Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

М.А. Демиденко

ВВЕДЕННЯ В СУЧАСНІ БАЗИ ДАНИХ

Навчальний посібник

Дніпро

НТУ «Дніпровська політехніка»

2020

УДК 338.22.021.4

БКК 32.973-018

Д30 *Рекомендовано вченою радою як навчальний посібник по дисципліні "Технології проектування та адміністрування баз даних та сховищ даних " для студентів напряму підготовки 051"Економіка", (Протокол № 22 від 27.12.2019).*

Рецензенти:

Шаповал В.М. - д-р. екон. наук, проф., завідувач кафедри Туризму та економіки підприємства (НТУ «Дніпровська політехніка»).

Паршина О.А. – д-р. екон. наук, проф., завідувач кафедри Аналітичної економіки та менеджменту (Дніпровський університет внутрішніх справ).

Демиденко М.А.

Д30 "Введення в сучасні бази даних": навч. посіб. / М.А. Демиденко; НТУ «Дніпровська політехніка». – Д. : 2020. – 38 с.

ISBN 978-912-350-293-9

Викладено основи і розглянуто поняття технології проектування та адміністрування баз даних та сховищ даних. Матеріал подано на рівні, доступному студентам, які знайомі з курсом інформатики для економістів. Методи, що розглядаю-ться в посі-бнику, ілюструються великою кількістю прикладів. Посібник має на меті на-вчити сту-дентів застосовувати технології проектування та адміністрування баз даних та сховищ даних, сучасне програмне забезпечення та комп'ютери для об-ґрунтування оптимальних економічних рішень.

Посібник базується на досвіді викладання дисципліни «Технології проектування та адміністрування баз даних та сховищ даних» в НТУ «Дніпровська політехніка», призначений для студентів вищих учбових закладів і може бути корисним для економістів, плановиків, менеджерів та маркетологів.

УДК 338.22.021.4

БКК 32.973-018

ISBN 978-912-350-293-9

Зміст

Тема 1. Основні відомості про бази даних та сховища даних

В темі викладено основні поняття, формулювання, основи функціонування , проектування та адміністрування баз даних та сховищ даних

1.1 Що таке база даних?

База даних (БД) — це засіб збирання й впорядкування даних. Бази даних можуть містити відомості про людей, товари, замовлення тощо. Багато баз даних починаються як список у текстовому редакторі або електронній таблиці. У разі збільшення розміру списку в даних починають з'являтися зайві або невідповідні елементи. Стає важко розуміти дані у формі списку, а способи пошуку та витягування наборів даних для перегляду є обмеженими. Коли ці проблеми починають з'являтися, слід перенести дані до бази даних, створеної за допомогою системи керування базами даних (DBMS), наприклад Office Access .

Комп'ютерна база даних — це контейнер об'єктів. Цими об'єктами є таблиці, запити, звіти, форми, макроси, модулі.

За допомогою бази даних (наприклад Access) можна:

додавати нові дані до бази даних, наприклад новий предмет до інвентарного списку;

редагувати наявні дані бази даних, наприклад змінювати поточне розташування предмета;

видаляти відомості, якщо, наприклад, предмет було продано або списано;

впорядковувати та переглядати дані різними способами;

спільно користуватися даними з іншими користувачами за допомогою звітів, повідомлень електронної пошти, інтрамережі або Інтернету.

Адміністратор бази даних (АБД) є особа або група осіб, які вирішують такі завдання:

розробка вимог до бази даних,

проектування бази даних,

створення бази даних,

Забезпечення ефективного використання і супроводу бази даних.

В процесі експлуатації АБД контролює функціонування інформаційної системи, забезпечує захист інформації від несанкціонованого доступу, контролює надмірність, неперечливість і достовірність інформації, що зберігається в бази даних.

Сучасні бази даних функціонують в мережах, тому АБД, як правило, взаємодіє з адміністратором мережі. В обов'язки останнього входять контроль за функціонуванням апаратно-програмних засобів мережі, реконфігурація мережі, відновлення програмного забезпечення після збоїв і відмов обладнання, профілактичні заходи та забезпечення розмежування доступу.

1.2 Архітектура баз даних

Ефективність інформаційної системи залежить від її структури . В сучасних системах широко розповсюджена архітектура клієнт-сервер. Сервером ресурсу в комп'ютерній мережі називають комп'ютер який управляє цим ресурсом - клієнтом В якості клієнта виступає комп'ютер або програма, наприклад бази даних або файлові системи.

На рис. 1 показана структура з файл-сервером. В таких інформаційних системах по запитах клієнтів файли передаються на персональні комп'ютери-клієнти де виконується їх обробка. Недоліком такої системи є надлишкова кількість даних які передаються між сервером та клієнтом.

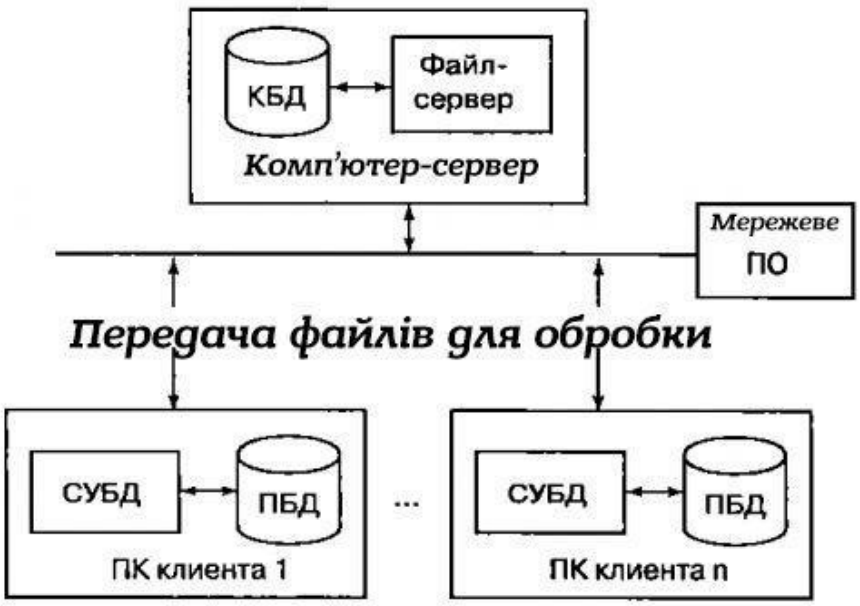


Рисунок 1. Архітектура бази даних з файл-сервером

Структура розподіленої ІС, побудованої по архітектурі клієнт-сервер з використанням сервера баз даних, показана на рис.2. При такій архітектурі сервер бази даних забезпечує виконання основного обсягу обробки даних. Користувачем формуються запити, які надходять до сервера бази даних у вигляді інструкцій мови SQL. Сервер бази даних виконує обробку даних, які потім передаються на комп'ютер користувача. Перевагою такого підходу в порівнянні попереднім є помітно менший обсяг даних, що передаються .

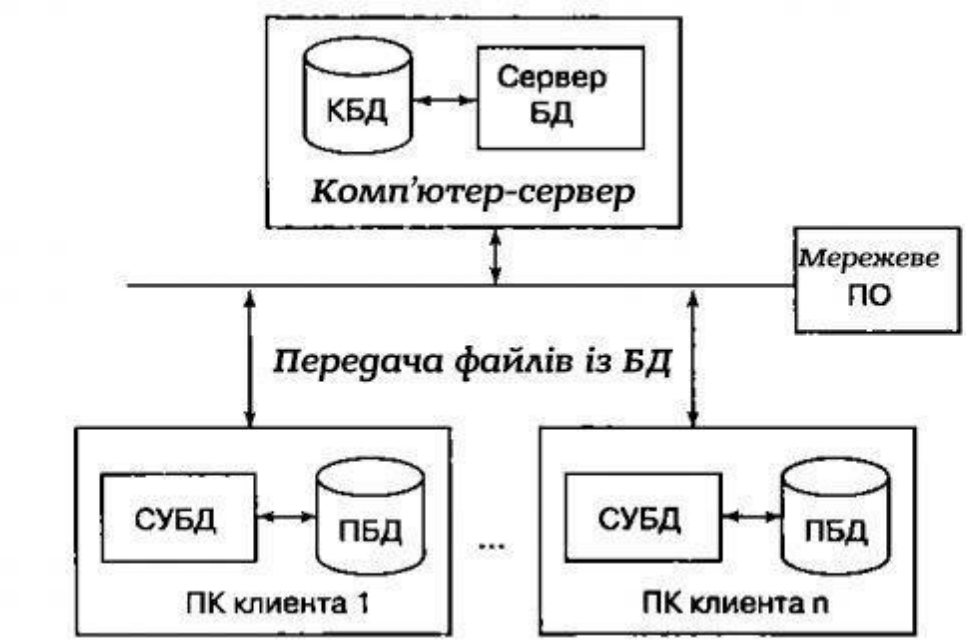


Рисунок 2. Архітектура розподіленої системи бази даних

Найважливішою перевагою бази даних в інформаційних системах є забезпечення незалежності даних від прикладних програм. Це дає можливість користувачам не займатися проблемами представлення даних на фізичному рівні: розміщення даних в пам'яті, методів доступу до них і т. д. Така незалежність досягається із застосуванням багаторівневих СУБД які забезпечують багаторівневе подання даних на логічному (який функціонує на мові яку розуміє користувач) і фізичному (який застосовує машинні команди зрозумілі лише для комп'ютера) рівнях. Завдяки СУБД і наявності логічного рівня представлення даних забезпечується відділення концептуальної (понятійної) моделі бази даних від її фізичного представлення в пам'яті комп'ютера.

1.3 Системи управління базами даних (СУБД)

У загальному випадку під СУБД розуміють будь-який програмний продукт, що підтримує процеси створення, підтримки функціонування та використання БД.

До СУБД відносяться такі основні види програм:

- повнофункціональні СУБД;
- сервери бази даних;
- клієнти бази даних;
- засоби розробки програм роботи з БД.

Повнофункціональні СУБД (ПФСУБД) представляють собою традиційні СУБД, які спочатку з'явилися для великих машин, потім для ПЕОМ. З числа всіх СУБД сучасні ПФСУБД є найбільш численними і потужними за своїми можливостями.

До ПФСУБД відносяться, наприклад, такі пакети, як DataFlex, dBase IV, Microsoft Access, Microsoft FoxPro і Paradox R: BASE.

ПФСУБД мають розвинений інтерфейс, що дозволяє за допомогою команд меню виконувати основні дії з БД: створювати і модифікувати структури таблиць, вводити дані, формувати запити, розробляти звіти, виводити їх на друк і т. п. Для створення запитів і звітів не обов'язково програмування, можливо застосування мови QBE (Query By Example – формулювання запитів за зразком). ПФСУБД мають засоби програмування для професійних розробників.

Сервери БД призначені для організації центрів обробки даних в комп'ютерних мережах. Ця група БД в даний час не чисельна, але кількість таких систем поступово зростає. Сервери БД реалізують функції управління базами даних, які запитуються іншими (клієнтськими) програмами допомогою операторів SQL.

Прикладами серверів БД є такі програми: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress).

У якості **клієнтів бази** даних використовуються різні програми, додатки : ПФСУБД, електронні таблиці, текстові процесори, програми електронної пошти і т. д. При цьому елементи пари «клієнт - сервер» можуть належати одному або різним виробникам програмного забезпечення. Якщо ми маємо систему з багатьма розробниками, тоді додатки інших розробників будуть клієнтами основної системи управління базами даних.

Засоби розробки програм роботи з БД Серед найбільш поширених засобів можна назвати наступні інструментальні системи:

- Delphi і Power Builder (Borland), Visual Studio (Microsoft), SILVERRUN (Compute r Advisers Inc.), S-Designor (SD P і Powersoft) і ERwin (LogicWorks).
- системи програмування (C ++,C #, Visual Basic, Java і ін.).

1.4 Реляційна модель баз даних

Реляційна модель даних запропонована співробітником фірми IBM Едгаром Коддом і ґрунтується на понятті відношення (relation).

Відношення це множина елементів, які називаються кортежами. У якості відношення використовують двовимірні таблиці.

Таблиця містить рядки (записи) і стовпці (поля). Кожен рядок таблиці має однакову структуру і складається з полів. Рядкам таблиці відповідають кортежі, а стовпцями - атрибути відносини.

Оскільки за допомогою однієї таблиці не можливо врахувати складні логічні структури даних, тому застосовують зв'язування таблиць.

Перевага реляційної моделі даних полягає в простоті, зрозумілості та зручності фізичної реалізації на комп'ютерах.

Основними недоліками реляційної моделі є: відсутність стандартних засобів ідентифікації окремих записів і складність опису ієрархічних і мережевих зв'язків.

Прикладами реляційних СУБД є: dBase! II Plus і dBase IY (фірма) Ashton-Tate), DB2 (IBM), R: BASE(Microrim), FoxBase (Fox Software), Paradox і dBASE for Windows (Borland), FoxPro пізніших версій, Visual FoxPro і Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) і Oracle (Oracle).

1.5 Частини баз даних

Існуючі реляційні бази даних мають декілька частин які властиві для всіх систем.

Комп'ютерна база даних — це контейнер об'єктів. За допомогою , програм бази даних можна:

- додавати нові дані до бази даних, наприклад новий предмет до інвентарного списку;

- редагувати наявні дані бази даних, наприклад змінювати поточне розташування предмета;
- видаляти відомості, якщо, наприклад, предмет було продано або списано;
- впорядковувати та переглядати дані різними способами;
- спільно користуватися даними з іншими користувачами за допомогою звітів, повідомлень електронної пошти, інтрамережі або Інтернету.

1.5.1 Таблиці

Таблиця бази даних схожа на електронну таблицю, в якій дані зберігаються в рядках і стовпцях. В результаті зазвичай досить легко імпортувати електронну таблицю до таблиці бази даних. Головна відмінність між збереженням даних в електронній таблиці та базі даних — це спосіб упорядкування даних.

Щоб забезпечити максимальну гнучкість бази даних, дані необхідно впорядкувати в таблицях, щоб позбутися зайвих елементів. Наприклад, якщо потрібно зберігати дані про працівників, відомості про кожного працівника необхідно *один раз* ввести в таблиці, яка буде використана і усіма документами в яких містяться відомості про працівників. Цей процес називається *оптимізацією*.

Кожний рядок у таблиці називається записом. Записи — це місце розташування окремих елементів даних. кожний запис складається з одного або кількох полів. Поля відповідають стовпцям у таблиці. Наприклад, можна створити таблицю «Працівники», де кожний запис (рядок) зберігає відомості про окремого працівника, а кожне поле (стовпець) містить власний тип даних, наприклад ім'я, прізвище, адресу тощо. Поля мають містити певний тип даних: текст, дату або час, число або інший тип.

Ще один спосіб опису записів і полів: уявіть старий картковий каталог у бібліотеці. Кожна картка у ящику відповідає запису бази даних. кожний елемент даних на окремій картці (автор, назва тощо) відповідає полю бази даних.

1.5.2 Форми

Форми іноді називаються «екранами вводу даних». Це інтерфейси, які використовуються під час роботи з даними, тому вони часто містять кнопки для виконання різних команд. Можна створити базу даних без використання форм, просто

редагуючи дані в таблицях даних. Проте більшість користувачів баз даних використовують форми для перегляду, введення та редагування даних у таблицях.

Форми пропонують простий у використанні формат роботи з даними, крім того, до них можна також додавати функціональні елементи, наприклад кнопки. Ці кнопки можна настроїти для визначення даних, що відображатимуться у формі, відкриття інших форм або звітів та для виконання низки інших завдань. Наприклад, є форма «Форма клієнта», у якій виконується робота з даними клієнта. Форма клієнта може містити кнопку, яка відкриває форму замовлення, де можна ввести нове замовлення цього клієнта.

Форми також дають змогу керувати способом взаємодії інших користувачів із даними бази даних. Наприклад, можна створити форму, яка відображає лише певні поля та дозволяє виконувати лише певні операції. Це допомагає захистити дані та забезпечує належне введення даних.

1.5.3 Звіти

Звіти використовуються для зведення та представлення даних у таблицях. Звіт зазвичай відповідає на певне питання, наприклад «Яку суму було отримано від кожного клієнта цього року?» або «У яких містах розташовані наші клієнти?». кожний звіт можна відформатувати таким чином, щоб він представляв дані найбільш зрозумілим способом.

Звіт можна запустити будь-коли, і він завжди відображатиме поточні дані в базі даних. Звіти зазвичай мають формат для друку, але їх також можна переглядати на екрані, експортувати до іншої програми або надсилати електронною поштою.

1.5.4 Запити

Запити — це справжні робочі коники бази даних, які можуть виконувати багато різних функцій. Їх найпоширеніша функція — отримання певних даних із таблиць. Дані, які потрібно переглянути, зазвичай розташовані в кількох таблицях, і запити дають змогу переглянути їх в одній таблиці даних. Також, оскільки зазвичай не потрібно бачити всі записи одночасно, запити дозволяють додавати критерії для «фільтрування» даних, щоб переглядати лише потрібні записи. Запити часто виконують роль джерела записів для форм і звітів.

Певні запити є «оновлюваними», тобто дані в базових таблицях можна редагувати за допомогою таблиці даних запиту. Якщо дії виконуються з оновлюваним запитом, слід пам'ятати, що зміни насправді виконуються в таблицях, а не лише в таблиці даних запиту.

Запити поділяються на дві основні групи: запити на вибірку і запити на дію. Запит на вибірку просто отримує дані й робить їх доступними для використання. Результати запиту можна переглянути на екрані, роздрукувати або скопіювати до буфера обміну. Або можна використати результат запиту як джерело записів для форми чи звіту.

Запит на змінення, згідно з назвою, виконує з даними певне завдання. Запити на змінення можна використовувати для створення нових таблиць, додавання даних до наявних таблиць, оновлення та видалення даних.

1.5.5 Макроси

Макроси в БД можна вважати спрощеною мовою програмування, яку можна використовувати для додавання функціональності до бази даних. Наприклад, можна вкласти макрос до кнопки форми, щоб запускати макрос у разі натискання цієї кнопки. Макроси містять дії, які виконують завдання, наприклад відкривають звіт, виконують запит або закривають базу даних. Більшість операцій із базою даних, які виконуються вручну, можна зробити автоматичними за допомогою макросів, тому вони можуть бути корисним засобом економії часу.

1.5.6 Модулі

Модулі, як і макроси, — це об'єкти, які можна використовувати для додавання функціональності до бази даних. Проте, якщо макроси Access створюються за допомогою вибору зі списку дій макросу, модулі пишуться мовою програмування Visual Basic for Applications (VBA). Модуль — це збірка декларацій(описів даних), інструкцій (команд мови програмування) і процедур(підпрограм), які зберігаються разом. Модуль може бути модулем класу або стандартним модулем.

Модулі класу додаються до форм або звітів і зазвичай містять процедури, характерні для форми чи звіту, до яких вони додаються. Стандартні модулі містять загальні процедури, не пов'язані з жодним іншим об'єктом. Стандартні модулі відображаються в області переходів у розділі.

Бази даних, створені у форматі Access мають розширення імені файлу, *.accdb .

1.6. Нормалізація таблиць в реляційних базах даних

Розглянемо причини , з яких база даних може бути недосконалою.

1. **Надмірність.** Дані в базі даних багаторазово повторюються.
2. **Потенційна суперечливість** (аномалії оновлення). Внаслідок надмірності можна оновити адрес постачальника в одному рядку, залишаючи його незмінним в інших. Отже, при оновленнях необхідно переглядати всю таблицю для знаходження і зміни всіх відповідних рядків.
3. **Аномалії включення.** В базу даних не може бути записаний новий постачальник, якщо продукт, що він постачає не використовується жодним споживачем.
4. **Аномалії видалення.** Проблема виникає при необхідності видалення з бази даних інформації про всі продукти, окремого постачальника. При таких видаленнях можуть бути втрачені відомості про самого постачальника.

Для того щоби уникнути недосконалостей бази даних, які не дозволяють їй успішно функціонувати, виконують нормалізацію таблиць бази даних.

Нормалізація - це розбиття таблиці на дві або більше, що мають кращі властивості при включенні, зміні і видаленні даних. Остаточна мета нормалізації зводиться до отримання такого проекту бази даних, в якому *кожен факт з'являється лише в одному* місці, тобто виключена надмірність інформації. Це робиться не стільки з метою економії пам'яті, скільки для виключення можливої суперечливості збережених даних.

Теорія нормалізації ґрунтується на наявності тієї чи іншої залежності між полями таблиці. Визначено два види таких залежностей: функціональні і багатозначні.

Функціональна залежність. Поле у таблиці функціонально залежить від поля А тієї ж таблиці в тому і тільки в тому випадку, коли в будь-який заданий момент часу для кожного з різних значень поля А обов'язково існує тільки одне з різних значень поля В. Тут допускається, що поля А і В можуть бути складовими.

Повна функціональна залежність. Поле В знаходиться в повній функціональної залежності від складеного поля А, якщо воно функціонально залежить від А і не залежить функціонально від будь-якої підмножини поля А.

Багатозначна залежність. Поле А багатозначно визначає поле У тій же таблиці, якщо для кожного значення поля А існує добре певну безліч відповідних значень В.

Для прикладу розглянемо таблицю 1. У ній є багатозначна залежність "Дисципліна-Викладач": дисципліна (у прикладі Інформатика) може може читатися кількома викладачами.

Багатозначні залежності Таблиця 1

Дисципліна	**Викладач**	**Підручник**
Інформатика	Шипілов П.А.	Форсайт Р. Паскаль для всіх
Інформатика	Шипілов П.А.	Уейт М. та ін. Мова Сі
Інформатика	Голованівський Г.Л.	Форсайт Р. Паскаль для всіх
Інформатика	Голованівський Г.Л.	Уейт М. та ін. Мова Сі
…	…	…

Нормальні форми

Таблиця знаходиться в *першій нормальній формі (1НФ)*тоді і тільки тоді, коли жодна з її рядків не містить в будь-якому своєму полі більше одного значення і жодне з її ключових полів не порожньо.

Таблиця знаходиться в *другій нормальній формі (2НФ),*якщо вона задовольняє визначенню 1НФ і всі її поля, що не входять в первинний ключ, пов'язані повної функціональної залежністю з первинним ключем.

Таблиця знаходиться в *третій нормальній формі (3НФ),*якщо вона задовольняє визначенню 2НФ і не одне з її неключових полів не залежить функціонально від будь-якого іншого неключових поля.

Таблиця знаходиться в *нормальній формі Бойса-Кодда (НФБК),*якщо і тільки якщо будь-яка функціональна залежність між його полями зводиться до повної функціональної залежності від *можливого* ключа.

Таблиця знаходиться в четвертій нормальній формі, якщо вона знаходиться в НФБК і все нетривіальні багатозначні залежності фактично є функціональними залежностями від її ключів.

Таблиця знаходиться в п'ятій нормальній формі (5НФ) тоді і тільки тоді, коли в кожній її повної декомпозиції все проекції містять можливий ключ.

1.6.1. Процедура нормалізації

Нормалізація - це процес послідовної заміни таблиці її повними декомпозиції до тих пір, поки всі вони не будуть знаходитися в 5НФ. На практиці ж досить привести таблиці до НФБК

Ця процедура ґрунтується на тому, що єдиними функціональними залежностями в будь-якій таблиці повинні бути залежності виду K-> F, де K - первинний ключ, а F - деяке інше поле.

1.7. Проектування бази даних

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

Визначивши основну частину даних, які замовник використовує, розпочинають розробку структури бази, тобто структури її основних таблиць.

1. Робота починається з визначення генерального переліку полів, який може нараховувати десятки і сотні позицій.
2. Відповідно до типу даних, що розміщуються в кожному полі, визначають тип кожного поля.
3. Розподіляють поля генерального списку по базових таблицях.
    a. На першому етапі розподіл здійснюють за функціональною ознакою. Мета - забезпечити одноразове введення даних в одну таблицю по можливості в рамках одного підрозділу, або (ще краще) - на одному робочому місці.
    b. На другому етапі розподілу полів здійснюють нормалізацію даних з метою вилучення повторів даних у таблицях бази даних.
4. Для кожної таблиці визначають ключове поле. Ключовим вибирають поле, дані в якому повторюватись не можуть. Наприклад, для таблиці даних про студентів таким полем може бути індивідуальний шифр студента. Для таблиць, у яких міститься розклад занять, такого поля можна і не знайти, але його можна створити штучно комбінуванням полів "Час заняття" і "Номер аудиторії". Ця комбінація унікальна, оскільки в певній аудиторії в певний час назагал не проводять двох різних занять. Якщо ж у таблиці взагалі немає полів, які можна було б використовувати як ключові,

завжди можна ввести додаткове поле типу лічильник - воно за визначенням не може містити дані, що повторюються.

5. На наступному етапі визначають зв'язки між таблицями (схему даних). Зв'язки між таблицями організуються на основі спільного поля, причому в одній із таблиць воно обов'язково має бути ключовим. Тобто на стороні "один" має бути ключове поле, яке не повторюється, значення на стороні "багато" можуть повторюватися.
6. "Паперовий" етап роботи над технічними пропозиціями закінчується розробкою схеми даних. Цю схему слід узгодити із замовником, після чого розпочати безпосереднє створення бази даних. Слід пам'ятати, що в ході розробки проекту замовникові неодмінно будуть надходити нові ідеї. Можливість гнучкого використання його побажань суттєво залежить від кваліфікації розробника бази даних. Якщо схема даних складена правильно, підключити до бази нові таблиці неважко. Якщо структура бази нераціональна, розробник може наштовхнутись на суттєві труднощі і дійти суперечності із замовником. Суперечка виконавця із замовником завжди свідчить про недостатню кваліфікацію виконавця. На цьому етапі завершується попереднє проектування бази даних, і на наступному етапі починається її безпосередня розробка бази даних на машинних носіях і її впровадження в експлуатацію.

1.8. Адміністрування баз даних

Адміністрування БД передбачає виконання функцій, спрямованих на забезпечення надійного та ефективного функціонування системи БД, адекватності змісту БД інформаційним потребам користувачів, відображення у БД актуального стану предметної області, рис. 3. Ці функції виконує адміністратор бази даних (АБД) – професіональний спеціаліст або група спеціалістів у галузі інформаційних технологій, що працюють на всіх

рівнях і відповідають за взаємодію між користувачами і системою.

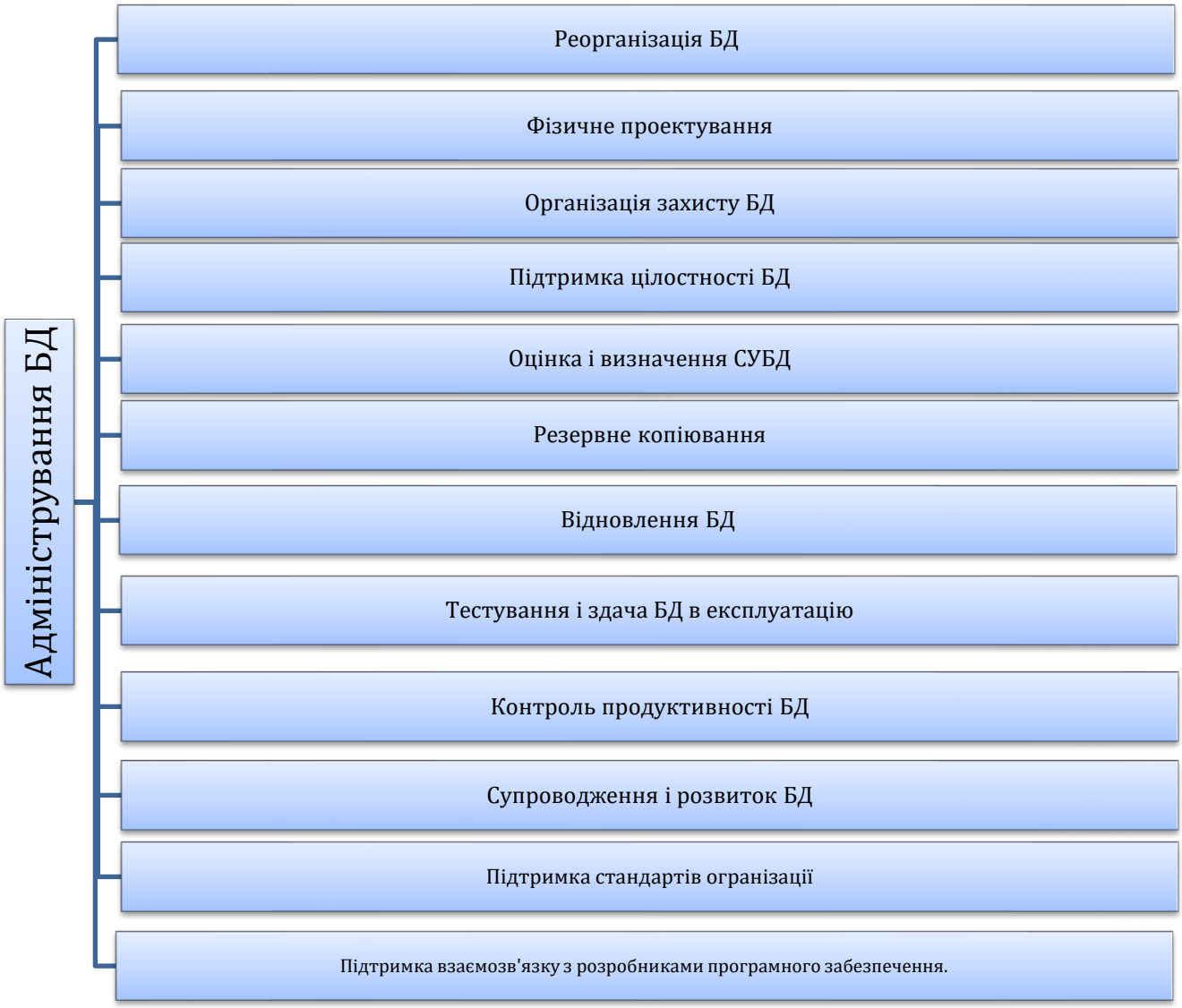


Рисунок 3. Задачі адміністрування бази даних

У діапазон відповідальності АБД входить:

1. – проектування та реалізація БД;
2. – визначення правил захисту та цілісності БД (як частини концептуальної схеми);
3. – спостереження за роботою користувачів (допомога та консультації),
4. управління ефективністю роботи системи .

У процесі своєї діяльності АБД взаємодіє з іншими категоріями користувачів БД, а також з іншими спеціалістами, які не є користувачами БД. Працює АБД на всіх рівнях представлення даних.

Існують три типи адміністраторів баз даних:

1. Системні адміністратори баз даних (також звані фізичні адміністратори баз даних, операційні адміністратори баз даних або адміністратори підтримки баз даних): зосереджуються на фізичних аспектах адміністрування баз даних, таких як встановлення СУБД, конфігурація, виправлення, покращення, резервне копіювання, відновлення, оновлення, оптимізація продуктивності, обслуговування та аварійне відновлення .
2. Адміністратори розробки баз даних: зосереджуються на логічних аспектах і аспектах управління базами даних, таких як розробка та підтримка моделі даних.
3. Адміністратори програмного забезпечення баз даних: зазвичай зустрічаються в організаціях, які придбали програмне забезпечення третьої сторони, такі як ERP (планування ресурсів підприємства) та CRM (системи управління взаємовідносинами з клієнтами). Прикладами таких прикладних програм є Oracle Applications, Siebel і PeopleSoft (обидві тепер є частиною Oracle Corp.) і SAP. Адміністратори програмного забезпечення баз даних керують усіма компонентами програми, які взаємодіють з базою даних і виконують такі дії, як встановлення та виправлення програм, оновлення програм, клонування баз даних, побудова та виконання процедур очищення даних, керування процесами завантаження даних, тощо.

В невеликих організаціях окрема особа або група працівників, які виконують декілька типів адміністрування баз даних.

1.9. Комерційні бази даних

Нині на ринку домінують DB2, SQL Server, Oracle та IBM. У ОС Windows SQL Server є звичайною базою даних вибору, тоді як Oracle і DB2 є верхівковими хижаками екосистеми Mainframe / Unix / Linux.

Microsoft SQL Server

Розроблено корпорацією Майкрософт, є виключно сумісним з Windows. З моменту її інтеграції з Microsoft Azure його гнучкість та продуктивність покращилися, а також тепер дозволяє отримувати інформацію з інших серверів, підвищуючи його зручність користування.

Порівняльний аналіз: http://www.microsoft.com/en-us/server-cloud/products/sql-server-benchmarks/industry.aspx

Oracle

Oracle може працювати практично в будь-якій системі. Також слід відзначити, що є багато інструментів, орієнтованих на моніторинг та адміністрування Oracle.

Oracle Benchmark: http://www.oracle.com/us/solutions/performance-scalability/index.html.

IBM DB2

Друга найбільш часто використовувана база даних для Unix / Linux, Windows за популярністю іде після Oracle, є найкращим вибором для Mainframe (суперкомп'ютерів). DB2 має своїх послідовників, навчених своїм мистецтвам, але їхні ініціативи менші, ніж для Oracle. Бенчмарк DB2: http://www-01.ibm.com/software/data/db2/performance.html

Teradata

Розроблена для Big Data, має велику потужність для зберігання та аналізу даних .

SAP Sybase

Використовується в автоматизованих системах управління. Залишається ефективною системою в плані масштабованості та продуктивності.

1.10. Контрольні запитання

1. Сформулюйте поняття бази даних і СУБД.
2. Сформулюйте поняття архітектури СУБД.
3. Сформулюйте поняття реляційної бази даних, особливості, відмінності.
4. Роз’ясніть нормалізацію, таблиць в реляційних базах даних.
5. Сформулювати поняття 1, 2, 3 нормальних форм.
6. Сформулювати поняття нормальної форми Кодда-Бойса
7. Об’єкти бази даних Access, їх особливості, призначення, взаємодія.
8. Створення і редагування зв'язків у таблицях.
9. Які існують сучасні комерційні бази даних?

10. Роз’ясніть типи адміністраторів БД.
11. Роз’ясніть основні задачі адміністрування баз даних
12. Які основні етапи проектування баз даних?

Тема 2. Нереляційні бази даних

В темі викладено основні поняття, формулювання, основи функціонування нереляційних баз даних.

Бази даних NoSQL, які називаються нереляційними , мають справу з дуже великими наборами розподілених даних. Ці бази охоплюють широкий спектр технологій і архітектур, які прагнуть вирішити питання продуктивності та масштабованості великих даних. *Великі дані* - термін, який описує будь-яку велику кількість структурованих, напівструктурованих та неструктурованих даних, які можуть бути використані для отримання інформації.

Нереляційні бази ефективні, коли підприємство потребує доступу і аналізу великих обсягів неструктурованих даних або даних, який зберігається віддалено на декількох хмарних серверах, рис.4 .

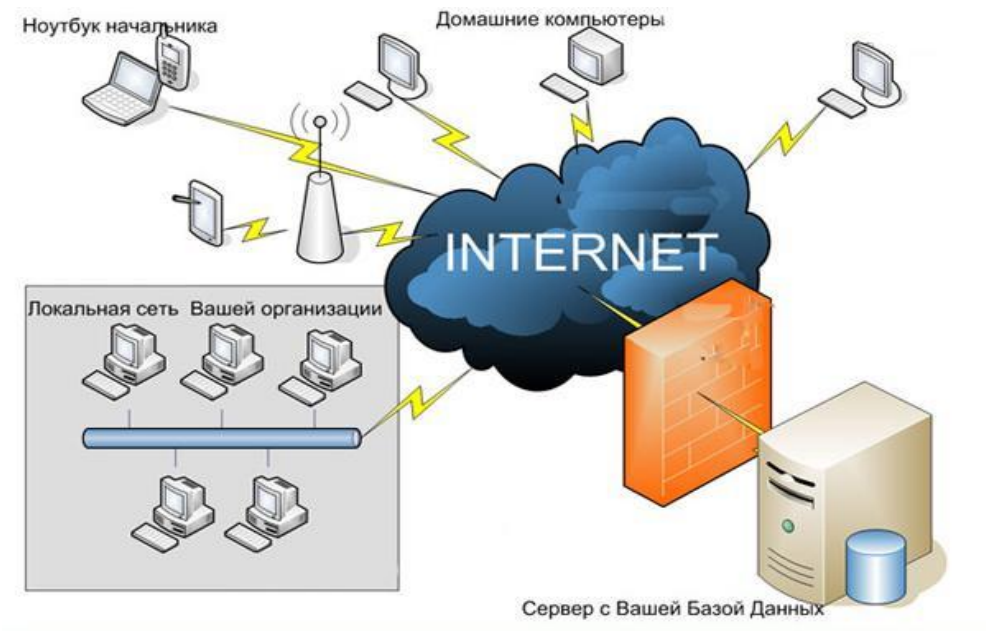


Рисунок 4. Нереляційна база даних

Найбільші нереляційні бази це Apache Cassandra, Google CLOUD BIGTABLE, Hadoop, ApacheMapReduce, Facebook, LinkedIn Netflix, Twitter.

NoSQL часто згадується в поєднанні з іншими великими інструментами масивної паралельної обробки даних, на основі стовпчастих (columnar database) баз даних і баз даних-як-послуга (DaaS).

2.1. Види NoSQL баз даних

Всього виділяють чотири основні типи NoSQL-сховищ. Вони розрізняються моделлю даних, підходом до розподіленності і реплікації, завдяки чому можуть в різній мірі підходити під ті чи інші види конкретних завдань.

Сховище виду "ключ-значення"

Сховища "ключ-значення" представляють собою найпростіший вид бази даних, будучи, по суті, асоціативним масивом - кожному значенню зіставляється свій унікальний ключ. Простота сховищ цього типу відкриває простори неймовірною масштабованості. Не потрібно ніяких схем побудови бази даних, немає ніякого зв'язку між значеннями, по суті кількість елементів асоціативного масиву обмежена лише обчислювальними потужностями. Саме тому даний вид сховищ цікавий в першу чергу компаніям, що надають послуги хмарного хостингу.

З іншого боку, простота сховищ "ключ-значення" дуже ускладнює або повністю відсікає більшість звичних операцій зі значеннями сховища - якщо ключами можна ще оперувати як завгодно, то спроба виконати пошук за значеннями може тривати на кілька порядків довше, ніж в реляційної базі даних. А разом з обмеженим набором маніпуляцій над значеннями осередків сховища йде і фактична неможливість швидко аналізувати наявну в базі даних інформацію і збирати статистику.

Саме тому подібні сховища використовуються в тих випадках, коли конкретне вміст окремої комірки не цікавий оператору бази даних - інакше кажучи, повністю відсутні зв'язки між окремими осередками сховища. Бази даних типу "ключ-значення" не підходять в якості повної заміни реляційних БД, але знайшли своє застосування в якості кешей для об'єктів.

Найбільш відомі приклади СУБД даного типу це Amazon DynamoDB, Berkeley DB, MemcacheDB, Redis і Riak.

Документоорієнтовані бази даних

Документоорієнтована БД являє собою систему зберігання ієрархічних структур даних (документів), що має структуру дерева або лісу. Структура дерева починається з кореневого вузла і може мати кілька внутрішніх і листових вузлів. Листові вузли містять кінцеві дані, які при додаванні заносяться в індекси бази, завдяки яким можна здійснювати швидкий пошук навіть при досить складною загальній структурі сховища, рис.5.

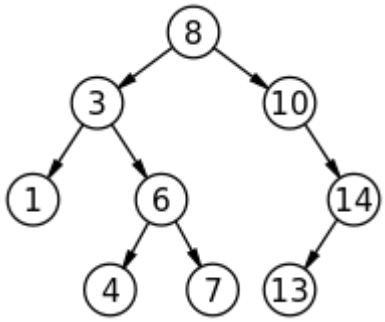


Рисунок 5 Структура документоорієнтованої бази даних

Фактично документоорієнтовані БД є більш складною версією сховищ "ключ-значення" - вони все не використовуються для систем, що мають багато зв'язків між елементами, але дозволяють здійснювати вибірку за запитом без повного завантаження окремих документів в оперативну пам'ять. Механізми пошуку дозволяють знаходити як документи цілком, так і частини документів, а деревоподібна структура дозволяє організовувати окремі колекції документів одного типу або схожої тематики.

Наприклад, при створенні музичного сховища можна створити колекцію музики 80-х років, в ній зробити окремі колекції по роках, а всередині них окремі документи з треками випущених в цей рік альбомів. Але якщо користувач побажає побачити рейтинг найпопулярніших композицій певного десятиліття - цей запит буде виконуватися досить довго, адже доведеться переглянути кожен документ всієї бази даних. Таким чином, можна зробити висновок що документоорієнтовані БД знайдуть своє застосування в задачах, де потрібно впорядковане зберігання інформації, але немає великої кількості зв'язків між даними і не потрібно постійно збирати статистику по ним.

Приклади СУБД даного типу: CouchDB, Couchbase, MarkLogic, MongoDB, eXist.

Графові бази даних

Графова модель бази даних являє собою узагальнення мережевої моделі даних і відрізняється сильними зв'язками між вузлами.

Графові бази даних найкраще підходять для реалізації проектів, які передбачають природну графову структуру даних - в першу чергу соціальних мереж, а так само для створення семантичного павутиння. У подібних завданнях вони сильно випереджають реляційні БД по продуктивності, простоті внесення змін і наочності подання інформації. У деяких баз даних існують механізми спеціальної оптимізації для роботи з SSD-накопичувачами. Для роботи з досить великими графами використовуються алгоритми, які передбачають часткове приміщення графа в оперативну пам'ять.

Найбільш відомі графові СУБД це ArangoDB, FlockDB, Giraph, HyperGraphDB, Neo4j, OrientDB.

Bigtable-подібні бази даних

Bigtable-подібні бази даних містять дані, впорядковані у вигляді розрідженій матриці, рядки і стовпці якої використовуються в якості ключів. Ці сховища мають багато спільного з документоорієнтованими БД і використовуються для системи керування вмістом, реєстрації подій, блоги.

Система керування вмістом - це комп'ютерне програмне забезпечення, яке підтримує створення та зміну цифрового вмісту. Система використовується для підтримки декількох користувачів, які працюють у спільному середовищі.

Як правило, ці сховища застосовуються для веб-індексування і рішення інших завдань, які передбачають величезні обсяги даних.

Прикладами СУБД даного типу є: HBase, Cassandra, Hypertable, SimpleDB.

2.2. Контрольні запитання

1. Охарактеризуйте нереляційних баз даних .
2. Основні принципи функціонування відомих нереляційних баз даних.

3. Які існують типи нереляційних баз даних.
4. Які відмінності реляційних і нереляційних баз даних?
5. В яких випадках доцільно використовувати нереляційні бази даних.
6. Що означає термін NoSQL.
7. Які основні типи NoSQL-сховищ існують?
8. Роз’ясніть поняття Bigtable-подібних баз даних

Тема 3. Сховища даних

В темі викладено основні поняття, формулювання, основи функціонування сховищ даних

Сховище даних (СД) — предметно орієнтований, інтегрований, незмінний набір даних, що підтримує хронологію і здатний бути комплексним джерелом достовірної інформації для оперативного аналізу та прийняття рішень. 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 — основа для побудови систем підтримки прийняття рішень. Основна мета створення сховища 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. Незважаючи на відмінності в підходах та реалізаціях, усім сховищам даних властиві такі спільні риси: предметна орієнтованість, інтегрованість, прив'язка до часу, незмінність.

3.1 Основні характеристики сховищ даних

Предметна орієнтованість

Інформація в сховищі даних організована відповідно до основних аспектів діяльності підприємства (замовники, продажі, склад тощо). Це відрізняє сховище даних від оперативної БД, де дані організовано відповідно до процесів (виписка рахунків, відвантаження товару тощо). Предметна організація даних в сховищі сприяє як значному спрощенню аналізу, так і підвищенню швидкості виконання аналітичних запитів. Вона виражається, зокрема, в використанні інших, порівняно з оперативними системами, систем організації даних.

Інтегрованість даних

Перш ніж потрапити до сховища даних оперативні дані перевіряють, очищають та певним чином агрегують. Вихідні дані отримуються із оперативних БД, перевіряються, очищаються, приводяться до єдиного виду, в потрібній мірі агрегуються (вираховуються сумарні та інші статистичні показники) і завантажуються в сховище. Такі інтегровані дані набагато простіше аналізувати.

Підтримка хронології даних

Дані в сховищі завжди напряму пов'язані з певним періодом часу. Дані, отримані із оперативних БД, накопичуються в сховищі у виді «історичних шарів», кожен з яких стосується конкретного періоду часу. Це дозволяє аналізувати тенденції в розвитку бізнесу.

Незмінність даних

Потрапивши в певний «історичний шар» сховища, дані вже ніколи не мінятимуться. Це також відрізняє сховище від оперативної БД, в якій дані постійно змінюються, у зв'язку з чим один і той же запит, виконаний в різні моменти часу, може дати різні результати. Стабільність даних також полегшує їх аналіз.

Мінімальна надлишковість

Не зважаючи на те, що інформація до сховища даних потрапляє від багатьох OLTP-систем, надлишковість інформації в сховищі даних зведена до мінімуму.

3.2. Види сховищ даних

При використанні СППР можуть застосовуватись два види сховищ даних:

Фізичне сховище даних

При реалізації моделі СППР з фізичним СД дані з різних джерел копіюються в єдине сховище. Зібрані дані приводяться до єдиного формату, узгоджуються та узагальнюються.

Аналітичні запити адресуються до сховища даних.

Така модель приводить до дублювання інформації в оперативних джерелах даних та в СД.

Проте така надлишковість не перевищує 1%. Це пояснюється такими причинами:

- при завантаженні інформації з оперативних джерел даних в сховище дані фільтруються і дані які не мають змісту не потрапляють в СД,
- під час завантаження в СД дані очищаються (видаляється непотрібна інформація) і приводяться до єдиного формату. Після такої обробки дані займають значно менший обсяг.

Віртуальне сховище даних

В даному випадку на відміну від фізичного СД дані з оперативних джерел даних не копіюються в єдине сховище. Вони витягуються, перетворюються та інтегруються безпосередньо при виконанні аналітичних запитів в оперативній пам'яті комп'ютера. Фактично такі запити напряму адресуються до оперативних джерел даних. Основними перевагами віртуального СД є:

- мінімізація обсягу пам'яті, який займають дані на носії інформації;
- робота з поточними, деталізованими даними.

Сховища даних використовуються в системах підтримки прийняття рішень для аналітичної обробки інформації для розробки нових проектів і прийняття рішень в умовах невизначеності.

Конструкція “Сховища даних підприємства” показані на рис. 6 .

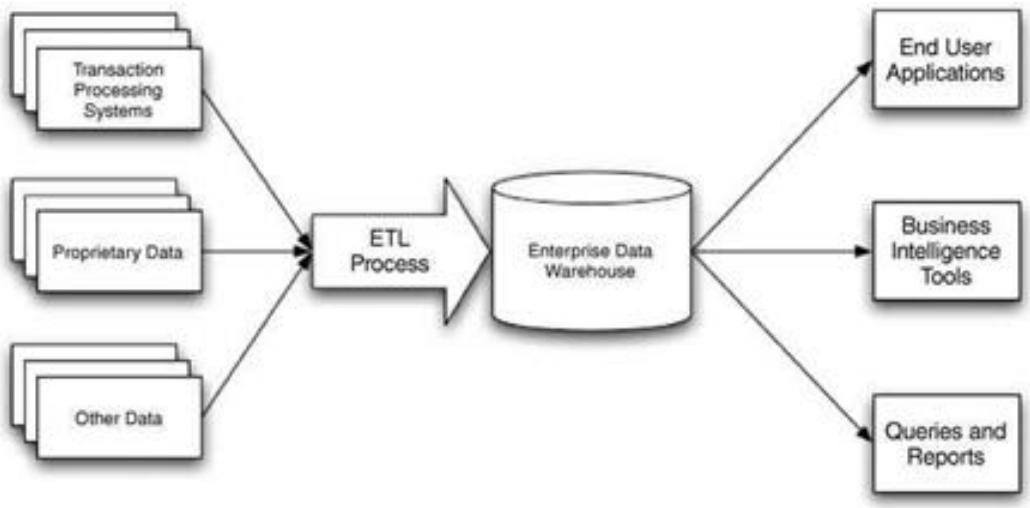
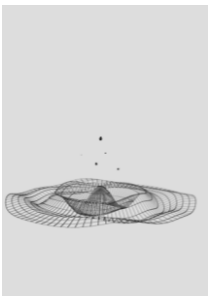


Рисунок 6. Схема сховища даних підприємства

- Extract, Transform, and Load (ETL) або Виймання, Перетворення, Завантаження — процес, який використовується в базах даних та, особливо, у сховищах даних.
- Він охоплює наступні етапи обробки даних:
- Виймання даних із зовнішніх джерел
- Перетворення даних, для зберігання даних у відповідній структурі або форматі, з метою подальшого аналізу. Перетворення даних пов'язане з різницею логічних структур даних, а також з такими проблемами:
    - багатомодельність представлення даних (ієрархічні, мережні, реляційні) в різних БД і СКБД;
    - різниця в логічних структурах даних, в довідниках, класифікаторах і в системах кодування інформації;
    - використання різних мов для представлення текстової інформації;
    - різні типи СКБД і постійний розвиток даних БД в процесі експлуатації.
- Завантаження даних у сховище даних.



Озера даних

Озеро даних - це сховище , яке містить величезну кількість необроблених даних у своєму власному форматі. Озеро даних використовує плоску архітектуру для зберігання даних. Кожному елементу даних озера присвоюється унікальний ідентифікатор і позначається набір розширених тегів метаданих. Коли виникає бізнес-питання, на озеро даних можна запитувати відповідні дані, і тоді менший набір даних може бути проаналізований, щоб допомогти відповісти на запитання.

Унікальний ідентифікатор (UID) - це числовий або буквено-цифровий рядок, пов'язаний з єдиним об'єктом у межах даної системи. Ідентифікатори UID дозволяють звертатися до цього об'єкта, так що до нього можна отримати доступ і взаємодіяти з ним.

Метадані - це дані, які описують інші дані. Мета є префіксом, який у більшості випадків використання інформаційних технологій означає "основне визначення або опис". Метадані узагальнюють основну інформацію про дані, яка спрощує пошук і роботу з окремими примірниками даних. Наприклад, автор, дата створення і дата зміни і розмір файлу є прикладами дуже простих метаданих документів. Наявність здатності фільтрувати ці метадані полегшує комусь знайти конкретний документ.

Метадані використовуються для зображень, відео, електронних таблиць і веб-сторінок. Використання метаданих на веб-сторінках може бути дуже важливим. Метадані для веб-сторінок містять опис вмісту сторінки, а також ключові слова, пов'язані з вмістом. Зазвичай вони виражаються у вигляді метатегів. Метадані, що містять опис та резюме веб-сторінки, часто відображаються в результатах пошуку пошуковими системами, завдяки чому їх точність і деталізація дуже важливі, оскільки вона може визначити, чи вирішує користувач користуватися сайтом чи ні. Метатеги часто оцінюються пошуковими системами, щоб допомогти вирішити актуальність веб-сторінки, і використовувалися як ключовий фактор у визначенні позиції в пошуку до кінця 1990-х років. Збільшення пошукової оптимізації (SEO) до кінця 1990-х років призвело до того, що багато веб-сайтів "набирають ключові слова" свої метадані, щоб обдурити пошукові системи, роблячи їхні сайти більш актуальними, ніж інші. З тих пір пошукові системи зменшили свою залежність від метатегів, хоча вони все ще враховуються при індексації сторінок. Багато пошукових систем також намагаються зупинити здатність веб-сторінок перешкодити їхній системі, регулярно змінюючи свої критерії рейтингу, оскільки Google був відомий тим, що часто змінював свої алгоритми ранжування.

Метадані можуть бути створені вручну або за допомогою автоматизованої обробки інформації. Ручне створення має тенденцію бути більш точним, дозволяючи користувачеві вводити будь-яку інформацію, яку вони вважають релевантною або необхідною для опису файлу. Автоматизоване створення метаданих може бути набагато більш елементарним, як правило, відображається тільки така інформація, як розмір файлу, розширення файлу, коли файл був створений і хто створив файл.

Мета-тег- це тег (вираз кодування) у мові розмітки гіпертексту (HTML), який описує певний аспект вмісту веб-сторінки. Інформація, яку ви надаєте в метатезі, використовується пошуковими системами для індексування сторінки. Мета-тег розміщується у верхній частині HTML у веб-сторінці як частина

заголовка. Найважливішим для індексації пошукової системи є мета-тег ключових слів і мета-тег опису. Мета-тег ключових слів перераховує слова або фрази, які найкраще описують вміст сторінки. Мета-тег опису містить короткий опис однієї або двох речень сторінки. І ключові слова, і опис використовуються пошуковими системами для додавання сторінки до свого індексу. Деякі пошукові системи також використовують опис, щоб показати пошуковій системі резюме вмісту сторінки.

Незважаючи на те, що більшість пошукових систем також використовують вміст сторінки, щоб визначити, як індексувати її, розробник повинен обов'язково включати мета-теги з відповідними ключовими словами та описом. Якісно написані мета-теги допомагають підвищити рейтинг сторінки в результатах пошуку.

3.3. Контрольні запитання

1. Сформулюйте поняття сховища даних.
2. Де використовуються сховища даних.
3. Яка типи сховищ даних існують?
4. Роз’ясніть схему сховища даних підприємства.
5. Як функціонує сховище даних?
6. В чому відмінність баз даних і сховищ даних?
7. Роз’ясніть поняття озера даних.
8. Яка відмінність озера даних від бази даних?
9. Що таке метадані і мета-тег?
10. Що таке унікальний ідентифікатор (UID)

Тема 4. Блокчейн та розподілені бази даних

В темі викладено основні поняття, формулювання, основи функціонування блокчейн

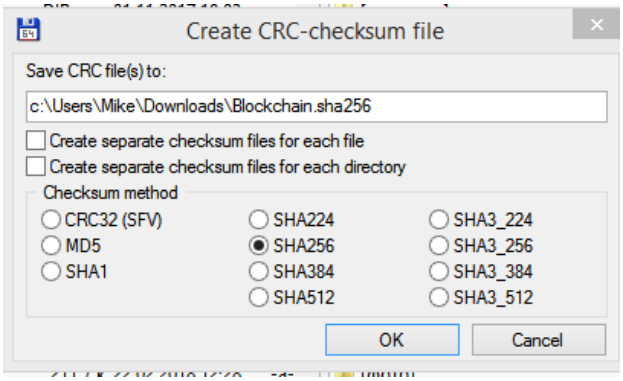
4.1. Основи функціонування блокчейн

Блокчейн – це розподілена база даних, у якій зберігається інформація про кожну транзакцію, вироблену в системі. Дані зберігаються у вигляді ланцюжка блоків (звідси і назва – blockchain) з записами про транзакції. Їх неможливо підробити, так як кожен новий запис здійснює підтвердження вже існуючих ланцюжків.

Транзакція— група послідовних операцій з базою даних, яка є логічною одиницею роботи з даними. Транзакція може бути виконана або цілком і успішно, дотримуючись цілісності даних і незалежно від інших транзакцій, що йдуть паралельно, або не виконана зовсім, і тоді вона не може справити ніякого ефекту. Транзакції обробляються транзакційними системами, в процесі роботи яких створюється історія транзакцій. Розрізняють послідовні (звичайні), паралельні і розподілені транзакції. Розподілені транзакції вбачають використання більш ніж однієї транзакційної системи і потребують набагато більш складної логіки.

Геш-функція — функція, що перетворює вхідні дані будь-якого розміру в дані фіксованого розміру. Гешування — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини.

SHA256 815f782fb4b1d519f4fe4c54ad3ef2928979b1700c3312001b042dc8ee90c6ed



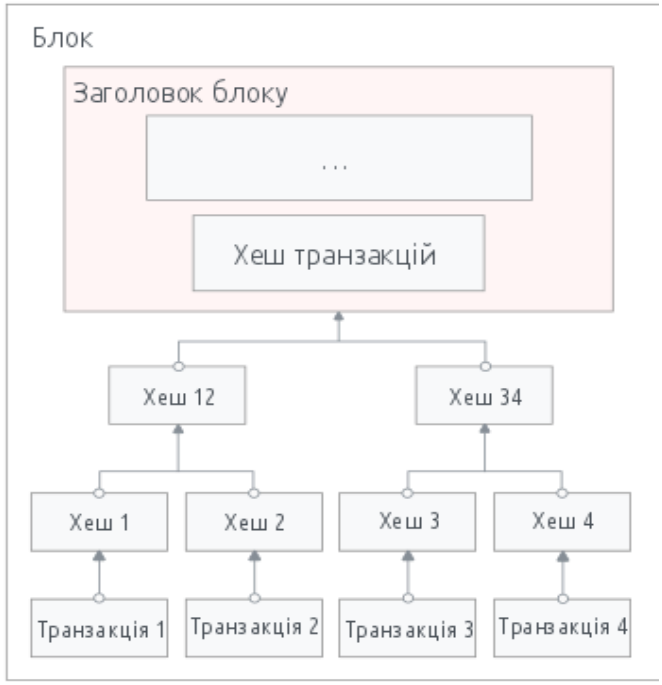


Рисунок 7. Блокчейн і геш-функція.

Кожен блок містить часову мітку та посилання на попередній блок хеш дерева.

Учасники мережі блокчейна обмінюються фактами. Факти, що зберігаються в блокчейне, не можуть бути загублені. Вони залишаються там назавжди, репліціручись на кожен вузол. Блокчейн не просто зберігає кінцеве стан, він зберігає і всі попередні стану. Тому кожен може перевірити правильність кінцевого стану, перераховуючи факти з самого початку. Фактам в блокчейне ми можемо довіряти, тому що вони технічно підтверджуються консенсусом. Навіть якщо в мережі знаходяться зловмисники, ви все одно можете довіряти її судження в цілому.

В блокчейне працює правило більшості. Блоки розсилаються усім учасникам , які мають підтвердити правильність цих даних (по гешам) рис.7.

Подібна проблема з'ясування істини в умовах, коли твої співрозмовники можуть брехати, була названа «Проблемою візантійських генералів», і вирішена в 1980 році. Було показано, що при n шпигунів, які можуть брехати і спотворювати інформацію, консенсус між учасниками може бути досягнутий при загальній кількості учасників 3n + 1. А якщо гарантувати, що шпигуни не можуть спотворювати передану через них повідомлення, то досить і 2n + 1. У блокчейне за рахунок електронного

підпису шкідливі вузли не можуть спотворювати інформацію, тому якщо в блокчейне менше половини шкідливих вузлів, то мережу стійка.

Стійкість мережі до шкідливим вузлів називається стійкістю до візантійської проблеми (Byzantine Fault Tolerance, BFT). BFT дуже важлива для публічних мережевих систем, в які можуть вільно додаватися довільні вузли. Саме такими системами є більшість проектів на блокчейне.

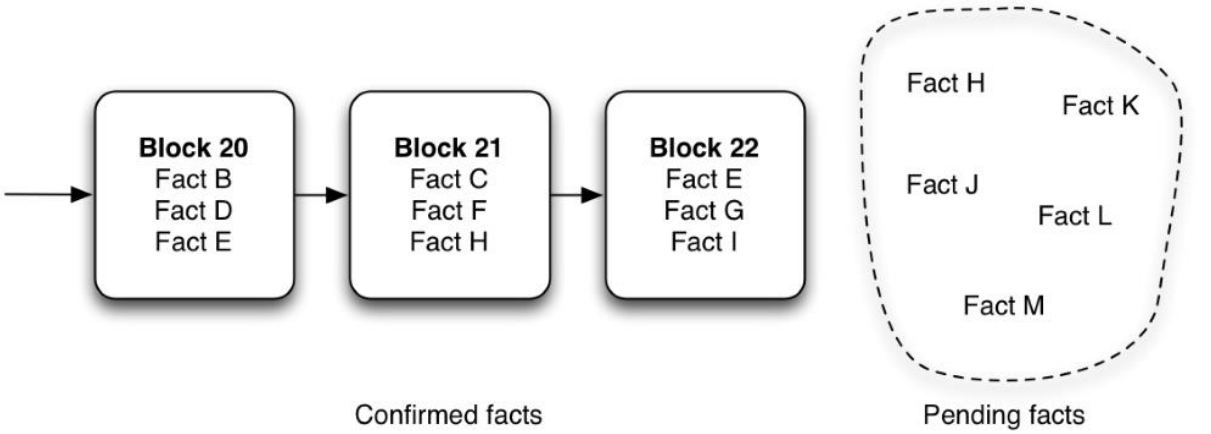


Рисунок 8. Блоки і факти в блокчейне

Блок складається із заголовка та списку транзакцій. Заголовок блоку включає в себе свій хеш, хеш попереднього блоку, геши транзакцій та додаткову службову інформацію. Першою транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок.Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки.

Створений блок буде прийнятий іншими користувачами, якщо числове значення хешу заголовка менше або дорівнює певному числу, величина якого періодично коригується. Оскільки результат хешування (функції SHA-256) необоротний, немає алгоритму отримання бажаного результату, окрім випадкового перебору. Якщо хеш не задовольняє умову, то довільно змінюється блок службової інформації в заголовку, і хеш перераховується. Зазвичай потрібна велика кількість перерахунків. Коли варіант знайдено, вузол розсилає отриманий блок іншим підключеним вузлам, які перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок, і наступний блок повинен включити в себе його хеш.

Блоки - засіб, щоб упорядкувати факти в мережі з недовірених вузлами. Ідея проста: факти групуються в блоки, і є тільки один ланцюжок блоків, які

репліцируються по всій мережі. Кожен блок посилається на попередній. Тобто, якщо факт F знаходиться в блоці 21, і факт E в блоці 22, то факт E розглядається всією мережею як наступний за фактом F. Перед додаванням до блоку, факти знаходяться на розгляді, тобто не підтверджені.

P2P-мережам, як і іншим розподіленим системам, доводиться вирішувати дуже складну проблему інформатики: розв'язання конфліктів між блоками при передачі в мережі. Реляційні бази даних пропонують цілісність за допомогою зв`язків, але такої особливості немає в розподілених системах. Якщо два несумісних факти прибувають в один і той же час, система повинна мати правила для визначення того, який факт вважати першим а який другим.

У P2P мережах, два факти відправлені приблизно в один час можуть прибути в різному порядку в віддалені вузли. Тоді як мережі узгодити який факт прийшов першим? Щоб гарантувати цілісність в P2P мережі, потрібен спосіб узгодження порядку фактів, потрібна система консенсусу.

Вузли в ланцюжку створюють нові локальні блоки з непідтвердженими фактами. Вони змагаються, щоб дізнатися, чи стане їх локальний блок наступним блоком в ланцюзі для всієї мережі, шляхом кидка гральних кісток. Якщо вузол викидає дві шістки, то він отримує можливість опублікувати його локальний блок, і всі факти в цьому блоці стають підтвердженими, рис. 9. Цей блок надсилається всім вузлам в мережі. Всі вузли перевіряють, що блок правильний, додають його до їх копії ланцюга і намагаються побудувати новий блок з новими непідтвердженими фактами.

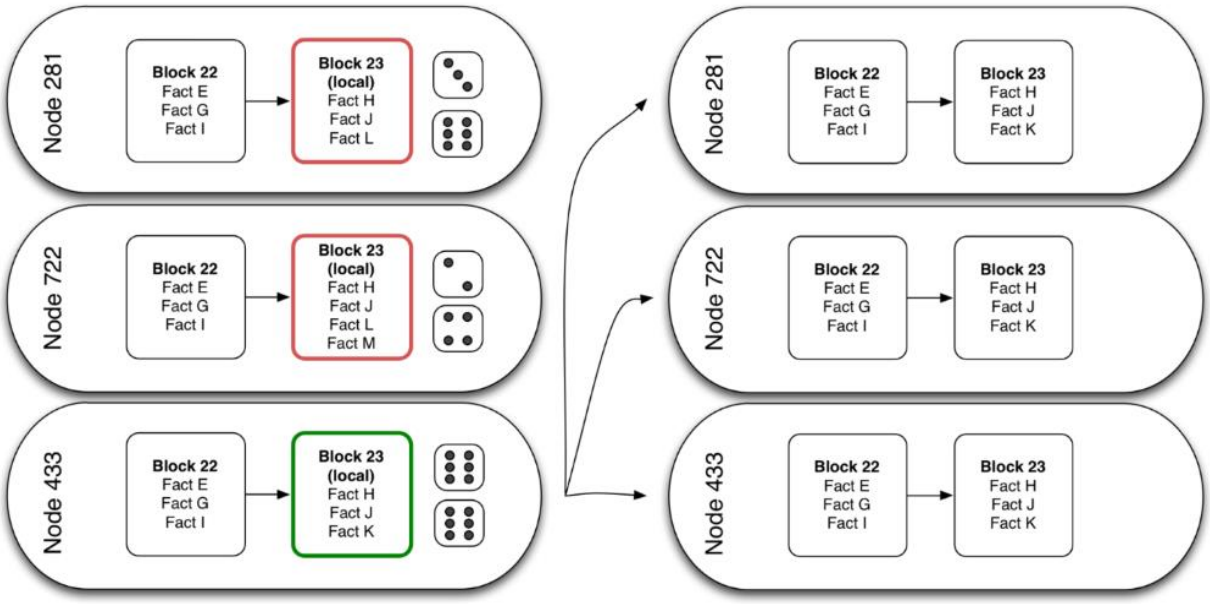


Рисунок 9. Змагання блоків з викиданням гральних кісток

Але насправді вузли не просто кидають пару гральних кісток. Завдання, яке вирішують вузли в блокчейне передбачає кидок величезної кількості гральних кісток. За задумом, виявлення випадкового ключа для перевірки блоку малоймовірно. Це запобігає шахрайству і робить мережу безпечною (до тих пір, поки зловмисник не має контролю більш ніж над половиною вузлів в мережі). Як наслідок, нові блоки будуть публікуватися в ланцюг через фіксований інтервал часу. У Bitcoin блоки публікуються, в середньому, кожні 10 хвилин.

Контракти

До сих пір, ми в основному говорили про блокчейне як про сховище фактів, але він також може виконувати програми. Деякі блокчейни дозволяють кожному факту містити міні програму. Такі програми реплицируются разом з фактом, і кожен вузол виконує їх, отримуючи факт. У Bitcoin це використовується для здійснення транзакцій з умовами, наприклад: А отримає 100 BTC від В тільки якщо сьогодні 29 лютого.

Інші блокчейни дозволяють більш складні контракти. Наприклад, в Ethereum кожен контракт несе в собі міні-базу даних і надає методи для зміни її даних. Оскільки контракти репліцируються по всіх вузлах, то і їх бази даних теж. Кожен раз, коли користувач викликає метод з контракту і, відповідно, змінює дані, ця команда реплицирується і повторюється всією мережею. Це дозволяє створити розподілений консенсус для виконання обіцянок.

Ця ідея сполучення блокчейна з реальним світом за допомогою заздалегідь запрограмованих умов і їх передачі всіх вузлів називається розумний контракт. Контракт - це обіцянка, яку сторони підписують, щоб закріпити його юридично. Розумний контракт - це те ж саме, тільки закріплення відбувається «технічно», а не «юридично». Завдяки цьому відпадає необхідність в нотаріуса або будь-якому іншому повноважного особі, визнаному обома сторонами.

Наприклад, ви хочете здати ваш будинок на тиждень за 1000 грн с 50% -ої передоплатою. Ви і орендар підписуєте контракт, найімовірніше, написаний юристом. Вам також потрібен банк для отримання платежу. На початку тижня ви вимагаєте проплату в 500 грн; орендар перераховує гроші . В кінці тижня він відмовляється сплатити решту 50%. Ви також дізнаєтеся, що він зламав вікно, і чек з проплатою веде на порожній рахунок. Зараз вам знадобиться адвокат, щоб передати ваш договір на оренду в суд.

Розумні контракти в блокчейне дозволяють вам обійтися без банку, юриста, адвоката і суду. Просто напишіть програму, яка визначає, скільки грошей має бути передано в разі певних умов:

- Два тижні перед орендою: передача 500 грн. від орендаря до власника
- Скасування власником оренди: передача 500 грн. від власника до орендаря;
- Кінець періоду оренди: передача 500 грн. від орендаря до власника
- Доказ механічних пошкоджень після періоду оренди: передача 5000 грн. від орендаря до власника

Додаємо розумний контракт в блокчейн. На час вказаний в контракті відбудеться передача грошей і, якщо власник зможе представити докази механічних пошкоджень, він автоматично отримає 5000 грн. (І немає ніякої потреби в передоплаті).

Ймовірно, ви ставите питанням, як отримати докази механічних пошкоджень. Тут в справу вступає «інтернет речей» (IoT). Для взаємодії з реальним світом блокчейну необхідні датчики , від яких він отримує інформацію про стан речей та їх пошкодження.

Такі додатки, що спираються на розумні контракти, називаються децентралізованими додатками або DApps.

Розумні контракти легко розширюються на розумну власність і багато інших розумних речей. Треба взяти до уваги що термін «розумні» означає «безпосередників» або «виконується автоматично». Блокчейн - це новий спосіб ведення бізнесу без посередників.

Сьогодні блокчейн є експериментальною технологією, яка розвивається і тому ще остаточно не підтверджено її ефективність. За думкою експертів і вчених ця прогресивна технологія має своє майбутнє.

4.2. Контрольні запитання

1. Що таке блокчейн?
2. Що таке транзакція?
3. Що таке геш-функція?
4. Чому можна довіряти фактам в блокчейн?
5. З чого складаються блоки в блокчейн?
6. Як створюються блоки в блокчейн?
7. Як узгоджуються факти в мережі блокчейн?
8. Як в блокчейн використовуються транзакції з умовами ?
9. Як в блокчейн використовуються транзакції з контрактами?
10. Що таке розумні контракти
11. Де доцільно використовувати технологію блокчейн

ПОКАЖЧИК

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ТА ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Томас Коннолли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика, Вільямс, 2010, -1439с.
2. Шаров С.В. , Осадчий В.В. Бази даних та інформаційні системи. Навчальний посібник / С.В. Шаров, В.В. Осадчий. – Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2014. – 352 с.
3. Виктор Пасько. Access 2007 . – К: Изд. группа BHV, 1999. - 384 с.
4. Демиденко М. А. Сучасні методи управління проектами інформатизації: навч. посіб. / М.А. Демиденко; НТУ «Дніпровська політехніка». – Д. : 2020. – 163 с. – Режим доступу до ресурсу:http://ir.nmu.org.ua/handle/123456789/154719
5. Демиденко М.А., Кочура Є.В. Вільне програмне забезпечення, офісний пакет Openoffice:Навч. посібник. – Дніпро: Національний гірничий університет, 2006.– 66 с.
6. Демиденко М. А. Управління проектами інформатизації / Михайло Андрійович Демиденко. – Дніпропетровськ: Національний гірничий університет, 2014. – 108 с.
7. M.A. Demydenko. Management of information projects. Scrum technology. Guidelines for the study of the subject "Informatics Project Management" – Dnipro University of Technology, 2019 - 21p. – Режим доступу до ресурсу: http://ir.nmu.org.ua/handle/123456789/154618
8. Демиденко М. А. Системи підтримки прийняття рішень / Михайло Андрійович Демиденко. – Дніпропетровськ: Національний гірничий університет, 2016. – 106 с. – (НГУ).
9. Demidenko M. Method of selection of ERP systems using multi-criterial optimization models / M.A. Demidenko. // Naukovyi Visnyk NHU. – 2018. – №5. – С. 132–137. DOI: 10/29202/nvngu/2018-5/21
10. Фридман С.Я., Бирицкая А.А. Microsoft Access. Часть 1. База данных. Таблицы. Запросы: Учеб. пособие. – Днепропетровск: Национальная горная академия Украины, 2002. – 127с.
11. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. – СПб: Корона принт, 2000. – 416 с.

Навчальне видання

Демиденко Михайло Андрійович,

ВВЕДЕННЯ В СУЧАСНІ БАЗИ ДАНИХ

Навчальний посібник

У редакції автора

Підписано до друку 28.12.2019. Формат 30 × 42/4.
Папір офсетний. Ризографія. Авт. Арк. 1,5.
Обл.–вид. арк. 13,7. Тираж 100 прим. Зам._№ 96/12.

Підготовлено до друку та видруковано
у Національному технічному університеті «Дніпровська політехніка
Свідоцтво про внесення до Державного реєстру ДК No 1842.
49600, м. Дніпро, просп. Д. Яворницького, 19.