створення й модифікація схем таблиць та маніпулювання даними в таблицях

З допомогою модифікуючи, так званих запитів дій, користувач може додати, знищити або актуалізувати записи і зберегти вихідний набір (Dynaset) запиту у вигляді нової таблиці. Виконанням макрокоманди RunSQL (запуск запиту SQL) ці процеси можна виконати прямо з макросу, не використовуючи спеціально спроектовані і окремо збережені запити.

Для створення і зміни схем таблиць у мові SQL використовують команди:
CREATE – для створення схеми таблиці,
ALTER – для зміни структури таблиці,
DROP – для вилучення створених елементів таблиці,
INSERT для вставки рядків атрибутів у таблицю
DELETE - застосовується для видалення рядків із таблиці
UPDATE - здійснює операцію модифікації рядків з таблиці.
MODYFY –змінює тип даних певного поля
ADD- додає певне поле в таблицю

Створення схем таблиць здійснюється з використанням оператора **CREATE TABLE** загальна структура якого має вигляд

CREATE TABLE <ім'я таблиці>
(<ім'я поля_1><тип даних>[NOT NULL],
<ім'я поля_2><тип даних>[NOT NULL]…);

Створену структуру таблиці можна змінювати, модифікувати, тобто додавати нові поля, видаляти існуючі, змінювати імена і типи даних поля тощо. Для цього слугує оператор **ALTER TABLE** загальна структура якого

ALTER TABLE<ім’я таблиці>[MODYFY(<ім’я поля><тип даних>)]
[ADD(<ім’я поля><тип даних>)] [DROP<ім’я поля>];

Операція INSERT використовується для вставки рядків атрибутів у таблицю і має наступний загальний синтаксис:

INSERT INTO<ім’я таблиці>[*(<поле_1>[, <поле_2>], ...)]*

VALUES*(<значення_1>, значення_2, ...).*

Якщо після ключового слова INTO не вказані імена полів, а вказане тільки ім’я таблиці, то заповнюються всі стовпці рядка.

Приклад застосування оператора I**NSERT**.

Нехай таблиці Tab1 і Tab2, сформовані пропозиціями:

CREATE TABLE Tabl (Atl CHAR(3), At2 NUMBER);

CREATE TABLE Tab2 (Atl NUMBER);

Найпростіший варіант введення даних є вставка рядка явним переліком списку значень.

INSERT INTO Tabl VALUES('A',1);

Дані, що вводяться в таблицю, можуть бути результатом запиту до іншої таблиці.

```
INSERT INTO Tab2
SELECT At2 FROM
Tabl WHERE Atl = ‘A';
```

Пропозиція **DELETE** застосовується для видалення рядків із таблиці або базової таблиці представлення і має такий синтаксис

DELETE FROM <ім'я_таблиці**> [WHERE <**умова**>]**

Якщо ключове слово WHERE відсутнє, то з таблиці видаляться всі рядки. Якщо ж ключове слово WHERE використовується, видаляться рядки, для яких умова виконується.

Всі рядки і відповідні індекси, що видаляються, звільняють пам'ять комп’ютера, що займається ними.

І. Питання до самостійного вивчення

Вимоги до запитів на вибірку у загальному запиті на поєднання.

Рекомендовані джерела:

Основні: 5.

Інтернет-ресурси: 14.

ІІ. Перелік індивідуальних завдань

Створити запити на оновлення та на створенн таблиці.

ІІІ. Питання для самоконтролю

1. Які ключові слова використовують для операцій маніпулювання даними?
2. Опишіть синтаксис та структуру запитів-дій.
3. Класифікація команд створення, видалення та модифікації об’єктів у БД.
4. Яка специфіка створення таблиць за допомогою команди CREATE TABLE?

5. Чим відрізняються команди DELETE та UPDATE?
6. З використанням якого оператора здійснюється створення схем таблиць?
7. У чому полягає різниця між знищенням таблиці, як цілого об’єкту та запитом на видалення усіх записів таблиці?

ТЕМА 9. ПІДТРИМКА ЦІЛІСНОСТІ ДАНИХ У СУБД

Підтримка цілісності даних у СКБД

Цілісність даних у базі даних пов'язана з визначенням правил перевірки та підвищенням достовірності введених даних (таких, як "процентне значення відсотку повинно знаходитися між 0 і 100"), які гарантують, що недійсні дані не потраплять у ваші таблиці.

Автоматичний контроль добре структурованих даних підвищує достовірність інформації в базі, оскільки найбільш поширеними помилками введення є пропуск символу, заміна або подвійний набір цифри.

Цей контроль здійснюється у формі:

- декларативних обмежень цілісності: NOT NULL, UNIQUE, CHECK, PRIMARY KEY, FOREIGN KEY;
- коду прикладної програми;
- тригерів бази даних (СКБД Oracle).

Розглянемо декларативні обмеження цілісності.

Декларативні обмеження цілісності встановлюють бізнес-правила на рівні бази даних, визначаючи набір перевірок для таблиць системи. Ці перевірки автоматично виконуються кожного разу, коли викликається оператор вставки, модифікації або видалення даних таблиці.

NOT NULL (не порожньо) встановлюється для стовпця, що повинен мати значення в кожному рядку; він ніколи не може бути NULL (порожнім). За замовчанням всі стовпці в таблиці можуть бути порожніми.

PRIMARY KEY (первинний ключ) визначає стовпець або групу стовпців, які можна використовувати для унікальної ідентифікації рядка. Ніякі два рядки в таблиці не можуть містити однакові значення стовпців первинного ключа. Крім того, стовпці первинного ключа завжди - NOT NULL. При накладенні обмеження PRIMARY KEY на таблицю після її створення за допомогою ALTER TABLE значення відповідних стовпців змінюються на NOT NULL. Для таблиці може бути заданий тільки один первинний ключ PRIMARY KEY.

UNIQUE (унікальний) визначає вторинний ключ для таблиці. Це стовпець або група стовпців, котрі можна використовувати, як інший спосіб унікальної ідентифікації рядка. Ніякі два рядки не можуть містити однакові значення для стовпця або стовпців ключа UNIQUE. Кожна таблиця має тільки один первинний ключ, але на неї можна накласти декілька обмежень UNIQUE. Якщо в таблиці визначено складений унікальний індекс, то комбінація значень включених у нього стовпців повинна бути унікальною для кожного запису таблиці, хоча окремі стовпці можуть мати однаковими значення.

Стовпці для обмеження UNIQUE не обов'язково повинні бути NOT NULL (хоча, як правило, так і є). Якщо значення для деяких стовпців, що формують унікальне обмеження, порожні, обмеження не перевіряється.

FOREIGN KEY (зовнішній ключ) або обмеження цілісності за посиланням, встановлює відношення цілісності між таблицями. Воно вимагає, щоб стовпець або набір стовпців в одній таблиці збігався з первинним або вторинним ключем іншої таблиці. Наприклад, можна установити обмеження FOREIGN KEY для таблиці замовлень, щоб зазначити, що кожного разу, коли вставляється або оновлюється запис замовлення, у таблиці замовників повинен існувати номер замовника. Це гарантує, що ви не введете замовлення для неіснуючих замовників.

Обмеження FOREIGN KEY використовуються для встановлення батьківсько-дочірніх відношень між таблицями. Якщо деякі стовпці зовнішнього ключа порожні, обмеження взагалі не перевіряється. Стовпці зовнішнього ключа звичайно оголошуються як NOT NULL.

Можна зазначити серверу БД, що, коли батьківський рядок видалиться, видалення повинно автоматично поширюватися на дочірні рядки – каскадне видалення. Це - небезпечна ситуація. Користувач інформується тільки про видалення рядків батьківської таблиці, і він може не знати про додаткові рядки, що були віддалені автоматично у фоновому режимі, бо ніщо не вказує, що каскадне видалення відбулося.

Подібне автоматичне видалення дочірніх рядків підтримується, тільки якщо задана фраза DELETE CASCADE наприкінці оператора створення зовнішнього ключа. Однак якщо ви змінюєте значення ключа головної таблиці, рядки нащадка не обновляються автоматично, щоб відповідати новому ключу.

Відзначимо, що обмеження FOREIGN KEY не перевіряється взагалі, якщо якийсь із стовпців у зовнішньому ключі порожній.

Обмеження **CHECK** визначає логіку додаткової перевірки, яка повинна дати результат TRUE (істина) для оператора вставки, модифікації або видалення з таблиці. Додаткова логіка повертає результат Boolean. Обмеження CHECK гарантує, що значення в змінюваному рядку що змінюється задовольняють заданому наборові перевірок цілісності. Синтаксис обмеження CHECK подібний синтаксису фрази WHERE оператора SELECT, однак тут не можна використовувати підзапити або стовпці, що змінюються з часом (такі, як SYSDATE).

Переваги декларативних правил цілісності:

1) підвищення продуктивності;
2) зручність задання;
3) централізована підтримка;
4) легкість модифікації;
5) безпосередній зворотній зв'язок із користувачем;
6) гнучкість (включення/відключення).

Недоліки декларативних правил цілісності:

- важко відслідковувати по користувачах і базах даних.

Визначення правил цілісності

Правила цілісності декларуються під час створення або модифікації таблиці. Для визначення правил цілісності при описі нової таблиці, її стовпців використовується інструкція CREATE TABLE, або ALTER TABLE.

Синтаксис команди CREATE TABLE:

CREATE TABLE *ім'я_таблиці* (*стовпець_1* тип [(розмір)] [NOT NULL] [*індекс_1*] [,*стовпець_2* тип [(розмір)] [NOT NULL] [*індекс_2*] [, ...]] [, CONSTRAINT *складений_індекс* [, ...]]),

Де і*м'я_таблиці* – ім'я таблиці, яка створюється;

стовець_1, *стовпець_2* – імена одного або декількох стовпців, що створюються у новій таблиці. Таблиця повинна містити хоча б один стовпець.

Тип – тип даних поля в новій таблиці.

розмір – розмір поля в символах (в Access тільки для текстових і бінарних полів).

індекс_1, індекс_2 – Пропозиція CONSTRAINT, призначена для створення простого індексу.

складений_індекс – Пропозиція CONSTRAINT, призначена для створення складеного індексу.

Пропозиція **CONSTRAINT** установлює різні обмеження на поле і використовується для визначення ключа. Крім того, для створення ключа або додаткового індексу для існуючої таблиці можна використовувати інструкцію CREATE INDEX. Створення обмеження за допомогою пропозиції CONSTRAINT подібно застосуванню індексу, хоча воно також застосовується для установлення відношень між таблицями.

Синтаксис пропозиції CONSTRAINT

CONSTRAINT ім'я {PRIMARY KEY (*ключове_1*[, *ключове_2* [, ...]]) | UNIQUE (*унікальне_1*[, *унікальне_2* [, ...]]) |
NOT NULL (*непорожнє_1*[, *непорожнє_2* [, ...]]) |
FOREIGN KEY (*посилання_1*[, *посилання_2* [, ...]])
REFERENCES *зовнішняТаблиця* [(*зовнішнєПоле_1* [, *зовнішнєПоле_2* [, ...]])]},

де *ключове_1, ключове_2* – імена ключових полів (стовпців);

унікальне_1, унікальне_2 – імена полів (стовпців), що є унікальними;

непорожнє_1, *непорожнє_2* – імена полів (стовпців), у яких забороняються значення NULL;

посилання_1, посилання_2 – імена полів, включених у зовнішній ключ, що містять посилання на поля в іншій таблиці;

зовнішняТаблиця – ім'я зовнішньої таблиці, що містить поля, зазначені за допомогою аргументу *зовнішнєПоле_1*;

зовнішнєПоле_1, *зовнішнєПоле_2* – імена полів(стовпців) у *зовнішняТаблиця*, на які посилаються поля, зазначені за допомогою аргументу *посилання_1*, *посилання_2*.

Отже, правила цілісності можуть бути оголошені на рівні стовпця або на рівні таблиці. На рівні стовпця правила цілісності діють тільки на стовпець, для якого вони оголошені. Правила цілісності на рівні стовпця задаються як

частина визначення стовпця. Пропозиція CONSTRAINT у даному випадку розміщується відразу після опису типу стовпця в команді інструкції CREATE TABLE або ALTER TABLE.

На рівні таблиці пропозиція CONSTRAINT створює складений індекс і розміщується поза пропозицією, що описує поля в інструкції ALTER TABLE або CREATE TABLE.

І. Питання до самостійного вивчення

Використання поєднань.

Рекомендовані джерела:

Основні: 5.

Інтернет-ресурси: 13, 14.

ІІ. Перелік індивідуальних завдань

1. Оголосити правила цілісності на рівні стовпця та на рівні таблиці.
2. Встановити на основі пропозиції **CONSTRAINT** різні обмеження на поля.

ІІІ. Питання для самоконтролю

1. Яким чином відбувається контроль цілісності даних засобами SQL?
2. З допомогою яких ключових слів задають декларативні обмеження цілісності?
3. Як визначити первинний ключ таблиці?
4. Як задається вторинний ключ таблиці?
5. Яке призначення зовнішнього ключа FOREIGN KEY?
6. Переваги декларативних правил цілісності.
7. Яка інструкція використовується при визначенні правил цілісності?
8. Типи обмежень.
9. Яке призначення пропозиції CONSTRAINT?

ТЕМА 10. ПРИНЦИПИ СТВОРЕННЯ КЛІЄНТ-СЕРВЕРНИХ БАЗ ДАНИХ З ВЕБ-ІНТЕРФЕЙСОМ

Будь-яке SQL-застосування реляційної БД складається з трьох частин: інтерфейса користувача, набору таблиць в БД і SQL-машини.

Microsoft SQL сервер підтримує доступ до БД в багатокористувацькому режимі за допомогою локальної мережі, та мережі Інтернет. Доступ до бази реалізованої на Microsoft SQL сервері, може здійснюватись за допомогою Web додатків, мов програмування Java, та C#. А також підтримується робота з мовою розмітки XML.

Microsoft SQL Server – це реляційна СУБД. У реляційних БД дані зберігаються в таблицях. Користувачі отримують доступ до даних на сервері через програми, а адміністратори, виконуючи завдання конфігурування, адміністрування та підтримки БД, виробляють безпосередній доступ до сервера. SQL Server є масштабованою БД, це означає, що вона може зберігати значні обсяги даних і підтримувати роботу багатьох користувачів, які здійснюють одночасний доступ до БД.

Під час створення БД необхідно визначити її ім'я, розмір, а також файли і групи файлів, у яких вона буде зберігатися.

SQL Server створює нову БД у два етапи:

1. Використовуючи копію бази Model, SQL Server ініціалізує нову та її метадані.

2. Після цього SQL Server заповнює частину, що залишилася, БД (крім сторінок із внутрішніми даними, що відбивають використання дискового простору, зайнятого БД) порожніми сторінками.

І. Питання до самостійного вивчення

Фонові (DBWO, LGWR, CKPT, SMON, PMON) та серверні (виділені та спільні) процеси СУБД Oracle.

Рекомендовані джерела:

Основні: 1, 4.
Додаткові: 6, 7, 8, 9, 10.
Інтернет-ресурси: 14, 15.

ІІ. Перелік індивідуальних завдань

1. В утиліті SQL Server Management Studio створити нову БД за допомогою оператора Create Database, назву БД визначити, виходячи з предметної області.
2. Закоментувати оператор (-- – однорядковий коментар,и/* */ – багаторядковий коментар).
3. Програмно зробити активною створену БД за допомогою оператора Use.
4. Створити перераховані таблиці за допомогою операторів Create table, причому самостійно визначити типи таблиць (батьківська чи підпорядкована), типи полів і їх розміри, знайти поля типу Primary key і Foreign key.
5. Зберегти файл програми.
6. У SQL Server Management Studio в розділі діаграм створеної БД згенерувати нову діаграму, перевірити зв'язки між таблицями.

ІІІ. Питання для самоконтролю

1. Яке призначення і склад оператора SELECT.
2. Назвіть вимоги до порядку розміщення стовпців в операторі SELECT.
3. Яка особливість використання символу (*) в операторі SELECT.
4. Охарактеризуйте призначення пропозиції оператора SELECT – FROM.
5. Яке призначення пропозиції оператора SELECT – WHERE.
6. Яка суть пошуку за шаблоном.
7. Які особливості використання ключових слів AND і OR.
8. Яке призначення пропозиції оператора SELECT – ORDER BY.
9. Яке призначення і склад оператора INSERT.
10. Яке призначення і структура оператора UPDATE.

ТЕМА 11. КОНЦЕПЦІЯ ПОБУДОВИ СХОВИЩ ДАНИХ

Концепція Сховища даних вперше запропонована спеціалістами IBM у формі «інформаційного складу» (складу інфо). В основі концепції сховища даних (СД) лежить розподіл інформації, що використовують в системах оперативної обробки даних (OLTP) і в системах підтримки прийняття рішень (СППР). Такий розподіл дозволяє оптимізувати як структури даних оперативного зберігання для виконання операцій введення, модифікації, знищення та пошуку, так і структури даних, що використовуються для аналізу.

В СППР ці два типи даних називаються відповідно оперативними джерелами даних (ОДД) та сховищем даних.

Сховища даних Data Waren House являють собою інтегровані колекції даних, зібрані з різних систем оперативного доступу до даних.

Основна мета створення Data Waren House в тому, щоб зробити усі значимі для управління бізнесом дані доступними в стандартизованій формі, придатними для аналізу та отримання необхідних звітів.

Для досягнення цього потрібно отримати дані із існуючих внутрішніх та зовнішніх, доступних для комп'ютера, джерел.

І. Питання до самостійного вивчення

1. Поняття сховищ даних та передумови їх створення.
2. Інтеграція даних.
3. Проблеми, що приводять до інтеграції даних.
4. Агрегація даних.
5. Простори даних

Рекомендовані джерела:

Основні: 4.
Додаткові: 6, 10.
Інтернет-ресурси: 14, 15.

ІІ. Перелік індивідуальних завдань

CASE-технології BPWin та ERWin.

ІІІ. Питання для самоконтролю

1. Яка мета створення і використання метаданих у сховищах даних?
2. Які відмінності проектування сховищ даних від проектування БД?

ТЕМА 12. ЗАСОБИ СТВОРЕННЯ СХОВИЩ ДАНИХ

Okacie — призначено для розробки систем багатовимірного аналізу даних. **Okacie express** побудований за за архітектурою “клієнт/сервер” і включає серверні і клієнтські програми.

Hewlett Packard – **Openwarc hause** забезпечує можливість побудови сховищ даних на основі могутніх комп’ютерів HP , апаратури інших виробників і програмних компонентів

NCR – спрямоване на розв'язання проблем корупцій, **Informix Software – Diremik Paranel Server.**

Архітектура сховища даних базується на чотирьох технологіях : реляційні бази даних . програмне забезпечення для управління сховищем даних , засоби доступу даних і платформи відкритих систем.

SAS institute Підхід заснований на такому:

- забезпечення доступу до даних

— перетворення даних і маніпулювання ними

-наявність сервера багатомірних баз даних

— великий набір методів і засобів для аналітичної обробки і статистичного аналізу.

Erwin підтримує методику вимірного проектування сховищ даних. Базовою моделлю яка є результатом вимірного проектування сховищ даних , є модель типу "зірка" та її різновид – "сніжинка" Модель типу "зірка" складається з центральної області фактів та раціонально зв'язаних з нею таблиць вимірювань.

Сховища даних (**Data Warehouse**) представляють собою спеціалізовані бази даних, призначені для зберігання даних, які рідко змінюються, але на основі яких часто вимагається виконання складних запитів. Зазвичай вони орієнтовані на виконання аналітичних запитів, які забезпечують підтримку прийняття рішень для керівників і менеджерів. Сховища даних дозволяють розвантажити промислові бази даних, і тим самим дозволяють користувачам більш ефективно і швидко знаходити необхідну інформацію. Як правило, сховища даних оперують з величезними обсягами інформації, що пред'являє до їх проектування і реалізації підвищені вимоги. Вибір в якості платформи сховища даних такої високопродуктивної РСУБД дозволяє істотно підвищити загальну ефективність створюваної інформаційної системи. У цьому випадку **ERwin** (CASE-засіб фірми PLATINUM Technology inc.) стає незамінним інструментом, оскільки з одного боку ефективно підтримує на фізичному рівні проектування об'єктів РСУБД, з іншого боку має спеціалізовані засоби моделювання сховищ даних.

До проектування сховищ даних звичайно пред'являються наступні вимоги:

- Структура даних сховища повинна бути зрозуміла користувачам.
- Повинні бути виділені статичні дані, які регулярно модифікуються: щодня, щотижня, щоквартально.
- Повинні бути спрощені вимоги до запитів, з метою виключення запитів, які могли б вимагати множинних тверджень SQL у традиційних реляційних СУБД.
- Повинна бути забезпечена підтримка складних запитів SQL, які потребують послідовної обробки тисяч або мільйонів записів.

Для ефективного проектування сховищ даних ERwin використовує розмірну (Dimensional) модель. **Dimensional** – методологія проектування, спеціально призначена для розробки сховищ даних. ERwin підтримує методологію розмірного моделювання завдяки використанню спеціальної нотації для фізичної моделі – Dimensional.

І. Питання до самостійного вивчення

Віртуалізація сховища.

Рекомендовані джерела:

Основні: 4.
Додаткові: 9, 10.
Інтернет-ресурси: 14, 15.

ІІ. Перелік індивідуальних завдань

Установка WEB-сервера APACHE, PHP та СУБД MYSQL

ІІІ. Питання для самоконтролю

У чому полягають відмінності сховищ даних і облікових систем.

ТЕМА 13. СУЧАСНІ СУБД ТА СХОВИЩА ДАНИХ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

І. Питання до самостійного вивчення

1. Інтелектуальний аналіз даних.
2. Моделі видобування даних.
3. Методи видобування даних.
4. Огляд ринку програмних продуктів видобування даних.

Рекомендовані джерела:

Основні: 1, 5.
Додаткові: 9, 10.
Інтернет-ресурси: 11, 12.

ІІ. Перелік індивідуальних завдань

Інтелектуальний аналіз даних Data Mining

ІІІ. Питання для самоконтролю

Сутність технології Data Mining.

ТЕМА 14. ІНФОРМАЦІЙНІ СИСТЕМИ ДІЛОВОГО АДМІНІСТРУВАННЯ

І. Питання до самостійного вивчення

Технологія блокчейн: проектування й застосування.

Рекомендовані джерела:

Основні: 3.
Додаткові: 8, 9.

ІІ. Перелік індивідуальних завдань

Реалізувати функції комплексної автоматизації завдань розробки, узгодження, розповсюдження, пошуку та архівного зберігання документів.

ІІІ. Питання для самоконтролю

1. Моделі та інформаційні потоки електронного офісу.
2. Інструменти автоматизації документообігу організації .

4. КРИТЕРІЇ ОЦІНЮВАННЯ САМОСТІЙНОЇ РОБОТИ. ЗАСОБИ ПРОВЕДЕННЯ ПІДСУМКОВОГО КОНТРОЛЮ

Самостійна робота здобувачів вищої освіти з дисципліни «Організація баз даних та знань» оцінюється у 30 балів. Зокрема, сюди входять наступні види робіт:

1) підготовка наукових статей, наукових публікацій та участь з доповідями у студентських конференціях і семінарах – 10 балів;

2) підготовка реферату та презентації – 20 балів.

Результати самостійної роботи здаються здобувачами вищої освіти поетапно протягом семестру відповідно до вивченого матеріалу згідно тематичного плану.

Підсумковий контроль – *екзамен*.

Якщо здобувач вищої освіти повністю виконав програму дисципліни та набрав протягом семестру 75 і більше балів, то підсумкова оцінка може бути виставлена без опитування чи виконання екзаменаційного завдання на момент проведення екзамену.

У разі, якщо здобувач вищої освіти бажає поліпшити свою оцінку, або не набрав 75 балів, він складає екзамен з усієї програми навчальної дисципліни у вигляді письмового опитування знань згідно завдань встановленого зразка. Результат виконання екзаменаційних завдань оцінюється з урахуванням результатів у співвідношенні 80:20, де 80 – максимальна оцінка за виконання екзаменаційного завдання, 20 – результат поточної успішності відповідно до шкали переводу поточної роботи для врахування її при підсумковій оцінці.

5. СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Основні

1. Гандерлой М. Автоматизация Microsoft Access с помощью VBA = Mike Gunderloy, Susan Sales Harkins; Automating Microsoft Access with VBA / Пер. с англ. С.А. Храмова. М, С.Пб., К : Вильямс, 2006. 416с.

2. Энсор Дейв. Oracle. Проектирование баз даних : пер. с англ. / Дейв Энсор, Йен Стивенсон. - К. : Издат. група BHV, 2000. - 560 с.

3. Макарова М. В., Карнаухова Г. В., Запара С. В. Інформатика та комп'ютерна техніка : навч. посіб. / за ред. М. В. Макарової. 3-тє вид., переробл. і допов. Суми : Університетська книга, 2008. 665 с.

4. Пасічник В. В., Шаховська Н. Б. Сховища даних : навч. посіб. / за ред. В. В. Пасічника. Львів : Магнолія 2006, 2008. 492 с.

5. Руденко В. Д. Бази даних в інформаційних системах : навч. посібник / за заг. ред. В. Ю. Бикова. К. : Фенікс, 2010. 240 с.

2. Додаткові

6. Gertz, M. Jajodia S. Handbook of Database Security: Applications and Trends. – Springer New York, NY, 2008. – 577 p.

7. Barry D. K. The Object Database Handbook: How to Select, Implement, and Use Object-Oriented Databases. – Wiley, 1996. – 352 p.

8. Garcia-Molina H., Ullman J., Widom J. Database Systems: The Complete Book 2nd Edition. – Pearson, 2008. – 1248 p.

9. Гужва В.М. Інформаційні системи і технології на підприємствах : навч. посіб. / В.М. Гужва. - К. : КНЕУ, 2001. - 400 с.

10. Romanuke V. V., Husak L. P. Expert databases and expert estimations in building mathematical interpretations and models for development of entrepreneurship subjects // Proceedings of the International Scientific Conference “Digitalization of the Economy as a Factor in the Sustainable Development of the State”, September 23-24, 2022, Kielce, Poland, The Jan Kochanowski University, pp. 219 – 222.

3. Internet-ресурси

11. Теорія розробки інтелектуальних агентів. - Режим доступу : http://www.williamspublishing.com

12. Додаткові матеріали та програмні засоби з розробки інтелектуальних систем. - Режим доступу : http://www.pearsoneduc/ com/computing

13. Методи проектування програмних систем. - Режим доступу : http://www.sdm.viptop.ru/articles/booch/'

14. СУБД MySQL. - Режим доступу : www.mysql.com

15. СУБД PostgreSQL - Режим доступу : www.postgresql.org