УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра спеціалізованих комп'ютерних систем

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять
з дисципліни

«ОРГАНІЗАЦІЯ ТА СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ»

Харків – 2021

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри спеціалізованих комп'ютерних систем 22 лютого2021 р., протокол № 9.

Описано методології, методи та інструментальні засоби розроблення моделей баз даних.

Методичні вказівки призначені для здобувачів вищої освіти напряму 123 — «Комп'ютерна інженерія», що вивчають курс «Організація та системи керування базами даних».

Укладачі:

доц. С. І. Доценко,
викл. О. Л. Некрасов

Рецензент

доц. Н. А. Корольова

ЗМІСТ

Вступ……... 4
Практичне заняття 1. Методології і технології проектування інформаційних систем…….. 4
1.1 Методологія RAD…… 6
1.2 Методологія IDEF0……. 9
1.3 Діаграми потоків даних DFD……. 16
1.4 Моделювання даних ERD. Case-метод Баркера…… 22
Практичне заняття 2. Нотації, що використовуються при побудові діаграм «сутність-зв'язок»…… 29
2.1 Нотація Чена…… 30
2.2 Нотація Мартіна……... 31
2.3 Нотація IDEF1X……... 32
2.4 Нотація Баркера……... 35
Практичне заняття 3. Ієрархічна і мережна моделі даних. Структура даних…… 36
3.1 Ієрархічна модель даних……. 36
3.2 Операції над даними, визначені в ієрархічній моделі…. 39
3.3 Мережна модель даних. Структура даних…… 47
Список літератури……. 43

ВСТУП

У методичних вказівках розглядається рішення типових завдань розроблення та застосування баз даних, що виконуються на практичних заняттях та в години самостійної роботи з дисципліни «Організація та системи управління базами даних».

Для кожного практичного завдання додаються варіанти питань, які мають бути опрацьовані здобувачами вищої освіти в години самостійної роботи.

У кожному практичному завданні наводяться додаткові джерела літератури, які можуть бути опрацьовані здобувачами вищої освіти в години самостійної роботи.

Завдання розроблення та застосування баз даних, приведені в методичних вказівках, також можуть бути використані при проведенні поточного контролю знань здобувачів вищої освіти, модульного контролю та на екзамені/заліку.

ПРАКТИЧНЕ ЗАНЯТТЯ 1. Методології і технології проектування інформаційних систем

Завдання: ознайомитися зі змістом методологій проектування, які застосовуються для розроблення інформаційних систем, а саме:

1.1 методологія RAD;

1.2 методологія IDEF0;

1.3 діаграми потоків даних DFD;

1.3.1 нотація Йордона-Де Марко;

1.3.2 методологія DFD в нотації Гейна-Сарсона;

1.3.3 порівняльний аналіз методологій функціонального моделювання.

Вступ. Методології, технології та інструментальні засоби проектування (CASE-засоби) складають основу проекту будь-якої інформаційної системи (рисунок 1).

За формою реалізації методології проектування поділяються на дві групи (рисунок 1.2).

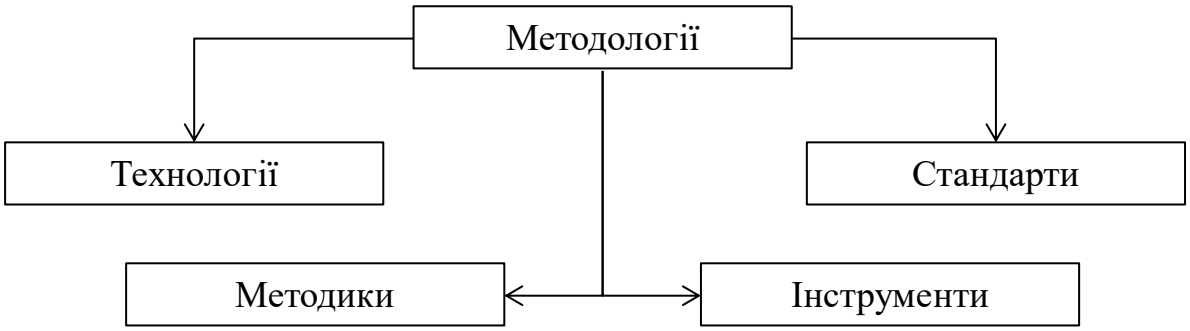


Рисунок 1.1 – Склад CASE-засобів

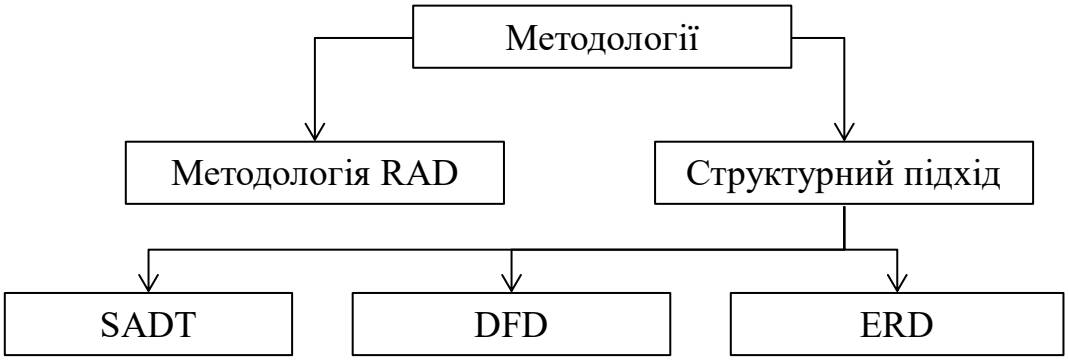


Рисунок 1.2 – Класифікація форм реалізації методологій проектування

До складу технологій проектування входять такі складові:

1 Покрокові процедури, які визначають послідовність технологічних операцій проектування (рисунок 1.3).

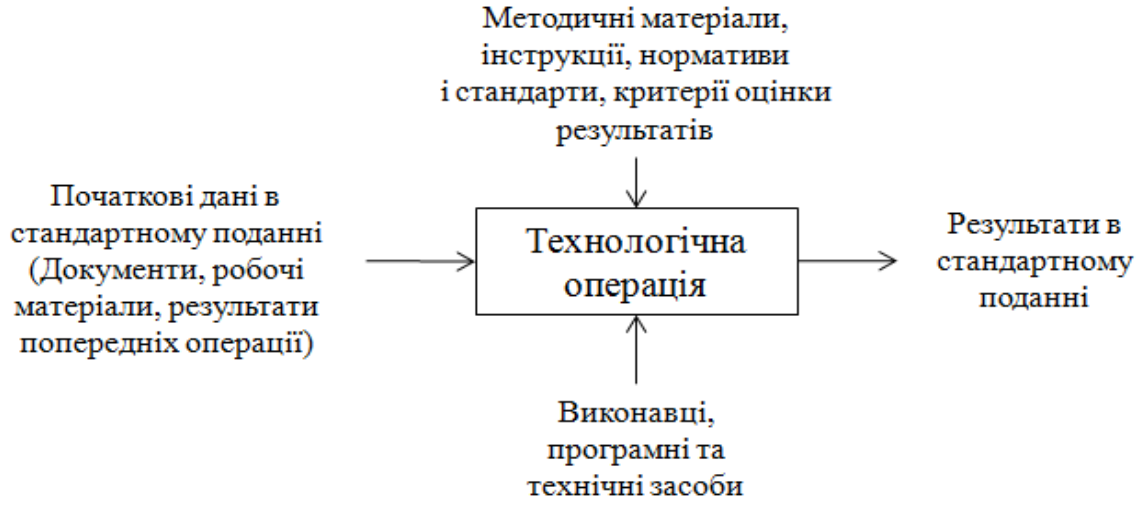


Рисунок 1.3 – Подання технологічної операції проектування

2 Критерії та правила, які застосовуються для оцінювання результатів виконання технологічних операцій.

3 Нотації (графічні і текстові засоби), які використовуються для опису проектованої системи.

1.1 Методологія RAD

Основні особливості методології RAD. Методологія створення інформаційних систем, заснована на використанні засобів швидкого розроблення додатків, останнім часом широко поширилася і отримала назву методології швидкого розроблення додатків (Rapid Application Development, RAD) (рисунок 1.4). Дана методологія охоплює всі етапи життєвого циклу сучасних інформаційних систем.

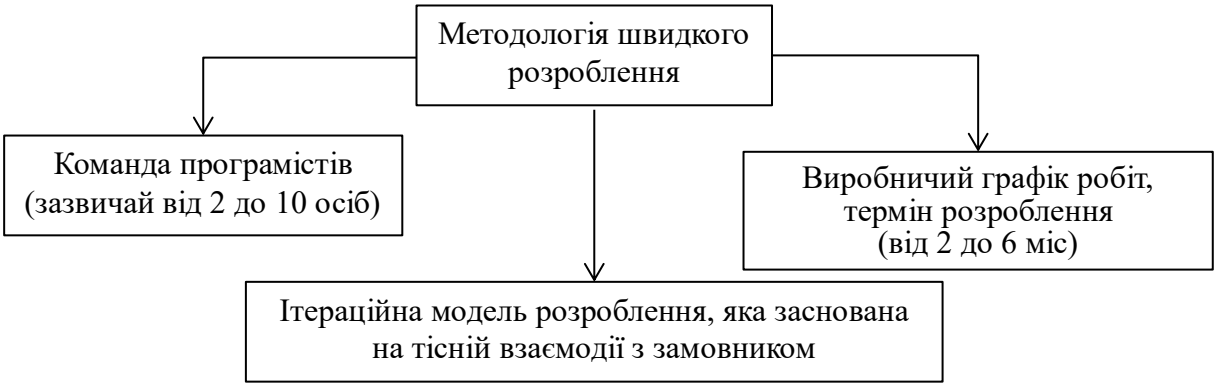


Рисунок 1.4 – Елементи методології швидкого розроблення додатків

Методологія RAD – це комплекс спеціальних інструментальних засобів, що дозволяють оперувати з певним набором графічних об'єктів, функціонально відображують окремі інформаційні компоненти додатків.

Основні принципи методології RAD можна звести до поданих на рисунку 1.5.

Засоби RAD дали можливість реалізовувати абсолютно іншу порівняно з традиційною технологію створення додатків.

Інформаційні об'єкти формуються як якісь *діючі моделі* (прототипи), чиє функціонування узгоджується з користувачем, а потім розробник може переходити безпосередньо до формування

закінчених додатків, не випускаючи з уваги загальної картини проектованої системи.

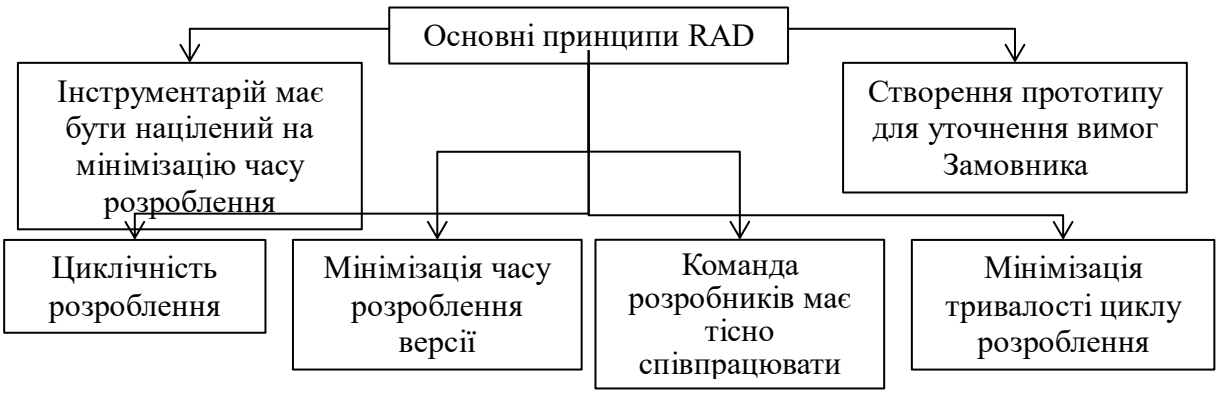


Рисунок 1.5 – Основні принципи методології RAD

Можливість використання подібного підходу значною мірою є результатом застосування принципів об'єктно-орієнтованого проектування.

Проблеми моделювання!

Застосування об'єктно-орієнтованих методів дозволяє подолати одну з головних труднощів, що виникають при розробленні складних систем, – колосальний розрив між реальним світом (предметною областю описуваної проблеми) та імітуючим середовищем.

Переваги об'єктно-орієнтованих методів моделювання!

Використання об'єктно-орієнтованих методів дозволяє створити опис (модель) предметної області у вигляді сукупності об'єктів – сутностей, які об'єднують дані і методи обробки цих даних (процедури). Кожен об'єкт має свою власну поведінку і моделює деякий об'єкт реального світу. *З цієї точки зору об'єкт є цілком відчутною річчю, яка демонструє певну поведінку*.

Цілісність об‘єктів!

Об'єкти характеризуються цілісністю, яка не може бути порушеною. Таким чином, властивості, що характеризують об'єкт і його поведінку, залишаються незмінними. *Об'єкт може тільки змінювати стан, управлятися або ставати в певне відношення до інших об'єктів*.

Візуалізація об‘єктів!!!

Широку популярність об'єктно-орієнтоване програмування отримало з появою візуальних засобів проектування, коли було забезпечено поєднання (інкапсуляцію) даних з процедурами, що описують поведінку реальних об'єктів, в об'єкти програм, які можуть бути відображені певним чином у графічному користувацькому середовищі. Це дозволило розпочати створення програмних систем, максимально схожих на реальні, і добиватися найвищого рівня абстракції. У свою чергу об'єктно-орієнтоване програмування дозволяє створювати більш надійні коди, оскільки в об'єктів програм існує точно визначений і жорстко контрольований інтерфейс.

Візуальні засоби розроблення оперують у першу чергу зі стандартними інтерфейсними об'єктами – вікнами, списками, текстами, які легко можна пов'язати з даними з бази даних і відобразити на екрані монітора. Інша група об'єктів являє собою стандартні елементи управління – кнопки, перемикачі, прапорці, меню і т. п., за допомогою яких здійснюється управління відображеними даними. Всі ці об'єкти можуть бути стандартним чином описані засобами мови, а самі описи збережені для подальшого повторного використання.

В даний час існує досить багато різних візуальних засобів розроблення додатків. Але всі вони можуть бути розділені на дві групи – універсальні і спеціалізовані (рисунок 1.6).

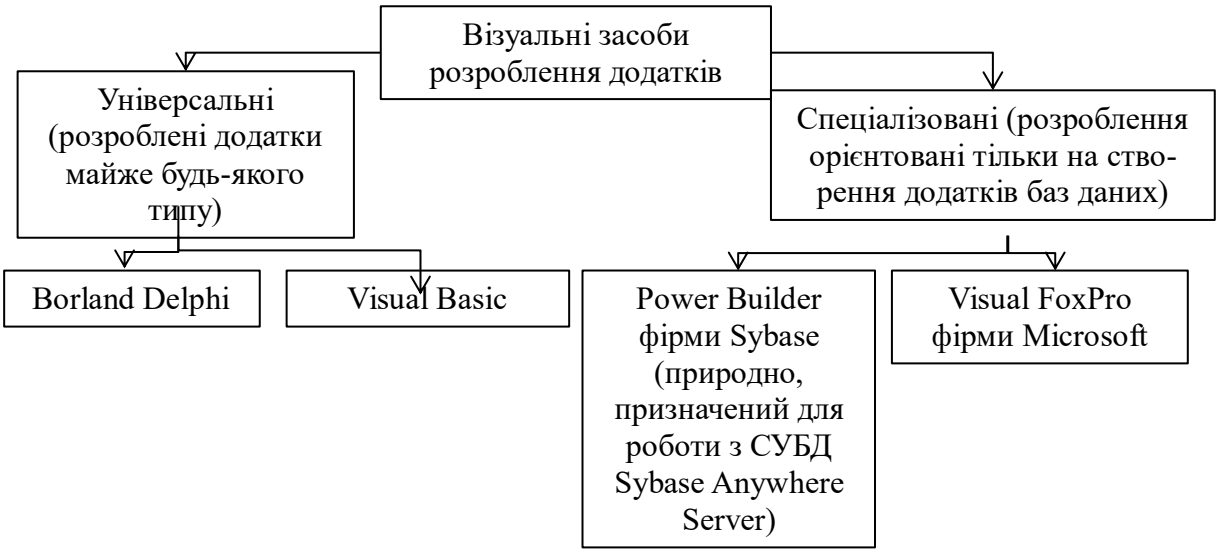


Рисунок 1.6 – Класифікація візуальних засобів розроблення додатків

Перелік джерел, рекомендованих для поглибленого вивчення маткріалу

1 http://en.wikipedia.org/wiki/Rapid_application_development

2 http://searchsoftwarequality.techtarget.com/definition/rapid-application-development

3 http://www.informicus.ru/default.aspx?SECTION=6&id=93

4 http://www.sqa.org.uk/e-learning/SDM01CD/page_09.htm

5 http://wysterdesir.com/2008/09/28/using-rapid-application-development-for-your-software-project/

1.2 Методологія IDEF0

Методологія IDEF0 запроваджена Федеральним стандартом США: INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0). Draft Federal Information Processing Standards Publication 183, 1993 December 21.

Також вона запроваджена як міжнародний стандарт IS: ISO/IEC/IEEE 31320-1:2012(en) Information technology — Modeling Languages.

Зі змістом цього стандарту можна ознайомитися за посиланням https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:31320:-1:ed-1:v1:en.

Матеріал стосовно змісту методології IDEF0 наведено в електронних ресурсах:

– «Теоретичні відомості про методологію idef0» за посиланням https://studfile.net/preview/5706328/page:4/

– «Лекція з методології IDEF0» https://uadoc.zavantag.com/text/1001/index-1.html.

Подані у цьому розділі навчальні матеріали викладено за змістом вказаного стандарту.

Загальна методологія IDEF складається з трьох приватних методологій моделювання, заснованих на графічному поданні систем:

- **IDEF0** – використовується для створення функціональної моделі, яка відображує структуру і функції системи, а також потоки інформації і матеріальних об'єктів, що зв'язують ці функції;

- **IDEF1** – застосовується для побудови інформаційної моделі, яка відображує структуру і зміст інформаційних потоків, необхідних для підтримки функцій системи;

- **IDEF2** – дозволяє побудувати динамічну модель мінливих у часі поведінки функцій, інформації та ресурсів системи.

Основу підходу і як наслідок методології IDEF0 становить **графічна мова** опису (моделювання) систем, що має такі властивості:

• **Графічна мова** – повний і виразний засіб, здатний наочно представляти широкий спектр ділових, виробничих та інших процесів і операцій підприємства на будь-якому рівні деталізації.

• **Мова забезпечує** точний і лаконічний опис модельованих об'єктів, зручність використання і інтерпретації цього опису.

• **Мова полегшує** взаємодію та взаєморозуміння системних аналітиків, розробників і персоналу досліджуваного об'єкта (фірми, підприємства), тобто служить засобом «інформаційного спілкування» великої кількості фахівців і робочих груп, зайнятих в одному проекті, процесі обговорення, рецензування, критики і затвердження результатів.

• **Мова** пройшла багаторічну перевірку і продемонструвала працездатність як у проектах ВПС США, так і в інших проектах, що виконувалися державними і приватними промисловими компаніями.

• **Мова** є легкою і простою у вивченні та освоєнні.

• **Мова** може генеруватися поруч з інструментальними засобами машинної графіки; відомі комерційні програмні продукти підтримують розроблення та аналіз моделей – діаграм IDEF0, наприклад продукт Design / IDEF 3.7 (і більш пізні версії) фірми Meta Software Corporation (США).

1.2.1 Концепція IDEF0

Методологія IDEF0 заснована на таких концептуальних положеннях.

1.2.1.1 **Модель** – штучний об'єкт, що являє собою відображення (образ) системи і її компонентів. М моделює А, якщо М відповідає на питання щодо А. Тут М – модель, А – модельований об'єкт (оригінал). Система є сукупністю взаємопов'язаних і взаємодіючих частин, що виконують деяку корисну роботу. Частинами (елементами) системи можуть бути будь-які комбінації різноманітних сутностей, що включають людей, інформацію, програмне забезпечення, обладнання, вироби, сировину або енергію (енергоносії). Модель описує, що відбувається в системі, як нею керують, які сутності вона перетворює, які кошти використовує для виконання своїх функцій і що виробляє.

1.2.1.2 **Блочне моделювання** та його графічне подання. Основний концептуальний принцип методології IDEF – подання будь-чого вивчається у вигляді набору взаємодіючих і взаємопов'язаних блоків, які відображують процеси, операції, дії (визначення дивись нижче), що відбуваються в системі, яка вивчається. В IDEF0 все, що відбувається в системі та її елементах, прийнято називати функціями. Кожній функції ставиться у відповідність блок.

На IDEF0-діаграмі основний документ при аналізі і проектуванні систем – це блок, що являє собою прямокутник. Інтерфейси, за допомогою яких блок взаємодіє з іншими блоками або з зовнішнім по відношенню до модельованої системи середовищем, подаються стрілками, що входять у блок або виходять з нього. Вхідні стрілки показують, які умови мають бути одночасно виконані, щоб функція, що описується блоком, здійснилася.

1.2.1.3 **Лаконічність і точність**. Документація, що описує систему, має бути точною і лаконічною.

1.2.1.4 **Передача інформації**. Засоби IDEF0 полегшують передачу інформації від одного учасника розроблення моделі (окремого розробника або робочої групи) до іншого. До таких засобів належать:

• діаграми, засновані на простій графіці блоків і стрілок, легко читаються і розуміються;

• мітки на природній мові для опису блоків і стрілок, а також глосарій і супровідний текст для уточнення сенсу елементів діаграми;

• послідовна декомпозиція діаграм, що будується за ієрархічним принципом, при якому на верхньому рівні відображуються основні функції, а потім відбувається їх деталізація та уточнення;

• деревоподібні схеми ієрархії діаграм і блоків, що забезпечують видимість моделі в цілому і деталей, які входять до неї.

1.2.1.5 **Строгість і формалізм**. Розроблення моделей IDEF0 вимагає дотримання ряду строгих формальних правил, що забезпечують переваги методології щодо однозначності, точності і цілісності складних багаторівневих моделей. Ці правила описуються нижче. Тут висвітлюється тільки основне з них: ***всі стадії та етапи розроблення і коригування моделі мають строго, формально документуватися з тим, щоб при її експлуатації не виникало питань, пов'язаних з неповнотою або некоректністю документації***.

1.2.1.6 **Ітеративне моделювання**. Розроблення моделі в IDEF0 – це покрокова, ітеративна процедура. На кожному кроці ітерації розробник пропонує варіант моделі, який піддають обговоренню, рецензуванню і подальшому редагуванню, після чого цикл повторюється. Така організація роботи сприяє оптимальному використанню знань системного аналітика, який володіє методологією і технікою IDEF0, і знань фахівців – експертів у предметній області, до якої належить об'єкт моделювання.

1.2.1.7 **Відокремлення «організації» від «функцій»**. При розробленні моделей слід уникати початкової «прив'язки»

функцій досліджуваної системи до існуючої організаційної структури об'єкта, що моделюється (підприємства, фірми). Це допомагає уникнути суб'єктивної точки зору, нав'язаної організацією і її керівництвом. Організаційна структура повинна бути результатом використання (застосування) моделі. Порівняння результату з існуючою структурою дозволяє, по-перше, оцінити адекватність моделі, а по-друге, запропонувати рішення, спрямовані на вдосконалення цієї структури.

1.2.2 Основні визначення (поняття) методології та мови IDEF0

1.2.2.1 **Блок**: прямокутник, що містить ім'я і номер і використовується для опису функції (рисунок 1.7).

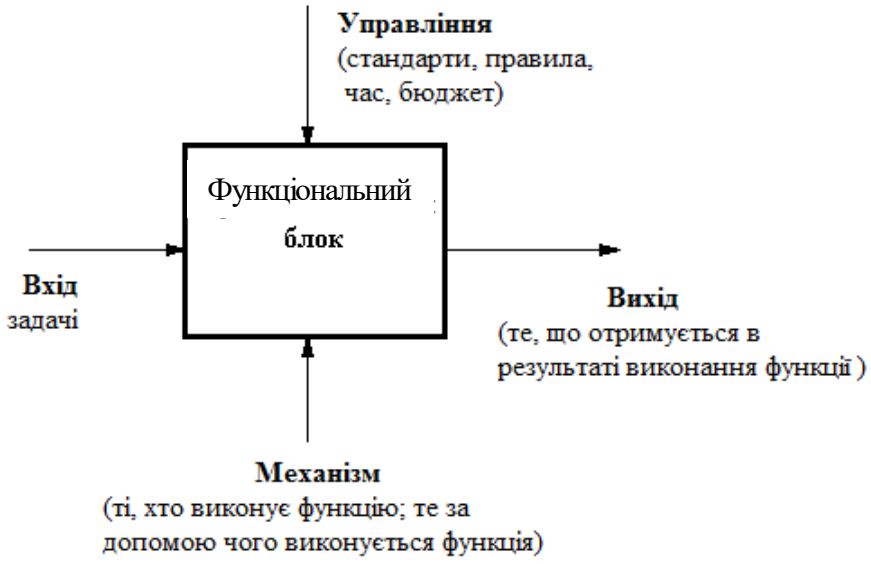


Рисунок 1.7 – Схема функціонального блока

1.2.2.2 **Розгалуження**: поділ стрілки на дві або більшу кількість сегментів. Може означати «розв'язання пучка» (рисунок 1.8).

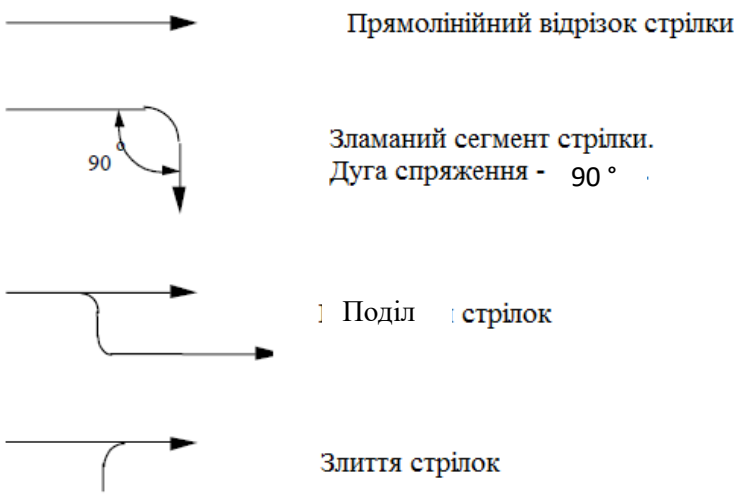


Рисунок 1.8 – Форми стрілок

1.2.2.3 **Внутрішня стрілка**: вхідні, керуюча або вихідна стрілки, кінці якої зв'язують джерело і споживача, є блоками однієї діаграми. Відрізняється від граничної стрілки.

1.2.2.4 **Вхідна стрілка**: клас стрілок, які відображують вхід IDEF0-блока, тобто дані або матеріальні об'єкти, що перетворюються функцією у вихід. Вхідні стрілки зв'язуються з лівою стороною блока IDEF0.

1.2.2.5 **Вихідна стрілка**: клас стрілок, які відображують вихід IDEF0-блока, тобто дані або матеріальні об'єкти, вироблені функцією. Вихідні стрілки зв'язуються з правою стороною блока IDEF0.

1.2.2.6 **Словник**: список визначень для ключових слів, фраз і аб-бревіатур, пов'язаних з вузлами, блоками, стрілками або з моделлю IDEF0 в цілому.

1.2.2.7 **Гранична стрілка**: стрілка, один з кінців якої зв'язаний з джерелом або споживачем, а інший не приєднаний до жодного блока на діаграмі. Відображує зв'язок діаграми з іншими блоками системи і відрізняється від внутрішньої стрілки.

1.2.2.8 **Декомпозиція**: поділ функції, що моделюється, на функції – компоненти.

1.2.2.9 **Дерево вузлів**: уявлення відносин між батьківськими і дочірніми вузлами моделі IDEF0 у формі деревоподібного графа. Має те саме значення і зміст, що і перелік вузлів.

1.2.2.10 **Діаграма А-0**: спеціальний вид (контекстної) діаграми IDEF0, що складається з одного блока, який описує функцію верхнього рівня, її входи, виходи, управління і механізми разом з формулами-острівцями мети моделі і точки зору, з якої будується модель.

1.2.2.11 **Діаграма**: частина моделі, яка описує декомпозицію блока.

1.2.2.12 **Діаграма-ілюстрація** (FEO): графічний опис, що використовується для повідомлення специфічних фактів про діаграми IDEF0. При побудові діаграм FEO можна не дотримуватися правила IDEF0.

1.2.2.13 **Дочірній блок**: блок на дочірній (породженій) діаграмі

1.2.2.14 **Дочірня діаграма**: діаграма, що деталізує (породжує) батьківський блок.

1.2.2.15 **Ім'я блока**: дієслово або дієслівний оборот, поміщений всередині блока, описує функцію, що моделюється.

1.2.2.16 **Інтерфейс**: розділяє межу, через яку проходять дані або матеріальні об'єкти; з'єднання між двома або більшою кількістю компонентів моделі, яка передає дані або матеріальні об'єкти від одного компонента до іншого.

1.2.3 Правила інтерпретації моделі

Розрізняють:

– функціональний блок (функція) перетворює вхідні об'єкти у вихідні;

– управління визначає, коли і як це перетворення може або повинно відбутися;

– виконавець здійснює це перетворення.

З дугами зв'язуються мітки на природній мові, що описують дані, які вони представляють. Дуги показують, як функції системи пов'язані між собою, як вони обмінюються даними і здійснюють управління один одним. Виходи однієї функції можуть бути входами, управлінням або виконавцями іншої.

Дуги можуть розгалужуватися і з'єднуватися. Розгалуження означає множинність (ідентичні копії одного об'єкта) або

розщеплення (різні частини одного об'єкта). З'єднання означає об'єднання або злиття об'єктів.

Кожен блок IDEF0-діаграми може бути поданий декількома блоками, з'єднаними інтерфейсними дугами, на діаграмі наступного рівня. Ці блоки представляють підфункції (підмодулі) вихідної функції. Кожен з підмодулів може бути декомпозований аналогічним чином. Кількість рівнів не обмежується, проте рекомендується на одній діаграмі використовувати не менше 3 і не більше 6 блоків.

На рисунку 1.9 зображена IDEF0-модель діяльності підприємства згідно з роботою [1].

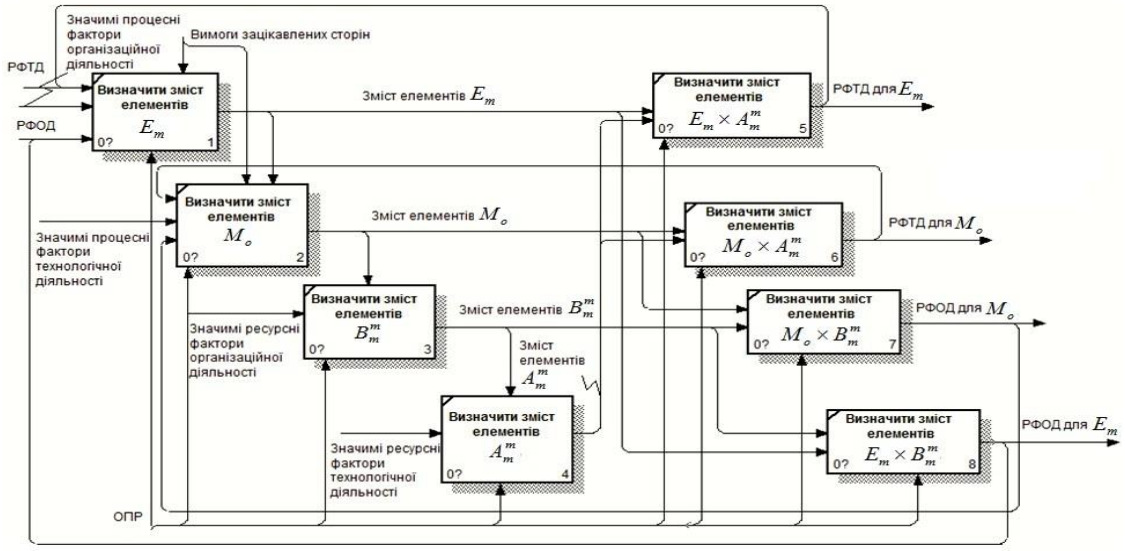


Рисунок 1.9 – IDEF0-модель діяльності підприємства

1.3 Діаграми потоків даних DFD

Діаграма потоків даних (англ. *Data Flow Diagram*) — модель проектування, графічне подання «потоків» даних в *інформаційній системі*. Діаграма потоків даних також може використовуватись для візуалізації процесів обробки даних структурного проектування [2].

Для розробника вважається звичним спочатку креслити діаграму потоків даних рівня контексту, завдяки чому буде показано взаємодію системи із зовнішніми модулями. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів і потоків даних з метою показати розлого розроблювану систему.

Діаграми потоків даних містять чотири типи графічних елементів:

- процеси – являють собою трансформацію даних в рамках описуваної системи;
- сховища даних (репозиторії);
- зовнішні по відношенню до системи сутності;
- потоки даних між трьох попередніх типів.

1.3.1 Нотація Йордона–Де Марко

Функції, сховища і зовнішні сутності на DFD-діаграмі зв'язуються дугами, що являють собою потоки даних. Дуги можуть розгалужуватися або зливатися, що означає, відповідно, поділ потоку даних на частини або злиття об'єктів.

Елементи, які позначають сутності на DFD-діаграмах наведені в таблиці 1.1.

Такий тип позначень елементів DFD-діаграми отримав назву «Нотація Йордона–Де Марко», за іменами фахівців, які його розробили.

Таблиця 1.1 – Склад елементів діаграми потоків даних (DFD – Data Flow Diagramm) [3]

Елемент	Опис	Позначення
Функція	Дія, що виконується системою, яка моделюється	Ім'я функції
Потік даних	Об'єкт, над яким виконується дія. Може бути інформаційним (логічним) або керуючим. (Керуючі потоки позначаються пунктирною лінією зі стрілкою).	Ім'я об'єкту
Сховище	Структура для зберігання інформаційних об'єктів	Ім'я об'єкту
Зовнішня сутність	Зовнішній по відношенню до системи об'єкт, обмінюється з нею потоками даних	Ім'я зовнішнього обєкту

При інтерпретації DFD-діаграми використовуються правила, яких дотримуються:

– функції перетворюють вхідні потоки даних у вихідні;

– сховища даних не змінюють потоки даних, а служать тільки для зберігання даних, що надходять з об'єктів;

– перетворення потоків даних у зовнішніх сутностях ігнорується.

Крім цього, для кожного інформаційного потоку і сховища визначаються пов'язані з ними елементи даних. Кожному елементу даних привласнюється ім'я, також для нього може бути зазначений тип даних і формат. Саме ця інформація є вихідною на наступному етапі проектування – побудові моделі «*сутність-зв'язок*». При цьому, як правило, інформаційні сховища перетворюються по суті, проектувальнику залишається тільки вирішити питання з використанням елементів даних, які зв'язані зі сховищами.

Побудуємо DFD-схему бізнес-процесу «Оформлення та видача трудової книжки працівникові при звільненні» (рисунок 1.10) [4].

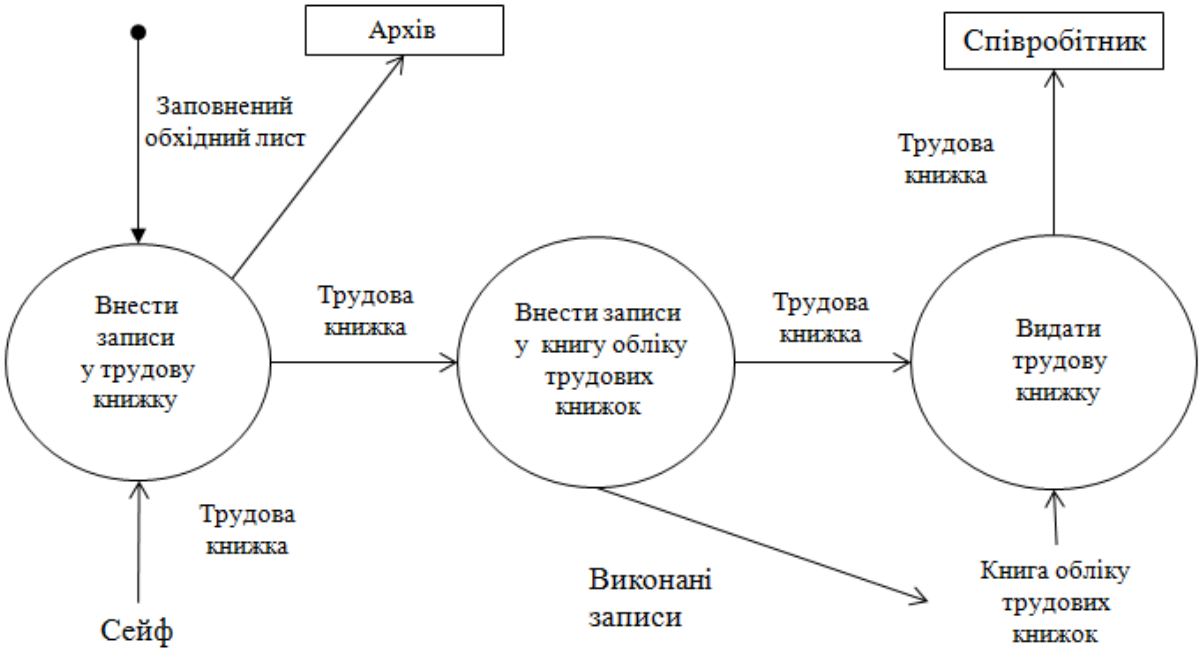


Рисунок 1.10 – DFD-схема бізнес-процесу «Оформлення та видача трудової книжки працівникові при звільненні» в нотації Йордона-Де Марко

Ця діаграма є найвищим верхнім рівенем функціональної моделі. Природно, це дуже грубий опис предметної області. Уточнення моделі проводиться шляхом деталізації необхідних функцій на DFD-діаграмі наступного рівня. Так, ми можемо розбити функцію «Визначення потреб і забезпечення матеріалами» на підфункції «Визначення потреб», «Пошук постачальників», «Висновок і аналіз договорів на поставку», «Контроль платежів», «Контроль поставок», пов'язані власними потоками даних, які будуть подані на окремій діаграмі. Деталізація моделі має здійснюватися до тих пір, поки вона не буде містити всю інформацію, необхідну для побудови інформаційної системи [4].

1.3.2 Методологія DFD в нотації Гейна-Сарсона [3]

Гейн і Сарсон запропонували класичну DFD-схему дещо ускладнити, а саме ввести додатковий об'єкт, за допомогою якого показуються місця бізнес-процесу, в яких зберігається інформація або матеріальні ресурси. Прикладами таких місць є архів, в якому зберігаються документи, база даних, в якій зберігається інформація, або склад, на якому зберігаються матеріальні ресурси. Даний об'єкт називається сховищем даних. На DFD-схемах в нотаціях Гейна-Сарсона і Йордона-Де Марко також використовуються об'єкти, за допомогою яких показують зовнішніх суб'єктів, з якими бізнес-процес взаємодіє. Дані об'єкти називають зовнішніми сутностями. На рисунку 1.11 наведено приклад DFD-схеми бізнес-процесу «Оформлення та видача трудової книжки працівникові при звільненні», розробленої в нотації Гейна-Сарсона [3].

На цій схемі сховищем даних є сейф, в якому зберігаються трудові книжки, і архів, в який поміщено заповнений обхідний лист. Зовнішньою суттю виступає співробітник, який звільняється та отримує вихід розглянутого бізнес-процесу – трудову книжку.

У таблиці 1.2 подані назви, позначення і зміст елементів, які використовуються при побудові DFD-схеми бізнес-процесу в нотаціях Гейна-Сарсона і Йордона-Де Марко [3].

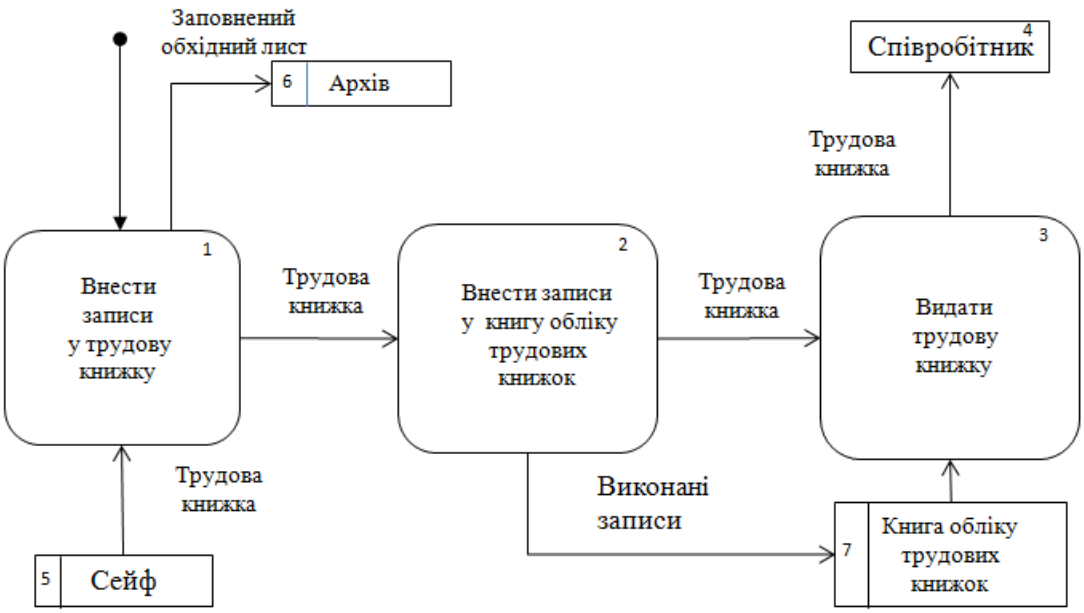


Рисунок 1.11 – DFD-схема бізнес-процесу «Оформлення та видача трудової книжки працівникові при звільненні» в нотації Гейна-Сарсона

Таблиця 1.2 – Елементи методології DFD в нотаціях Гейна-Сарсона і Йордона-Де Марко

Елемент	Опис	Нотація Йордана-де Марко	Нотація Гейна-Сарсона
Функція	Робота	Ім‘я функції номер	Ім‘я функції / Номер
Потік даних	Об‘єкт, над яким виконується робота. Може бути логічним чи управляючим	Ім‘я об‘єкту	Ім‘я об‘єкту / Поняття управляючого потоку відсутнє
Сховище даних	Структура для зберігання інформаційних об‘єктів	Ім‘я об‘єкту	Ім‘я об‘єкту
Зовнішня сутність	Зовнішній по відношенню до системи об‘єкт, який обмінюється з нею потоками	Ім‘я зовнішнього об‘єкту	Ім‘я зовнішнього об‘єкту

Крім нотацій Йордона-Де Марко та Гейна-Сарсона, для елементів DFD-діаграм можуть використовуватися й інші умовні позначення (OMT, SSADM і т. д.). Всі вони мають практично однакову базову функціональність і розрізняються лише в деталях.

Інструментальні засоби проектування (CASE-системи), як правило, підтримують кілька нотацій поданя DFD-діаграм. Однією з таких систем є **Power Designer** компанії Sybase, що включає такі модулі:

– **Process Analyst** – побудова діаграм потоків даних з використанням будь-якої з вищезазначених нотацій;

– **Data Analyst** – побудова діаграм «сутність-зв'язок» і перетворення її в реляційну модель;

– **Application Modeller** – засіб для генерації додатків.

Навчальну копію Power Designer, в якій заблокована функція збереження побудованих моделей, можна скачати з web-сервера компанії **Sybase**.

1.3.3 Порівняльний аналіз методологій функціонального моделювання [3]

Незважаючи на те, що методологія IDEF0 набула значного поширення в компаніях, на наш погляд, DFD набагато більше підходить для проектування інформаційних систем взагалі і баз даних зокрема. DFD дозволяє вже на стадії функціонального моделювання визначити базові вимоги до даних (цьому сприяє поділ потоків даних на матеріальні, інформаційні та керуючі). Взагалі інтеграція моделей DFD і ER (entity-relationship, «сутність-зв'язок») не викликає ускладнень. Наприклад, можна визначити список атрибутів сховищ даних, останні на стадії інформаційного моделювання однозначно відображуються у моделі «сутність-зв'язок».

У свою чергу, як вже зазначалося, IDEF0 більше підходить для вирішення завдань, пов'язаних з управлінським консультуванням (перепроектуванням ділових процесів, бізнес-реінжинірингом). Цьому сприяє також тісний зв'язок IDEF0 з методом функціонально-вартісного аналізу ABC (Activity Based Costing), що дозволяє визначити схему розрахунку вартості виконання тієї чи іншої ділової процедури. Однак існує ряд CASE-

систем, що пропонують методологію IDEF0 на етапі функціонального обстеження предметної області. У таких системах на наступний етап передається просто список всіх об'єктів IDEF0-моделі (входи, виходи, механізми, управління), які потім розглядаються на предмет включення в інформаційну модель.

Перелік джерел, рекомендованих для поглибленого вивчення матеріалу

1 Стаття «Data Flow Diagrams (DFD)» автора Vicki L. Sauter.

2 Стаття «Data Flow Diagrams» автора Tony Drewry

3 Стаття «The Semantics of Data Flow Diagrams» авторів P. D. Bruza та Th. P. van der Weide.

4 Стаття «How to draw data flow diagrams» авторство Smartdraw.

5 Розділ «Dataflow Diagrams» автора Ed Yourdon.

6 Розділ «Just Enough Structured Analysis – Chapter 9» автора Ed Yourdon.

1.4 Моделювання даних ERD. Case-метод Баркера

Матеріал даного пункту викладено на основі джерела [5].

Мета моделювання даних полягає в забезпеченні розробника ІС концептуальною схемою бази даних у формі однієї моделі або кількох локальних моделей, які відносно легко можуть бути відображені в будь-якій системі баз даних.

Найпоширенішим засобом моделювання даних є діаграми «сутність-зв'язок» (ERD). З їх допомогою визначаються важливі для предметної області об'єкти (сутності), їх властивості (атрибути) і відношення один з одним (зв'язок). ERD безпосередньо використовуються для проектування реляційних баз даних.

Нотація ERD була вперше введена П. Ченом (Chen) і набула подальшого розвитку в роботах Баркера. Метод Баркера буде викладатися на прикладі моделювання діяльності компанії з торгівлі автомобілями. Нижче наведено витяги з інтерв'ю, проведеного з персоналом компанії.

Перший крок моделювання – вилучення інформації з інтерв'ю і виділення сутностей.

Сутність (Entity) – реальний або уявний об'єкт, що має суттєве значення для розглянутої предметної області, інформація про нього підлягає зберіганню (рисунок 1.12) [5].

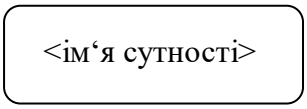


Рисунок 1.12 – Графічне зображення сутності

Кожна сутність повинна мати унікальний ідентифікатор. Кожен екземпляр сутності повинен однозначно ідентифікуватися і відрізнятися від усіх інших примірників даного типу сутності. Кожна сутність повинна мати деякі властивості:

– кожна сутність повинна мати унікальне ім'я, до одного і того самого імені має завжди застосовуватися одна й та сама інтерпретація. Одна і та ж інтерпретація не може застосовуватися до різних імен, якщо тільки вони не є псевдонімами;

– сутність має один або декілька атрибутів, які або належать сутності, або успадковуються через зв'язок;

– сутність має один або декілька атрибутів, які однозначно ідентифікують кожен екземпляр сутності;

– кожна сутність може мати будь-яку кількість зв'язків з іншими сутностями моделі.

Звертаючись до наведених вище уривків з інтерв'ю, видно, що сутності, які можуть бути ідентифіковані з головним менеджером, – це автомашини і продавці. Продавцю важливі автомашини та пов'язані з їх продажем дані. Для адміністратора важливі покупці, автомашини, продавці і контракти. Виходячи з цього, виділяються чотири сутності (автомашина, продавець, покупець, контракт), які зображено на діаграмі (рисунок 1.13) [5].

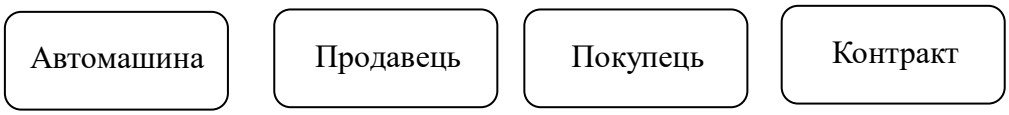


Рисунок 1.13 – Сутності адміністратора, важливі для нього

Наступним кроком моделювання є ідентифікація зв'язків.

Зв'язок (Relationship) – найменована асоціація між двома сутностями, значима для розглянутої предметної області. Зв'язок – це асоціація між сутностями, при якій, як правило, кожен екземпляр однієї сутності, названої батьківською сутністю, асоційований з довільною (в тому числі нульовою) кількістю примірників іншої сутності, названої сутністю-нащадком, а кожен екземпляр сутності-нащадка асоційований в точності з одним екземпляром сутності-батька. Таким чином, примірник сутності-нащадка може існувати тільки при існуванні сутності батька.

Зв'язку може даватися ім'я, яке виражається граматичним зворотом дієслова і поміщається біля лінії зв'язку. Ім'я кожного зв'язку між двома даними сутностями має бути унікальним, але імена зв'язків у моделі не зобов'язані бути унікальними. Ім'я зв'язку завжди формується з точки зору батьків, тому пропозиція може бути утворена з'єднанням імені сутності-батька, імені зв'язку, вираження ступеня та імені сутності-нащадка.

Наприклад, зв'язок продавця з контрактом може бути виражено таким чином:

– продавець може отримати винагороду за один або більше контрактів;

– контракт має бути ініційований рівно одним продавцем.

Ступінь зв'язку та обов'язковість графічно зображуються на рисунку 1.14 [5].

Рисунок 1.14 – Графічне зображення ступеня зв'язку та обов'язковості

Таким чином, два речення, що описують зв'язок продавця з контрактом, графічно будуть виражені як на рисунку 1.15.

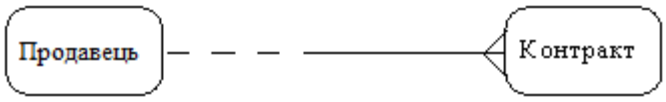


Рисунок 1.15 – Зв'язок продавця з контрактом

Описавши також зв'язки інших сутностей, отримаємо схему, зображену на рисунку 1.16 [5].

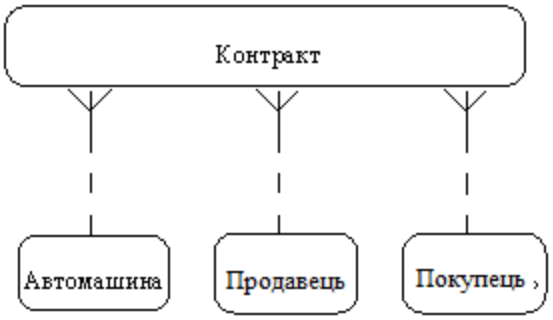


Рисунок 1.16 – Загальна схема відношення

Останнім кроком моделювання є ідентифікація атрибутів.

Атрибут – будь-яка характеристика сутності, значима для розглянутої предметної області і призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або вираження стану сутності.

Атрибут є типом характеристик або властивостей, асоційованих з безліччю реальних або абстрактних об'єктів (людей, місць, подій, станів, ідей, пар предметів і т. д.). Примірник атрибута – це певна характеристика окремого елемента множини. Примірник атрибута визначається типом характеристики і її значенням, що називається значенням атрибута. В ER-моделі атрибути асоціюються з конкретними сутностями. Таким чином, примірник сутності повинен мати єдине певне значення для асоційованого атрибута.

Атрибут може бути або обов'язковим, або необов'язковим (рисунок 1.17) [5]. Обов'язковість означає, що атрибут не може набувати невизначених значень (null values). Атрибут може бути або описовим (тобто звичайним дескриптором сутності), або входити до складу унікального ідентифікатора (первинного ключа).

Унікальний ідентифікатор – це атрибут або сукупність атрибутів і / або зв'язків, призначена для унікальної ідентифікації кожного примірника даного типу сутності. У разі повної ідентифікації кожен примірник даного типу сутності повністю ідентифікується своїми власними ключовими атрибутами, в іншому випадку в його ідентифікації беруть участь також атрибути іншої сутності-батька (рисунок 1.18) [5].

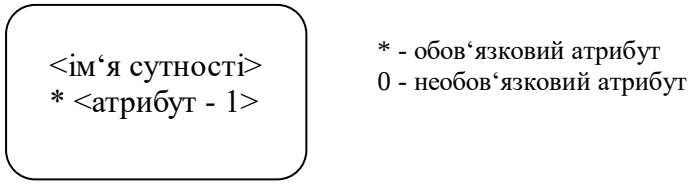


Рисунок 1.17 – Атрибут обов'язковий або необов'язковий

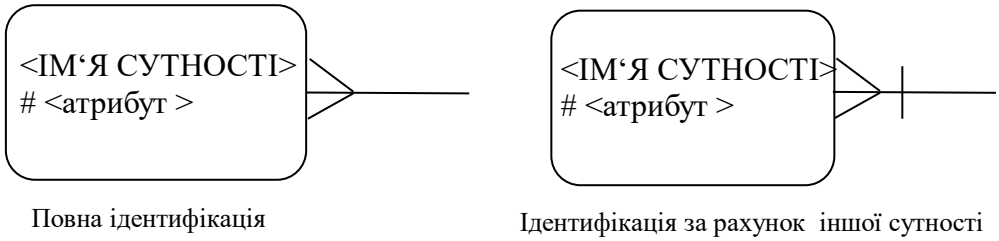


Рисунок 1.18 – Ідентифікація за участі атрибута іншої сутності-батька

Кожен атрибут ідентифікується унікальним ім'ям, вираженим граматичним зворотом іменника, що описує характеристику, яка представляється атрибутом. Атрибути зображуються у вигляді списку імен усередині блока асоційованої сутності, причому кожен атрибут займає окремий рядок. Атрибути, що визначають первинний ключ, розміщуються вгорі списку і виділяються знаком «#».

Кожна сутність повинна мати хоча б одним можливий ключ. Можливий ключ сутності – це один або декілька атрибутів, чиї значення однозначно визначають кожен екземпляр сутності. При існуванні декількох можливих ключів один з них позначається як первинний ключ, а решта – як альтернативні ключі.

З урахуванням наявної інформації доповнимо побудовану раніше діаграму (рисунок 1.19) [5].

Крім перерахованих основних конструкцій, модель даних може отримати ряд додаткових.

Підтипи і супертипи: одна сутність є узагальнюючим поняттям для групи подібних сутностей (рисунок 1.20) [5].

Взаємовиключаючі зв'язки: кожен екземпляр сутності бере участь тільки в одному зв'язку з групи взаємовиключаючих зв'язків (рисунок 1.21) [5].

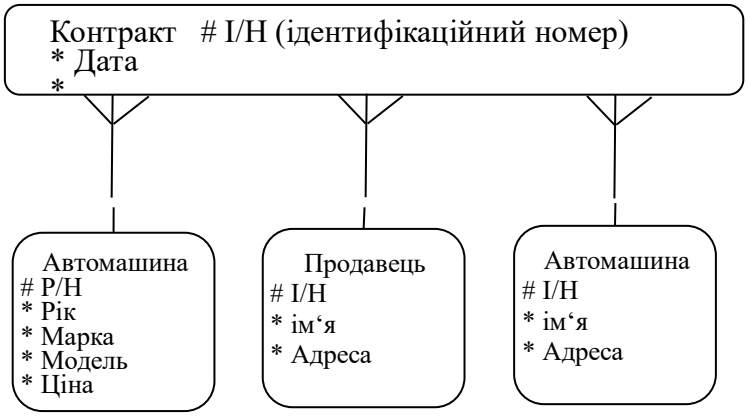


Рисунок 1.19 – Позначення первинного ключа

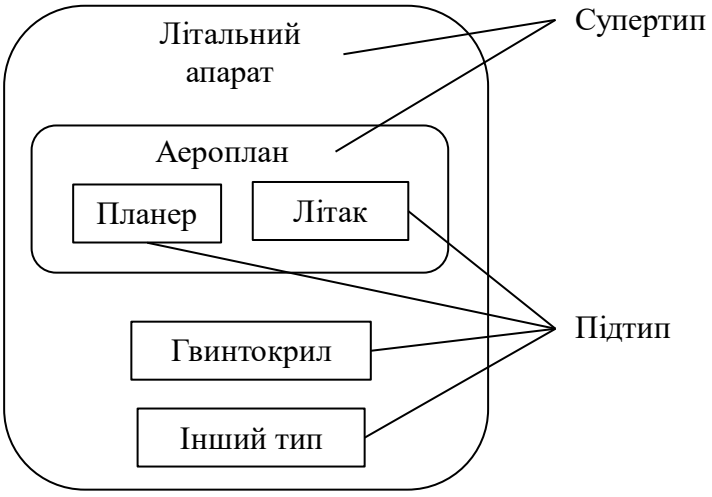


Рисунок 1.20 – Підтипи і супертипи

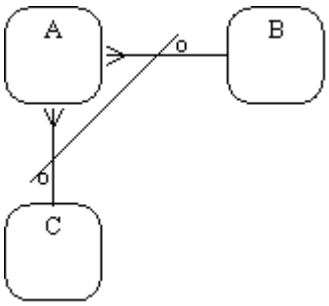


Рисунок 1.21 – Взаємовиключаючий зв'язок

Рекурсивний зв'язок: сутність може бути пов'язана сама з собою (рисунок 1.22) [5].

Непереміщувані (non-transferrable) зв'язки: примірник сутності не може бути перенесений з одного примірника зв'язку в інший (рисунок 1.23) [5].

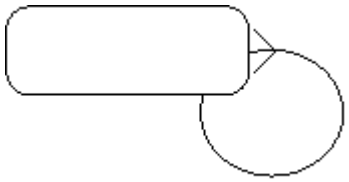


Рисунок 1.22 – Рекурсивний зв'язок

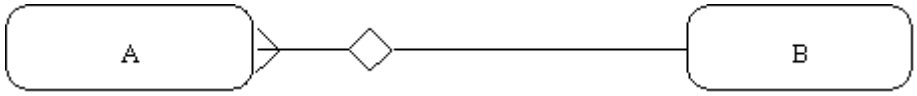


Рисунок 1.23 – Непереміщуваний зв'язок

Контрольні питання

1 Визначте завдання, які вирішуються із застосуванням методології RAD.

2 Наведіть послідовність розроблення функціональної моделі за допомогою методології SADT (IDEF0).

3 Наведіть основні елементи, які застосовуються у діаграмах потоків даних DFD, а саме:

– в нотації Йордона-Де Марко;

– в нотації Гейна-Сарсона.

4 Наведіть результати порівняльного аналізу методологій функціонального моделювання.

5 Дайте визначення моделювання потоків даних (процесів).

6 Які основні компоненти діаграм потоків даних?

7 В чому полягає побудова ієрархії діаграм потоків даних?

8 У чому полягає Case-метод Баркера?

9 Наведіть кроки моделювання даних.

ПРАКТИЧНЕ ЗАНЯТТЯ 2. Нотації, що використовуються при побудові діаграм «сутність-зв'язок»

Завдання: вивчити склад, зміст та особливості нотацій, які застосовуються при побудові діаграм «сутність-зв'язок», а саме:

2.1 нотація Чена;

2.2 нотація Мартіна;

2.3 нотація IDEF1X;

2.4 нотація Баркера.

Детальний опис вказаних нотацій наведено у роботі [6].

Перелік джерел, рекомендованих для поглибленого вивчення матеріалу

1 Batini C., Ceri S., Kant S. and Navathe B. Conceptual Database Design: An Entity Relational Approach. The Benjamin/Cummings Publishing Company, 1991.

2 Date C. J. An Introduction to Database Systems, 5th ed. Addison-Wesley, 1990.

3 Fleming Candace C. and Barbara von Halle. Handbook of Relational Database Design. Addison-Wesley, 1989.

4 Kroenke David. Database Processing, 2nd ed. Science Research Associates, 1983.

5 Martin James. Information Engineering. Prentice-Hall, 1989.

6 Reingruber Michael C. and William W. Gregory. The Data Modeling Handbook: A Best-Practice Approach to Building Quality Data Models. John Wiley & Sons, Inc., 1994.

7 Simsion Graeme. Data Modeling Essentials: Analysis, Design, and Innovation. International Thompson Computer Press, 1994.

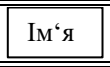
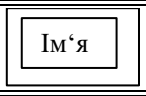
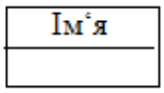
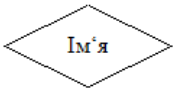
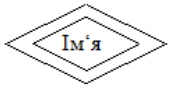
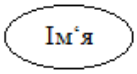
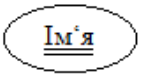
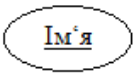
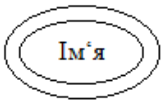
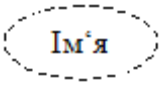
8 Teory Toby J. Database Modeling & Design: The Basic Principles, 2nd ed. Morgan Kaufmann Publishers, Inc., 1994.

2.1 Нотація Чена

Історію розроблення нотації Чена наведено у його статті за посиланням (http://bit.csc.lsu.edu/~chen/pdf/Chen_Pioneers.pdf) [7].

У талиці 2.1 подано умовні позначення алементів діаграм «сутність-зв’язок» у нотації Чена [6].

Таблиця 2.1 – Позначення елементів діаграм «сутність-зв’язок» у нотації Чена

Елемент діаграми	Означення
	незалежна сутність
	залежна сутність
	батьківська сутність в ієрархічному зв'язку
	зв'язок
	ідентифікуючий зв'язок
	атрибут
	первинний ключ
	зовнішній ключ (поняття зовнішнього ключа вводиться в реляційній моделі даних)
	багатозначний атрибут
	одержуваний (успадкований) атрибут в ієрархічних зв'язках

Зв'язок з'єднується з асоційованими сутностями лініями. Біля кожної сутності на лінії, що з'єднує її зі зв'язком, цифрами вказується клас приналежності. Приклад наведено на рисунку 2.1 [6].

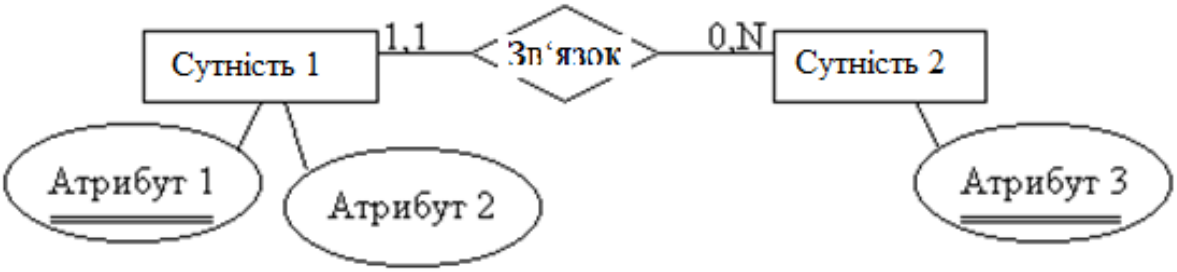


Рисунок 2.1 – Зображення зв'язків сутностей в нотації Чена

2.2 Нотація Мартіна

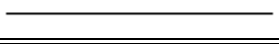
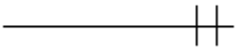
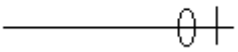
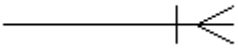
Елемент діаграми означає (таблиця 2.2) [6]:
– незалежну сутність;
– залежну сутність;
– батьківську сутність в ієрархічному зв'язку.

Таблиця 2.2 – Позначення елементів діаграм «сутність-зв’язок» у нотації Мартіна

Елемент діаграми	Означення
Ім'я	незалежна сутність
Ім'я	залежна сутність
Ім'я	батьківська сутність в ієрархічному зв'язку

Список атрибутів наводиться всередині прямокутника, що позначає сутність. Ключові атрибути підкреслюються. Відношення зображують лініями, що з'єднують сутності, вид лінії в місці з'єднання з сутністю визначає кардинальність зв'язку (таблиця 2.3) [6].

Таблиця 2.3 – Позначення зв’язків у діаграмі сутність-зв’язок нотації Мартіна

Означення	Кардинальність
	нема
	1,1
	0,1
	M,N
	0,N
	1,N

Ім'я зв'язку вказується на лінії, яка її позначає. Приклад подано на рисунку 2.2.

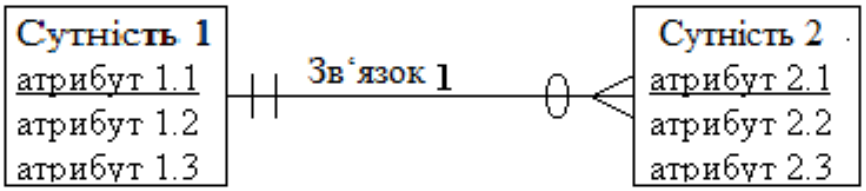


Рисунок 2.2 – Зображення зв'язків сутностей в нотації Мартіна

2.3 Нотація IDEF1X

Означення сутностей (таблиця 2.4) [6]:
– незалежна сутність;
– залежна сутність.

Таблиця 2.4 – Позначення сутностей діаграми сутність-зв’язок у нотації IDEF1X

Елемент діаграми	Означення
Ім‘я	незалежна сутність
Ім‘я	залежна сутність

Список атрибутів наводиться всередині прямокутника, що позначає сутність. Атрибути, що становлять ключ сутності, групуються у верхній частині прямокутника і відокремлюються горизонтальною лінією.

Означення зв'язків (таблиця 2.5) [6]:

– ідентифікуючий зв'язок;

– неідентифікуючий зв'язок.

Таблиця 2.5 – Позначення зв’язків діаграм сутність-зв’язок у нотації IDEF1X

Елемент діаграми	Означення
———————	ідентифікуючий зв'язок
- - - - - - - - - - - -	неідентифікуючий зв'язок

Позначення кардинальності зв'язків (таблиця 2.6) [6]:

1,1; 0, M; 0,1; 1, M; точно N (N – довільне число).

Таблиця 2.6 – Позначення кардинальних зв’язків діаграм «сутність-зв’язок» у нотації IDEF1X

Елемент діаграми	Означення
———————	1,1
———●—	0,M
———●— Z	0,1
———●— P	1,M
———●— N	точно N (N - довільне число)

Приклад наведено на рисунку 2.3 [6].

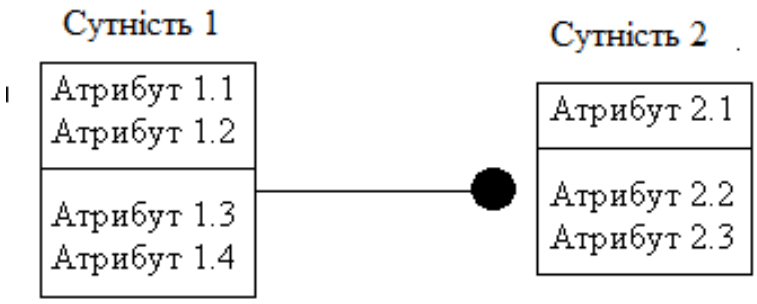


Рисунок 2.3 – Зображення зв'язків сутностей в нотації IDEF1X

Крім того, в IDEF1X вводиться поняття «відношення категоризації», за змістом еквівалентне розглянутому нами ієрархічному зв'язку. Відношення повної категоризації (сутності-категорії складають повну множину нащадка батьківської сутності) позначається згідно з рисунком 2.4 [6].

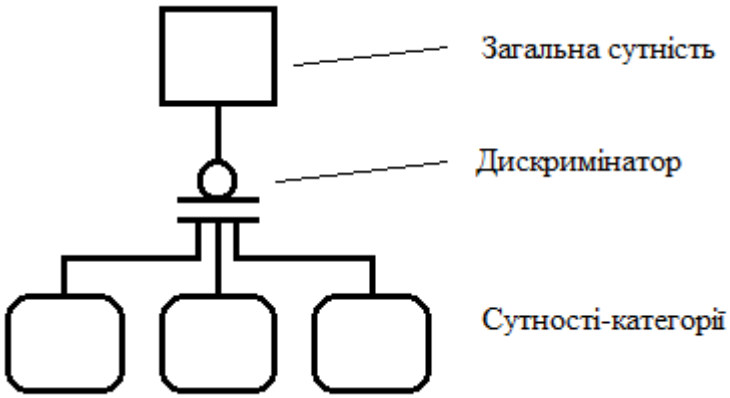


Рисунок 2.4 – Зображення зв'язків сутності-категорії
в нотації IDEF1X

Також може існувати відношення неповної категоризації (сутності-категорії складають неповну множину нащадків загальної сутності) (рисунок 2.5) [6].

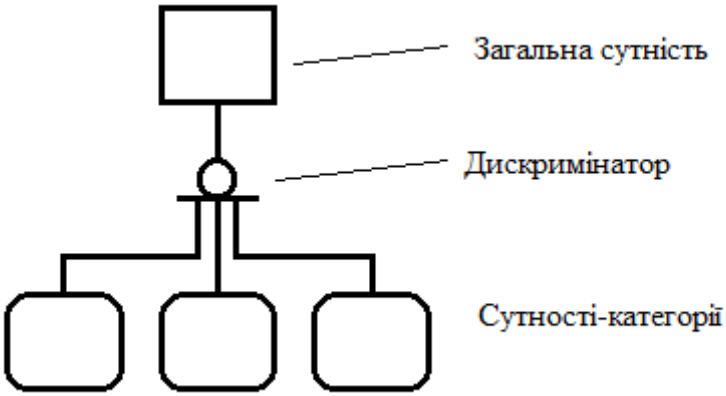
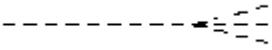
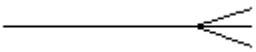


Рисунок 2.5 – Зображення зв'язків неповної категоризації сутностей в нотації IDEF1X

2.4 Нотація Баркера

Сутності позначаються прямокутниками, всередині яких наводиться список атрибутів; ключові атрибути – символом «#» (решітка); зв'язки – лініями з іменами, місце з'єднання зв'язку і сутності визначає кардинальність зв'язку: 0,1; 1,1; 0, N; 1, N (таблиця 2.7) [6].

Таблиця 2.7 – Позначення зв’язків діаграм «сутність-зв’язок» у нотації Баркера

Позначення	Кардинальність
	0,1
	1,1
	0,N
	1,N

Приклад подано на рисунку 2.6.

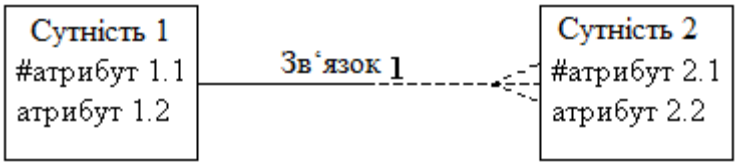


Рисунок 2.6 – Зображення зв'язків сутностей в нотації Баркера

Для позначення відношення категоризації вводиться елемент «дуга» (рисунок 2.7).

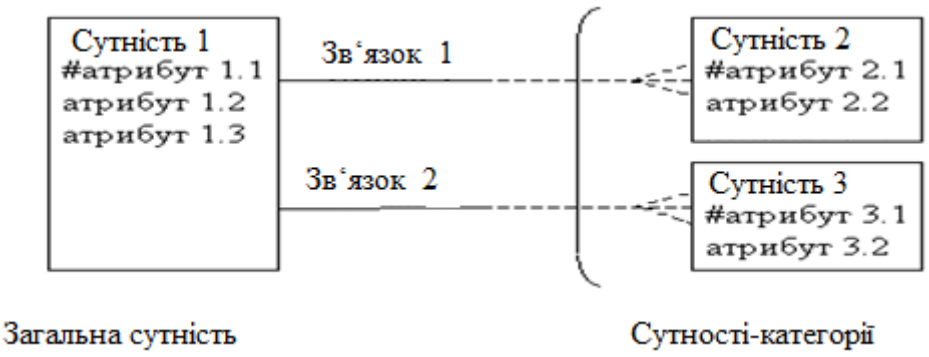


Рисунок 2.7 – Зображення зв'язків категоризації сутностей в нотації Баркера

Контрольні питання

1 Нотація Чена. Наведіть склад і зміст елементів діаграми.
2 Нотація Мартіна. Наведіть склад і зміст елементів діаграми.
3 Нотація IDEF1X. Наведіть склад і зміст елементів діаграми.
4 Нотація Баркера. Наведіть склад і зміст елементів діаграми.

ПРАКТИЧНЕ ЗАНЯТТЯ 3. Ієрархічна і мережна моделі даних. Структура даних

Завдання: набути практичних навичок побудови ієрархічної моделі даних, (мережної моделі даних) і розробити структуру даних для цих моделей.

3.1 Ієрархічна модель даних

Організація даних в СУБД ієрархічного типу визначається в таких термінах: елемент, агрегат, запис (група), групове відношення, база даних [8-10].

Статус (елемент даних) – найменша одиниця структури даних. Зазвичай кожному елементу при описі бази даних присвоюється унікальне ім'я. За цим іменем до нього звертаються при обробці. Елемент даних також часто називають полем.

Запис – іменована сукупність атрибутів. Використання записів дозволяє за одне звернення до бази отримати деяку логічно

зв'язану сукупність даних. Саме записи змінюються, додаються і видаляються. Тип запису визначається складом її атрибутів. Примірник запису – конкретний запис з конкретним значенням елементів [8].

Групове відношення – ієрархічне відношення між записами двох типів. Батьківський запис (власник групових відношень) називається вихідним, а дочірні записи (члени групових відношень) – підлеглими. Ієрархічна база даних може зберігати тільки такі деревоподібні структури.

Кореневий запис кожного дерева обов'язково повинен містити ключ з унікальним значенням. Ключі некореневих записів повинні мати унікальне значення тільки в рамках групових відношень. Кожен запис ідентифікується повним зчепленим ключем, під яким розуміється сукупність ключів усіх записів від кореневого за ієрархічним напрямом [8].

При графічному зображенні групові відношення зображують дугами орієнтованого графа, а типи записів – вершинами (діаграма Бахмана).

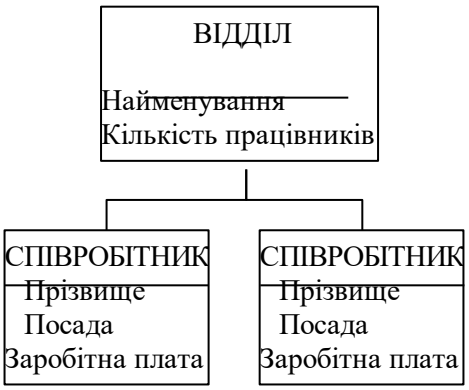
Для групових відношень в ієрархічній моделі забезпечується автоматичний режим включення і фіксоване членство. Це означає, що для запам'ятовування будь-якого некореневого запису в БД має існувати його батьківський запис. При видаленні батьківського запису автоматично видаляються всі підлеглі.

Приклад. Розглянемо таку модель даних підприємства (дивись рисунок 3.1): підприємство складається з відділів, у яких працюють співробітники. У кожному відділі може працювати кілька співробітників, але співробітник не може працювати більш ніж в одному відділі.

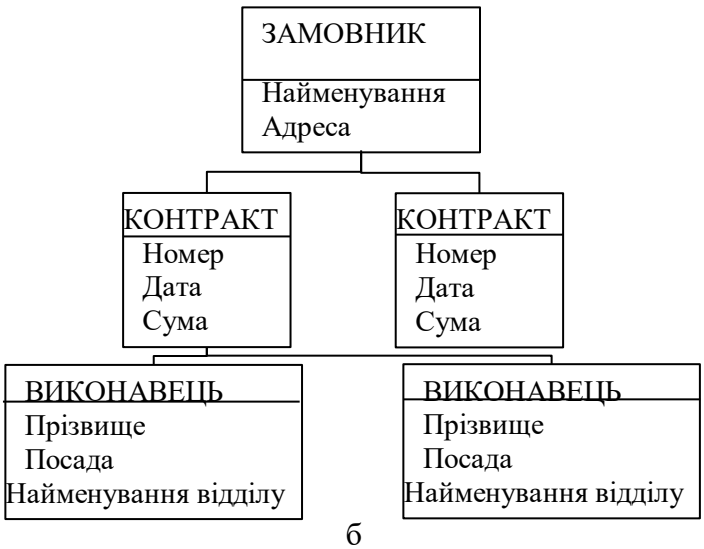
Тому для інформаційної системи управління персоналом необхідно створити групове відношення, що складається з батьківського запису ВІДДІЛ (НАЙМЕНУВАННЯ_ВІДДІЛУ, КІЛЬКІСТЬ_ПРАЦІВНИКІВ) і дочірнього запису СПІВРОБІТНИК (ПРІЗВИЩЕ, ПОСАДА, ОКЛАД). Це відношення показано на рисунку 3.1, а. Для простоти приймається, що є тільки два дочірніх записи.

Для автоматизації обліку контрактів з замовниками необхідно створення ще однієї ієрархічної структури: замовник – контракти з ним – співробітники, задіяні в роботі над контрактом.

Це дерево буде включати записи ЗАМОВНИК (НАЙМЕНУВАННЯ_ЗАМОВНИКА, АДРЕСА), КОНТРАКТ (НОМЕР, ДАТА, СУМА), ВИКОНАВЕЦЬ (ПРІЗВИЩЕ, ПОСАДА, НАЙМЕНУВАННЯ_ВІДДІЛУ) (рисунок 3.1, б) [8].



а



б

в

Рисунок 3.1 – Модель даних підприємства

З цього прикладу видно недоліки ієрархічних БД [8]:

– частково дублюється інформація між записами СПІВРОБІТНИК і ВИКОНАВЕЦЬ (такі записи називають парними), причому в ієрархічній моделі даних не передбачена підтримка відповідності між парними записами;

– ієрархічна модель реалізує відношення між вихідним і дочірнім записом за схемою 1:N, тобто одному батьківському запису може відповідати будь-яка кількість дочірніх. Припустимо тепер, що виконавець може брати участь більш ніж в одному контракті (тобто виникає зв'язок типу M:N). В цьому випадку в базу даних необхідно ввести ще одне групове відношення, в якому ВИКОНАВЕЦЬ буде вихідним записом, а КОНТРАКТ – дочірнім (рисунок 3.1, в). Таким чином, ми знову змушені дублювати інформацію.

3.2 Операції над даними, визначені в ієрархічній моделі [8]

ДОДАТИ в базу даних новий запис. Для кореневого запису обов'язково формування значення ключа.

ЗМІНИТИ значення даних попередньо витягнутого запису. Ключові дані не повинні зазнавати змін.

ВИДАЛИТИ деякий запис і всі підлеглі йому записи.

ВИТЯГТИ:

– кореневий запис за ключовим значенням, допускається також послідовний перегляд кореневих записів;

– наступний запис (такий запис витягується в порядку лівостороннього обходу дерева).

В операції витягти допускається завдання умов вибірки (наприклад, витягти співробітників з окладом понад 1 тис. грн).

Як бачимо, всі операції зміни застосовуються тільки до одного «поточного» запису (який попередньо витягнуто з бази даних). Такий підхід до маніпулювання даними отримав назву «навігаційного».

Обмеження цілісності. Підтримується тільки цілісність зв'язків між власниками і членами групового відношення (жоден нащадок не може існувати без предка). Як уже зазначалося, не

забезпечується автоматична підтримка відповідності парних записів, що входять до різних ієрархій.

3.3 Мережна модель даних. Структура даних

3.3.1 Мережна модель даних

На розроблення цього стандарту значний вплив зробив американський вчений Ч. Бахман. Основні принципи мережної моделі даних було розроблено в середині 1960-х рр., еталонний варіант мережної моделі даних описано у звітах робочої групи з мов баз даних (COnference on DAta SYstem Languages) CODASYL (1971) [11].

Мережна модель даних визначається в тих самих термінах, що й ієрархічна. Вона складається з безлічі записів, які можуть бути власниками або членами групових відношень. Зв'язок між записом-власником і записом-членом також має вигляд 1:N.

Основна відмінність цих моделей полягає в тому, що в мережній моделі запис може бути членом більш ніж одного групового відношення. Відповідно до цієї моделі кожне групове відношення іменується і позначається відмінність між його типом та екземпляром. Тип групового відношення задається його ім'ям і визначає властивості, загальні для всіх екземплярів даного типу. Примірник групового зв'язку є записом-власником і множиною (можливо, порожньою) підлеглих записів. При цьому мається на увазі таке обмеження: екземпляр запису не може бути членом двох примірників групових відношень одного типу (тобто співробітник наприклад, не може працювати в двох відділах).

Ієрархічна структура перетворюється в мережну таким чином:

– дерева замінюються однією мережною структурою, в якій запис СПІВРОБІТНИК входить у два групових відношення;

– для відображення типу M:N вводиться запис СПІВРОБІТНИК_КОНТРАКТ, який не має полів і служить тільки для зв'язку записів КОНТРАКТ і СПІВРОБІТИК (дивись рисунок 3.2) [11]. (Відзначимо, що в цьому записі може зберігатися і корисна інформація, наприклад, частка даного співробітника в загальній винагороді за даним контрактом.)

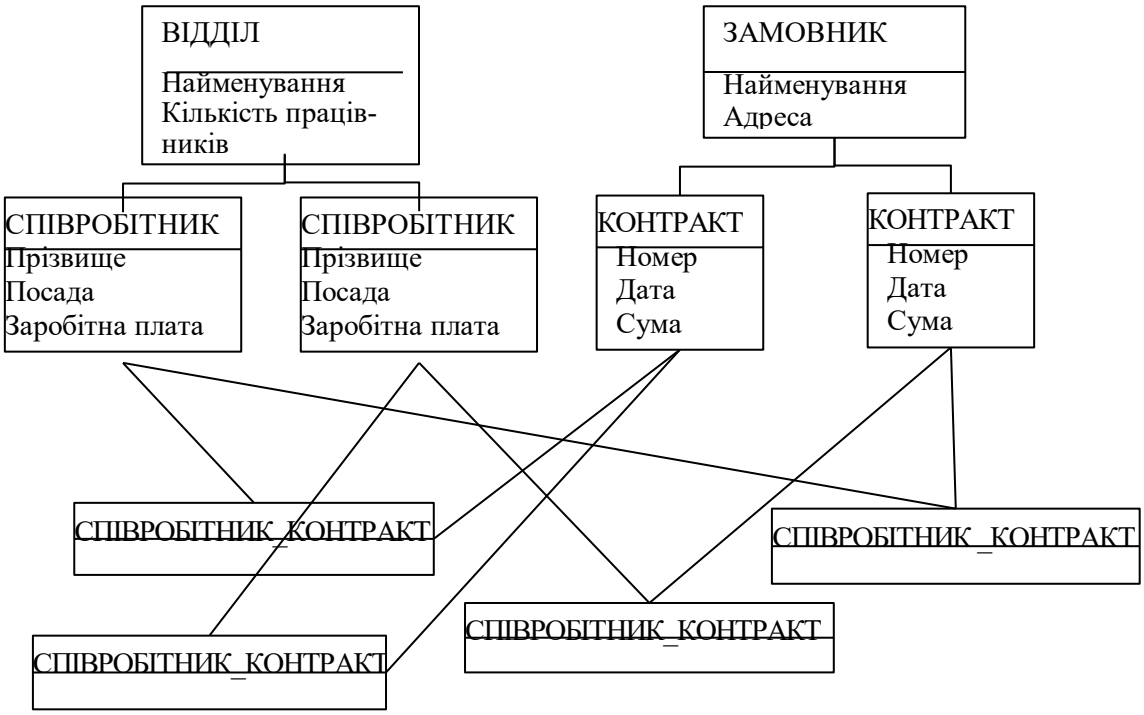


Рисунок 3.2 – Дерево ієрархічної структури

Кожен екземпляр групового відношення характеризується способами упорядкування підлеглих записів:

– довільний;

– хронологічний / черга /;

– зворотний хронологічний / стек /;

– сортований.

Якщо запис оголошено підпорядкованим в декількох групових відношеннях, то в кожному з них може бути призначений свій спосіб упорядкування.

Режим включення підлеглих записів [11]:

– автоматичний – неможливо занести в БД запис без того, щоб він був одразу ж закріплений за певним власником;

– ручний – дозволяє запам'ятати в БД підпорядкований запис і не включати його негайно в екземпляр групового відношення. Ця операція пізніше ініціюється користувачем;

– режим виключення.

Прийнято виділяти три класи членства підлеглих записів в групових відношеннях:

1) фіксований. Підпорядкований запис жорстко пов'язано із записом власників і його можна виключити з групового

відношення тільки видаливши. При видаленні запису-власника всі підлеглі записи автоматично теж видаляються. У розглянутому вище прикладі фіксоване членство передбачає групове відношення між записами КОНТРАКТ і ЗАМОВНИК, оскільки контракт не може існувати без замовника;

2) обов'язковий. Допускається перехід підпорядкованого запису на іншого власника, але неможливо його існування без власника. Для видалення запису-власника необхідно, щоб він не мав підлеглих записів з обов'язковим членством. Таким відношенням пов'язані записи СПІВРОБІТНИК і ВІДДІЛ. Якщо відділ розформовується, всі його співробітники мають бути або переведені в інші відділи, або звільнені;

3) обов'язковий. Можна виключити запис з групового відношення, але зберегти його в базі даних, не прикріплюючи до іншого власника. При видаленні запису-власника його підлеглі записи – необов'язкові члени – зберігаються в базі, більше не беручи участі в груповому відношенні такого типу. Прикладом такого групового відношення може служити ВИКОНУЄ між СПІВРОБІТНИКИ і КОНТРАКТ, оскільки в організації можуть існувати працівники, чия діяльність не пов'язана з виконанням будь-яких договірних зобов'язань перед замовниками.

3.3.2 Операції над даними в мережевій моделі даних [11]

ДОДАТИ – внести запис до БД і залежно від режиму включення або включити його до групового відношення, де він оголошений підлеглим, або не включати до жодного групового відношення.

ВКЛЮЧИТИ ДО ГРУПОВОГО ВІДНОШЕННЯ – зв'язати існуючий підпорядкований запис із записом-власником.

ПОМІНЯТИ – зв'язати існуючий підпорядкований запис з іншим записом-власником у тому самому груповому відношенні.

ОНОВИТИ – змінити значення елементів попередньо вилученого запису.

ВИЛУЧИТИ – вилучити записи послідовно за значенням ключа, а також використовуючи групові відношення: від власника можна перейти до записів-членів, а від підпорядкованого запису – до власника набору.

• ВИДАЛИТИ – прибрати з БД запис. Якщо цей запис є власником групового відношення, то аналізується клас членства підлеглих записів. Обов'язкові члени мають бути попередньо виключені з групового відношення, фіксовані видалені разом з власником, необов'язкові залишаться в БД.

ВИКЛЮЧИТИ З ГРУПОВОГО ВІДНОШЕННЯ – розірвати зв'язок між записом-власником і записом-членом.

3.3.3 Обмеження цілісності

Як і в ієрархічній моделі забезпечується тільки підтримання цілісності за посиланням (власник відношення – член відношення).

Контрольні питання

1 Наведіть приклад ієрархічної моделі даних.

2 Назвіть операції над даними, які визначені в ієрархічній моделі даних.

3 Наведіть приклад мережної моделі даних.

4 Назвіть операції над даними, які визначені в мережній моделі даних.

СПИСОК ЛІТЕРАТУРИ

1 Доценко С. І. Теоретичні основи створення інтелектуальних систем комп’ютерної підтримки рішень при управлінні енергозбереженням організацій : дис. д-ра техн. наук: 05.13.06 / Харківський національний технічний університет сільського господарства імені Петра Василенка. Харків, 2017. 369 с.

2 Діаграма потоків даних. URL: http://uk.wikipedia.org/wiki /%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC %D0%B0_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D1% 96%D0%B2_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85.

3 Методологии функционального моделирования. URL: http://www.mstu.edu.ru/study/materials/zelenkov/ch_5_3.html.

4 Методология DFD в нотациях Гейна — Сарсона. URL: http://bstudy.net/700472/ekonomika/metodologiya_notatsiyah_geyna_sarsona_yordana_marko.

5 Моделювання даних Case-метод Баркера. URL: http://studfile.net/preview/14511093/page:2/

6 Буй Д. Б., Сільвейструк Л. М. Формалізація моделі «Сутність-Зв'язок»: монографія. URL: http://csc.knu.ua/en/library/books/bui-silveistruk-33.pdf/.

7 Нотації Чена. URL: http://bit.csc.lsu.edu/~chen/pdf/Chen_Pioneers.pdf.

8 Ієрархічна модель даних. URL: http://www.mstu.edu.ru/study/materials/zelenkov/ch_3_1.html/.

9 Codd E.F. A relational model of data for large shared data banks. URL: https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf (дата звернення: 24.09.2021).

10 Introducing databases by Stephen Chu, in Conrick, M. (2006) Health informatics: transforming healthcare with technology, Thomson. *Eprints*. URL: https://eprints.usq.edu.au/1771/3/docs.pdf (дата звернення: 24.09.2018).

11 Мережева модель даних. URL: https://beasthackerz.ru/uk/odnoklassniki/kakie-modeli-dannyh-ispolzuyutsya-v-bd-shifrovanie.html.

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять
з дисципліни

«ОРГАНІЗАЦІЯ ТА СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ»

Відповідальний за випуск Доценко С. І.

Редактор Третьякова К. А.

Підписано до друку 15.06.21 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 2,25. Тираж 5. Замовлення №

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.