Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

_____Інститут електроенергетики_____
(інститут)
_____факультет інформаційних технологій_____
(факультет)
Кафедра__інформаційних систем та технологій та комп'ютерної інженерії___
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ____бакалавра___
(бакалавра, спеціаліста, магістра)

Студента_____Коваленко Олександр Артурович_____
(ПІБ)
академічної групи_____126-17-1_____
(шифр)
спеціальності _____126 « Інформаційні системи та технології»____
(код і назва спеціальності)
за освітньо-професійною програмою
_____«Інформаційні системи та технології»_____
(офіційна назва)
на тему " Розробка_АІС_контрольно-пропускного_пункту_підприємства**"**
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доцент Сергєєва К. Л.			
розділів:				

Рецензент	професор Алексєєв М.О.			

Нормоконтролер	професор Коротенко Г.М.			

Дніпро
2021

ЗАТВЕРДЖЕНО:
завідувач кафедри
___інформаційних технологій___
___та комп'ютерної інженерії____
(повна назва)
_____ __Гнатушенко В.В.__
(підпис) (прізвище, ініціали)

«_____»_____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____**бакалавр**_____
(бакалавра, спеціалістара)

студенту Коваленку О. А. **академічної групи** ____126-17-1___
(прізвище та ініціали) (шифр)

спеціальності ___126 « Інформаційні системи та технології »_____

за освітньою-професійною програмою _____

_____« Інформаційні системи та технології»_____

на тему ____Розробка_АІС_контрольно-пропускного_пункту_підприємства_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 07.06.2021р. №317-с

Розділ	Зміст	Термін виконання
Розділ 1. Огляд й аналіз методів створення звітів	На основі матеріалів з відповідних джерел та даних проаналізувати стан області задачі	30.03.2021 р.
Розділ 2. Проектна частина	На основі матеріалів передатестаційної практики та інших науково-технічних джерел проаналізувати методи і засоби по створенню АІС та реалізувати відповідну розробку	24.06.2021 р.

Завдання видано _____ Сергєєва К.Л.
(підпис керівника) (прізвище, ініціали)

Дата видачі 09.02.2021

Дата подання до екзаменаційної комісії___24.06.2021 р._____

Прийнято до виконання _____ Коваленко О. А.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 68 с., 16 рис., 7 табл., 5 додатків, 27 джерела.

Об’єкт розробки: створення АІС для контрольно-пропускного пункту підприємства.

Мета роботи: основною ціллю бакалаврської роботи є розробка та налагодження програмного забезпечення для організації роботи КПП на підприємстві.

У першому розділі бакалаврської роботи розглянуті теоретичні аспекти роботи, а саме: поняття баз даних, властивості баз даних, призначення та класифікація систем управління базами даних.

У другому розділі описано вибір программного забезпечення, створення бази даних, розробка методів та алгоритмів генерації звітів

Практичне значення кваліфікаційної роботи полягає у основним завданням програми клієнта є обробка подій та відправлення їх на сервер. До подій відносять події відкриття тих чи інших ресурсів, а також події обробки, збереження та видалення інформації, а також зміна ім'я користувача. Дані події є тригерами, які говорять про час початку і закінчення роботи співробітника, так як при загальній схожості, кожна має свої унікальні якості, які в певних ситуаціях можуть виступати як плюсами, так і мінусами. Перед прийняттям рішення про впровадження такої системи необхідно провести повний аналіз існуючих процесів в компанії, зрозуміти, які джерела даних можуть бути використані для обліку робочого часу співробітника. Створено повноцінну базу даних. Для цього запропоновано набір таблиць, необхідних для функціонування програмного засобу.

Розроблене технологічне рішення може бути запроваджено на всіх підприємствах де використовують пропускний пункт КПП

ESSAY

Explaining note: 68 p., 16 fig., 7 tab., 5 additional documents, 27 dzherela.

Ob'єkt development: gate AIC for the checkpoint of the enterprise.

Meta robots: the main goal of bachelor's robotics is the development and improvement of software security for organizing a checkpoint robot at an enterprise.

For the first bachelor robotics, the theoretical aspects of the robot will be understood, and the same: the understanding of the data bases, the power of the data bases, the recognition of the classification of the systems for managing the data bases.

The other section describes the vibration of the software security, the base of the data, the distribution of methods and algorithms for the generation of results.

Practically significant value for the quality of robots in the main client's programs є processing of podiy and sending them to the server. Until the end of the introduction of the report of quiet resources, as well as the process of processing, saving that visual information, as well as changing the name of the corystuvach. These are the triggers, like talking about an hour of ear and finishing of the robot and the athlete, so that in case of zagalny similarity, the skin has its own unique qualities, like in singing situations there can be pluses, so in my opinion. Before taking decisions about the implementation of such a system, it is necessary to conduct a reanalysis of the current processes in the company, intelligence, as dzherela danikh can be a victorian for the work hour of a sportsman. Created a base of tributes. For the whole, a set of tables, which are necessary for the function of the software program, are proponated.

The development of technological solutions can be provided at all enterprises de vikoristovoyt checkpoint checkpoint.

Зміст

ВСТУП

Бази даних виконують функцію систематизації знань. На основі цієї систематизації можуть створюватися нові знання. Так чи інакше, будь-яка база даних служить людині саме для опису подій, що відбулися у минулому, і на основі знання цих подій допомагає ухвалити те або інше рішення на майбутнє.

Технологія баз даних використовується в безлічі додатків. Деякі з них призначені для єдиного користувача з єдиним комп'ютером, інші використовуються робочими групами в кількості 20-30 чоловік через локальну мережу, треті служать сотням користувачів і містять трильйони байтів даних. В час цифрових технологій, що супроводжується великою кількістю оцифрованого матеріалу, не можливо обійтись без систематизації зібраної інформації. Саме тому виникла необхідність створення електронних баз даних для оперативного збору та пошуку необхідної інформації, для будь-якого користувача даної бази. База знань – це, перш за все, сховище об'єктів даних, тобто набору можливих понять або подій, що описуються базою даних, з можливістю пошуку цих об'єктів за ознаками. Невід'ємною межею бази даних є можливість скріплення об'єктів між собою.

Основною ціллю дипломної роботи є розробка та налагодження програмного забезпечення для організації роботи кпп на підприємстві. Для роботи над дипломним проектом було обрано досить вузьку предметну область, що дозволяє розробити програму для кпп.

1. Огляд й аналіз методів створення звітів в інформаційних системах

Інформаційна система — сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів. В наш час, при наявності комп’ютерів, практично кожну інформаційну систему можна вважати автоматизованими – взаємозв'язаною сукупністю даних, обладнання, програмних засобів, персоналу, стандартних процедур, які призначені для збору, обробки, розподілу, зберігання, представлення інформації згідно з вимогами, які випливають з цілей організації.

Документ, у якому в письмовій формі подається повідомлення про виконання певної роботи називають звітом. Загалом розрізняють кілька видів таких документів:

- статистичні (цифрові) — на спеціальних, виготовлених друкарським способом бланках;
- текстові — на звичайному папері за встановленою формою;
- періодичні - затверджуються керівником, який раніше підписав план ;
- разові - не затверджуються, а лише адресуються посадовим особам.

За допомогою звітів відповідальна особа має можливість без зайвих зусиль *вивчити*, *перевірити* й *узагальнити* чиюсь роботу, знайти в ній позитивне й негативне, зробити висновки, окреслити перспективи.

У час цифрових технологій та насиченої мережі програмного забезпечення існує чимала кількість автоматизованих систем для створення звітів у будь-якій предметній області. Нижче наведено основні комплекси автоматичної генерації звітів.

Висновки до розділу «Огляд й аналіз методів створення звітів в інформаційних системах»

Розглянуті в розділі сервіси для електронної звітності та бухгалтерії мають повний функціонал та інструментарій для формування, зберігання та статистичних електронних форм звітності, що подаються до державних контролюючих органів. Всі електронні звітні форми, що готуються в програмах автоматичної звітності повністю відповідають затвердженим контролюючими органами специфікаціям та форматам. Переважна більшість таких сервісів за допомогою телекомунікаційних каналів зв’язку мають можливість здійснювати обмін електронними документами між підзвітними організаціями та державними контролюючими органами.

Використовуючи сервіси автоматичної звітності, платники податків можуть вести бізнес без вивчення обліку на кпп.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1 Поняття бази даних

База даних (БД) — це організована структура, призначена для зберігання інформації: даних і методів, за допомогою яких відбувається взаємодія з іншими програмно-апаратними комплексами.

Головне завдання баз даних — гарантоване збереження значних обсягів інформації (так звані записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, база даних складається з двох частин: збереженої інформації та системи керування нею.

З метою забезпечення ефективності доступу записи даних організовують як множину фактів – елемент даних.

Бази даних класифікують:

1. За способом організації даних:
   - фактографічні(оперують фактами (даними) різних типів, що зв'язані в системі в більш чи менш складні структури);
   - документальні (зорієнтовані на обробку та зберігання документа, який неподільний з точки зору системи).
2. За технологією обробки даних:
   - централізована база даних зберігається в пам'яті однієї комп'ютерної системи. (часто застосовують у локальних мережах);
   - розподілена база даних складається з декількох частин, збережених у пам'яті різних комп'ютерів обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).
3. За способом доступу до даних:
   - локальні – доступ до даних здійснюється з одного комп'ютера;
   - мережні – доступ можливий із різних ПК комп'ютерної мережі.
4. За структурою:

•структуровані – використовують структури даних, тобто структурований опис типу фактів за допомогою схеми даних, більш відомої як модель даних. Модель даних описує об'єкти та взаємовідношення між ними. Існує декілька моделей баз даних, основні: плоска, ієрархічна, мережна та реляційна. Приблизно з 2000 року більше половини БД використовують реляційну модель.

- Неструкторавані – повнотекстові бази даних, які містять неструктуровані тексти статей чи книг у формі, що дозволяє здійснювати швидкий пошук [6].

1.2 Моделі баз даних

Основою бази даних є модель даних — фіксована система понять і правил для представлення даних структури, стану і динаміки проблемної області в базі даних.

База даних може бути заснована на одній моделі або на сукупності декількох.

Існує три основні типи моделей даних:

- реляційна;
- ієрархічна;
- мережева;

Термін «*реляційний*» (від латин. relatio — відношення) указує передусім на те, що така модель зберігання даних побудована на взаємовідношенні частин, що її складають. У найпростішому випадку реляційна модель являє собою двовимірний масив або двовимірну таблицю, а при створенні складних інформаційних моделей складає сукупність взаємопов’язаних таблиць. Кожний рядок такої таблиці називається записом, кожний стовпець – полем.

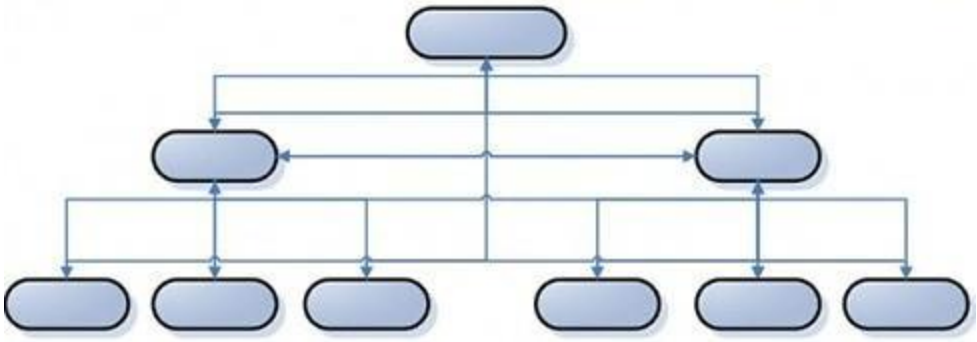


Рис. 1.1 Структура реляційної моделі бази даних

Реляційна модель бази даних має такі властивості:

- кожний елемент таблиці – один елемент даних;
- усі стовпці в таблиці є однорідними, тобто мають однаковий тип;
- кожний стовпець (поле) має унікальне ім’я;
- однакові рядки в таблиці відсутні;
- порядок слідування рядків у таблиці може бути довільним і може характеризуватися кількістю полів, кількістю записів, типом даних.

Таку модель бази даних зручно використовувати для:

- сортування даних (наприклад за алфавітом);
- вибірка даних за групами (наприклад класами);
- пошук записів (наприклад за прізвищами) і т. д.

Реляційна модель даних, як правило, складається з декількох таблиць, які зв’язуються між собою ключами. Ключ – поле, яке однозначно визначає відповідний запис. Необхідно зазначити, що зараз реляційна модель даних є найбільш зручною і застосовною моделлю зберігання даних.

Основною відмінністю пошуку даних в *ієрархічних*, *мережних* і *реляційних* базах даних є те, що ієрархічні і мережні моделі даних здійснюють зв'язок і пошук між різними об'єктами за структурою, а реляційні – за значенням ключових атрибутів.

Оскільки реляційна структура концептуально проста, вона дозволяє реалізовувати невеликі і прості (і тому легкі для створення) бази даних, навіть персональні, сама можливість реалізації яких ніколи навіть і не розглядалася в системах з ієрархічною чи мережною моделлю.

Недоліком реляційної моделі даних є надмірність по полях (для створення зв'язків між різними об'єктами бази даних).

Практично всі існуючі на сьогоднішній день комерційні бази даних і програмні продукти для їх створення використовують реляційну модель даних.

Ієрархічна модель бази даних являє собою сукупність елементів, які розташовані у порядку їх підкорення від загального до часткового і створюють обернене дерево – граф. Ця модель характеризується такими параметрами, як рівні, вузли, зв’язки. Вузол — це інформаційна модель елемента, що міститься на даному рівні ієрархії.

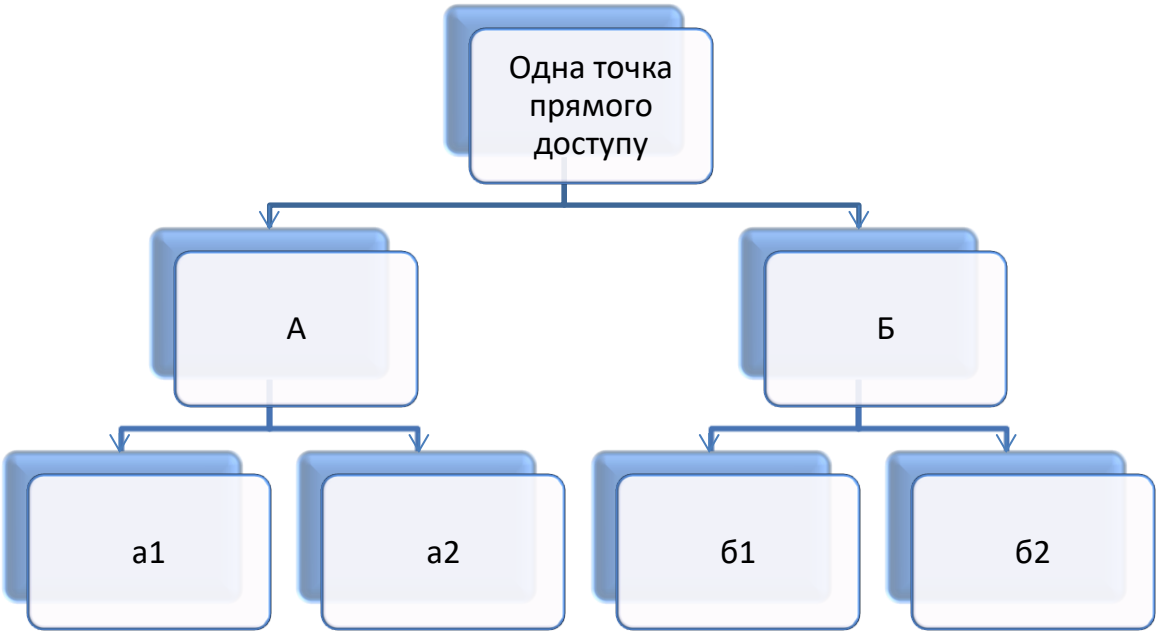


Рис. 1.2 Схема відношень між об'єктами в ієрархічній базі даних

Об'єкти, що перебувають в ієрархічних відношеннях, утворюють дерево «орієнтований граф», у якого є тільки одна вершина, не підлегла жодній іншій вершині (цю вершину називають коренем дерева); будь-яка інша вершина графа підлегла лише одній іншій вершині (рис. 6).

Концептуальна схема ієрархічної моделі являє собою сукупність типів записів, пов'язаних типами зв'язків в одне чи кілька дерев. Усі типи зв'язків цієї моделі належать до виду «один до декількох» і зображуються у вигляді стрілок.

Таким чином, взаємозв'язки між об'єктами нагадують взаємозв'язки в генеалогічному дереві, за єдиним винятком: для кожного породженого (підлеглого) типу об'єкта може бути тільки один вхідний (головний) тип об'єкта. Тобто ієрархічна модель даних допускає тільки два типи зв'язків між об'єктами: «один до одного» і «один до декількох». Ієрархічні бази даних є навігаційними, тобто доступ можливий тільки за допомогою заздалегідь визначених зв'язків.

При моделюванні подій, як правило, необхідні зв'язки типу «багато до декількох». Як одне з можливих рішень зняття цього обмеження можна запропонувати дублювання об'єктів. Однак дублювання об'єктів створює можливості неузгодженості даних.

Достоїнство ієрархічної бази даних полягає в тому, що її навігаційна природа забезпечує швидкий доступ при проходженні вздовж заздалегідь визначених зв'язків. Однак негнучкість моделі даних і, зокрема , неможливість наявності в об'єкта декількох батьків, а також відсутність прямого доступу до даних роблять її непридатною в умовах частого виконання запитів, не запланованих заздалегідь. Ще одним недоліком ієрархічної моделі даних є те, що інформаційний пошук з нижніх рівнів ієрархії не можна спрямувати по вище розміщених вузлах.

Мережева модель даних схожа на ієрархічну. Вона має ті самі основні складові (вузол, рівень, зв’язок), однак характер їх відносин принципово інший. У мережевій моделі прийнятий вільний зв’язок між елементами різних рівнів.

У моделях даних такого типу поняття головних і підлеглих об'єктів дещо розширені. Будь який об'єкт може бути і головним, і підлеглим. Той самий об'єкт може одночасно виконувати і роль власника, і роль члена набору. Це означає, що кожний об'єкт може брати участь у будь-якій кількості взаємозв'язків.

Подібно до ієрархічної, мережну модель також можна подати у вигляді орієнтованого графа. Але в цьому випадку граф може містити цикли, тобто вершина може мати кілька батьківських вершин.

Така структура набагато гнучкіша і виразніша від попередньої і придатна для моделювання більш ширшого класу завдань. У цій моделі вершини є сутностями, а ребра, що їх з'єднують, — відношеннями між ними (рис. 1.3).

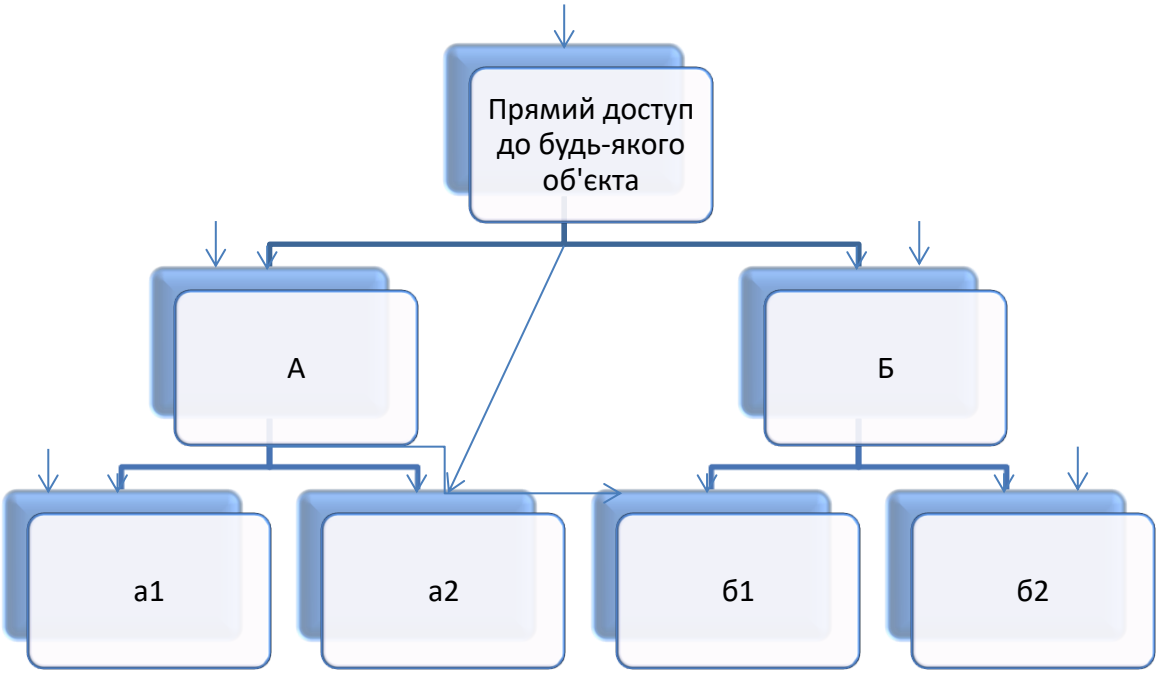


Рис. 1.3 Схема відношень між об'єктами в мережній базі даних

Ієрархічні і мережні бази даних часто називають базами даних з навігацією. Ця назва відбиває технологію доступу до даних, використовувану при написанні програм обробки мовою маніпулювання даними. При цьому доступ до даних по шляхах, не передбачених при створенні бази даних, може потребувати нерозумно тривалого часу.

Підвищуючи ефективність доступу до даних і скорочуючи таким чином час відповіді на запит, принцип навігації разом з цим підвищує і ступінь залежності програм і даних. Програми обробки даних виявляються жорстко прив'язаними до поточного стану структури бази даних і повинні бути переписані при її змінах. Операції модифікації і видалення даних вимагають переустановлення покажчиків, а маніпулювання даними залишається

записоорієнтованим. Крім того, принцип навігації не дозволяє істотно підвищувати рівень мови маніпулювання даними, щоб зробити його доступним користувачу-непрограмісту чи навіть програмісту-непрофесіоналу. Для пошуку запису-мети в ієрархічній або мережній структурі програміст повинен спочатку визначити шлях доступу, а потім – крок за кроком переглянути всі записи, що трапляються на цьому шляху [7].

Наскільки складними є схеми представлення ієрархічних і мережних баз даних, настільки і трудомістким є проектування конкретних прикладних систем на їхній основі. Як показує досвід, тривалі терміни розроблення прикладних систем нерідко призводять до того, що вони постійно перебувають на стадії розроблення і доопрацювання. Складність практичної реалізації баз даних на основі ієрархічної і мережної моделей визначила створення реляційної моделі даних.

1.3 Властивості бази даних

Властивості бази даних випливають з основних властивостей керуючої інформаційної системи і з властивостей системи управління базою даних. У відповідності з цим сформулюємо наступне визначення: бази даних - комплекс масивів даних, які логічно пов'язані один з одним, і їх існування має сенс лише в даному організаційному варіанті. База даних обумовлює основні властивості інформаційної системи, складовою частиною якої вона є.

Основними та невід'ємними властивостями БД є :

- для даних допускається така мінімальна надлишковість, яка сприяє їх оптимальному використанню в одному чи кількох застосуваннях;
- незалежність даних від програм;
- для пошуку та модифікації даних використовуються спільні механізми;
- як правило, у складі БД існують засоби для підтримки її цілісності та захисту від неавторизованого доступу
- дані логічно пов'язані між собою і несуть відповідну інформацію;

- структура баз даних звичайно відповідає тому специфічному набору даних, які вона містить;
- бази даних відображають тільки окремі аспекти реального світу, що дає змогу визначити їх як "мікросвіт" [8].

На відміну від файлових систем база даних зорієнтована для підтримки даних для кількох застосувань. На практиці ця властивість інколи порушується. Часом таке порушення можна пояснити тим, що проект вводиться в дію поетапно, і у певний момент дійсно функціонує тільки одне застосування. Іноді відхід від вказаної властивості зумовлений іншими важливими причинами, але, на жаль, не є рідкістю просто помилка у виборі засобів для реалізації проекту.

Взаємозв'язаність даних полягає в тому, що доступ до певної групи даних якогось застосування загалом полегшує доступ до інших груп даних цього ж застосування. В умовах орієнтації БД на велику кількість застосувань виникає необхідність у підтримці значного числа різноманітних зв'язків між даними. Саме у розумінні тісного логічного зв'язку використані слова про збереження разом даних.[3]

1.4 Організація інформаційних масивів бази даних

Ефективне функціонування інформаційної системи об’єкта можливе лише при відповідній організації інформаційної бази - сукупності впорядкованої інформації, яка використовується при функціонуванні ІС і поділяється на зовнішньо – і внутрішньомашинну (машинну) бази.

Зовнішньомашинна інформаційна база – частина інформаційної бази, яка являє собою сукупність повідомлень, сигналів і документів, призначених для безпосереднього сприйняття людиною без застосування засобів обчислювальної техніки.

Внутрішньомашинна інформаційна база - частина інформаційної бази, що використовується в ІС на носіях даних.

Така зовнішньомашинна ІБ має багато модифікацій від подання у вигляді повідомлень на паперовому носії, запитів на екрані дисплея, мовного спілкування з ЕОМ та ін.

Внутрішньомашинна ІБ пройшла *три етапи еволюції*.

Перший етап характеризується роз'єднаним фондом даних:

- програми розв'язання кожної окремої задачі становили одне ціле з масивами, які оброблялися;
- використання будь-якого масиву для іншої задачі забезпечувалось індивідуально пристосуванням до форм подання даних, структур елементів масивів і. т. ін. ;
- опис даних не потрібний, оскільки структура раніше була відома;
- коригування масивів виконувалось індивідуальними засобами;
- задача розв'язувалася в пакетному режимі, користувач отримував результати винятково у вигляді машинограм і виробничих документів через групу підготовки і оформлення даних.

Другий етап централізований фонд даних.

- Дані відокремлені від процедур їх обробки і організовані в бібліотеці масивів загального користування. Подання інформації, формати елементів даних і структура масивів уніфіковані і не залежать від конфігурації пам'яті та організації .
- Опис даних відокремлено як від програм, так і від самих даних, тому дані і програми їх обробки стають значною мірою незалежними. Це полегшує зміну структур даних і програм. Але реорганізація бібліотеки і її окремих груп компонентів потребує зміни програми обробки.

Третій етап - організація баз даних характеризується:

- об'єднанням не лише інформації, а й апаратно програмних засобів її поповнення, коригування і видачі користувачеві;

- повним відокремленням функцій нагромадження, ведення і реорганізації даних від функцій їх обробки. Дані коригуються поза рівнем програм користувача за допомогою власного апарату бази даних;
- появою логічного буферу, системи управління базою даних, розв'язки між програмами користувача і базою даних;
- можливістю оперативної реалізації довільних запитів у режимі безпосереднього зв'язку з ЕОМ;
- високим ступенем централізації загальносистемних масивів, яка передбачає спільне використання загальних даних;
- різноманітністю даних і поєднанням в довільні логічні структури;
- наявністю потужного програмного забезпечення і мовних засобів [9].

Масив даних – це конструкція даних, компоненти якої ідентичні за своїми характеристиками і є значенням функції від фіксованої кількості цілочисельних аргументів.

Відомі два підходи до організації інформаційних масивів:

- файлова організація
- організація у вигляді бази даних.

Файлова організація передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується рядом недоліків.

Характерна риса файлового підходу - вузька спеціалізація як обробних програм, так і файлів даних, що служить причиною великої надлишковості, тому що ті самі елементи даних зберігаються в різних системах. Оскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для

задоволення запитів користувачів, які перекривають дві і більше області. Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів передання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зв'язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань. Вказані причини обумовили появу баз даних.

База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні «файл-програма», вона явним чином декларується в базі.

База даних орієнтована на інтегровані запити, а не на одну програму, як у випадку файлового підходу, і використовується для інформаційних потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації [10].

1.5 Сутність системи управління базами даних

Перехід від структури бази даних до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

Такі системи проробляють операції різної складності, пов'язані з керуваннями даних у спільній базі з інтерфейсом і різною кількістю її користувачів.

Системи управлення базами даних має такі можливості:

- Дозволяє визначати базу даних – це зазвичай здійснюється за допомогою мови визначення даних (DDL- Data Definition Languge). Мова DDL надає користувачеві засоби для зазначення типу даних та їх структури, а також засоби створення обмежень для інформації, яка зберігається у базі.

- Дозволяє додавати, оновлявати, видаляти та вилучати інформацію з БД, що звичайно здійснюється за допомогою мови керування даними (Data Manipulation Language). Наявність централізованого сховища усіх даних та їх опис дозволяє використовувати мову DML, яку часто називають мовою запитів, як загальний інструмент організації запитів. Саме завдяки мові запитів усуваються обмеження, притаманні файловим системам, при яких користувачі мали справу лише з фіксованим набором запитів чи з постійно зростаючою кількістю прикладних програм, що породжували ще більші проблеми управління ресурсами та прогармним забезпеченням. Існує два різновиди мов запитів DML – процедурні та непроцедурні мови, які відрізняються між собою способом вилучення даних. Основна відмінність полягає в тому, що процедурні мови звичайно обробляють інформацію у базі даних послідовно, запис за записом, а непроцедурні оперують одразу цілими наборами даних. Тому за допомогою процедурних мов DML звичайно вказують, як можна одержати бажаний результат, тоді як непроцедурні мови DML використовуються для опису того, що слід одержати. Найбільш поширеним типом пепроцедурних мов DML є мова структурованих запитів (Structured Query Language- SQL), яка в даний час фактично є обов'язковою мовою для реляційних СУБД.
- Надає контрольований доступ до бази даних за допомогою засобів, що перераховуються нижче:

- Системи забезпечення безпеки, що попереджають несанкціонований доступ до БД з боку користувачів;
- Системи підтримки цілісності даних, яка забезпечує несуперечливий стан даних, що зберігаються;
- Системи управління паралельною роботою прикладних програм, яка котролює процеси їх сумісного доступу до БД;
- Системи відновлення, які дозволяють відновити БД після порушення, викликаного відмовою апаратного чи програмного забезпечення;
- Доступного для користувачів каталога, який містить опис інформації, що зберігається у БД [11].

Основні функції СУБД

- управління даними у зовнішній пам'яті (на дисках);

Ця функція включає забезпечення необхідних структур зовнішньої пам’яті як для зберігання безпосередньо даних, що належать до БД, так і для службових цілей, наприклад, для прискорення доступу до даних, журналів транзакцій тощо. Системи управлення БД підтримує власну систему іменування об’єктів.

- керування даними в оперативній пам'яті з використанням дискового кеша;

Системи управлення базами даних звичайно працюють з БД значного розміру; у будь-якому випадку цей розмір значно перевищує доступний об’єм оперативної пам’яті. Зрозуміло, якщо при зверненні до деякого елементу даних буде виконуватися обмін зі зовнішньою пам’яттю, тоді вся система буде працювати зі швидкістю пристрою зовнішньої пам’яті. Єдиним способом реально підвищити цю швидкість є буферизація даних в оперативній пам’яті. І навіть якщо операційна система виконує загальносистемну буферизацію (як у випадку ОС UNIX), цього недостатньо для цілей СУБД, яка володіє набагато більшою інформацією про корисність буферизації тої чи іншої частини БД. Існують окремі напрямки СУБД, які орієнтовані на постійну присутність в оперативній пам’яті всієї БД. Цей напрямок оснований на припущенні, що в майбутньому об’єм оперативної пам’яті комп’ютерів може бити настільки великим, що дозволить не турбуватися про буферизацію.

- журналізація змін, резервне копіювання і відновлення бази даних після збоїв;

Однією з основних вимог до СУБД є надійне зберігання даних у зовнішній пам’яті. Під надійністю зберігання розуміють те, що СУБД повинна бути в змозі відновити останній узгоджений стан БД після будь-якої апаратної або програмної відмови. В будь-якому випадку для відновлення БД потрібно володіти деякою додатковою інформацією. Інакше кажучи, забезпечення надійного збереження даних у БД вимагає надлишкового збереження даних, причому та їх частина, яка використовується для відновлення, повинна зберігатися особливо ретельно. Найбільш поширеним методом збереження такої надлишкової інформації – є ведення журналу змін БД [12].

Журнал - це особлива частина БД, недоступна користувачам СУБД і підтримується особливо ретельно (деколи підтримується дві копії журналу, які розташовуються на різних фізичних дисках), в яку потрапляють записи про всі зміни основної частини БД.

- Керування транзакціями

Транзакція - це послідовність операцій над БД, які розглядаються СУБД як єдине ціле. Будь-яку транзакцію, що успішно виконується, СУБД фіксує зміни у БД, у зовнішній пам’яті, або жодна з цих змін ніяк не відбивається на стані БД. Поняття транзакції необхідне для підтримки логічної цілісності бази даних.

При відповідному керуванні транзакціями, що паралельно виконуються, з боку СУБД кожен користувач може в принципі відчувати себе єдиним користувачем СУБД, якщо не зважати на деяку загальмованість роботи для кожного користувача.

Для керування транзакціями у багатокористувацьких СУБД використовується поняття серійності транзакцій і серійного плану виконання суміші транзакцій. Під серійністю транзакції, що виконуються паралельно, розуміють таку послідовність планування їх роботи, при якій сумарний ефект суміші транзакцій еквівалентний ефекту їх деякого послідовного виконання.

- підтримка мов БД (мова визначення даних, мова маніпулювання даними).

Зазвичай сучасна СУБД містить наступні компоненти:

- ядро, яке відповідає за управління даними у зовнішній і оперативної пам'яті, і журналізацію,
- процесор мови бази даних, що забезпечує оптимізацію запитів на вилучення та зміна даних і створення, як правило, машинно-незалежного виконуваного внутрішнього коду,
- підсистему підтримки часу виконання, яка інтерпретує програми маніпуляції даними, що створюють користувальницький інтерфейс із СУБД
- сервісні програми (зовнішні утиліти), що забезпечують ряд додаткових можливостей по обслуговуванню інформаційної системи.

За характером використання СУБД поділяються на:

- персональні (сукупність мовних і програмних засобів, які потрібні для створення й керування базами даних - VISUAL FOXPRO, ACCESS)
- загальні (використовують різні операційні системи і містять у собі сервер бази даних і клієнтську частину - ORACLE, INFORMIX).

Технологія баз даних використовується в безлічі додатків. Деякі з них призначені для єдиного користувача з єдиним комп'ютером, інші використовуються робочими групами в кількості 20-30 чоловік через локальну мережу, треті служать сотням користувачів і містять трильйони байтів даних. Останнім часом технологія баз даних застосовується в поєднанні з інтернет технологією для підтримки мультимедійних додатків у відкритих і закритих мережах.

Компонентами додатку бази даних є сама база даних, система управління базою даних і прикладні програми.

База даних - це само документовані збори інтегрованих записів. Вона є само документованою, оскільки містить опис самої себе в словнику даних. Словник даних відомий також як каталог даних, або метадані. База даних є зборами інтегрованих записів, оскільки зв'язки між записами також зберігаються в базі даних. Така організація дозволяє СУБД конструювати навіть складні об'єкти, комбінуючи дані на підставі зв'язків, що зберігаються.

В загальному випадку дані в системі баз даних є інтегрованими та розділювальними.

Під поняттям інтегрованості даних слід розуміти можливість представити базу даних як об’єднання декількох окремих файлів даних, в яких повністю, або частково виключає надлишковість інформації [13].

Під поняттям роздільністю даних розуміють можливість використання окремих елементів, що зберігаються у базі даних, декількома користувачами. Мається на увазі, що кожен бажаючий може одержати доступ до одного і того

самого елементу даних в той самий момент часу для досягнення різних цілей (паралельний доступ).

1.6 Огляд можливостей інтерфейсу системи управління базами даних

1.6.1 Призначення та класифікація систем управління базами даних

При створенні бази даних необхідно підготувати декілька файлів даних операційної системи, які будуть використовуватися разом як єдина база даних. База даних створюється один раз, незалежно від того, скільки файлів даних вона має, і скільки екземплярів будуть звертатися до неї. Процедуру створення бази даних можна також використовувати для того, щоб стерти інформацію в існуючій базі даних і створити нову базу даних з тим же ім'ям і фізичною структурою [15].

Створення бази даних у системі ORACLE включає наступні операції:

- створення нових файлів даних, або стирання даних, що зберігалися в попередніх файлах даних
- створення структур, потрібних ORACLE для доступу та роботи з базою даних (словника даних)
- створення і ініціалізація керуючих файлів і файлів журналу повторення для бази даних

Словник даних

Словник даних - централізований набір таблиць і уявлень, що використовуються в режимі "тільки читання" для отримання даних про БД. У словнику зберігається, наприклад:

- логічна і фізична структура БД;
- інформація про користувачів БД;
- обмеження цілісності;
- дані про виділений для об'єктів схем просторі і скільки з цього простору використовується.

Словник створюється, коли створюється база даних, і автоматично змінюється при зміні структур бази даних. Enterprise Manager отримує інформацію про об'єкти БД зі словника. Ви можете тільки виконувати запити інформації з таблиць словника даних. Enterprise Manager робить теж саме для вас і представляє інформацію у зручному для використання вигляді. Подання DICTIONARY містить опис таблиць і уявлень словника даних. В іменах уявлень зазвичай є один з трьох префіксів:

- USER- інформація, що відноситься до об'єктів, що належать користувачеві.
- ALL - інформація, що відноситься до об'єктів, доступним користувачеві.
- DBA - інформація про всі об'єкти бази даних.

1.6.2 Архітектура системи управлення базами даних Oracle

Система управління базами даних - це складна програмна система накопичення та з наступним маніпулюванням даними, що представляють інтерес для користувача. Кожній прикладній програмі СУБД надає інтерфейс з базою даних та має засоби безпосереднього доступу до неї. Таким чином, СУБД відіграє центральну роль в функціонуванні автоматизованого банку даних.

Архітектурно СУБД складається з двох великих компонент (рис. 8). За допомогою мови опису даних (МОД) створюються описи елементів, груп та записів даних, а також взаємозв'язки між ними, які, як правило, задаються у вигляді таблиць. В залежності від конкретної реалізації СУБД мову опису даних підрозділяють на мову опису схеми бази даних (МОС) та мову опису підсхем бази даних (МОП). Слід особливо зазначити, що МОД дозволяє створити не саму базу даних, а лише її опис.

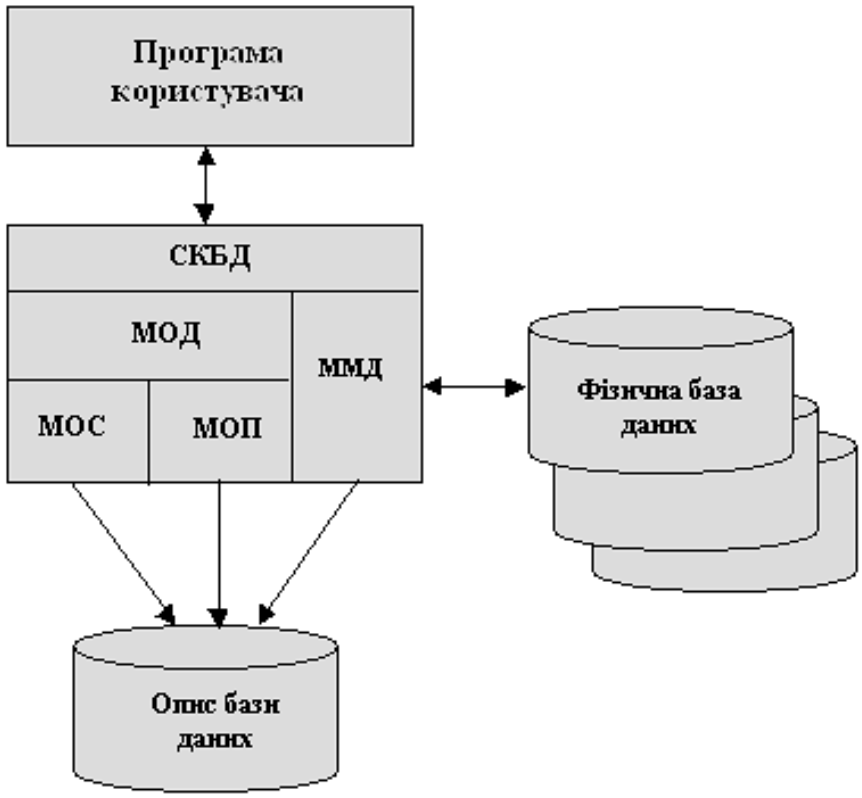


Рис. 1.4 Архітектура СУБД

Для виконання операцій з базою даних в прикладних програмах використовується мова маніпулювання даними (ММД). Фактична структура фізичного зберігання даних відома тільки СУБД.

З метою забезпечення зв'язків між програмами користувачів і СУБД (що особливо важливо при мультипрограмному режимі роботи операційної системи) в СУБД виділяють особливу складову - резидентний модуль системи керування базами даних. Цей модуль значно менший від всієї СУБД, тому на час функціонування автоматизованого банку інформації він може постійно знаходитись в основній пам'яті ЕОМ та забезпечувати взаємодію всіх складових СУБД і програм, які до неї звертаються.

Приведена структура притаманна усім СУБД, котрі розрізняються обмеженнями та можливостями по виконанню відповідних функцій. Отже, процес порівняння і оцінки таких систем для одного конкретного застосування зводиться до співставлення можливостей наявних СУБД з вимогами користувачів.

До недавнього часу при організації обробки інформації на ЕОМ застосовувався підхід, при якому на основі інформації одного і того ж об'єкту управління (наприклад, матеріальних ресурсів) в залежності від її вигляду і ступеню постійності формувались масиви лінійної структури двох типів: умовно-постійні (з інформацією, яка використовувалась багато разів протягом довгого часу) і умовно-перемінні (з фактичною або поточною інформацією). Створення і багаторазове використання масивів з умовно-постійною інформацією має ті переваги, які дозволяють значно спростити первинну документацію шляхом виведення з її складу ряд постійних реквізитів. знизити трудомісткість робіт на стадії заповнення первинних документів, підготовки і вроду фактичної або поточної інформації до ЕОМ. Недоліком таких масивів, які мають лінійну структуру, є то що інформація одного і того ж об'єкту управління розосереджується поміж багатьох різних масивів (нормативних, планових та ін.), що неминуче веде до дублювання деяких реквізитів, ускладненню при спільній їх обробці тощо, а головне - не дає змоги реалізувати принцип незалежності від прикладних програм користувача. Лінійні масиви, сформовані традиційним способом, ефективні, як правило, а позиції одного застосування.

З розвитком інформаційного забезпечення систем автоматизованої обробки інформації, прагненням забезпечити виконання нових режимів обробки даних у реальному часі і з мультидоступом до схованих даних позначилась нова тенденція до складення інформаційного забезпечення розподілених баз даних. В умовах використання таких баз створюються комплексні масиви нелінійної структури, які мають усі дані про ту чи іншу предметну область або про керований об'єкт як постійного, так і перемінного характеру.

Oracle проектувалася як максимально переносима СУБД, - вона доступна на всіх поширених платформах. Тому фізична архітектура Oracle різна в різних операційних системах. Наприклад, в ОС UNIX СУБД Oracle реалізована у вигляді декількох окремих процесів операційної системи - практично кожна

суттєва функція реалізована окремим процесом. Для UNIX така реалізація підходить, оскільки основою багатозадачності в ній є процес. Для Windows, однак, подібна реалізація не підходить і працювала б не дуже добре (система вийшла б повільною і погано масштабується). На цій платформі СУБД Oracle реалізована як один багатопоточний процес, тобто з використанням відповідних для цієї платформи механізмів реалізації. На мейнфреймах IBM, що працюють під управлінням OS / 390 і zOS, СУБД Oracle використовує кілька адресних просторів OS / 390, спільно утворюють екземпляр Oracle. Для одного примірника бази даних можна конфігурувати до 255 адресних просторів. Більш того, СУБД Oracle взаємодіє з диспетчером завантаження OS / 390 WorkLoad Manager (WLM) для установки пріоритетності виконання певних компонентів Oracle по відношенню один до одного і до інших завдань, які працюють в системі OS / 390. В ОС Netware теж використовується многопоточная модель. Хоча фізичні засоби реалізації СУБД Oracle на різних платформах можуть відрізнятися, архітектура системи - досить загальна, щоб можна було зрозуміти, як СУБД Oracle працює на всіх платформах.

Архітектура Oracle складається з:

- файлів БД
- процесів
- області оперативної пам'яті

База даних Oracle містить наступні види файлів:

- Контрольні файли (Control files) - містять метадані про самій базі даних. Ці файли дуже важливі для бази даних. Без них не можуть бути відкриті файли даних і тому не може бути відкритий доступ до інформації бази даних. Містять керуючу інформацію про всі файлах бази даних; підтримують внутрішню цілісність бази даних; керують операціями відновлення; зазвичай зберігаються на різних дисках, щоб звести до мінімуму їх можливе пошкодження при збої диска.

- Файли даних (Data files) - містять інформацію бази даних. База даних розділена на логічні структурні одиниці, звані табличними просторами. Вони використовуються для об'єднання збережених в них логічно пов'язаних структур. Кожна база даних містить одне або кілька табличних просторів. Для зберігання інформації, що міститься в логічних структурах табличного простору, створюється один або декілька файлів даних.
- Оперативні журнали (оперативні файли повторного виконання) - дозволяють відновити базу даних після збою екземпляра. Коли робота бази даних завершується аварійно і при цьому не втрачаються ніякі файли даних, екземпляр може відновити базу даних на основі інформації в цих файлах. Файли журналів повтору (журнали транзакцій) містять відомості про виконання транзакцій; використовуються для відновлення транзакцій бази даних в належному порядку у разі збою БД; збереження інформації журналів повтору є зовнішнім по відношенню до файлів даних; надають Oracle спосіб запису даних на диск [16].

1.6.3 Основа організації БД

Створення бази даних вручну включає в себе кілька кроків. Деякі з них залежать від операційної системи. Наприклад, в середовищі Windows, перш ніж створювати базу даних, спочатку необхідно виконати Oracle-програму, використовувану для створення служби бази даних. Кроки по створенню бази даних вручну:

1. Написаты сценарій створення бази даних. Зразок такого сценарію наведено на кроці 6.
2. Створіты структуру каталогів, в яких буде розміщуватися нова база даних. Дотримуйтесь інструкцій по створенню оптимальної гнучкою архітектури.
3. Змініть існуючий зразок файлу init.ora, підтримуваний Oracle, щоб у ньому відбивалися параметри для нової бази даних.

4. Описаты SID-ім'я для Oracle. На платформі Windows на запрошення операційної системи необхідно ввести:

   set ORACLE_SID = mydb

   В UNIX:

   export ORACLE_SID = mydb

5. Встановлюется з'єднання з базою даних через SQL * Plus як SYSTEM / MANAGER as sysdba або як / as sysdba і введіть наступну команду запуску бази даних в режимі nomount:

   startup nomount pfile = D:/oracleadmin/mydbscripts/initMYDB.ora;

   Підставте свої параметри ініціалізації замість наведених тут значень параметрів pfile.

6. Після запуску бази даних використовуйте написаний сценарій створення бази даних. Ось зразок:

   create database MYNEW maxinstances 1 maxloghistory 1 maxlogfiles 5 maxlogmembers 5 maxdatafiles 100

   datafile d: /oracle/oradata/mydb/system01.dbf size 325M reuse autoextend on next 10240K maxsize unlimited

   character set WE8MSWIN1252 national character set AL16UTF16

   Logfile group 1 (d: /oracle/oradata/mydb/edo01.log) size 100M, group 2 (d: /oracle/oradata/mydb/edo02.log) size 100M, group 3 (d: / oracle / oradata / mydb /edo03.log) size 100M

   default temporary tablespace TEMP tempfile d: /oracle/oradata/mydbemp01.dbf

   extent management local uniform size 1M undo tablespace UND0_TS datafile d oracleoradatamydb emp01 dbf size 150M reuse autextend on next 10240K maxsize unlimited;

7. Після створення бази даних виконайте сценарії catalog.sql, catproc.sql, catexp.sqli все нові сценарії, необхідні для підтримки встановлених вами продуктів. В системі UNIX сценарії розміщуються в каталозі $ ORACLE_HOME \ rdbms \ admin, а в середовищі Windows- в $ ORACLE_HOME

/ rdbms / admin. Перш ніж виконувати сценарії, перегляньте їх, оскільки багато сценарії каталогу викликають інші сценарії.

8. Для забезпечення підвищеної безпеки введіть, як мінімум, якісь інші паролі для SYS і SYSTEM, а не залишаєте паролі за замовчуванням MANAGER і CHANGE_ON_INSTALL.

У прикладі сценарію, наведеного на кроці 6, створюється табличний простір UNDO. Параметрами ініціалізації для нього є:

undo_management = AUTO undo_tablespce = UNDOTBS

Єдиний параметр, який ви не можете змінити після створення бази даних, це розмір блоку бази даних, який був описаний вами у файлі init.ora до її створення. Для завдання цього значення використовується параметр DB_BLOCK_SIZE. Наприклад, в наступному рядку задається розмір блоку бази даних за замовчуванням, який становить 8 Кбайт.

DB_BL0CK_SIZE = 8k

Для того щоб побачити параметри, що діють у вашій базі даних, запросите динамічний перегляд V $ PARAMETER:

select Name, Value, IsDefault from V $ PARAMETER;

Засоби для автоматичного створення БД

- Oracle Database 11g R1

Для створення бази даних використовується утиліта Database Configuration Assistant (dbca). Вона, як і більшість утиліт розташована в каталозі bin сервера Oracle. В консолі виконуємо:

oracle @ test: cd /u01/app/oracle/product/11.1.0/db_1/bin

oracle @ test: ./ dbca

Відбудеться запуск графічного додатку, в якому і належить працювати. Oracle дозволяє створювати базу даних "руками", без використання різних утиліт, але про це поговоримо пізніше.

- Oracle SQL Developer

Це безкоштовна графічне середовище управління базами даних і розробки додатків на мовах програмування SQL і PL / SQL, розроблена спеціально для СУБД Oracle Database.

Дане середовище написана мовою програмування Java і вона працює на всіх платформах де є Java SE.

SQL Developer, дозволяє переглядати об'єкти бази даних, запускати різні SQL інструкції, створювати і редагувати об'єкти бази даних, імпортувати і експортувати дані, а також створювати всілякі звіти.

оригінальний і надійний інструмент для розробки програмного забезпечення доступу до баз даних. Він об'єднує можливості додатків клієнт / сервер з перевагами об'єктно-орієнтованої моделі розробки.

Для побудови програми розробник повинен мати у своєму розпорядженні деякі конструктивні елементи - об'єкти.

Це безкоштовна, менша за обсягом версія Oracle Database. Oracle Database XE проста в установці і управлінні.

Для роботи з Oracle Database XE використовується інтуїтивно-зрозумілий web-інтерфейс, що дозволяє:

- Адмініструвати базу даних
- Створювати таблиці, подання та інші об'єкти бази даних
- Імпортувати, експортувати і переглядати табличні дані
- Виконувати запити і запускати SQL-скрипти генерувати звіти
- Вибір та аналіз специфіки предметної області

«Огляд можливостей інтерфейсу системи управління базами даних»

В даному розділі розглянуто основні можливості СУБД та встановлено її переваги над іншими системами управління базами даних. До них відносяться:

- Продуктивність СУДБ, що дозволяє автоматично управляти рівнями сервісу і тиражувати еталонні конфігурації в рамках всієї мережі.
- Прості засоби розробки, що надає можливість простому користувачеві створювати ефективні програми для роботи з базами даних в короткі терміни та без зайвих затрат часу.
- Спеціальні механізми дозволяють самостійно перерозподіляти навантаження на систему, оптимізувати і коректувати SQL-запити, виявляти і прогнозувати помилки, що можуть бути допущеними користувачами реляційної бази даних.
- Можливість створення баз даних великих об’ємів. На сьогодні, максимальний розмір одного екземпляра бази даних Oracle може досягати 8 екзабайт, що рівно 10^{18} байт.

Постановка задачі

Завдання бакалаврської роботи полягає у розробці програмного інтерфейсу для створення універсальних звітів на підприємстві на прикладі БД.

- використовуючи мову Oracle реалізувати фізичний проект - створити базу даних, яка містить інформацію про компанію та її базу даних;
- наповнити створену базу даних таблицями, в яких зберігається інформація щодо витрат та прибутків підприємства;
- під′єднати базу до середовища програмування Microsoft Visual Studio;
- створити програмний інтерфейс..
- розробити можливість створення звіту та його конвертування у файли типу зображень.

ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМИ ОСОБЛИВОСТІ КОНТРОЛЬНО-ПРОПУСКНОГО ПУНКТУНА ПРИКЛАДІ БАЗИ ДАНИХ

Висновки до розділу «Поняття баз даних»

Впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів називаю базами даних.

У розділі «Поняття баз даних» було розглянуто основні поняття про бази даних та управління ними. Визначено також головне завдання баз даних – гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі.

З метою забезпечення ефективності доступу записи даних організовують як множину фактів – елементів даних.

Висновки до розділу до розділу

Розділ 2 Проектні рішення

2.1Вибір програмного забезпечення для розробки

Щодо мови програмування, обрано мову Microsoft Visual Studio 2010, яка надає можливість створення, редагування, запису та зчитування даних із створеної бази даних.

У складі Visual Studio, нажаль, нема готової програми-клієнта для роботи з базами даних.Проте, програмне забезпечення дозволяє підключитися до попередньо створеної БД та проводити різноманітні маніпуляції над нею.

Розподілена обробка інформації та обробка запитів від декількох клієнтів зумовила вибір в якості архітектури розробленого додатка клієнт-серверну архітектуру. Вона дозволила реалізувати розподілений веб-додаток для браузеру з незалежним доступом до загальних ресурсів реляційної бази даних. Також були розглянуті основні сценарії, призначені для користувача та доступні системи. Створена модель даних предметної області і на її основі розроблена логічна схема бази даних.

Отримана на даному етапі проектна документація слугувала основною розробки програмного коду системи. Сутність впровадження автоматизованої системи обліку робочого часу полягає в усуненні нагальних проблем, що виникають під час ручного виконання більшості рутинних процесів, людського фактору, впровадження неефективних інструментів контролю робочого часу та оптимізації бізнес-процесів компанії в цілому. Також у процесі Висновки до розділу «Поняття баз даних»ідження були розглянуті різні підходи до автоматизації робочого часу співробітників із впровадженням систем різного типу, проаналізовано системи-аналоги, виявлені їх недоліки, над якими необхідно було попрацювати для створення більш інноваційної та зручної системи для автоматизації обліку та аудиту робочого часу співробітників на підприємстві. Задоволені основні вимоги, що пред'являються до системи обліку часу, розробленої в рамках МНР: − використання реляційної бази даних, для

зберігання інформації про робочий час співробітників; 16 − облік і аналіз інформації з різних джерел: облік роботи комп'ютера співробітника, облік даних, а також інформації, наданої самим співробітником (інформація про відпустку, лікарняному і інші причини відсутності на робочому місці); − розробка механізмів вирішення суперечностей даних про робочий час; − розробка користувальницького інтерфейсу, що дозволяє працювати з інформацією з бази даних. Для досягнення вказаної мети роботи виконано наступні завдання: − проведено аналіз існуючих алгоритмів та методів обліку робочого часу працівників; − запропоновано використання модифікованого алгоритму обліку робочого часу працівників, який би враховував та усував недоліки роботи його існуючих аналогів; − розроблено програмне забезпечення, яке дозволить проводить контроль робочого часу працівників та підвищить ефективність їх роботи; − досліджена робота системи обліку та аудиту робочого часу працівників на прикладі компанії, що здійснює реальну професійну діяльність в IT-сфері; − оцінено рентабельність впровадження системи. Як результат бакалаврської роботи отримано автоматизовану систему обліку робочого часу на основі сучасних технологічних рішень і відповідно до сформульованих вимог щодо функціонального складу подібних систем. У вигляді спроектованого програмного забезпечення представлено веб-застосунок, вирішальне завдання якого полягає в автоматизації обліку робочого часу співробітників організацій, установ і підприємств.

2.2 Розробка методів та алгоритмів генерації звітів

Моделювання - це добре вивчена та широко вживана інженерна методика. Уніфікована мова моделювання (UML) є стандартним інструментом для створення моделей програмного забезпечення. За допомогою UML можна візуалізувати, специфікувати, конструювати й документувати артефакти програмних систем.

UML придатна для моделювання будь-яких систем: від інформаційних систем масштабу підприємства до розподілених Web-додатків і навіть убудованих систем реального часу.

У інфраструктурі UML важливе місце займають засоби проектування і реалізації. До них відносяться:

- мови програмування.

UML – мова візуального моделювання, яка непризначена для візуального програмування, хоча інструментальний засіб, що підтримує нотації UML, може реалізовувати кодогенерацію у конкретну мову програмування (C++, Delphi, Java та ін.) на основі побудованих моделей;

- інструментальні засоби.

UML не є специфікацією для розробки інструментальних середовищ. В UML визначена тільки модель семантики, але не визначені моделі інтерфейсу, зберігання і підтримки часу виконання. Зараз вже є декілька CASE – засобів, виробники котрих підтримують моделювання засобами UML. Серед них можна виділити: Rational Rose, Select Enterprise, Platinum і Visual Modeler.

- моделі процесу.

Деякі підприємства використовують UML як єдину мову при розробці усіх компонентів проекту, хоч конкретні етапи і сам процес розробки можуть бути реалізовані по-різному. При цьому спостерігається збіжність моделей процесів для різних організацій, але загальної згоди по цьому питанню поки ще не знайдено. UML не нав'язує свою модель процесу,але при розробці мови автори мали на увазі процес, що має наступні властивості: той, що керується прецедентами, ітеративний та інкрементний.

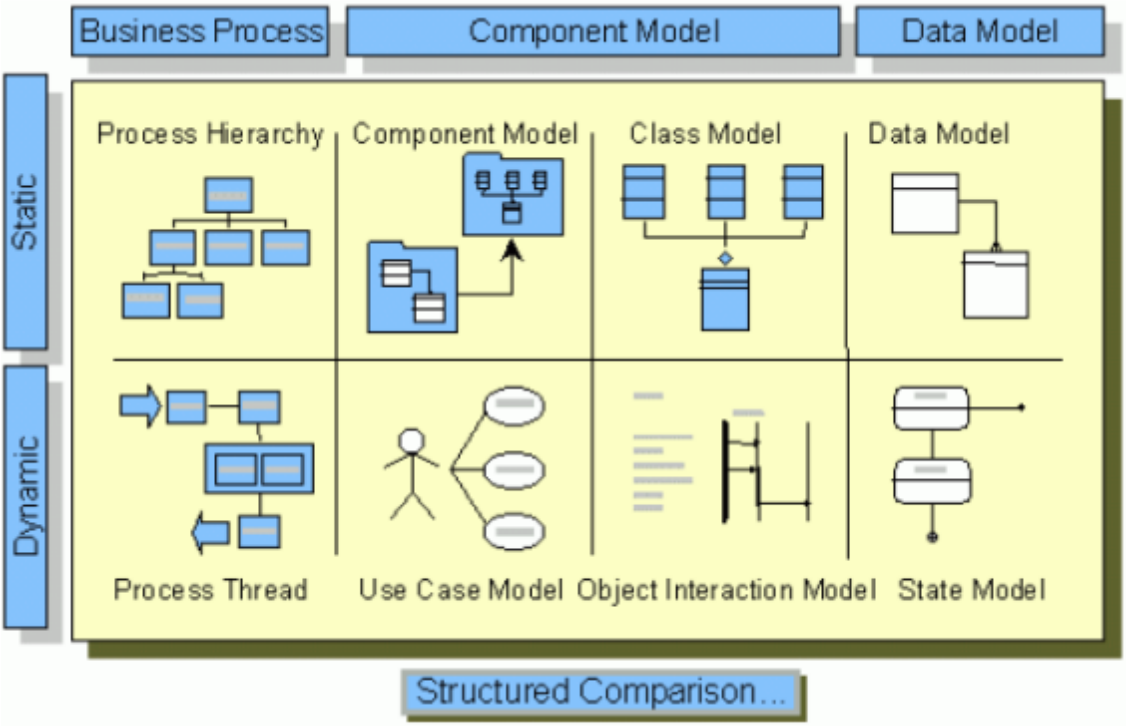


Рис. 2.1 Застосування моделей UML при розробці бізнес додатків за допомогою CASE-засобів системи фірми Select Enterprise

Наступна діаграма являє собою короткий опис функцій, з допомогою яких планується програма створення універсальних звітів.

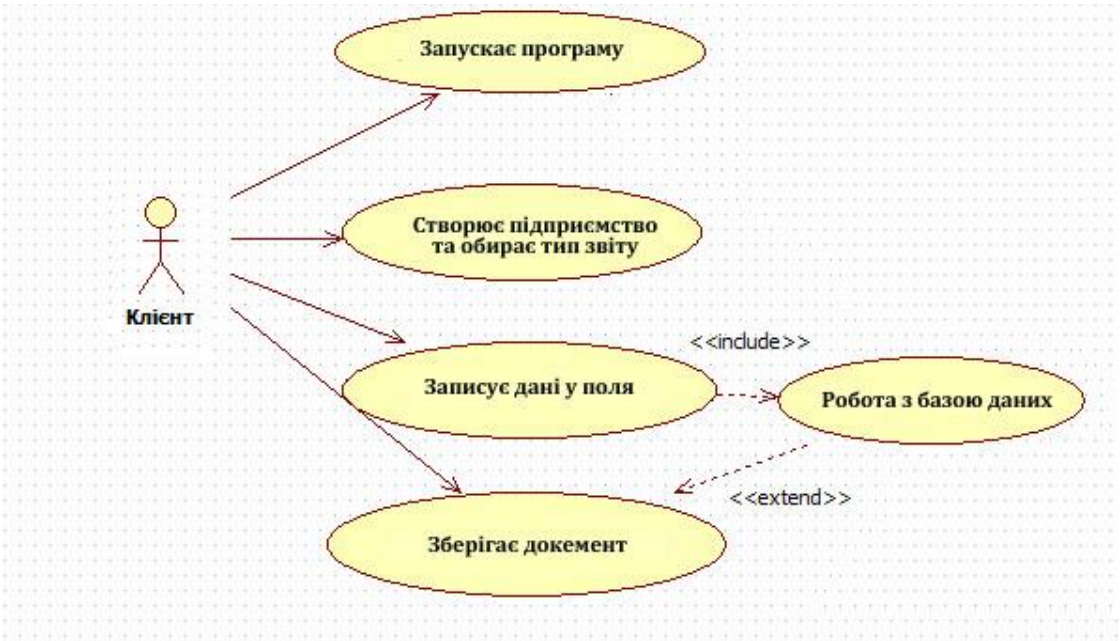


Рис. 2.2 Візуальне представлення графу діяльностей (діаграма блоків-дій)

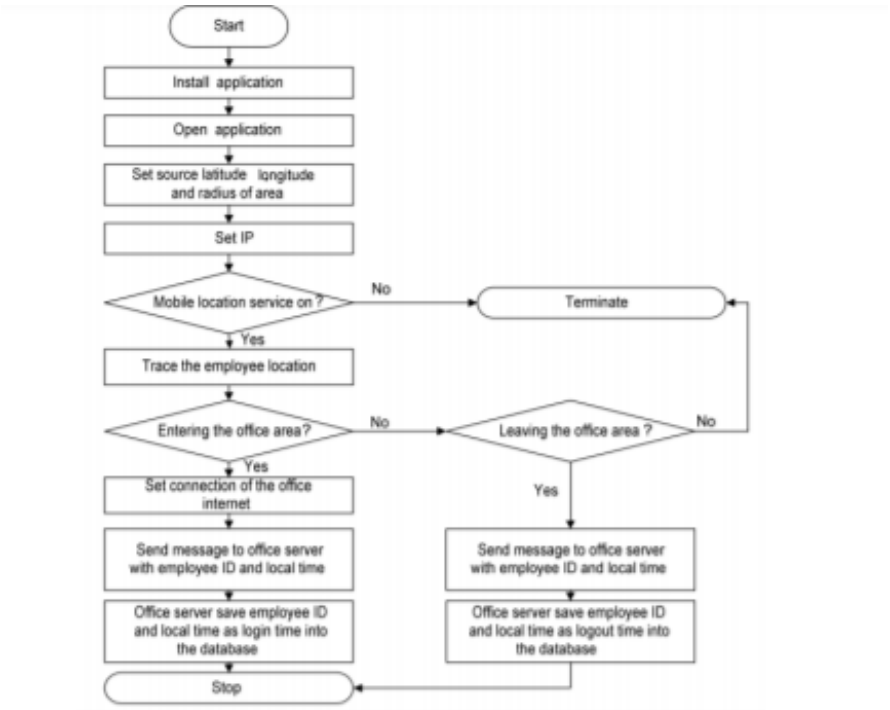
На діаграмі зображено як відбувається процес створення звіту; взаємозв’язок між актором та системою, його діями та можливостями. Відношення асоціації (*directed association* →) говорить про те, що один клас пов'язаний з іншим, тобто зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного). Включення (*include*) — це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. Відношення розширення (*extend*) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Порядок розробки програмного забезпечення можна також представити в наступному узагальненому алгоритмі.

2.3 Фізичне проектування бази даних

2.3.1 Створення бази даних

Перевага подібних систем в їх простоті і малому обсязі даних для зберігання і передачі. Оскільки конфіденційна інформація не передається, то керівнику можна не боятися за її збереження. Однак не можна не відзначити цілий набір мінусів. Поперше, такий функціонал реалізований в системах першої і другої. По-друге, подібного роду програми не приносять користі для служби безпеки та IT, оскільки такі системи не можуть дати відповідь, що саме робила людина в тій чи іншій програмі або на сайті. І, по-третє, ці програми дуже просто обдурити, симулюючи активність.



Назва системи	Гнучкий графік	Заяви на відпустку	User-friendly інтерфейс	Доступ до статистики
Staff Cop	+/-	-	+	+/-
Manic Time	+/-	-	+/-	+/-
Skype Time	+	-	+/-	+
Спроектована система	+	+	+	+

Проаналізувавши порівняльну таблицю, можна прийти до висновку, що розроблена система найбільшою мірою задовольняє вимогам, що пред'являються до неї. Одними з головних функціональних вимог були можливість виставляти гнучкий графік присутності на робочому місці, а також самостійного відзначення співробітниками своєї відсутності на роботі. Розглянуті системи в більшості не пропонують необхідного функціоналу. Варто зазначити, що необхідно відповідально підходити до вибору системи обліку часу, так як при загальній схожості, кожна має свої унікальні якості, які в певних ситуаціях можуть виступати як плюсами, так і мінусами. Концептуальною засадою розробки інтерфейсу користувача є визначення того,

що клієнтський додаток не повинен бути ресурсоємким, складним у використанні. Для розробки інтерфейсу користувача одним з найбільш поширених підходів, які використовують при розробці клієнтських програм для взаємодії з реляційними базами даних під ОС Windows, є використання інструментарію Visual Studio корпорації Microsoft

2.3.2 Створення таблиць у базі даних

Є декілька способів підключення до БД в додатку. Найпростіше це зробити за допомогою графічних інструментів Visual Studio .NET під час розробки. Для управління поточними з'єднаннями з джерелами даних служить вікно Server Explorer. Воно зазвичай розташоване біля лівої межі вікна IDE поруч з панеллю Toolbox.

У Visual Studio .NET є багато вбудованих майстрів і дизайнерів, які допоможуть швидко і ефективно створити архітектуру доступу даним під час розробки і оснастити додаток надійним механізмом доступу даним, витративши мінімум зусиль на написання коди. Разом з цим всі можливості об'єктної моделі ADO.NET доступні програмно, що дозволяє реалізувати нестандартні функції або створювати застосування, орієнтовані на потреби користувача. Ми дізнаємося, як за допомогою ADO.NET підключитися до БД, витягувати з неї дані і передати їх застосуванню. Навчимося робити це як за допомогою інструментів з графічним інтерфейсом з Visual Studio .NET, так і з використанням прямого програмного доступу.

2.3.3 Підключення до сервера

Поточні з'єднання з джерелами даних, доступні в Visual Studio .NET, відображаються у вікні Server Explorer у вигляді вузлів дерева *Data Connections*. Щоб додати до проекту з'єднання необхідно:

1. Перейти за пунктом меню *Data →Add new data sourse*

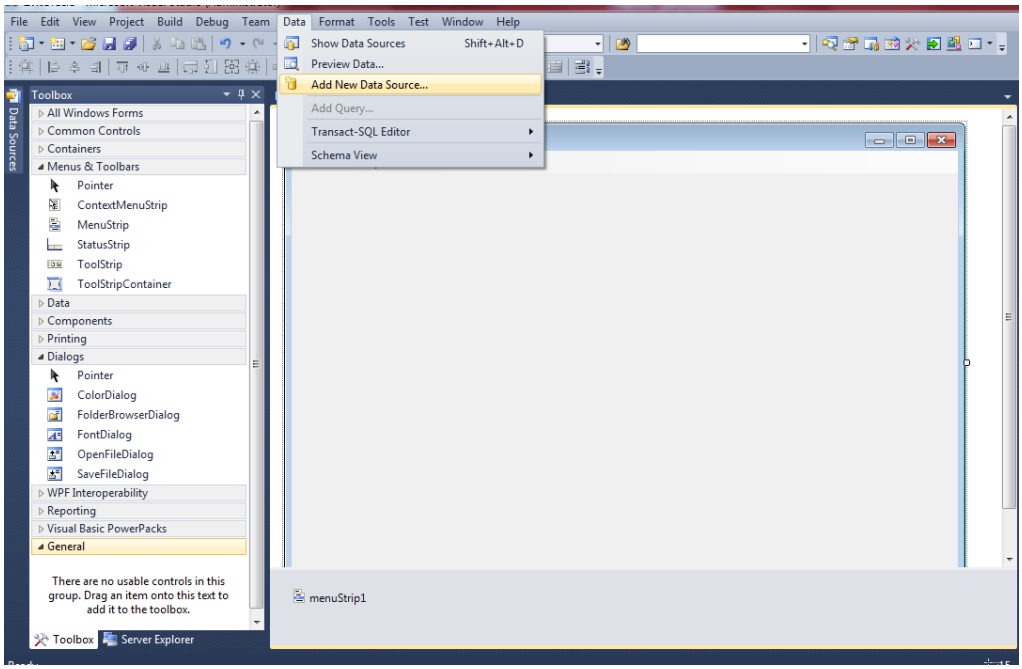
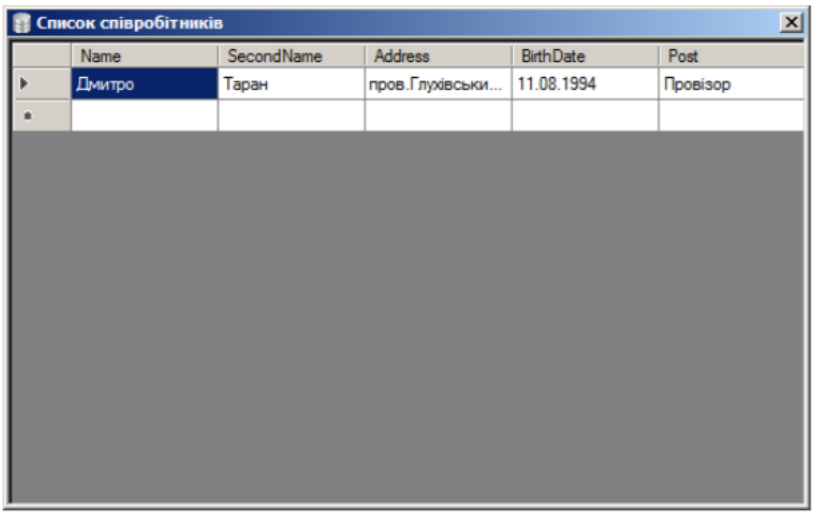


Рис. 1 Підключення нового ресурсу даних до проекту в Visual Studio 2010



При цьому відкривається діалогове вікно Data Link Properties, що надає графічний інтерфейс для налаштування з'єднань з джерелом даних.

Перевіряємо вірність введених даних – Test Connection та, вразі відсутності помилок, тиснемо кнопку Ok.

У випадку коректного створення та під’єднання ресурсів бази даних Oracle розробник має можливість переглянути наявні ресурси БД через вікно Server Explorer у вигляді дерева.

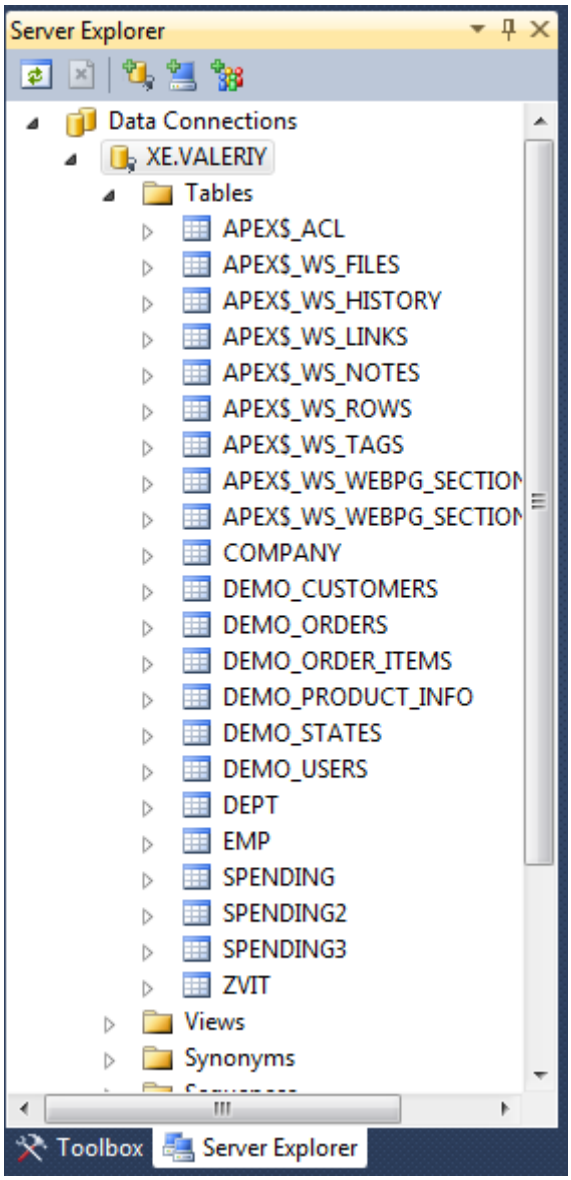
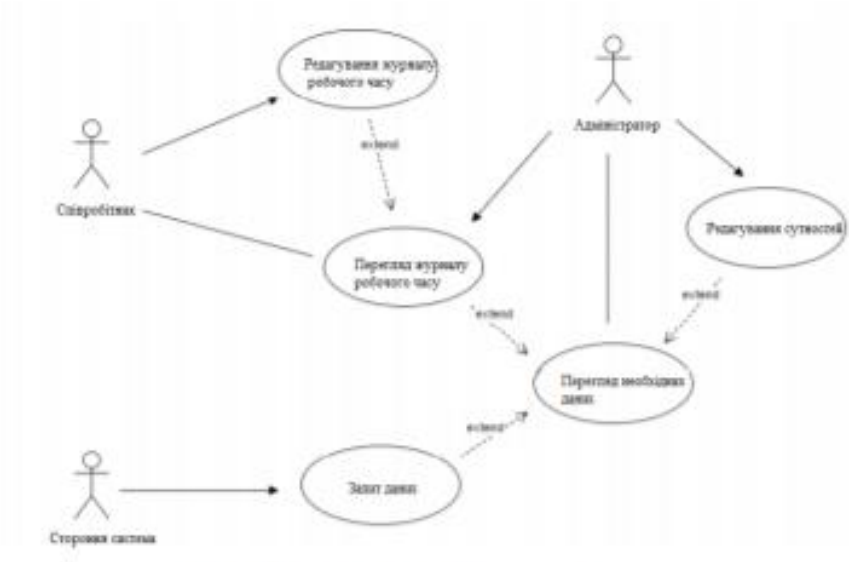


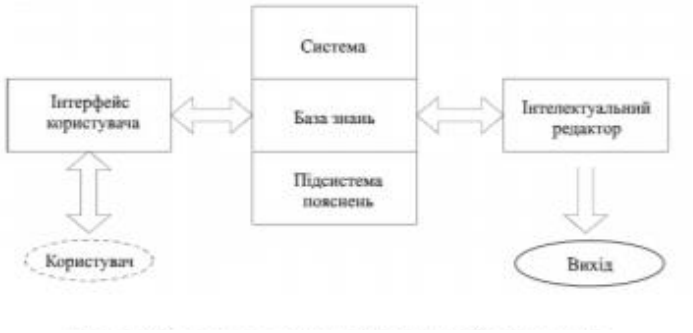
Рис. 2.4 Ієрархія під’єднаної БД до проекту Visual Studio

Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи і керують усім процесом розробки. Всі основні види діяльності, такі як аналіз, проектування, тестування виконуються на основі варіантів використання. Під час аналізу і проектування варіанти використання дозволяють зрозуміти як результати, які хоче отримати користувач впливають на архітектуру системи і як повинні поводитися компоненти системи, для того щоб реалізувати потрібну для користувача функціональність. Діаграма варіантів використання складається з акторів, для яких система виробляє дію і власне дії Use Case, яке описує те, що актор хоче отримати від системи. Відповіді на такі питання дозволять визначити акторів, що взаємодіють з системою: − хто взаємодіє з системою або використовує систему; 9 − хто передає чи приймає інформацію в / із системи; − хто є зовнішнім по відношенню до системи. Кожен варіант використання показує, як конкретний актор використовує систему і надалі розширюється діаграмами станів і послідовності дій.



Функціонування ІС представлено на рисунку 1.3. Користувач, який бажає отримати необхідну інформацію, через призначений для користувача інтерфейс надсилає запит до системі. Система, користуючись базою знань, генерує і видає користувачеві відповідну рекомендацію, пояснюючи хід своїх міркувань за

допомогою підсистеми пояснень. У свою чергу інженер знань та експерт, володіючи знаннями через інтелектуальний редактор, формує базу знань ІС.

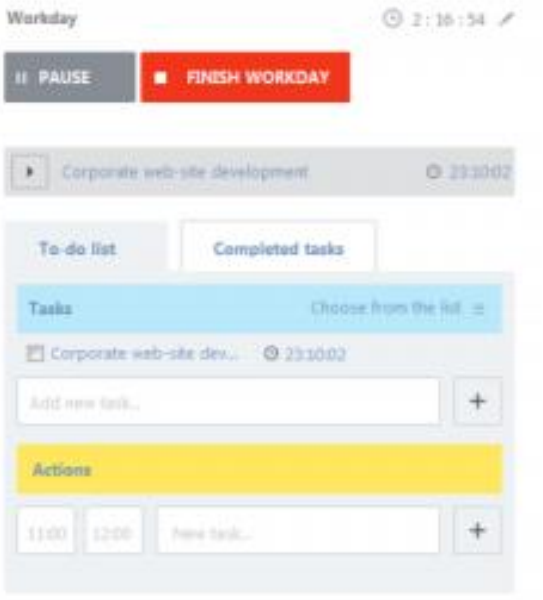


Процес проектування користувальницького інтерфейсу складається з п'яти ітераційних кроків, проте вони не є строго концептуальними у фіксованій послідовності. Тепер коротко розглянемо кожен з цих кроків. На першому етапі ми визначаємо, проектуємо та оцінюємо відповідні діаграми потоків даних (DFD). Далі ми визначаємо і призначаємо їм сценарії використання, у співпраці з кінцевими користувачами.

Результатом другого етапу є шаблонні структури інтерфейсу користувача (UI), візуалізовані у вигляді діаграм, які визначають його первинні компоненти та взаємозв'язки, що відбуваються між ними. На третьому етапі ми визначаємо стандарти інтерфейсу користувача в трьох різних категоріях, таких як: навігація, структурні шаблони, звіти та документація. Виконання четвертого етапу демонструє прототип UI – візуальне поєднання елементів, попередньо визначених і спроектованих. Нарешті, проводиться оцінка ергономіки UI і зручності користування, і при оцінці фактів: − дозволяє затверджувати інтерфейс користувача, після чого команда розробників може реалізувати та підключити функціональність системи до відповідних компонентів інтерфейсу або 11 − не дозволяє затвердити UI, після чого його додаткові модифікації повинні бути зроблені для уточнення даного прототипу, перевірки вимог користувачів, а при необхідності - перегляду продуктів перших трьох кроків. Наступний рисунок графічно зображує ідею, яка лежить в основі обраної моделі прототипування. Завжди є модератор, людина, що відіграє центральну роль у успішному розвитку інтерфейсу користувача. Це користувач, який

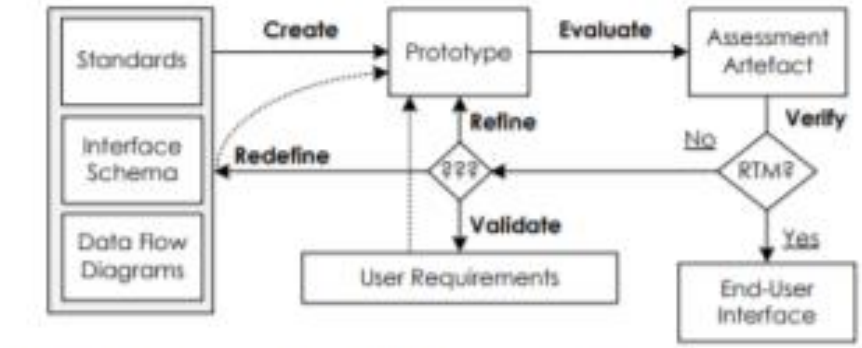
встановлює контекст, веде дискусію в правильному напрямку і залучає учасників до інтерактивного діалогу щодо прототипування, якщо в процесі приймає участь група користувачів (тестувальників) або окремо обраний користувач.

Процес прототипування інтерфейсу користувача передбачає певну внутрішню залежність: для реалізації функціоналу системи інтерфейс користувача повинен бути повністю прийнятий учасниками проекту. Слід зазначити, що це твердження має своє походження та обґрунтування в сильному впливі шляхів потоків даних по всьому інтерфейсу, особливо помітним і об'єктивно підтверджуваним у всіх компонентах, пов'язаних з даними.



Основним інтерфейсом для роботи користувача з системою обліку робочого часу буде web-інтерфейс. При розробці інтерфейсу використано компонентний підхід – сторінка управління додатком розбита на кілька незалежних функціональних компонентів. Скориставшись раніше зазначеними принципами про проектування інтерфейсу користувача та наявними технічними знаннями, було створено головну 12 сторінку (рис. 1.5.) системи обліку робочого часу, на якій користувач може виконувати певні дії, сторінку

адміністратора (керівника проектів) з можливістю відслідковування активності співробітників, формування звітів, що представлена на рис. 1.5.



Всі основні види діяльності, такі як аналіз, проектування, тестування виконуються на основі варіантів використання. Під час аналізу і проектування варіанти використання дозволяють зрозуміти як результати, які хоче отримати користувач впливають на архітектуру системи і як повинні поводитися компоненти системи, для того щоб реалізувати потрібну для користувача функціональність.

2.4 Створення програми в MS Visual Studio

2.4.1 Створення форм. Їх призначення

Для проектування форми Створення універсальних звітів було використано такі компоненти, як:

- *menuStrip*

Компонента *menuStrip* використовується для створення користувацького меню. При розробці такого меню було створено

- Файл
  - Створити

Використовується для створення нового звіту з попередньо вибраним типом.

Діаграма варіантів використання складається з акторів, для яких система виробляє дію і власне дії Use Case, яке описує те, що актор хоче отримати від системи. Відповіді на такі питання дозволять визначити акторів, що взаємодіють з системою: − хто взаємодіє з системою або використовує систему; 9 − хто передає чи приймає інформацію в / із системи; − хто є зовнішнім по відношенню до системи. Кожен варіант використання показує, як конкретний актор використовує систему і надалі розширюється діаграмами станів і послідовності дій. Проаналізувавши порівняльну таблицю, можна прийти до висновку, що розроблена система найбільшою мірою задовольняє вимогам, що пред'являються до неї.

Одними з головних функціональних вимог були можливість виставляти гнучкий графік присутності на робочому місці, а також самостійного відзначення співробітниками своєї відсутності на роботі. Розглянуті системи в більшості не пропонують необхідного функціоналу. Варто зазначити, що необхідно відповідально підходити до вибору системи обліку часу, так як при загальній схожості, кожна має свої унікальні якості, які в певних ситуаціях можуть виступати як плюсами, так і мінусами. Перед прийняттям рішення про впровадження такої системи необхідно провести повний аналіз існуючих процесів в компанії, зрозуміти, які джерела даних можуть бути використані для обліку робочого часу співробітника.

Для того, щоб більш точно зрозуміти як повинна працювати система, використовується опис функціональності системи через варіанти використання (Use Case або прецеденти). Варіанти використання – це опис послідовності дій, які може здійснювати система у відповідь на зовнішні впливи користувачів або інших програмних систем. Варіанти використання відображають функціональність системи з точки зору отримання відчутного результату для користувача, тому вони точніше дозволяють ранжувати функції за значимістю одержуваного результату. Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи і керують усім процесом розробки.

2.4.2 Огляд створеної програми

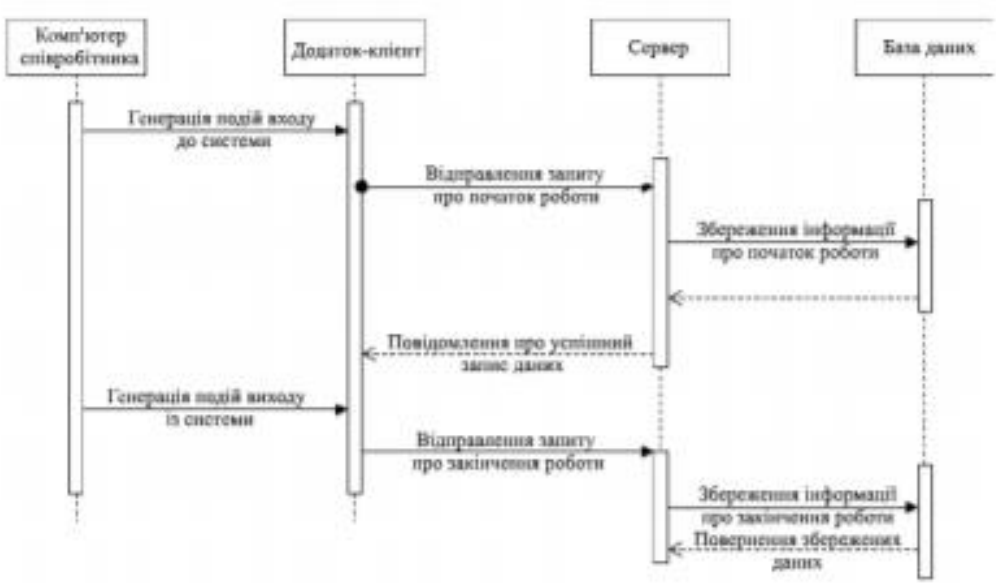
Система обліку і контролю робочого часу призначена для швидкої реєстрації часу приходу і відходу співробітників з роботи. Ця система забезпечує керівництво підприємства і його окремих підрозділів оперативною інформацією про відсутніх або залишили робоче місце співробітників. Система веде облік загальної кількості відпрацьованих годин кожним співробітником, проводить облік відряджень, відгулів, лікарняних, відпусток. У будь-який момент система дозволяє отримувати звіти по всіх перерахованих параметрах і складати табель робочого часу.

Системи контролю співробітників діляться на три основних типи. Перші – роблять основну ставку на записи дій персоналу за комп'ютером на відео, онлайнспостереження і контроль порушень. Безумовним плюсом такого підходу є той факт, що від відео сховатися неможливо. Мінус – для зберігання відео необхідна достатнє місце на диску. Інші – збирають максимальний обсяг даних про дії користувача (листи, файли, повідомлення) і пропонують використовувати для їх аналізу звіти. Плюс такого походу в тому, що шляхом пошуку за отриманими і відправленим листам, повідомленнями можна знайти можливі порушення співробітника. Мінус – дуже важко проаналізувати

великий обсяг інформації. Третя група (системи УРВ) зберігають мінімум даних: тільки відвідані сайти і запущені програми. Такого роду програми ділять активність на продуктивну, непродуктивну, нейтральну і формують звіти, в яких роботодавець зможе побачити, на що витрачали час його підлеглі.

Перед прийняттям рішення про впровадження такої системи необхідно провести повний аналіз існуючих процесів в компанії, зрозуміти, які джерела даних можуть бути використані для обліку робочого часу співробітника. Для того, щоб більш точно зрозуміти як повинна працювати система, використовується опис функціональності системи через варіанти використання (Use Case або прецеденти). Варіанти використання – це опис послідовності дій, які може здійснювати система у відповідь на зовнішні впливи користувачів або інших програмних систем. Варіанти використання відображають функціональність системи з точки зору отримання відчутного результату для користувача, тому вони точніше дозволяють ранжувати функції за значимістю одержуваного результату.

Основним завданням програми клієнта є обробка подій та відправлення їх на сервер. До подій відносять події відкриття тих чи інших ресурсів, а також події обробки, збереження та видалення інформації, а також зміна ім'я користувача. Дані події є тригерами, які говорять про час початку і закінчення роботи співробітника. Схема взаємодії клієнта і сервера показана на діаграмі на рисунку 1.8.



У момент входу до системи відбувається автозавантаження програми-клієнта. При старті програми вона відправляє на сервер запит про те, що співробітник почав роботу із системою. Сервер відправляє запит про збереження інформації про початок роботи в базу даних. Після успішного збереження, база даних повертає інформацію про збережену сутність сервера, який, в свою чергу, повертає клієнту інформацію про успішну операцію збереження даних. При виході із системи програма-клієнт відправляє повідомлення про це серверу, який відправляє запит до бази даних про збереження часу закінчення роботи співробітника.

Висновки

У дипломній роботі розглянуто класичний підхід до створення програмних засобів для роботи з базами даних. На теперішній час існує велике різноманіття технологій для створення програмних додатків, в тому числі і тих, що тісно взаємодіють з системами управління базами даних чи з базами даних безпосередньо. Створено повноцінну базу даних. Для цього запропоновано набір таблиць, необхідних для функціонування програмного засобу.

У вступі описаний стан сучасної проблеми, яка постає перед керівниками підприємств, які хочуть покращити показники ефективності роботи свого персоналу, впровадивши автоматизовану систему обліку та аудиту робочого часу працівників. У першому розділі бакалаврської роботи розглянуті теоретичні аспекти роботи, а саме: аналіз ринку використання подібної системи, наведені основні положення та поняття щодо обліку робочого часу, проведено порівняльний аналіз існуючих прототипів та систем, висунені загальні вимоги до розробки систем обліку робочого часу.

У другому розділі представлено моделювання предметної області із використанням діаграм різного типу (IDEF0, IDEF3, DFD), наведена загальна структура системи, що підлягає розробці, висунені вимоги до розробки програмного забезпечення, описані модель збереження даних, база даних, а також проведено опис сценаріїв користувача за допомогою мови UML. Наведена інформація щодо загальної архітектури застосунку для веб-браузеру, структурна модель розробленого програмного продукту з описом кожного елементу (файлу) розширення та описані можливі види створення користувацького інтерфейсу. У розділі роботи представлені дані про реалізацію прототипування інтерфейсу користувача, способи створення інтерфейсу, описано проектування та розробка клієнтської та серверної частини системи, наведено програмну реалізацію спроектованої системи обліку та аудиту робочого часу працівників на підприємстві.

Необхідно відповідально підходити до вибору системи обліку часу, так як при загальній схожості, кожна має свої унікальні якості, які в певних ситуаціях

можуть виступати як плюсами, так і мінусами. Перед прийняттям рішення про впровадження такої системи необхідно провести повний аналіз існуючих процесів в компанії, зрозуміти, які джерела даних можуть бути використані для обліку робочого часу співробітника. Реалізовано ряд функцій, які відповідають за дії над введеними користувачем дані та їх обробку.

.

Список використаної літератури

1. Андрющенко Г. Актуальні питання формування облікової політики малих підприємств. Бухгалтерський облік і аудит. – 2008. - № 11. - С. 10-14
2. Закон України “Про облік та звітність в Україні” від 16.07.1999 р. № 996-XIV із змінами і доповненнями, внесеними Законами України від 11.05.2000 р. № 1707-III, від 08.06.2000 р. - № 1829- II.
3. Козярук О.А. Організаційний процес бухгалтерського обліку на підприємствах малого бізнесу, 30 Березеня 2010.
4. Литвин, І.С. Інформаційні технології в економіці / І.С. Литвин. – Тернопіль: навч. посібник "Економічна думка", 2011. – 296 с. – ISBN 5-7763- 0459.
5. Методические основы создания информационных систем и информационных технологий в управлении организацией [Електронний ресурс]. – Режим доступу: http://miemp-mi-gor.narod.ru/utcheba/itu/glava/ 002.htm
6. Нотация IDEF0 [Електронний ресурс] // Business Studio Wiki. – Режим доступу до ресурсу: http://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0.
7. Особенности клиент-сервер [Електронний ресурс]. – Режим доступу до ресурсу: http://citforum.ck.ua/programming/application/builder_cs3.shtml.
8. Прайс-лист продуктів Microsoft [Електронний ресурс]. – Режим доступу до ресурсу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2017- pricing.
9. Ресурсне забезпечення менеджменту персоналу [Електронний ресурс]. – Режим доступу до ресурсу: http://lubbook.org/book_336_glava_17_Rozd%D1%96l_3._Resursne_zabezpeche.html.
10. Розробка ІС [Електронний ресурс]. – Режим доступу: https://xreferat.com/33/6794-3-razrabotka-informacionnoiy-sistemy-sluzhbazanyatosti.html.
11. Суперключ [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Суперключ.

12. Юрченко І.В. Інформатика та програмування. Частина 1. Навчальний посібник / І.В. Юрченко – Чернівці: Книги–XXI, 2011.– 203 с.
13. Юрченко І.В. Інформатика та програмування. Частина 2 / І.В. Юрченко, В.С. Сікора. – Чернівці: Видавець Яворський С.Н., 2015. – 210 с.
14. NET Framework (.NET) [Електронний ресурс].– Режим доступу до ресурсу: https://www.techopedia.com/definition/3734/net-framework-net. 23. ADO.NET [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ADO.NET.
15. Шилова Т.С., Балюк Т.К. Особливості організації обліку на підприємствах малого бізнесу – http://intkonf.org/shilova-ts-balyuk-tk-osoblivosti-organizatsiyi-obliku-na-pidpriemstvah-malogo-biznesu.
16. Вінницький торговельно-економічний інститут КНЕУ. - Рубрика: Соціум. Наука. Культура, Економіка. – 2010.
17. Долин П.А. Основы техники безопасности в электроустановках. - М.: Энергоатомиздат, 1984. - 448с.
18. Ткачук І.Н., Слонченко А.В., Степанов А.Г., Сабарко Р.В., Охрана труда в приборостроении.-К.:Вища шк., 1980.-192ст.
19. Правила безпечної експлуатації електроустановок споживачів К.: Основа, 1998. – 348с.
20. Пожежна безпека. Нормативні акти та інші документи. В 2-х т. К.: Основа, 1997. – Т.1. – 446с., Т.2 – 448с.
21. Кобевник В.Ф. Охрана труда. - К.: Вища школа, 1990, - 286 с.
22. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
23. Overview of the .NET Framework [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/ getstarted/overview.
24. Skonnard, A. XML in .NET: .NET Framework XML Classes and C# Offer Simple, Scalable Data Manipulation [Електронний ресурс]. – Режим доступу до ресурсу: http://msdn.microsoft.com/en-us/magazine/cc302158.aspx.

25. SQL CREATE TABLE Statement [Електронний ресурс]. – Режим доступу до ресурсу: https://www.w3schools.com/sql/ sql_create_table.asp.
26. Visual Studio 2017 [Електронний ресурс]. – Режим доступу до ресурсу: https://www.microsoft.com/ru-ru/SoftMicrosoft/vs2017.
27. Windows [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Windows
28. https://lmsapi.plagiat.pl/report/?language=uk&auth=eyJraWQiOiJyZXBvcnQyMDIwIiwiYWxnIjoiUlMyNTYifQ.eyJ2aWV3T25seSI6dHJ1ZSwiZXZhbHVhdGlvbkVuYWJsZWQiOnRydWUsImRvY3VtZW50SWQiOjgzMTMwMzEsImV4cCI6MTYyNDUzMzc4NCwiaWF0IjoxNjI0NDQ3Mzg0LCJleHBpcmF0aW9uUmVkaXJlY3Rpb24iOiJodHRwczpcL1wvcGFuZWwuc3RyaWtlcGxhZ2lhcmlzbS5jb21cLyNcL2RvY3VtZW50c1wvODMxMzAzMVwvcmVwb3J0IiwicGxhZ2lhdFVzZXJJZCI6OTMwMTA1fQ.aGoRNFdbvu3XjC13O-SQC6UpiEYSEJe2crZJD9zvEVMa5KOvF1EfBNOf7gf2wveFb_z9YawBLVzhU--JTi4ANfyBEK1xKU4kAv5XaT9zJygJxiEFZ-3Ksi7ipWYIykzfNfRyeFHv35UAvgK4S4gs7KSe1P3aMztSnrL5qvMs_ALDWxo9PrgjaXnNsDkIzy6DEu8ra4FZuXUhN3H6uB5pQv9Q8_r05pGJYCkCGAGKOA04oZQ_-9CJwZbxWFfH2miXRw9qzowRW4TAjkGedgBmmSLSJLLy8lJuTpY4BPmFpdhm51ILTvBBYwSMmKrjuBKyZtkDR3TvyfRxaGKB9kMHFg

Додатки

Програмний код створення бази даних та базових таблиць

Форма Zvit:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows;
using System.Data.OracleClient;
using System.IO;
using System.Threading;

namespace ZvitOracle
{
    public partial class Zvit : Form
    {
        public Zvit()
        {
            InitializeComponent();
        }

        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void довідкаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Програма створена, як приклад використання бази даних Oracle
для зчитування та запису інформації через MS Studio");
        }

        private void створитиНовеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Company firm = new Company();
            firm.Show();
        }

        private void створитиToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            ZvitType gh = new ZvitType();
            gh.Owner = this;
            gh.ShowDialog();
        }

        private void Zvit_Load(object sender, EventArgs e)
        {
            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();

            OracleDataAdapter da = new OracleDataAdapter("SELECT COMPANY_NAME FROM COMPANY",
con);
            DataSet ds = new DataSet();
```

```
            da.Fill(ds);
            foreach (DataRow item in ds.Tables[0].Rows)
            {
                company.Items.Add(item[0]);
                company1.Items.Add(item[0]);
                company2.Items.Add(item[0]);
            }

            con.Close();
        }

        private void company_SelectedIndexChanged(object sender, EventArgs e)
        {
            Object selectedItem = company.SelectedItem;

            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();

            OracleCommand comm = new OracleCommand("SELECT COMPANY_CODE FROM COMPANY WHERE
COMPANY_NAME='" + selectedItem.ToString() + "'", con);
            OracleDataReader reader = comm.ExecuteReader();
            while (reader.Read())
            {
                code.Text = reader[0].ToString();
                code1.Text = reader[0].ToString();
                code2.Text = reader[0].ToString();
            }

            OracleCommand comm1 = new OracleCommand("SELECT COMPANY_HEAD FROM COMPANY WHERE
COMPANY_NAME='" + selectedItem.ToString() + "'", con);
            OracleDataReader reader1 = comm1.ExecuteReader();
            while (reader1.Read())
            {
                head.Text = reader1[0].ToString();
                head1.Text = reader1[0].ToString();
                head2.Text = reader1[0].ToString();
            }

            OracleCommand comm2 = new OracleCommand("SELECT COMPANY_BOOKER FROM COMPANY
WHERE COMPANY_NAME='" + selectedItem.ToString() + "'", con);
            OracleDataReader reader2 = comm2.ExecuteReader();
            while (reader2.Read())
            {
                booker.Text = reader2[0].ToString();
                booker1.Text = reader2[0].ToString();
                booker2.Text = reader2[0].ToString();
            }

            con.Close();

        }

        private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();

            OracleCommand cmd = new OracleCommand();
            cmd.Connection = con;
            cmd.CommandText = string.Format("INSERT INTO SPENDING (FILD_VALUE) VALUES
('{0}')",
```

```
                textBox3000.Text+ textBox2000.Text+ textBox2050.Text+ textBox2090.Text+
textBox2095.Text+
                textBox3000.Text+ textBox3005.Text+ textBox3006.Text+ textBox3010.Text+
textBox3095.Text+
                textBox3100.Text+ textBox3105.Text+ textBox3110.Text+ textBox3115.Text+
textBox3190.Text+
                textBox3200.Text+ textBox3205.Text+ textBox3215.Text+ textBox3220.Text+
textBox3225.Text+
                textBox3250.Text+ textBox3255.Text+ textBox3260.Text+ textBox3270.Text+
textBox3290.Text+
                textBox3295.Text+ textBox3300.Text+ textBox3305.Text+ textBox3340.Text+
textBox3345.Text+
                textBox3350.Text+ textBox3355.Text+ textBox3390.Text+ textBox3395.Text+
textBox3400.Text+
                textBox3405.Text+ textBox3410.Text+ textBox3415.Text+ textBox3500.Text+
textBox3505.Text+
                textBox3510.Text+ textBox3515.Text+ textBox3520.Text+ textBox3550.Text+
textBox3560.Text+
                textBox3570.Text+ textBox3580.Text+ textBox3595.Text);
            cmd.CommandText = "INSERT INTO SPENDING (ZVIT_DATA, ZVIT_COMPANY, ZVIT_CODE,
ZVIT_HEAD, ZVIT_BOOKER, ZVIT_NAME) VALUES ('"+ dateTimePicker1.Value.ToShortDateString() +
"','" +company.Text + "', '" + code.Text + "','" + head.Text + "','" + booker.Text + "','" +
zv_name.Text + "')";
            cmd.CommandType = CommandType.Text;

            con.Close();
        }


        private void зберегтиЯкToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "JPeg Image|*.jpg|Bitmap Image|*.bmp|Gif Image|*.gif";
            saveFileDialog1.Title = "Save an Image File";
            saveFileDialog1.ShowDialog();
            if (saveFileDialog1.FileName != "")
            {
                System.IO.FileStream fs = (System.IO.FileStream)saveFileDialog1.OpenFile();
                switch (saveFileDialog1.FilterIndex)
                {
                    case 1:
                        this.зберегтиЯкToolStripMenuItem.Image.Save(fs,
System.Drawing.Imaging.ImageFormat.Jpeg);
                        break;

                    case 2:
                        this.зберегтиЯкToolStripMenuItem.Image.Save(fs,
System.Drawing.Imaging.ImageFormat.Bmp);
                        break;

                    case 3:
                        this.зберегтиЯкToolStripMenuItem.Image.Save(fs,
System.Drawing.Imaging.ImageFormat.Gif);
                        break;
                }
                fs.Close();
            }
        }

        private void відкритиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ZvitOpen op = new ZvitOpen();
            op.Owner = this;
            op.ShowDialog();
```

```
            Object selectedName = zv_name.Text;

            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();

            OracleCommand com = new OracleCommand("SELECT ZVIT_COMPANY FROM SPENDING WHERE
ZVIT_NAME='" + selectedName.ToString() + "'", con);
            OracleDataReader reader = com.ExecuteReader();
            while (reader.Read())
            {
                company.Text = reader[0].ToString();
            }

            OracleCommand comm = new OracleCommand("SELECT ZVIT_CODE FROM SPENDING WHERE
ZVIT_NAME='" + selectedName.ToString() + "'", con);
            OracleDataReader reader1 = comm.ExecuteReader();
            while (reader1.Read())
            {
                code.Text = reader1[0].ToString();
            }

            OracleCommand commm = new OracleCommand("SELECT ZVIT_HEAD FROM SPENDING WHERE
ZVIT_NAME='" + selectedName.ToString() + "'", con);
            OracleDataReader reader2 = commm.ExecuteReader();
            while (reader2.Read())
            {
                head.Text = reader2[0].ToString();
            }

            OracleCommand commmm = new OracleCommand("SELECT ZVIT_BOOKER FROM SPENDING WHERE
ZVIT_NAME='" + selectedName.ToString() + "'", con);
            OracleDataReader reader3 = commmm.ExecuteReader();
            while (reader3.Read())
            {
                booker.Text = reader3[0].ToString();
            }
            con.Close();
        }

    }
}
```

Форма ZvitType:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OracleClient;

namespace ZvitOracle
{
    public partial class ZvitType : Form
    {
        public ZvitType()
        {
            InitializeComponent();
        }

        private void selectType_Click(object sender, EventArgs e)
        {
            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();

            if (checkBox1.Checked)
            {
                OracleCommand cmd = new OracleCommand();
                cmd.Connection = con;
                cmd.CommandText = "INSERT INTO SPENDING (ZVIT_TYPE) VALUES ('forma-3')";
                cmd.CommandType = CommandType.Text;

                Zvit main = this.Owner as Zvit;
                if (main != null)
                {
                    main.tabControl1.SelectedIndex = 0;
                }
            }
            if (checkBox2.Checked)
            {
                OracleCommand cmd = new OracleCommand();
                cmd.Connection = con;
                cmd.CommandText = "INSERT INTO SPENDING (ZVIT_TYPE) VALUES ('forma-3n')";
                cmd.CommandType = CommandType.Text;

                Zvit main = this.Owner as Zvit;
                if (main != null)
                {
                    main.tabControl1.SelectedIndex = 1;
                }
            }
            if (checkBox3.Checked)
            {
                OracleCommand cmd = new OracleCommand();
                cmd.Connection = con;
                cmd.CommandText = "INSERT INTO SPENDING (ZVIT_TYPE) VALUES ('forma-2')";
                cmd.CommandType = CommandType.Text;

                Zvit main = this.Owner as Zvit;
                if (main != null)
                {
                    main.tabControl1.SelectedIndex = 2;
```

```
                }
            }
            con.Close();
            this.Focus();
            this.Close();
        }
    }
}
```

Форма Company:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OracleClient;

namespace ZvitOracle
{
    public partial class Company : Form
    {
        public Company()
        {
            InitializeComponent();
        }

        private void createCompany_Click(object sender, EventArgs e)
        {
            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();

            OracleCommand cmd = new OracleCommand();
            cmd.Connection = con;
            cmd.CommandText = "INSERT INTO COMPANY (COMPANY_NUMBER, COMPANY_NAME,
COMPANY_CODE, COMPANY_DATE, COMPANY_HEAD, COMPANY_BOOKER) VALUES ('" + id_company.Value +
"','" + nameCompany.Text + "', '" + codeCompany.Text + "','" + DateCompany.Text + "','" +
headCompany.Text + "','" + bookerCompany.Text + "')";
            cmd.CommandType = CommandType.Text;

            cmd.ExecuteNonQuery();
            con.Dispose();

            Close();
        }

    }
}
```

Форма ZvitOpen:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows;
using System.Data.OracleClient;

namespace ZvitOracle
{
    public partial class ZvitOpen : Form
    {
        public ZvitOpen()
        {
            InitializeComponent();
        }

        private void ZvitOpen_Load(object sender, EventArgs e)
        {
            string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
            OracleConnection con = new OracleConnection(oradb);
            con.Open();
            OracleDataAdapter da = new OracleDataAdapter("SELECT ZVIT_NAME FROM SPENDING",
con);
            DataSet ds = new DataSet();
            da.Fill(ds);
            foreach (DataRow item in ds.Tables[0].Rows)
            {
                comboBox1.Items.Add(item[0]);
            }
            con.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Zvit lable = this.Owner as Zvit;
            if (lable != null)
            {
                Object selectedItem = comboBox1.SelectedItem;
                lable.zv_name.Text = selectedItem.ToString();
                Object selectedName = lable.zv_name.Text;
                string oradb = "Data Source=XE;User ID=valeriy;Password=user;";
                OracleConnection con = new OracleConnection(oradb);
                con.Open();
                OracleDataAdapter da = new OracleDataAdapter("SELECT FILD_VALUE FROM
SPENDING WHERE ZVIT_NAME='" + selectedName.ToString() + "'", con);
                DataSet ds = new DataSet();
                da.Fill(ds);
                if (ds.Tables[0].Rows.Count > 0)
                {
                    lable.textBox3000.Text = ds.Tables[0].Rows[0][0].ToString();
                    lable.textBox3005.Text = ds.Tables[0].Rows[1][0].ToString();
                    lable.textBox3006.Text = ds.Tables[0].Rows[2][0].ToString();
                    lable.textBox3010.Text = ds.Tables[0].Rows[3][0].ToString();
                    lable.textBox3095.Text = ds.Tables[0].Rows[4][0].ToString();
                    lable.textBox3100.Text = ds.Tables[0].Rows[5][0].ToString();
                    lable.textBox3105.Text = ds.Tables[0].Rows[6][0].ToString();
                    lable.textBox3110.Text = ds.Tables[0].Rows[7][0].ToString();
                    lable.textBox3115.Text = ds.Tables[0].Rows[8][0].ToString();
                    lable.textBox3190.Text = ds.Tables[0].Rows[9][0].ToString();
                    lable.textBox3200.Text = ds.Tables[0].Rows[10][0].ToString();
                    lable.textBox3205.Text = ds.Tables[0].Rows[11][0].ToString();
                    lable.textBox3215.Text = ds.Tables[0].Rows[12][0].ToString();
                    lable.textBox3225.Text = ds.Tables[0].Rows[13][0].ToString();
```

```
                    lable.textBox3250.Text = ds.Tables[0].Rows[14][0].ToString();
                    lable.textBox3255.Text = ds.Tables[0].Rows[15][0].ToString();
                    lable.textBox3295.Text = ds.Tables[0].Rows[16][0].ToString();
                    lable.textBox3300.Text = ds.Tables[0].Rows[17][0].ToString();
                    lable.textBox3305.Text = ds.Tables[0].Rows[18][0].ToString();
                    lable.textBox3340.Text = ds.Tables[0].Rows[19][0].ToString();
                    lable.textBox3345.Text = ds.Tables[0].Rows[20][0].ToString();
                    lable.textBox3350.Text = ds.Tables[0].Rows[21][0].ToString();
                    lable.textBox3355.Text = ds.Tables[0].Rows[22][0].ToString();
                    lable.textBox3390.Text = ds.Tables[0].Rows[23][0].ToString();
                    lable.textBox3395.Text = ds.Tables[0].Rows[24][0].ToString();
                    lable.textBox3400.Text = ds.Tables[0].Rows[25][0].ToString();
                    lable.textBox3405.Text = ds.Tables[0].Rows[26][0].ToString();
                    lable.textBox3410.Text = ds.Tables[0].Rows[27][0].ToString();
                    lable.textBox3415.Text = ds.Tables[0].Rows[28][0].ToString();
                }

                con.Close();
        }

            this.Focus();
            this.Close();
        }
    }
}
```