Міністерство освіти і науки України
Державний ВНЗ «Національний гірничий університет»

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

напрям підготовки *121 «Інженерія програмного забезпечення»*
(код і назва напряму підготовки)

спеціальність *122 Комп'ютерні науки*
(код і назва спеціальності)

освітній рівень *магістр*
(назва освітнього рівня)

кваліфікація *інженер з комп'ютерних систем*
(назва кваліфікації)

на тему: *Дослідження функціонування моделі даних за стандартом UML із застосуванням технології liquibase*

Виконавець:
студент *2* **курсу , групи** 122м-16-1

(підпис) Шарапат В.Є.
(прізвище та ініціали)

Керівник	**Посада, прізвище, ініціали**	**Оцінка**	**Підпис**
проекту	*проф. Куваєв В.М.*.		
розділів:			
Економічний	*доц. Касьяненко Л.В.*		

Рецензент			

Нормоконтроль	*доц. Коротенко Л.М.*		

Дніпро
2018

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Національний гірничий університет»

ЗАТВЕРДЖЕНО:
завідувач кафедри

програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 20 ____ року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту 122м-16-1 *Шарапат В.Є.*
(група) (прізвище та ініціали)

Тема дипломної роботи Дослідження функціонування моделі даних за стандартом UML із застосуванням технології liquibase

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора Державного ВНЗ «НГУ» від *26 . 12 .2017 р.* № *2127 -л*

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – технологія розробки програмного забезпечення за стандартом UML на основі організації "Укрпромавтоматика".

Предмет досліджень – технологія підходу до розробки програмного забезпечення, проектування інформаційних систем - моделі даних за стандартом UML на основі організації "Укрпромавтоматика".

Мета НДР – вивчення бізнес-процесів підприємства та створення моделі даних за стандартом UML з використанням технології Liquibase.

Вихідні дані для проведення роботи: сценарій організації ПВТП «Укрпромавтоматика», яка потребує ефективніше використовувати обмежені ресурси.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна результатів, що очікуються, полягає у :

Проектуванні інформаційної систем на основі використання технології

проектування моделі даних UML і відстеження змін моделі.

Практична цінність результатів полягає у:

Скорочені часу на прийняття важливих та стратегічних рішень організації завдяки ретельному розгляду бізнес-процесів підприємства та створені гнучкої архітектури БД.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати магістерської роботи повинні відповідати вимогам паспорту наукової спеціальності 05.13.06 – «Інформаційні технології».

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання методики розробки інформаційних систем з використанням UML. Згідно виробничих функцій та професійних задач магістра, які виносяться на кваліфікаційну роботу, повинні бути розроблені програмна документація та комплекс компонентів для розробки програмного забезпечення та інформаційних систем з використанням UML.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз стану питання «Розробка моделі даних за стандартом UML».	1.08.2017 15.12.2017
Використання технології «UML» при проектуванні інформаційної системи.	1.08.2017 15.12.2011
Використання методики проектування інформаційних систем з використанням технології UML.	15.12.2017 15.01.2011

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів може бути отриманий за рахунок скорочення часу та затрат на розробку програмного забезпечення, інформаційних систем.

Соціальний ефект від реалізації результатів роботи не очікується, так як створена архітектура розрахована на обмежену групу осіб, які будуть її використовувати.

7 ДОДАТКОВІ ВИМОГИ

Відповідність оформлення ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення, ЕДИНОЙ СИСТЕМЕ ПРОГРАММНОЙ ДОКУМЕНТАЦИИ (ЕСПД – ГОСТ 19.101-77, ГОСТ 19.102-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.201-78, ГОСТ 19.202-78, ГОСТ 19.401-78, ГОСТ 19.402-78, ГОСТ 19.404-79)

Завдання видав _____ *Куваєв В.М.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Шарапат В.Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: _____

Термін подання дипломного проекту до ДЕК _____

Реферат

Пояснювальна записка: 89 с., 32 рис., 3 дод., 19 джерел.

Об'єкт дослідження: технологія підходу до розробки програмного забезпечення, проектування інформаційних систем - моделі даних за стандартом UML на основі організації "Укрпромавтоматика".

Мета магістерської роботи: вивчення бізнес-процесів підприємства та створення моделі даних за стандартом UML з використанням технології Liquibase.

Методи дослідження. При вирішенні поставленого завдання використовувалися наукові досягнення в областях розробки інформаційних систем і програмного забезпечення.

Наукова новизна результатів, що очікуються, полягає у: створенні методики проектування інформаційних систем на основі використання технології проектування моделі даних UML і відстеження змін моделі.

Практичне значення роботи полягає в створенні програмних модулів, програмного продукту, які дозволяють оцінити переваги проектування інформаційних систем, розробки програмного забезпечення з використанням технології MDA.

Галузь застосування. Розроблена інформаційна система може застосовуватися для вирішення широкого спектра завдань в організації «Укрпромавтоматика», зокрема, для створення програмного забезпечення, проектування інформаційних систем.

Значення роботи і висновки. Удосконалена методика дозволяє проектувати інформаційні системи зі значним скороченням як матеріальних витрат, так і тимчасових, що підтверджується розробленим програмним продуктом в даній магістерській роботі.

Прогнози щодо розвитку досліджень. Розробити універсальні програмні модулі, які можуть бути використані для підтримки проектування інформаційних систем з різних програмних платформ. Розробити комплекс програмних засобів і призначений для користувача інтерфейс для графічного представлення результатів, порівняльного аналізу розробки з використанням технології MDA.

Список ключових слів: АИС, БД, ЕОМ, ПВТП, СУБД, CDM, FK, PDM, SQL, UML, MDA, RATIONAL ROSE, UML.

The abstract

Explanatory note:89 p., 32 fig., 3 applications, 19 sources.

Object of research: Object of research: technology approach to software development, design of information systems - data models according to the UML standard based on the organization Ukrpromavtomatika.

The purpose of the degree project: the study of business processes of the enterprise and the creation of a data model under the UML standard using Liquibase technology.

Methods of research. Solving the task used scientific advances in the areas of development of information systems and software.

The scientific novelty consists in conducting elimination a methodology for designing information systems based on the use of technology for designing the UML data model and tracking changes in the model.

The practical value of work is to create software modules, software products that allow you to evaluate the benefits of designing information systems, software development using MDA technology.

The scope. The developed information system can be used to solve a wide range of tasks in the organization "Ukrpromavtomatika", in particular, for software development, designing of information systems.

The value of the work and conclusions. Advanced technique allows the design of information systems with significant reductions in both material costs and time, as evidenced by the developed software in the master's work.

Projections on development research. Develop a universal software modules that can be used to support the design of information systems from various platforms. Develop a set of software tools and user interface for graphical presentation of results of comparative analysis of the development of technology with MDA.

List of keywords: АИС, БД, ЕОМ, ПВТП, СУБД, CDM, FK, PDM, SQL, UML, MDA, RATIONAL ROSE, UML.

ЗМІСТ

ВСТУП

Останнім часом все частіше можна почути про збільшення масштабів розробки програмного забезпечення орієнтованого на веб. Веб індустрія займає велику долю ринку порівняно з іншими напрямками. Це багатомільярдні компанії, що займаються розробкою розважальних продуктів та програмних забезпечень, що складаються з мільйонів рядків програмного коду і файлів мультимедіа.

Цьому сприяло масштабне зростання числа користувачів інтернету. Веб-додатки стали складніші і масштабніші. Сучасні веб-додатки - великі і складні програмні комплекси. Витрати тільки на програмну розробку часто вимірюються сотнями людино-місяців. За обсягом задіяних технологій, залучених коштів і рівня професійної підготовки розробників, веб індустрія давно зайняла аж ніяк не останнє місце в світі IT.

Спочатку інтернету вся інформація була тільки в текстовому вигляді. Разом з збільшенням швидкості інтернет з’єднання виростають і можливості користувачів. Зараз в найпопулярніших сайтах української частини мереж зареєстрована велика кількість людей це наприклад - є rozetka (10 млн користувачів) і olx (20 млн). Світовий лідер - Facebook (більше 500 млн користувачів по всьому світу). Facebook дуже активно розвивається на нашому ринку, і зараз в мережі зареєстровано 3,7 млн співвітчизників. Інший популярний в світі сервіс Twitter (мікроблоги) в Україні зібрав вже понад 600 тис. користувачів.

Саме через таку велику кількість людей і виникає потреба в ще більшій кількості професійних програмістів, які змогли би впоратися з вимогами. А для таких професіоналів гостро стоїть питання вибору технологій, мов програмування, алгоритмів та засобів вирушення цих завдань.

З збільшення кількості людей ростуть і їх потреби, такі гіганти веб напрямку, як Facebook, Instagram, Twitter вводять штучний інтелект для своїх стрічок новин. Починають використовуватись такі шаблони програмування, як мікросервіси, маркетплейс платформи застосовують в своїй роботі блокчейн. В

такому різноманітності потреб та вимог стає дуже важливим створення об’ємного та обґрунтованого технічно порівняння мов та інструментів, які б максимально швидко вирішили поставлені перед ними завдання.

Тому існує досить велика кількість мов програмування орієнтованих на розробку web-додатків і, очевидно, що за процесом активізація їх використання ховається все більш складний етап порівняння та виявлення більш підходящого до вирішуваної задачі. Головна причина цього явища - відсутність єдиного механізму порівняння. Технічним керівникам команд доводиться витрачати значну кількість годин для виявлення, порівняння, та тестування, мов програмування які підходять для поставлених завдань. А згодом, в разі переходу до іншої задачі, доводиться, по суті, все повторювати заново, знову витрачати час та нерви. На великих підприємствах нерідко виникає необхідність впровадження і одночасного використання декількох мов програмування, а поверхневе порівняння яке може провести технічний лідер не дасть достатньо об’ємного результату. В цьому випадку необхідні витрати на висококваліфікованих програмістів можуть легко вийти за рамки допустимих, оскільки фахівці, які володіють одночасно декількома мовами програмування, зустрічаються досить рідко, а їх «ціна» росте аж ніяк не пропорційно кількості виконаних завдань. Перерахування подібних ситуацій можна продовжити, але вже зі сказаного, очевидно, що необхідно якимось чином вирішувати проблеми систематизації, профілювання та порівняння, щоб уніфікувати всі наявні нині мови розробки програмного забезпечення.

На даний час існує багато порівняльних графіків, діаграм та текстів, але ні один з них не описує ефективність тої чи іншої мови в певній задачі. Часто мови програмування порівнюють тільки за їх популярністю за кількістю цитування цих мов в сучасних засобах Інтернет мовлення та кореспонденції. Більшість використовує це як індикатор для вибору тої чи іншої мови. Сама ця ідея є непослідовною і навіть дивною. Таким чином виходить, що старі мови
типу Сі, які просто через вельми поважний вік свого існування отримали більше всіх документацій та посилань, як приведено в даній дипломній роботі – це не є

досить об’ємним фактором при виборі технології для реалізації сучасних запитів та задач користувачів розроблюваних програмних додатків.

Необхідність в виявленні параметрів за якими можна порівняти мови саме як мови, а не платформи стала дуже гостро. Популярність, інструменти розробки, позиція на ринку, максимальні заробітні плати - це важливо, але важливо саме подивитися на мови самі по собі.

РОЗДІЛ 1.

Аналіз предметної області і постановка завдання

1.1. Постановка задачі

Завдання дипломної роботи полягає у створенні моделі даних та дослідження її функціонування підприємства ПВТП «Укрпромавтоматика» за стандартом UML. Для створення моделі даних необхідно дослідити функціонування системи Liquibase та виконати наступне:

1) Відтворити сценарій діяльності підприємства;
2) Розробити діаграми використання;
3) Розробити діаграми кооперації;
4) Розробити логічну модель даних;
5) Розробити фізичну модель даних;
6) Згенерувати MySQL код;
7) Створити схему даних на сервері.

1.2. Стратегічний аналіз діяльності компанії

ПВТП (приватне виробничо-торгове підприємство) «УкрПромАвтоматика» за замовленнями промислових підприємств, торгівельних організацій і приватних осіб надає послуги з налагодження, установки і монтажу електрообладнання. Підприємство має у своєму розпорядженні численний обслуговуючий персонал. У складі підприємства є такі служби:

- керівництво підприємства;
- бухгалтерія;
- відділ кадрів;
- матеріальний склад;
- відділ охорони праці;
- техвідділ.

Для забезпечення інформаційних потреб ПВТП потрібно створити інформаційну систему (ІС) з централізованою базою даних (БД) і робітниками в підрозділах директора, головного бухгалтера, відділу постачання та відділу кадрів.

Робоча станція директора повинна обслуговувати потреби свого керівника, його заступника, головного інженера, відділу кадрів і бухгалтерії. На робочій станції постачання повинні здійснюватися також операції матеріального складу.

Функціональні обов'язки підрозділів полягають в наступному:

1. Директор керує роботою всіх служб підприємства, вирішує кадрові

питання, контролює банківські операції бухгалтерії і укладення договорів з виконання послуг по налагодженню, установці і монтажу електрообладнання.

2. Бухгалтерія - виконує взаєморозрахунок із замовниками за виконання роботи, нарахування зарплатні і переведення нарахованої суми на зарплатні картки працівників. Також стежить за поповненням і витратою матеріалів на складі, за своєчасною оплатою, а також реєструє договори на виконання послуг з налагодження, встановлення електрообладнання.

3. Відділ кадрів - прийом / звільнення співробітників, табельний облік працюючих, реєстрація лікарняних і відпусток, подача реклами в газети, на радіо і телебачення.

4. Відділ постачання за заявками відповідних служб здійснює придбання на склад необхідних запчастин. Ведеться суворий облік. Всі операції відділу підзвітні бухгалтерії.[1]

1.3. Бізнес-процеси ПВТП

При укладанні договору з клієнтом співробітник фіксує реквізити підприємства в БД, встановлює календарну дату виконання послуг. У бухгалтерії проводиться розрахунок вартості виконання послуг і завершується оформленням договору із замовником. Після затвердження договору з директором і його реєстрації в БД ІС один примірник договору передається замовнику для оплати.

В БД ІС реєструються такі відомості: тип робіт, дата початку виконання послуг, хто виконує послуги, вартість послуг, термін закінчення послуг.[2]

Територія складу охороняється: вдень двома охоронцями, в нічний час доби (з 18-00 до 8-00) – бригадою озброєних охоронців із мисливськими собаками.

Начальник охорони на робочій станції директора заносить в БД прізвища всіх присутніх в зміні співробітників. Вранці фіксуються всі події за минулу добу.

В кінці місяця бухгалтерія за даними табельного обліку нараховує зарплатню співробітникам ПВТП, підраховує витрати і обчислює остаточний прибуток.

1.3.1. Аналіз вимог

Аналіз предметної області спрямований на виявлення сутності в організації бізнес-процесу, з'ясування інтересів замовника і визначення їх у формі функціональних вимог для подальшої реалізації. В першу чергу виявляється масштаб розв'язуваної задачі. У разі великого обсягу роботи увагу акцентують на найбільш значущих функціях предметної області і займаються їх аналізом і деталізацією.

Згідно з завданням проектованої системи в першу чергу необхідно створити робочі станції - директору, головному бухгалтеру та у відділі постачання, складу, підрозділу охорони.[3]

Директор ПВТП управляє роботою всіх служб на підприємстві і робить накази, вирішує кадрові питання, контролює дисципліну праці, банківські операції бухгалтерії і укладення договорів на виконання послуг. Його функції розділені між заступником (ВК., бухгалтерія, матеріальний склад, відділ постачання). Для реалізації керуючих функцій директор повинен мати таку інформацію:

1) перелік укладених договорів на виконання послуг на майбутній місяць (реквізити замовників, види послуг, терміни, розміри оплати);

2) перелік виконаних договорів поза минулий місяць (реквізити замовників, види послуг, терміни, розмір оплати);

3) щомісячний звіт бухгалтерії про вступні платежі від замовників за виконані послуги, запчастини.

4) щомісячний звіт бухгалтерії про заробітну платню та преміювання працівників ПВТП;

5) щомісячний звіт ВК про прийнятих і звільнених з роботи фахівців, відомості про пішовших у відпустку, хворих, пенсіонерів.

6) щомісячна відповідь підрозділу охорони (стан об'єкта, зауваження до об'єктів, випадки порушення і т.д).

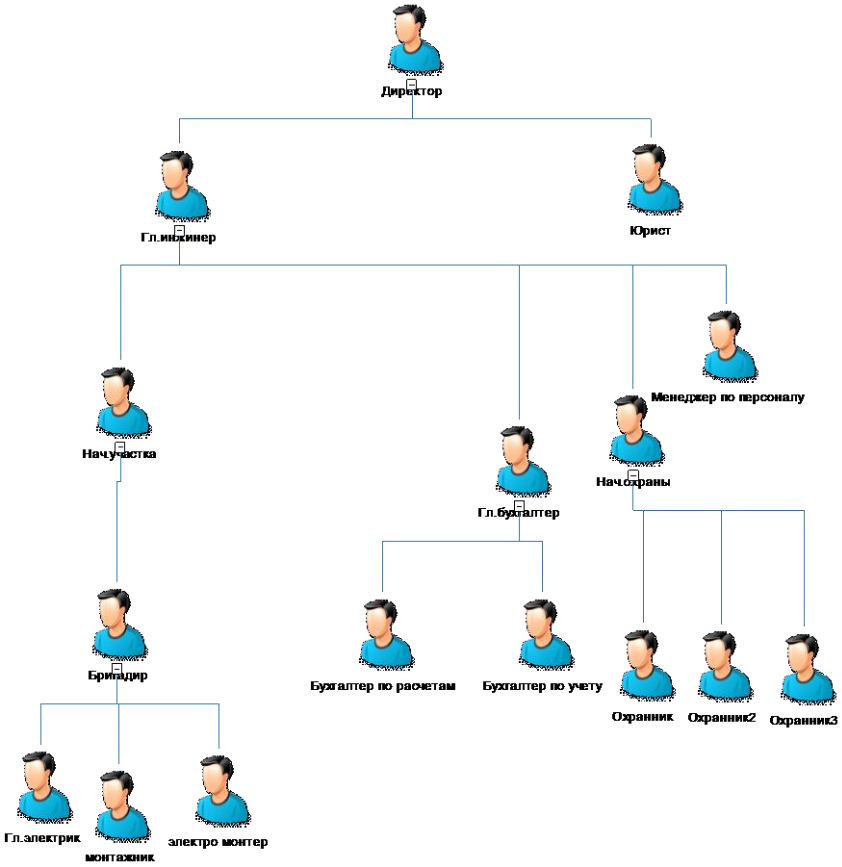

Рис. 1 Організаційно-штатна структура

РОЗДІЛ 2

Аналіз технологій та методів розробки БД

2.1. Короткі відомості про UML

UML (англ. Unified Modeling Language - уніфікована мова моделювання) - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

UML є мовою широкого профілю, це - відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, так званої UML-моделі. UML був створений для визначення, візуалізації, проектування та документування, в основному, програмних систем. UML не є мовою програмування, але на підставі UML-моделей можлива генерація коду.[4]

UML дозволяє також розробникам програмного забезпечення досягти угоди в графічних позначеннях для представлення загальних понять (таких як клас, компонент, узагальнення, агрегація і поведінка) і більше сконцентруватися на проектуванні та архітектурі.

В UML використовуються наступні види діаграм (для виключення неоднозначності приведені також позначення англійською мовою):

Структурні діаграми:

1. Діаграма класів;
2. діаграма компонентів;
3. діаграма композитної / складової структури;
4. діаграма кооперації;
5. діаграма розгортання;
6. діаграма об'єктів;
7. діаграма пакетів;
8. діаграма профілів.

Діаграми поведінки:

1. Діаграма діяльності;
2. діаграма станів;

3. діаграма варіантів використання;

Діаграми взаємодії:

1. Діаграма комунікації (UML2.0) / Діаграма кооперації (UML1.x);
2. діаграма огляду взаємодії;
3. діаграма послідовності;
4. діаграма синхронізації;

Діаграма класів (Class diagram) - статична структурна діаграма, що описує структуру системи, яка демонструє класи системи, їх атрибути, методи і залежності між класами.[5]

Існують різні точки зору на побудову діаграм класів в залежності від цілей їх застосування:

1. Концептуальна точка зору - діаграма класів описує модель предметної області;
2. точка зору специфікації - діаграма класів застосовується при проектуванні інформаційних систем;
3. точка зору реалізації - діаграма класів містить класи, використовувані безпосередньо в програмному коді (при використанні об'єктно-орієнтованих мов програмування).

Діаграма компонентів (Component diagram) - статична структурна діаграма, показує розбиття програмної системи на структурні компоненти та зв'язки (залежності) між компонентами. Як фізичні компоненти можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети і т. п.

Діаграма композитної / складової структури (Composite structure diagram) - статична структурна діаграма, демонструє внутрішню структуру класів і, по можливості, взаємодію елементів (частин) внутрішньої структури класу.[6]

Підвидом діаграми композитної структури є діаграми кооперації (Collaboration diagram, введені в UML 2.0), які показують роль і взаємодію класів в рамках кооперації. Кооперації зручні при моделюванні шаблонів проектування. Діаграми композитної структури можуть використовуватися спільно з діаграмами класів.

Діаграма розгортання (Deployment diagram, діаграма розміщення) - слугує для моделювання працюючих вузлів і артефактів, розгорнутих на них. В UML 2 на вузлах розгортаються артефакти, в той час як в UML 1 на вузлах розгорталися компоненти. Між артефактом і логічним елементом (компонентом), який він реалізує, встановлюється залежність маніфестації.

Діаграма об'єктів (Object diagram) - демонструє повний або частковий знімок модельованої системи в заданий момент часу. На діаграмі об'єктів відображаються екземпляри класів (об'єкти) системи із зазначенням поточних значень їх атрибутів і зв'язків між об'єктами.[7]

Діаграма пакетів (Package diagram) - структурна діаграма, основним змістом якої є пакети і відносини між ними. Жорсткого поділу між різними структурними діаграмами не проводиться, тому дана назва пропонується виключно для зручності і не має семантичного значення (пакети і діаграми пакетів можуть бути присутніми на інших структурних діаграмах). Діаграми пакетів служать, в першу чергу, для організації елементів в групи за будь-якою ознакою з метою спрощення структури і організації роботи з моделлю системи.[8]

Діаграма діяльності (Activity diagram) - діаграма, на якій показано розкладання деякої діяльності на її складові частини. Під діяльністю (англ. Activity) розуміється специфікація виконуваної поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів - вкладених видів діяльності і окремих, з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого.

Діаграми діяльності використовуються при моделюванні бізнес-процесів, технологічних процесів, послідовних і паралельних обчислень.

Діаграма станів (State Machine diagram, діаграма кінцевого автомата, діаграма станів) - діаграма, на якій представлений кінцевий автомат з простими станами, переходами і композитними станами. State Machine (автомат) в мові UML є певний формалізм для моделювання поведінки елементів моделі і системи в цілому. В метамоделі UML автомат є пакетом, в якому визначено безліч понять, необхідних для подання поведінки модельованої сутності у вигляді дискретного

простору з кінцевим числом станів і переходів. З іншого боку, автомат описує поведінку окремого об'єкта в формі послідовності станів, які охоплюють всі етапи його життєвого циклу, починаючи від створення об'єкта і закінчуючи його знищенням. Кожна діаграма станів представляє деякий автомат.[9]

Основними поняттями, що входять в формалізм автомата, є зі стояння і перехід. Головна відмінність між ними полягає в тому, що тривалість перебування системи в окремому стані істотно перевищує час, який витрачається на перехід з одного стану в інший. Передбачається, що в межі час переходу з одного стану в інший дорівнює нулю (якщо додатково нічого не сказано). Іншими словами, перехід об'єкта зі стану в стан відбувається миттєво.

Діаграма станів (англ. State machine) - специфікація послідовності станів, через які проходить об'єкт або взаємодія у відповідь на події свого життя, а також відповідні дії об'єкта на ці події. Кінцевий автомат прикріплений до вихідного елементу (класу, кооперації або методу) і служить для визначення поведінки його примірників.[10]

Діаграма варіантів використання (Use case diagram, діаграма прецедентів) - діаграма, на якій відображені відносини, що існують між акторами і варіантами використання. Основна мета створення будь-якої програмної системи - створення такого програмного продукту, який допомагає користувачеві виконувати свої повсякденні завдання. Для створення таких програм насамперед визначаються вимоги, яким повинна задовольняти система. Однак якщо дати користувачам написати ці вимоги на папері, то часто можна отримати список функцій, за яким важко судити чи майбутня система виконувати своє призначення і чи зможе вона полегшити користувачеві виконання його роботи взагалі. Незрозуміло які з виконуваних функцій важливіші і для кого. Для того, щоб більш точно зрозуміти як повинна працювати система, все частіше 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. 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. В процесі тестування, описані раніше варіанти використання дозволяють простіше оцінити точність реалізації вимог користувачів і дозволяють провести покрокову перевірку цих вимог.

Стратегія використання прецедентів при визначенні вимог визначає необхідність додатково до питання "що користувачі чекають від системи?" задавати питання "що система повинна зробити для учасників форуму?". Такий підхід дозволяє шукати функції, які потрібні багатьом користувачам і виключати ті можливості, які не можуть допомогти користувачам виконувати свої повсякденні завдання.[11]

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. Актор позначається значком чоловічка, а Use Case - овалом. Додатково в діаграми можуть бути додані коментарі. Окремий варіант використання позначається на діаграмі еліпсом, усередині якого міститься його коротка назва або ім'я у формі дієслова з пояснювальними словами.Мета варіанту використання полягає в тому, щоб визначити закінчений аспект або фрагмент поведінки деякої сутності без розкриття її внутрішньої структури. В якості такої сутності може виступати система або будь-який елемент моделі, який володіє власною поведінкою. Кожен варіант використання відповідає окремому сервісу, який надає моделююча сутність за запитом актора, тобто визначає спосіб застосування цієї сутності. Сервіс, який ініціалізується за запитом актора, є закінченою неподільну

послідовність дій. Це означає, що після того як система закінчить обробку запиту, вона повинна повернутися в початковий стан, щоб бути готовою до виконання наступних запитів.

Актори взаємодіють з системою за допомогою обміну повідомленнями з варіантами використання. [12] Повідомлення являє собою запит актором певного сервісу системи і отримання цього сервісу. Ця взаємодія може бути виражено за допомогою асоціацій між окремими акторами і варіантами використання або класами. Крім цього, з акторами можуть бути пов'язані інтерфейси, які визначають, яким чином інші елементи моделі взаємодіють з цими акторами.

Між акторами і варіантами використання можуть бути різні види взаємодії. Основні види взаємодії наступні:

1. Проста асоціація - відбивається лінією між актором і варіантом використання (без стрілки). Відображає зв'язок актора і варіанти використання. На малюнку між актором адміністратор і варіантом використання переглядати замовлення.

2. Спрямована асоціація - то ж що і проста асоціація, але показує, що варіант використання инициализируется актором. Позначається стрілкою.

3. Спадкування - показує, що нащадок успадкує атрибути і поведінку свого прямого предка. Може застосовуватися як для акторів, так для варіантів використання.

4. Розширення (extend) - показує, що варіант використання розширює базову послідовність дій і вставляє власну послідовність. При цьому на відміну від типу відносин "включення" розширена послідовність може здійснюватися в залежності від певних умов.

5. Включення - показує, що варіант використання включається в базову послідовність і виконується завжди (на малюнку не показаний).

Діаграма комунікації (Communication diagram, в UML 1.x - діаграма кооперації, collaboration diagram) - діаграма, на якій зображуються взаємодії між частинами композитної структури або ролями кооперації. На відміну від діаграми послідовності, на діаграмі комунікації явно вказуються відносини між елементами

(об'єктами), а час як окремий вимір не використовується (застосовуються порядкові номери викликів). Діаграми послідовності, і діаграми комунікації представляють схожі базові концепції, але з різних точок зору. Діаграми послідовності описують тимчасову послідовність, а комунікаційні діаграми - структури даних, через які проходить потік повідомлень. Діаграма комунікацій призначена для представлення взаємодії в контексті внутрішньої архітектури системи і переданих повідомлень. Діаграма комунікації має вигляд графа, вершинами якого є частини композитного класу або ролі взаємодії, зображені у вигляді прямокутників. Ці вершини відповідають лініям життя і зображуються в своєму структурному контексті. Ребрами графа є зв'язку, за якими проходять маршрути комунікації. Лінії життя можуть обмінюватися повідомленнями, які зображуються у вигляді невеликих стрілок з деяким ім'ям, розташованих біля ліній зв'язків.[13]

Діаграма послідовності (Sequence diagram) - діаграма, на якій показані взаємодії об'єктів, впорядковані за часом їх прояву. Зокрема, на ній зображуються об'єкти, що беруть участь у взаємодії і послідовність повідомлень, якими вони обмінюються. На діаграмі послідовності зображаються виключно ті об'єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об'єктами. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об'єктів в часі. При цьому діаграма послідовності має як би два виміри. Одне - зліва направо у вигляді вертикальних ліній, кожна з яких зображує лінію життя окремого об'єкта, який бере участь у взаємодії. Графічно кожен об'єкт зображується прямокутником і розташовується у верхній частині своєї лінії життя. Усередині прямокутника записуються ім'я об'єкту і ім'я класу, розділені двокрапкою. При цьому вся запис підкреслюється, що є ознакою об'єкта, який, як відомо, являє собою екземпляр класу.

Діаграма співпраці - цей тип діаграм дозволяє описати взаємодію об'єктів, абстрагуючись від послідовності передачі повідомлень. На цьому типі діаграм в компактному вигляді відображаються всі прийняті і передані повідомлення конкретного об'єкта і типи цих повідомлень.

Діаграма огляду взаємодії (Interaction over view diagram) - різновид діаграми діяльності, що включає фрагменти діаграми послідовності і конструкції потоку управління. Цей тип діаграм включає в себе діаграми Sequence diagram (діаграми послідовностей дій) і Collaboration diagram (діаграми співробітництва). Ці діаграми дозволяють з різних точок зору розглянути взаємодію об'єктів в створюваній системі.

Діаграма синхронізації (Timing diagram) - альтернативне подання діаграми послідовності, явно показує зміни стану на лінії життя із заданою шкалою часу. Може бути корисна в додатках реального часу.[14]

2.1.1. Можливості UML

- UML об'єктно-орієнтована, в результаті чого методи опису результатів аналізу і проектування семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах;
- UML дозволяє описати систему практично з усіх можливих точок зору і різні аспекти поведінки системи;
- Діаграми UML порівняно прості для читання після досить швидкого ознайомлення з його синтаксисом;
- UML розширює і дозволяє вводити власні текстові та графічні стереотипи, що сприяє його застосування не тільки в сфері програмної інженерії;[15]
- UML набув широкого поширення і динамічно розвивається.

2.2. Аналіз середовища PowerDesigner

PowerDesigner - це засіб проектування, що сполучить можливості об’єктно-орієнтованого, концептуального й фізичного моделювання об'єктів у єдиному, інтегрованому середовищі. Система підтримує роботу більш ніж з 3О найбільш популярними СУБД, дозволяючи дотримуватися єдиної стратегії створення бізнес-логіки й проектування баз даних.[16]

PowerDesigner є комплексним рішенням для моделювання й розробки

додатків і бізнес-процесів для організацій, які мають потребу у швидкому, послідовному й ефективному з погляду витрат створенні або реінжинірингу бізнесів-додатків. PowerDesigner вирішує такі проблеми як: розходження в професійній підготовці учасників проекту, різнорідні платформи мов розробки. Це дозволяє фокусуватися на бізнес-потребах створення додатків протягом усього процесу розробки - від системного аналізу й дизайну й аж до безпосередньої генерації коду для додатка. Остання версія продукту, PowerDesigner 16.1.0367, має нові можливості по моделюванню бізнес-процесів, об'єктному моделюванню, що базується на UML, і підтримує як традиційні, так і знову, що з'являються технології, моделювання в рамках одного розвитий графічного середовища. Одним з основних переваг PowerDesigner є також використання репозиторія масштабу підприємства для зберігання й керування всією інформацією, що стосується моделювання й дизайну додатків на всіх рівнях ведення бізнесу в компанії. Це дозволяє правильно організувати робочий процес і кардинальний образ підвищити ефективність роботи розроблювача.

PowerDesigner поставляється з концептуально новим репозиторієм, що дозволяє здійснювати моделювання систем у масштабі підприємства.

Проектувальники, аналітики, адміністратори баз даних, будь-які інші IT-фахівці можуть тепер спільно використовувати інформацію й метадані по створенню інформаційної системи підприємства, що зберігаються в єдиному репозиторії. Надаючи можливості розмежування прав доступу й повноважень, підтримки повторного використання об'єктів, пошуку об'єктів і керування версіями моделей, репозиторій PowerDesigner фактично є рішенням масштабу підприємства для надійного зберігання й централізованого керування всією інформацією з ведення проекту. Подібна архітектура дозволяє легко нарощувати можливості створюваної системи, додаючи нові модулі й діаграми в міру появи нових потреб і подальшого розвитку технологій моделювання. Крім того, потужний репозиторій PowerDesigner може поповнюватися в міру розширення вашої команди розроблювачів.

PowerDesigner поєднує. можливості проектування реляційних баз даних і моделювання об'єктів на базі мови UML, дозволяючи аналітикам, проектувальникам і розроблювачам працювати в спільно використовуваному середовищі й створювати погоджені й надійні додатки. От лише деякі із численних переваг, надаваних PowerDesigner.

– Моделювання бізнес-процесів: PowerDesigner дозволяє нетехнічним фахівцям компанії розробляти й моделювати бізнес-процеси, орієнтуючись на бізнес-завдання й опираючись на відомі їм терміни, використовуючи просту й інтуїтивно зрозумілу графічну нетехнічну модель. Включено підтримку генерації й реінжинірингу ebXML-коцу.

– Моделювання даних: PowerDesigner дозволяє розробляти й генерувати схему БД за допомогою дворівневого (концептуального й фізичного) моделювання реляційної БД, що підтримує класичні методики проектування баз даних. PowerDesigner має також убудовані засоби моделювання сховища даних.

– Об'єктне моделювання: PowerDesigner пропонує закінчену технологію аналізу й проектування систем з використанням стандарту UML (діаграми бізнес-процесів, послідовності виконання, класів і компонентів). На основі діаграми класів PowerDesigner автоматично здійснює генерацію й реінжиніринг коду для популярних

інструментальних середовищ, таких як JavaTM, XML, Web Servicies, C++, PowerBuilder, Visual Basic і інших, за допомогою генератора, що набудовується.

- Репозиторій підприємства: Enterprise- версія

PowerDesigner містить функціональність репозиторія класу підприємства. Репозиторій дозволяє всім членам вашої команди легко переглядати моделі й іншу інформацію, а також здійснювати обмін ними. Репозиторій має високу масштабованість і підтримує систему безпеки, засновану на ролі користувача, контроль версій, пошук і можливості складання звітів. 3 початком використання переваг електронного бізнесу кваліфіковані IT-фахівці створюють правильно спроектовані IT-системи, які можуть надати гарантовано стійку платформу для

збільшення ефективності ведення бізнесу поза залежністю від прийнятих вищим менеджментом рішень. IT-фахівці сьогодні створюють інформаційні системи, які дозволяють підтримувати й повторно використовувати існуючий бізнес-логікові відповідно до завдань, що змінюються бізнес-підприємства. Однак IT- фахівці переконуються в тім, що традиційні технології проектування й моделювання не відповідають потребам динамічною, зв'язаною мережею електронних комунікацій. На даний момент, мову об’єктно- орієнтованого проектування UML (Unified Modeling Language) надає потужне й гнучке рішення, але специфікація мови сама по собі не визначає структуру засобів проектування й розробки. PowerDesigner є тією практичною реалізацією об'єктної парадигми мови UML, що дозволяє на практиці реалізувати всі сучасні методи розробки й створення інформаційних систем масштабу підприємства. [17]

Загальний інтерфейс середовища PowerDesigner можна побачити рис. 2:

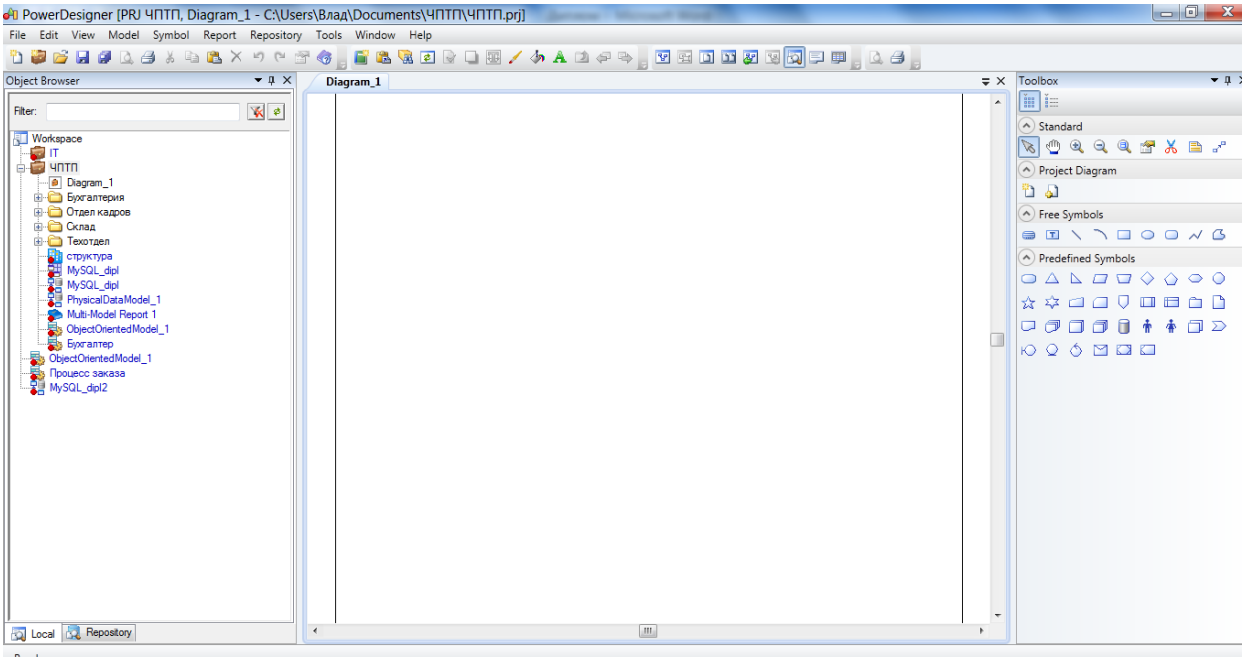

Рис. 2 Скріншот інтерфейсу PowerDesigner

Browser. Інформація про всі об'єкти додатка у вікні перегляду представлена використанням зручної для сприйняття деревоподібної структури (мал. 1.3.2). Ви можете використовувати вікно перегляду для навігації по моделі, відображуваної у вікні редагування діаграм. Об'єкт у вікні редагування діаграм PowerDesigner вибирається простим клацанням миші на об'єкті у вікні перегляду. Навіть якщо

діаграма включає більше тисячі об'єктів, вікно перегляду надасть швидкий і зручний спосіб для переміщення до необхідного об'єкта й вибору будь-якого іншого об'єкта в діаграмі.

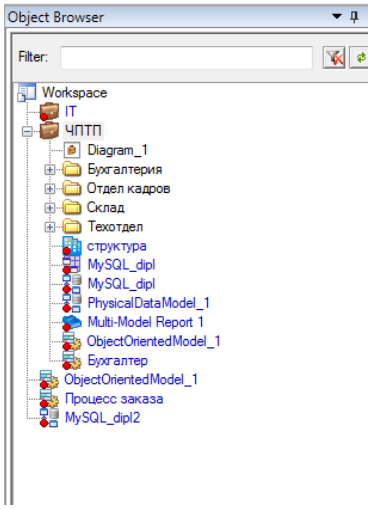

Рис. 3 вигляд браузера

Вікно редагування діаграм дозволяє переглядати діаграми всіх об'єктів моделі (рис. 3). Це набудоване вікно, що може бути використане для відображення концептуальних, об'єктно-орієнтованих, фізичних моделей даних або сукупних багато модельних звітів. Є також можливість розбивати
вікно на кілька сегментів для відображення двох і більше моделей того самого додатка.[18]

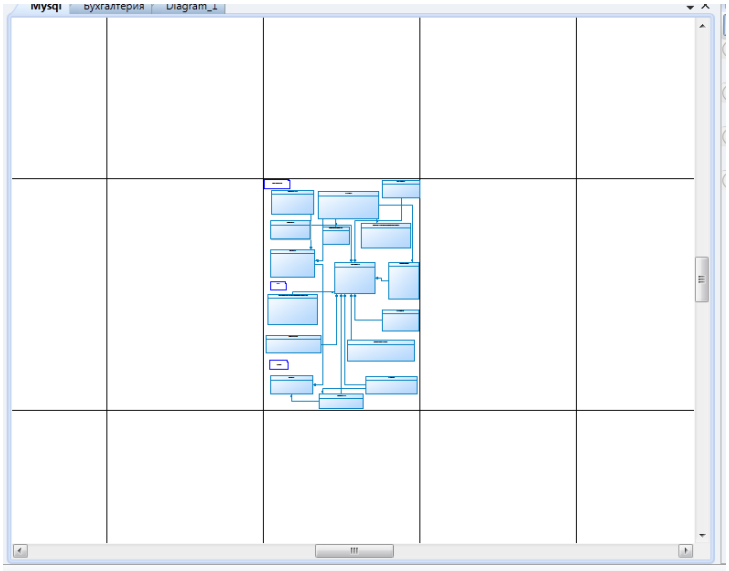

Рис. 4 Вікно редагування діаграм

Панель інструментів. Панель інструментів відображає ті інструменти проектування , які вам потрібні, тоді, коли вони вам потрібні, динамічно змінюючись відповідно до тому типу моделі, над яким ведеться робота. Ви можете переміщати панель інструментів по екрані, вибираючи найбільш зручне й доступне її розташування, або розмістити її на лінійці інструментів.

Моделювання баз даних в PowerDesigner здійснюється з використанням двухрівневого підходу, що дозволяє вести розробку баз даних ітеративно. Для розроблювачів - це насамперед можливість одночасної роботи на концептуальному й фізичному рівнях моделей баз даних з автоматичною синхронізацією внесених змін. Існує можливість вибору між роботою з незалежної від бази даних позначка-інформацією на концептуальному рів або роботою 3 фізичною структурою моделі, реалізованої для конкретно СУБД.

PowerDesigner також надає можливість об’єктно-орієнтованої моделювання за допомогою UML-Діаграм. Діаграми UML дозволяю користувачам проектувати бізнес-логікові інформаційної системи. Ієрархія класів може створюватися на високому концептуальному рівні використанням режиму "аналіз" або більше специфічним образом для робо із цільовими мовами програмування, такими як Java або PowerBuilder. При розробці інформаційної системи пакет PowerDesigner дозволяє зв'язувати моделі даних з діаграмами класів, що дає можливість розроблювачам баз даних і розроблювачам додатків використовувати єдине програмне середовище при розробці складних баз даних.

PowerDesigner максимально спрощує процеси імпорту існуючих файл прямого й зворотного проектування додатків і генерування коду:

- Пряме проектування перетворить діаграму класів або концептуальну модель у фізичну модель або переносить бізнес-логікові з діаграми класів у базу даних, сервер додатків або клієнтські додатки.
- Зворотне проектування генерує фізичні моделі з інформації про структуру баз даних або створює діаграму класів на основі існуючої бізнесу-логіки.[19]

- Генератор коду створює стандартні класи Java, компоненти JavaBeans або PowerBuilder з діаграм класів. Генератор коду також здійснює реверс-інжиніринг вихідних текстів мовою Java, відкомпільованих Java класів, Java-Архівів (файли JAR) або коду PowerBuilder у діаграму класів PowerDesigner.
- Імпортування із програми ERwin переводить ERX-Файли ERWIN у фізичні або концептуальні моделі даних PowerDesigner.
- Моделі Rational Rose також можуть бути без додаткової переробки завантажені як діаграми класів PowerDesigner.

Опція PowerDesigner MetaWorks дозволяє збільшити ефективність спільного використання моделей і спільної роботи, надаючи централізований репозиторій для моделей PowerDesigner і інших документів, створених поза середовищем PowerDesigner.

MetaWorks надає розроблювачам переваги наочного подання й зручності організації процесу колективного моделювання при роботі з усіма наявними в продукті PowerDesigner моделями, дозволяючи здійснювати централізоване й ефективне керування всіма проектами, що ведуться, і розробками.

MetaWorks надає наступні можливості:

- Імпорт умісту репозиторія у ваш локальний робочий простір.
- Експорт моделей і документів з вашого робочого простору в репозиторій.
- Порівняння й об'єднання різних моделей і версій у локальному робочому просторі й репозиторії або усередині репозиторія.
- Пошук об'єктів у моделі, і їхнє повторне використання в інших і моделях.

Керування доступом користувачів до вмісту репозиторія, що забезпечує гарантовану безпеку загального робітничого середовища. PowerDesigner MetaWorkss надає розширені кошти контролю версій і PowerDesigner має модульну структуру, що робить продукт надзвичайно ефективним за вартістю засобом проектування. Можливість вибору різних рівнів функціональності дозволяє здобувати тільки ті можливості, які дійсно необхідні. Модуль

PowerDesigner DataArchitect (PDM, CDM) підтримує інтегроване фізичне й концептуальне моделювання даних, надаючи аналітикам даних, проектувальникам баз даних і адміністраторам баз даних
можливість проектування й генерації баз даних для більш ніж 30 серверних і настільних платформ СУБД. ObjectArchitect поєднує в одному модулі можливості DataArchitect і можливості побудови діаграм класів. Це полегшує спільну роботу проектувальників баз даних і розроблювачів додатків, забезпечуючи координацію їхніх зусиль при роботі над спільним проектом і обмін інформацією між учасниками різних проектів.

Модуль PowerDesigner Developer (PDM, DOM), створений спеціально для розроблювачів, генерує класи Java, JavaBeans і PowerBuilder з діаграм класів мови UML, а також дозволяє здійснювати реверс-інжиніринг із коду мови Java або PowerBuilder і існуючих баз даних у діаграми класу.

2.2.1. Призначення PowerDesigner

Sybase PowerDesigner пропонує методи, необхідні для збору бізнес-вимог, і дозволяє створювати моделі бізнес-процесів, необхідні аналітикам. PowerDesigner перетворює ці моделі в моделі бізнес-процесів, моделі даних і моделі архітектури додатків, слідуючи принципам розробки, заснованим на застосуванні моделей. Потім PowerDesigner можна використовувати для автоматичного створення більш детальних моделей на основі вже сформованих архітектурних уявлень і для створення базису для розробки програмного коду.[20]

Якщо розробка ведеться в середовищах на основі Microsoft .NET або Java / J2EE, PowerDesigner може інтегруватися з Eclipse і Eclipse JDT, або може взаємодіяти з Microsoft Visual Studio .NET для розробки програм на мовах C # і VB.NET. Оптимальний цикл проектування є запорукою синхронізації моделей з фактично розроблюваним кодом програми, і відображенням застосованих структур і шаблонів у моделях PowerDesigner.

2.3. Дослідження функціонування фреймворку Liquibase та СУБД

Велика частина додатків, з якими доводиться стикатися по життю, є корпоративними додатками, які оперують великими масивами даних. Команди розробників, що працюють над такими проектами, часто розглядають базу даних як окрему сутність, незалежну від додатка. Така ситуація може виникати через організаційну структуру проекту, коли розробкою бази даних і розробкою програми займаються окремі команди розробників. Іноді це просто звичка розробників. Як би там не було, я помітила, що такий поділ проектів призводить до наступних проблем:

1. ручне внесення змін до БД;
2. різні версії БД у різних учасників команди розробників;
3. непослідовні підходи до внесення змін (в базу даних або дані);
4. неефективні механізми ручного управління змінами при переходах між версіями баз даних.

Liquibase - це система управління міграціями бази даних. Всі перераховані вище неефективні дії заважають розробникам синхронізувати БД. Крім того, через них у кінцевих користувачів додатка можуть виникнути проблеми, пов'язані з несумісністю або пошкодженням даних. Уникнути незручностей ручного підходу дозволяє стратегія зміни бази даних, при якій втручання людини мінімізується. За допомогою комбінації методик та інструментів можна організувати несуперечливе і повторюване управління змінами в структурі бази даних і в самих даних.

Продукт Liquibase, що розвивається з 2006 року, - це безкоштовний інструмент з відкритим вихідним кодом, призначений для міграції з однієї версії бази даних на іншу. Також Liquibase підтримує відкати бази данних. Liquibase підтримує більше 30 команд для рефакторінга баз даних. Цей розділ зачіпає чотири з цих команд: додавання стовпця, видалення стовпця, створення таблиці і управління даними. Liquibase підтримує багато інших команд рефакторінга бази даних, включаючи додавання таблиці перетворення і злиття стовпців.

Liquibase - кросплатформний Java-додаток, а це означає, що файл можна використовувати на Windows, Mac або Linux.

При використанні Liquibase зміни структури бази даних будуть зберігатися в окремих файлах (changelogs), підтримуються формати XML, YAML, JSON або SQL, що дуже зручно. Зміни можна зберігати в одному або безлічі файлів з наступним включенням в основний файл. Другий варіант кращий, тому що дозволяє гнучко організувати застосування і зберігання чейнджлогів.

Liquibase використовує так звані «чейнджсети» (changeset - набір змін), XML-коду для опису операторів DDL. Вони складають файли чейнджлогів (changelog). Використовується Liquibase для версіонування структури бази даних, це дуже зручно, якщо над проектом працює ціла команда розробників.

Для того, щоб внести зміни в БД, нам необхідно створити файл міграції (changeset), посилання на який потрібно буде вказати у файлі журналу змін (changelog), після чого міграція може бути успішно застосована до цільової БД. Беззаперечною перевагою такого підходу є можливість виконання відкату створених змін. Liquibase має вбудовану підтримку відкату деяких типів чейнджсетів, наприклад "createTable". Якщо ми викликаємо Liquibase через командний рядок з аргументом "rollbackCount 1" замість "update", відбудеться відкат останнього чейнджсету, однак інші типи чейнджсетів не можуть бути видаленими автоматично.

При виконанні команди "update" для кожного з чейнджсетів відбувається перевірка, чи був він застосований до схеми. Якщо чейнджсет ще не був застосований, відбувається його виконання. Для цього Liquibase намагається зберегти дані в допоміжній таблиці DATABASECHANGELOGS, що містить вже застосовані чейнджсети, а також їх хеш-значення. Хеши використовуються для того, щоб ніхто не зміг змінити вже виконані чейнджсети.

Виходячи з вищезазначених описів, можна сміливо робити вибір на користь Java – мережеву платформонезалежну мову програмування та її потужних інструментів: Liquibase та Activity, так як вони зможуть автоматизувати, вдосконалити та оптимізувати всі бізнес-процеси на будь-якому підприємстві. А

якщо використати фреймворки Liquibase та Activity у зв'язці для моделювання бізнес-процесів, то таку систему можна буде вважати складною, тому нижче ми розглянемо, що собою представляє у сукупності саме СПС та подивимось на її життєвий цикл.

Вибір системи управління базами даних (СУБД) являє собою складне абл. параметричне завдання і є одним з важливих етапів при розробці додатків баз даних. Обраний програмний продукт повинен задовольняти як поточним, так і майбутнім потребам підприємства, при цьому слід враховувати фінансові витрати на придбання необхідного обладнання, самої системи, розробку необхідного програмного забезпечення на її основі, а також навчання персоналу.

Для порівняння були обрані три СУБД: InterBase, MySQL і MS SQL Server. Порівняння проводилося за п’ятьма основними параметрами: підтримка СУБД механізму тригерів і збережених процедур, зручність і доступність засобів розробки додатків СУБД, перелік підтримуваних операційних систем, мінімальні вимоги до сервера баз даних, і продуктивність.

Для серйозних інформаційних систем найбільш поширеною є клієнт-серверна архітектура, тобто архітектура, в якій під БД виділяється окремий, досить потужний і надійний сервер, мережевий доступ до якого здійснюють кілька клієнтів. Підтримка СУБД механізму тригерів і збережених процедур дозволяє перенести частину обчислювального навантаження по обробці даних на сервер, знижує мережевий трафік, полегшує модернізацію ПО.

MySQL на відміну від Microsoft SQL Server і InterBase не підтримує тригери що є певною мірою недоліком, так як в додатках інформаційної системи більшу частину необхідних перевірок введених даних, а також забезпечення цілісності бази даних доводиться виконувати на рівні клієнтського додатку , що дещо ускладнює процес створення програмного продукту.

2.3.1. Засоби розробки додатків СУБД

Багато виробників СУБД випускають засоби розробки додатків для своїх систем. Як правило, ці кошти дозволяють найкращим чином реалізувати всі

можливості сервера БД, тому при аналізі СУБД варто розглянути також можливості засобів розробки додатків.

У MSSQL Server слід звернути особливу увагу на основний засіб розробки та адміністрування, яке включене до складу дистрибутива, - Enterprise Manager, який дозволяє вирішувати практично всі завдання адміністрування MS SQL Server і, крім того, зручний для розробника.

У InterBase, на жаль, засіб розробки та адміністрування, що поставляється в складі дистрибутива (Interbase Console), недостатньо зручно, але володіє необхідною функціональністю. Тому існують більш зручні засоби розробки і адміністрування, створені сторонніми розробниками, такі як IB Expert, EMS IB Manager.

Схожим чином йде справа і з засобами адміністрування та розробки MySQL, але при бажанні також можна скористатися продуктами сторонніх розробників: EMS MySQL Manager, WinSQL, PHP Admin.

2.3.2. Перелік операційних систем

Під керуванням яких здатна працювати СУБД. В цьому розділі, безумовно, лідирує MySQL, яка здатна працювати під управлінням більшості з наявних на даний час операційних систем.

Таблиця 1 - Сумісність СУБД і ОС

СУБД	ОС
InterBase	Windows 95/98/ME/NT/2000/XP/7 и Linux-системы
MS SQL Server	Windows NT, 2000, XP/7 (Intel и Alpha)

MySQL	Linux (x86, libc6, S/390, IA64, Alpha, Sparc), Windows 95/98/NT/2000/XP/7, Solaris 2.9 (Sparc, 64-bit, 32-bit), FreeBSD 4.x ELF (x86), Mac OS X v10.2, HP-UX 10.20 (RISC 1.0), HP-UX 11.11 (PA-RISC 1.1 или 2.0), AIX 5.1 (RS6000), QNX 6.2.0 (x86), Novell NetWare 6 (x86), SCO OpenUnix 8.0 (x86), м SGI Irix 6.5, Dec OSF 5.1 (Alpha)

Мінімальні вимоги до сервера БД представлені в табл. 1.2. З даної таблиці видно, що найменш вимоглива до ресурсів сервера - СУБД InterBase.

Таблиця 2 - Мінімальні вимоги до сервера БД

СУБД	Сервер
InterBase 7.0	Процесор x86 з тактовою частотою 1,0 ГГц, процесор x64 з тактовою частотою 1,4 ГГ, ОЗУ - 32 Мбайт, 50 Мбайт вільного місця на диску.
MS SQL Server 10.00	Процесор x64: AMD Opteron, AMD Athlon 64, Intel Xeon з підтримкою Intel EM64T, ОЗУ - 512 Мбайт, 6 Гб вільного місця на диску
MySQL 5.5.20	Процесор з тактовою частотою 1200 MHz, або більш потужний. Оперативна пам'ять 256 Мб або більше.Вільне місце на жорсткому диску від 540 Мб. Архітектура з розрядністю 32 біт або 64 біт (x86 або x64).

Отже, розглянуті тут СУБД мають свої переваги і недоліки. Враховуючи мінімальні вимоги до сервера БД і вартість продажу, для проекту було обрано СУБД MySQL.

Для рішень на основі MySQL також характерні легкість використання і управління, продуктивність, масштабованість, переносимість, ефективне використання ресурсів і відновлення після збою. MySQL розроблений саме з метою задовольняти всім цим вимогам.

РОЗДІЛ 3.

Розробка діаграм UML та моделі даних

3.1. Діаграми використання

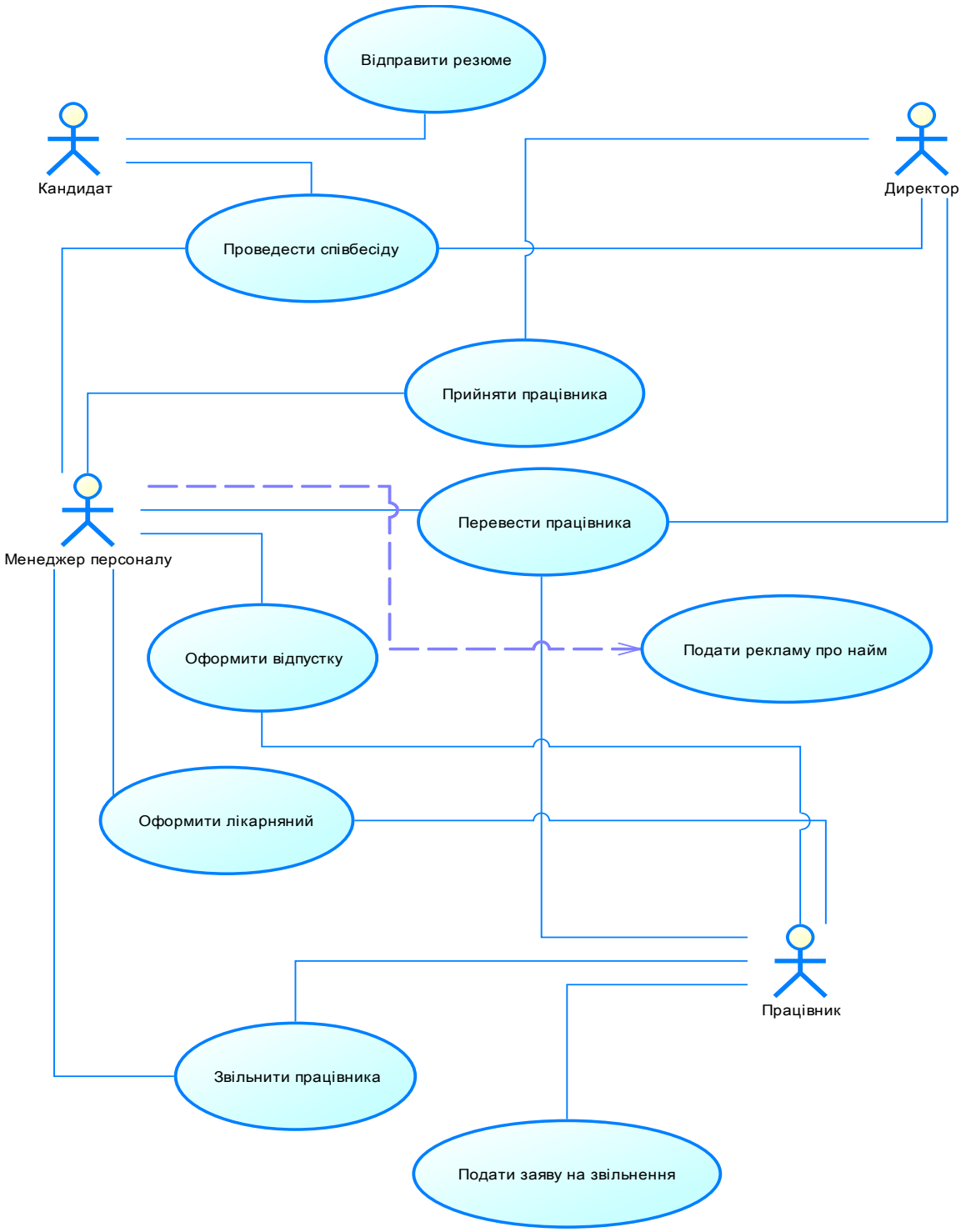

Рис. 5 Діаграма використання для відділу кадрів

Відділ кадрів – це структурний підрозділ системи управління, на який покладаються обов’язки реалізації кадрової політики підприємства. Спеціалісти

цього відділу проводять аналітичну й оперативну роботи, здійснюють виконавчі, розпорядчі, контролюючі та координаційні функції в сфері управління персоналом.

Серед основних функцій, які виконують відділи кадрів можна назвати наступні:

1. Адміністративна діяльність, яка забезпечує виконання діючого законодавства в галузі праці.
2. Організація працевлаштування, яка реалізується через підбір, прийом на роботу, знайомство з робочим місцем, інструктаж щодо умов праці та техніки безпеки на виробництві, перехід з одного робочого місця (посади) на інше, звільнення працівника та ін.
3. Формування резервного потенціалу на підприємстві.
4. Допомога керівництву в розробці форм та систем заробітної плати, системи преміювання, визначення розміру соціальних пільг.
5. Проведення постійного контролю за умовами праці, трудовою дисципліною, технікою безпеки.
6. Розробка положення та проведення атестації персоналу, організація конкурсів на заміщення вакантних посад.
7. Організація системної підготовки, перепідготовки та підвищення кваліфікації персоналу.
8. Співпраця з профспілками та керівництвом організації в розробці та укладанні колективного договору.

Кожен прецедент має пояснення як зображено на рисунку 6.

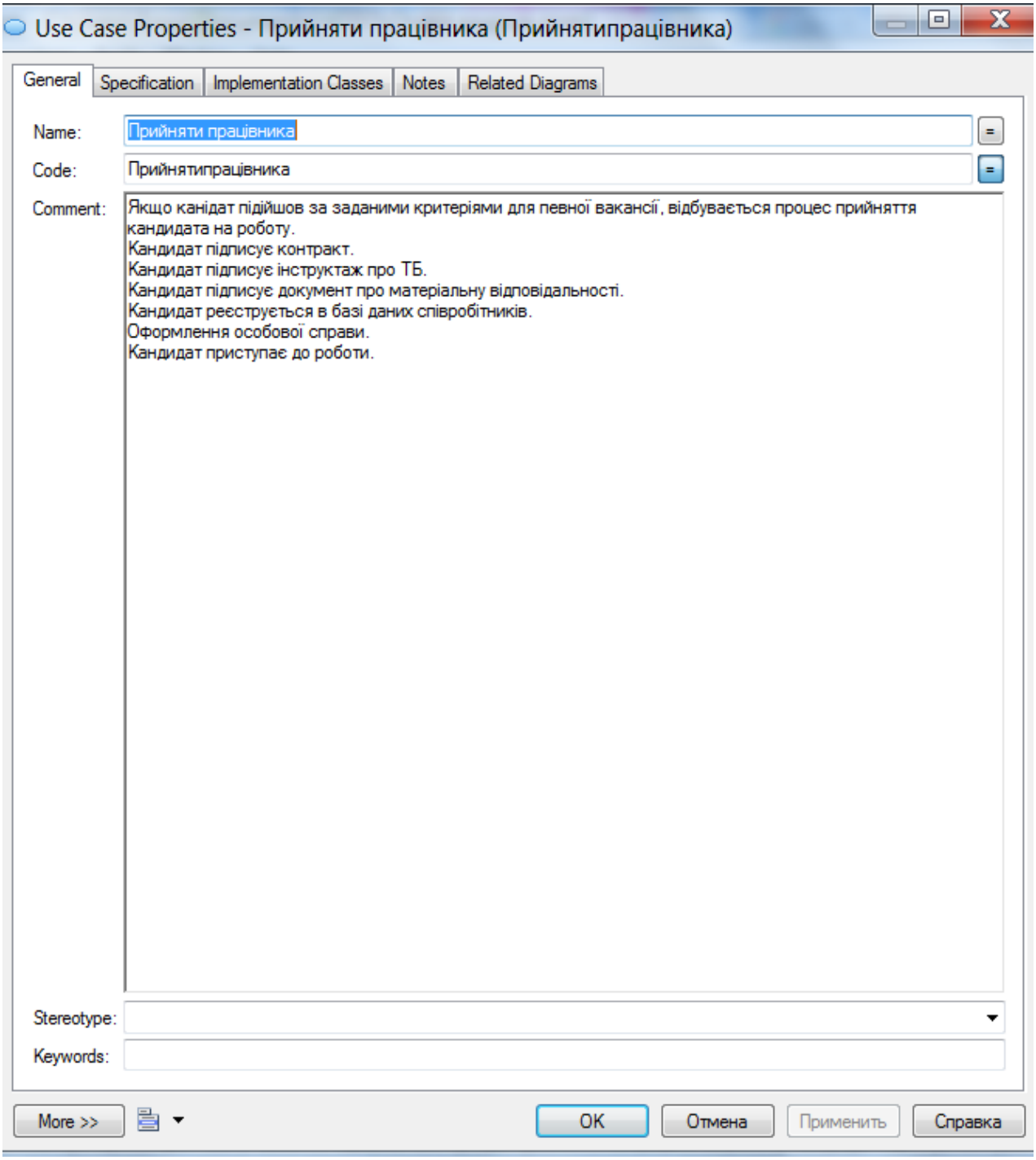

Рис. 6 Пояснення до прецеденту «прийняти працівника» діаграми використання

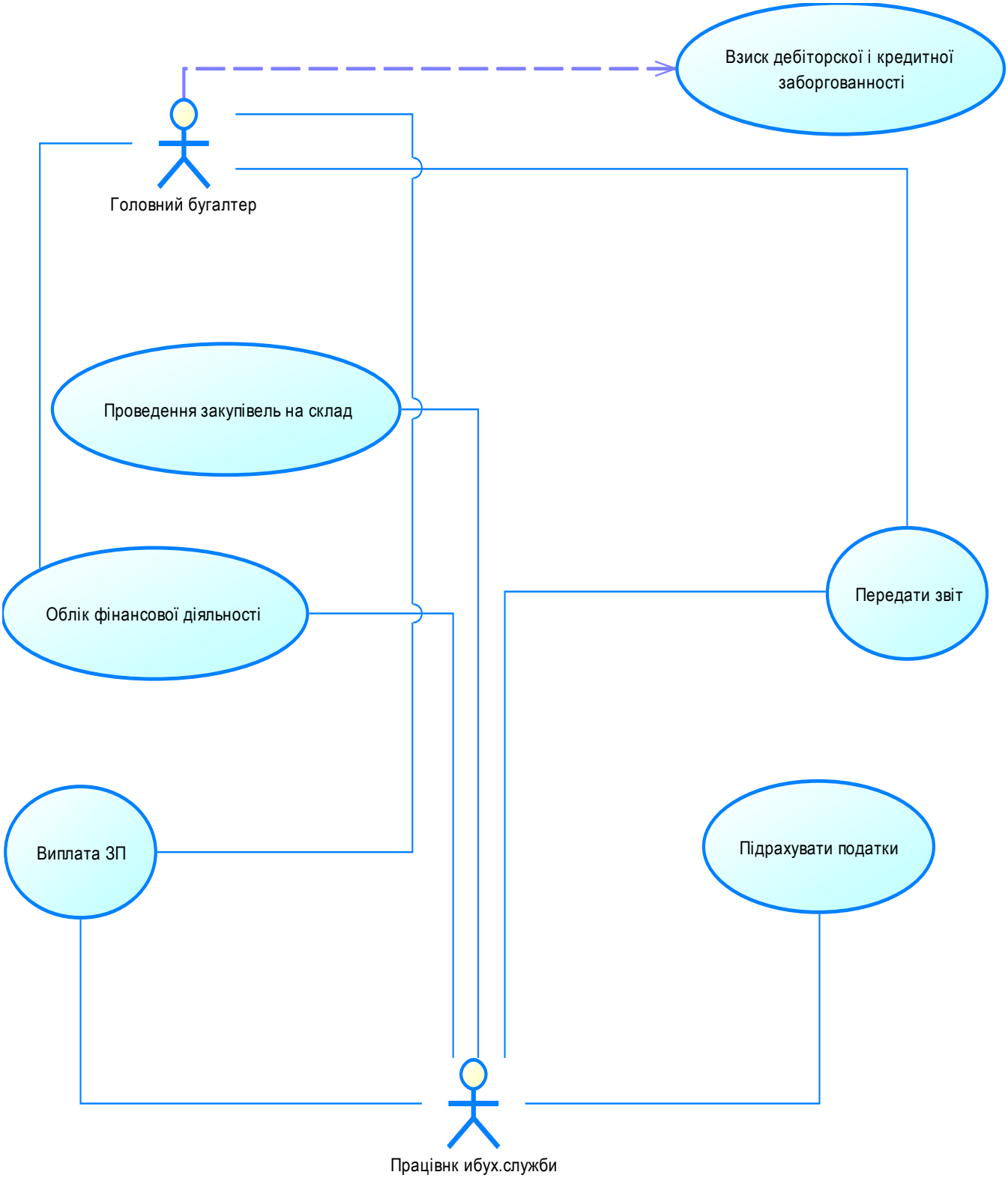

Рис. 7 Діаграма використання для бухгалтерії

Бухгалтерія - штатно-структурний підрозділ господарюючого суб'єкта, призначений для акумулювання даних про його майно і зобов'язання. Бухгалтерія є джерелом документально обґрунтованої і структурованої економічної

інформації, необхідної для прийняття управлінських рішень з метою забезпечення ефективного господарювання.

Основний управлінським завданням бухгалтерського обліку (інакше кажучи, завданням, яке зазнає бухгалтерію) є збір і обробка повної і достовірної інформації про діяльність господарюючого суб'єкта. Така інформація, в основному, використовується в двох цілях:

- Прийняття рішень на основі економічного аналізу такої інформації.
- Здійснення фінансового контролю.

Фінансовий контроль, будучи складовою частиною контролю в діяльності господарюючого суб'єкта, є однією з важливих функцій управління. Він являє собою перевірку дотримання господарюючим суб'єктом в цілому і його працівниками:

- норм законодавства (тобто інтересів держави, суспільства, контрагентів, співробітників підприємства),
- дотримання інтересів власника.

Одним з основних умов реалізації функцій фінансового контролю є наявність об'єктивної та актуальної фінансової інформації. Носіями такої інформації (в докладному і узагальненому вигляді) є бухгалтерські документи:

- первинні облікові документи,
- облікові регістри,
- бухгалтерська звітність,
- матеріали інвентаризацій.

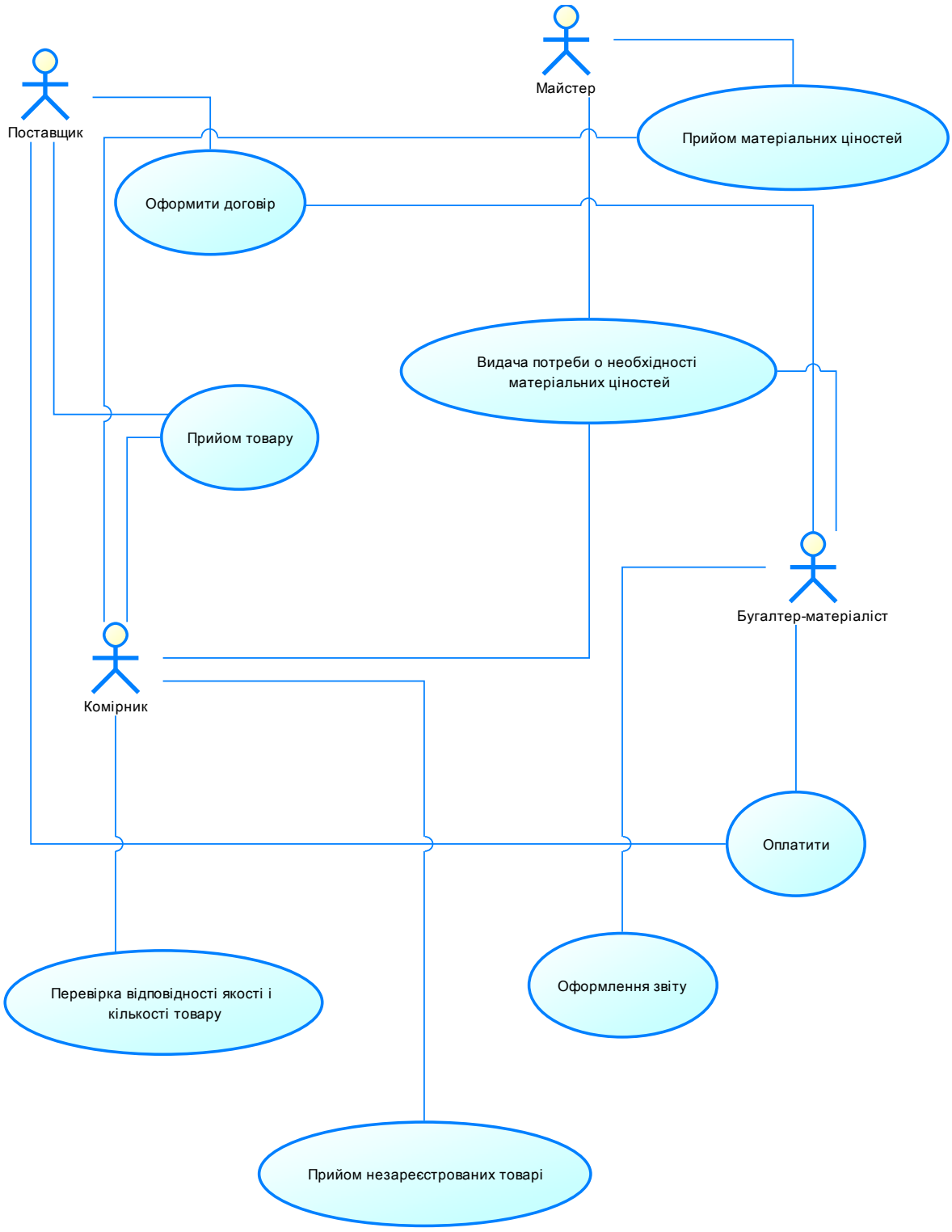

Рис. 8 Діаграма використання для складу

Склад - це нежитлове приміщення, призначене для зберігання сировини, продукції, товарів та інших вантажів, що забезпечує дотримання необхідних умов зберігання і оснащене обладнанням для зберігання та зручними для розвантаження-навантаження конструкціями і спорудами.

Основні зони складу:

- Ділянка розвантаження - механічна або ручна розвантаження транспортних засобів.
- Приймальних експедиція - приймання в неробочий час вантажу і його короткострокове зберігання до відкриття складу.
- Ділянка приймання - прийом вантажу. Перевірка відповідності документів за кількістю і за якістю.
- Зона зберігання - зона складу для розміщення вантажу, зберігання відбору товарів згідно із замовленнями.
- Ділянка комплектування - формування вантажних одиниць згідно із замовленнями.
- Відправних експедиція - короткострокове зберігання, розробка маршрутів, відпуск продукції на вивіз.
- Ділянка навантаження - завантаження транспортного засобу товаром і забезпечення документами.

Комплекс складських операцій, за які відповідає комірник:

1. розвантаження транспорту;
2. прийом товару;
3. розміщення на зберігання;
4. відбір товару з місць зберігання;
5. комплектація і відбір товару;
6. навантаження транспортного засобу;
7. Внутрішньо-складські переміщення вантажу.

На підприємстві на складі зберігаються деталі для виконання проектів. Якщо майстру необхідні деталі, він звертається до комірника з документом про необхідність матеріальних цінностей, якщо товар у наявності він видається майстру, якщо ні – комірник замовляє товар в бухгалтерію.

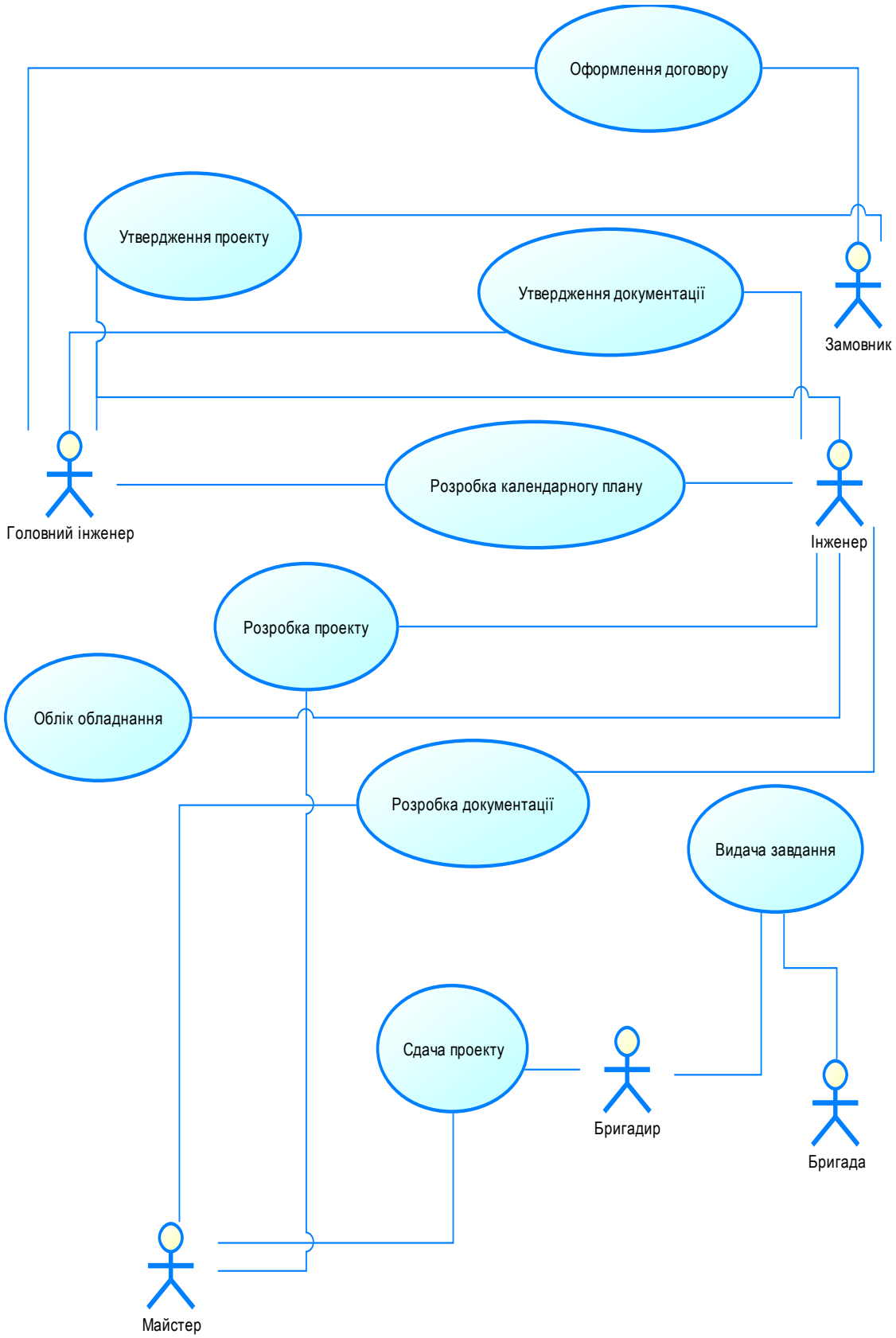

Рис. 9 Діаграма використання для технічного відділу

Функціональні обов'язки технічного відділу полягають у наступному:

Організація технічної підготовки виробництва або інших видів основної діяльності підприємства, забезпечення поліпшення якості продукції, робіт (послуг) і підвищення її конкурентоспроможності, скорочення матеріальних і трудових витрат на виготовлення продукції, виконання робіт (послуг).

Координація роботи технічних служб підприємства по випробуванню нових технічних засобів, створення і освоєння нових видів продукції, комплексної автоматизації та механізації виробництва, плануванню впровадження науково-технічних досягнень, нової техніки і прогресивної технології.

Здійснює поточним і перспективним плануванням технічного розвитку підприємства, його виробничої бази.

Розгляд і погодження проектно-конструкторську документацію з модернізації устаткування та раціоналізації робочих місць.

Контроль за укладанням і виконанням договорів, пов'язаних з впровадженням нової техніки, а також за фінансуванням і правильністю розрахунків економічної ефективності заходів щодо освоєння нової техніки і технології, нових видів сировини і готової продукції.

Бере участь в розробці і впровадженні у виробництво ресурсозберігаючих технологій, прогресивних норм витрати основних видів сировини і матеріалів, у вивченні причин браку і випуску продукції знижених сортів, в розробці заходів щодо підвищення якості продукції (робіт, послуг) і більш ефективному використанню виробничих потужностей. Виконує при відсутності самостійних конструкторських і технологічних відділів функції їх керівників.

Спрямовує діяльність підрозділів, що займаються питаннями стандартизації продукції, науково-технічної інформації, а також організацією патентно-винахідницької роботи.

3.2 Діаграми послідовності

На основі діаграми діяльності відділу кадрів була розроблена діаграма послідовності. Були виділені наступні об'єкти:

- Кандидат;
- менеджер персоналу;

- влаштування;
- працівник;
- неробочі дні;
- звільнені;

Відділ кадрів займається підбором персоналу, прийомом, переводом та звільненням працівників. Кандидат на вільну вакансію відсилає резюме та приходить до менеджеру персоналу, який проводе з ним співбесіду. Якщо менеджер визначив рівень кваліфікації кандидата, і він підходе підприємству, менеджер повідомляє директора про кандидата, який приймає рішення. Після чого менеджер проводе процедуру прийняття на роботу кандидата. Тепер працівник може перевестись, піти у відпустку, лікарняний або звільнитись.

Тож, якщо кандидат підійшов підприємству, то він підписує контракт, інструктаж про ТБ, документ про матеріальну відповідальність. Після чого менеджер персоналу реєструє нового працівника в БД та оформлює особисту справу.

У випадку необхідності менеджер персоналу може перевести працівника на іншу посаду. Працівник подає заяву, і якщо є вільна вакансія і працівник підходить критеріям заявленої вакансії, працівника переводять на іншу посаду.

Працівник має право на відпустку. Якщо йому потрібна відпустка, і він пише відповідну заяву менеджеру персоналу, який визначає строк відпустки. Після чого працівник виходе у відпустку.

Якщо працівник не може в силу фізичного стану вийти на роботу, то він виходе на лікарняний. Після лікарняного працівник видає довідку від лікаря.

У випадку необхідності працівник може бути звільненим. Для того щоб звільнити працівника, йому необхідно не виконувати свої обов’язки або написати відповідну заяву, відпрацювати 14 дні і працівник може бути звільненим.

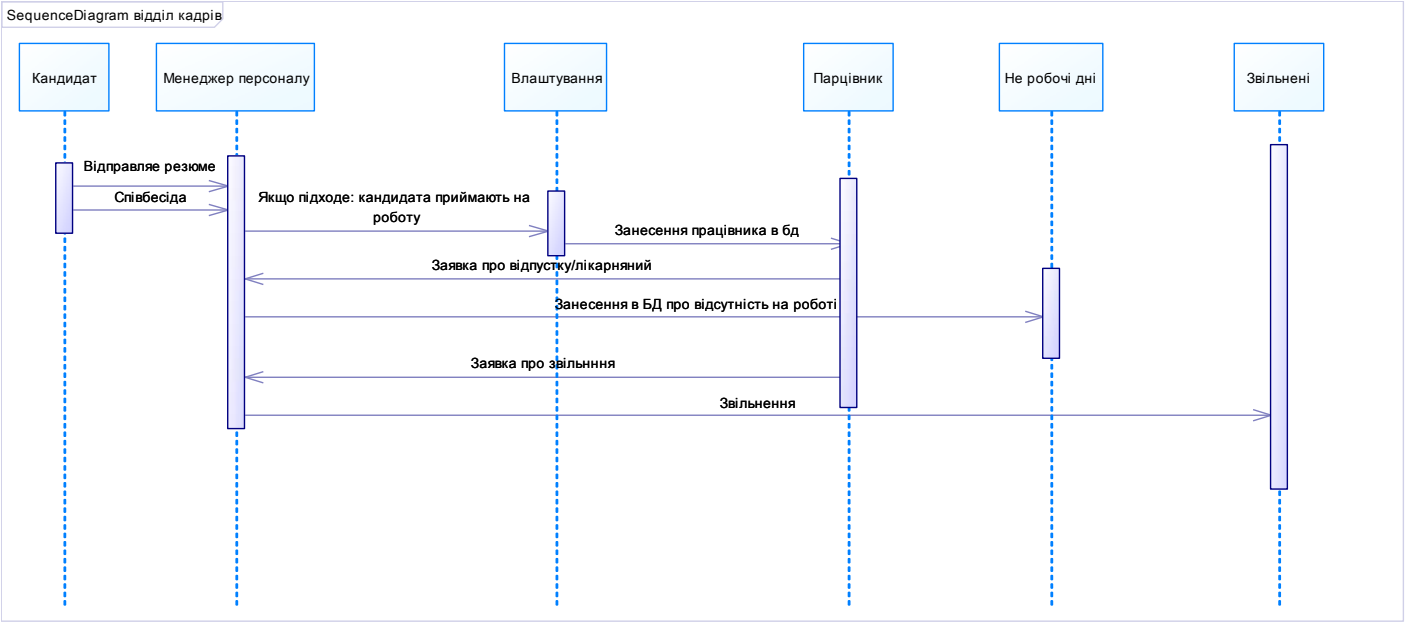

Рис. 10 діаграма послідовності для відділу кадрів

На основі діаграми діяльності бухгалтерії була розроблена діаграма послідовності. Були виділені наступні об'єкти:

1. Бухгалтер;
2. закупівлі;
3. облік фінансової діяльності;
4. виплата ЗП;
5. підрахунок податків;
6. головний бухгалтер.

Працівник бухгалтерії займається:

1. Проведенням закупівель – у випадку необхідності комірник видає вимогу про необхідність придбання товару, після чого бухгалтер купує товар;
2. обліком фінансової діяльності – бухгалтер введе облік про фінансову діяльність підприємства – реалізація, прибуток, фонд, резерв;
3. виплатою зарплати – бухгалтер підраховує зарплату для всіх працівників підприємства і передає звіт головному бухгалтеру;
4. підрахунком податків – бухгалтер перераховує усілякі що тільки можна податки для підприємства.

Після чого складає звіт про виконання роботи та передає головному бухгалтеру.

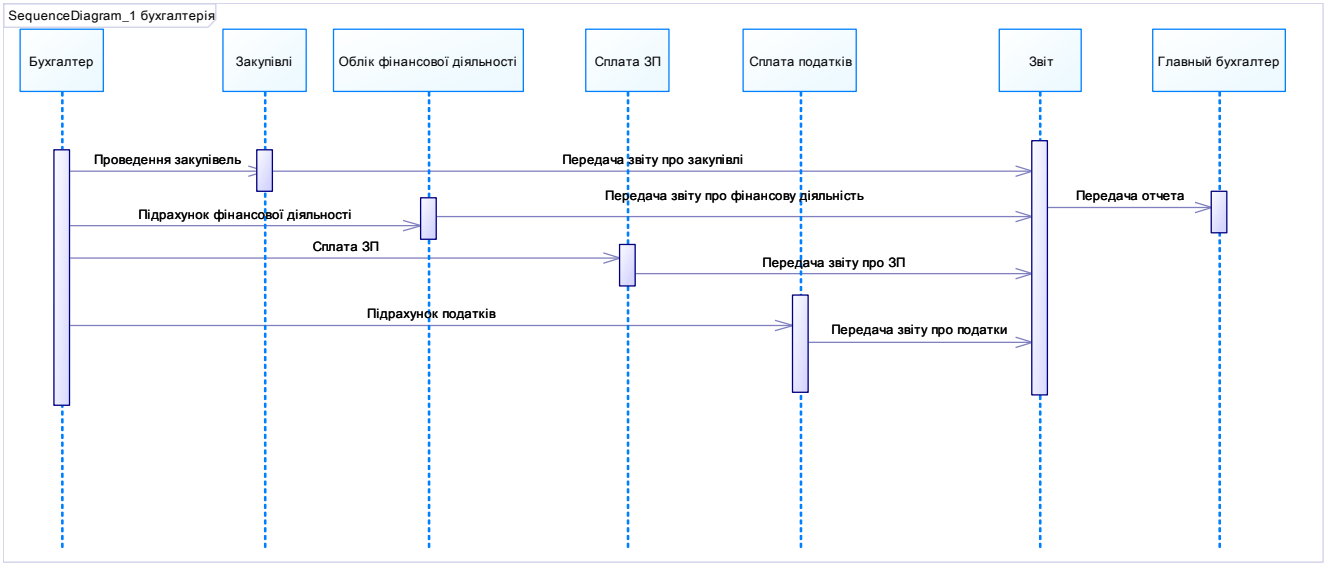

Рис. 11 Діаграма послідовності для бухгалтерії

На основі діаграми використання можна виділити наступні об'єкти:

1. Майстер;
2. Комірник;
3. Склад;
4. Видача;
5. Бухгалтер.

Коли майстер видає вимогу про необхідність матеріальної цінності, комірник перевіряє наявність необхідного товару на складі. Після підтвердження формується запис про видачу, котрому вказується номер виданого екземпляра, ПІБ майстра, поточна дата. Якщо товару немає в наявності на складі, комірник робить замовлення в бухгалтерію про необхідність придбання матеріальної цінності.

Якщо підприємство влаштовує ціна послуг постачальника, оформлюється договір про покупку необхідних товарів. Після прибуття товару на склад, товар оплачується. Якщо постачальник привіз замовлені товари, комірник перевіряє відповідність та якість замовленого товару. Якщо замовлений товар відповідає договору, комірник повідомляє бухгалтера, після чого бухгалтер проводе оплату .

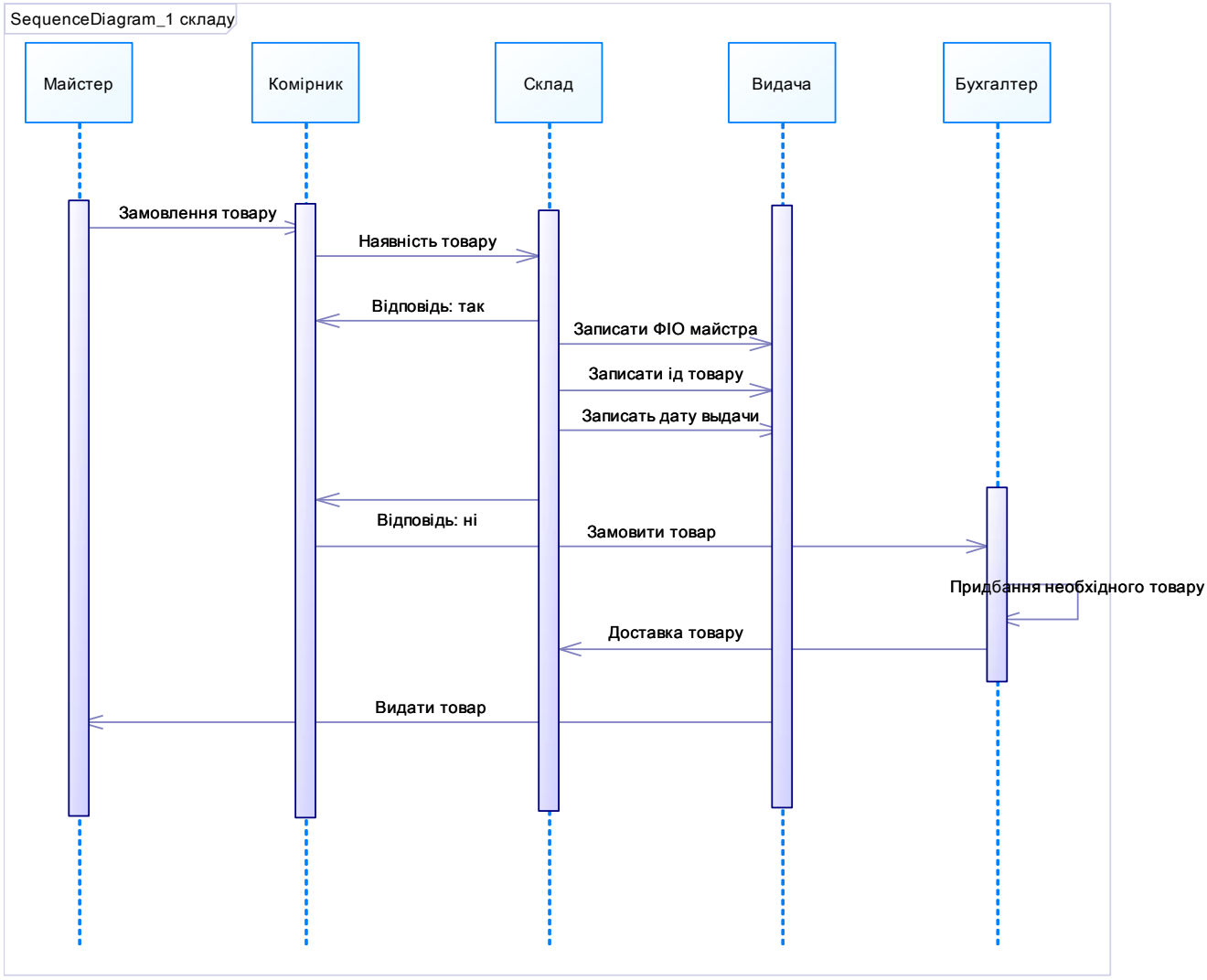

Рис. 12 Діаграма послідовності для складу

На основі діаграми використання можна виділити наступні об'єкти:

1. Інженер;
2. майстер;
3. склад;
4. видача;
5. бухгалтер;

Технічний відділ займається розробкою документацій для проектів та їх реалізацій. Для того щоб виконати проект на підприємстві працюють інженери, які розробляють документацію. Виробничо-технічний відділ веде підготовку виробництва, узгодження проектно-кошторисної документації, підготовку договорів з субпідрядниками, а також спрямовує і систематично контролює

виробничу діяльність будівельно-монтажних управлінь і ділянок. На основі проектно-кошторисної документації виробничо-технічний відділ розробляє робочі проекти і технологічні карти виробництва будівельних робіт, забезпечуючи використання найбільш раціональних новітніх будівельних машин, механізмів та пристосувань, а також передових методів праці. В умовах широкого розвитку спеціалізації і кооперування в будівництві виробничо-технічний відділ здійснює ув'язку роботи спеціалізованих організацій, а також контроль за роботою субпідрядників. При виконанні всіх зазначених функцій виробничо-технічний відділ здійснює інструктаж по відповідній роботі в управліннях та на ділянках.

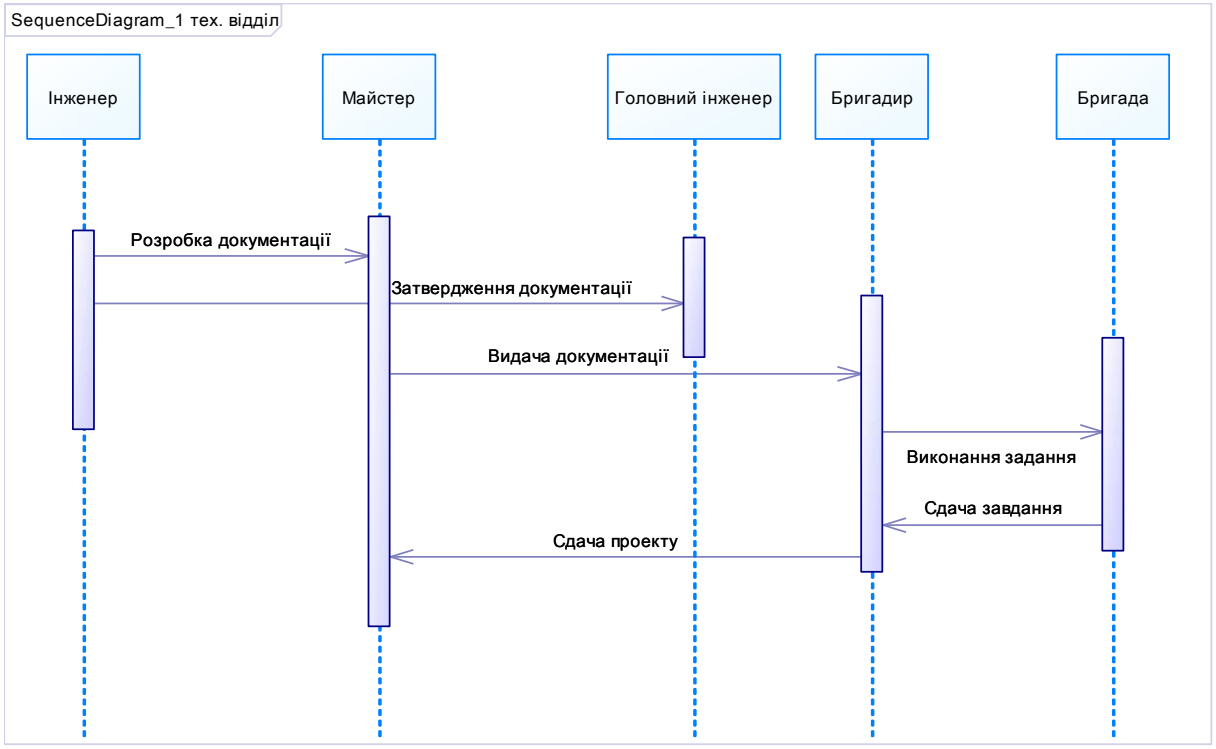

Рис.13 Діаграма послідовності для тех.віділу

3.3 Діаграми кооперацій

Відділ кадрів займається підбором персоналу, прийомом, перекладом і звільненням співробітників. Кандидат на вільну посаду приходить до менеджера персоналу, який проводить з ним співбесіду. Якщо менеджер персоналу визначив рівень кваліфікації кандидата, і він підходить підприємству, менеджер повідомляє директору про кандидата, який приймає рішення про прийом на роботу. Після

чого менеджер проводить процедуру прийняття на роботу кандидата. Тепер співробітник може перевестися, піти у відпустку, на лікарняний або звільниться.

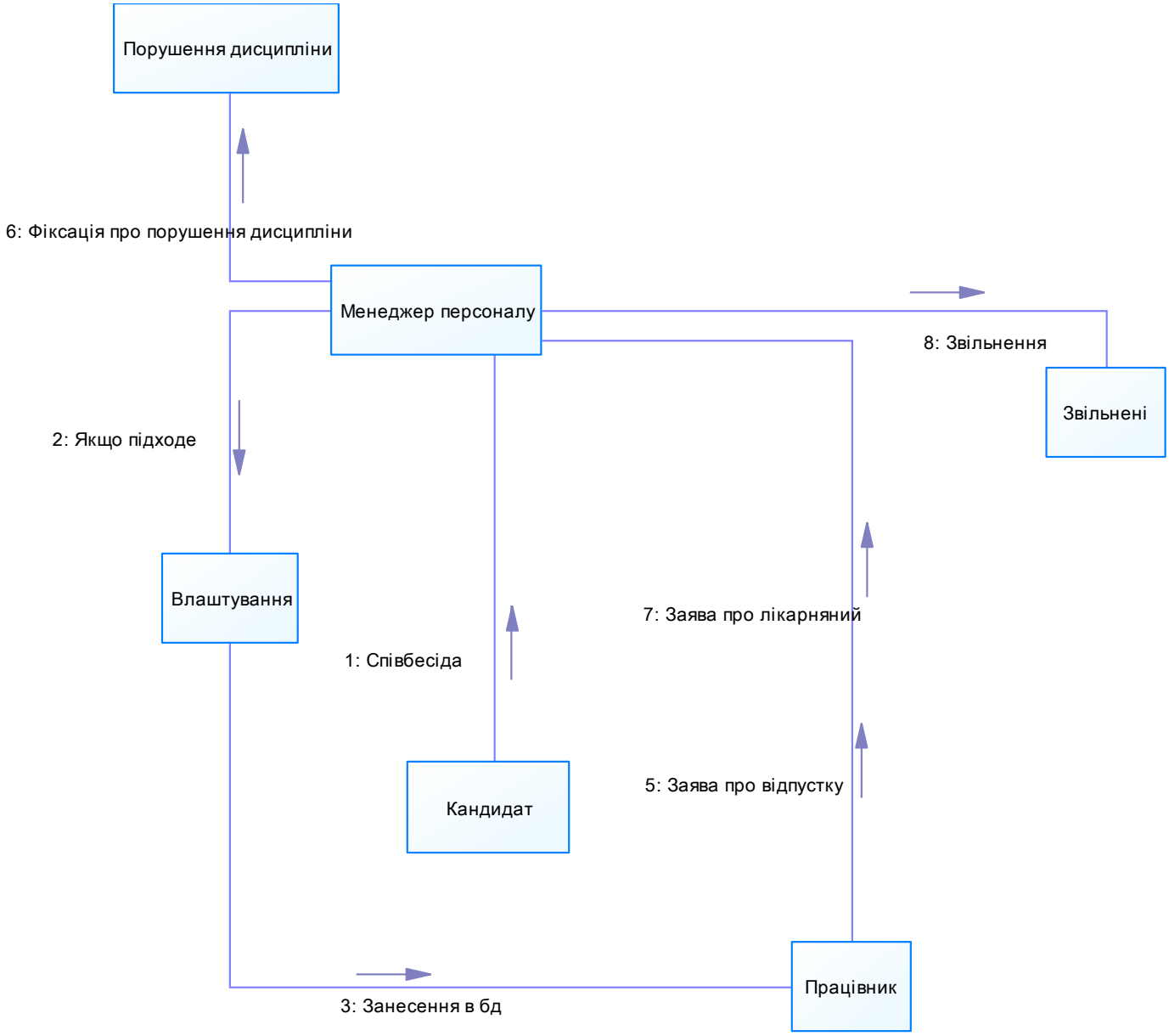

Рис. 14 Діаграма кооперацій для відділу кадрів

Бухгалтерія підприємства проводить облік фінансової діяльності. Бухгалтерія займається проведенням закупок з комірником та майстром, підрахунком усієї фінансової діяльності, сплатою зарплати, підрахунком податків та звітує головного бухгалтера про закупівлі, сплату податків, сплату зарплатні з урахуванням усіх дисциплінарних порушень або премій та всю іншу виконану роботу.

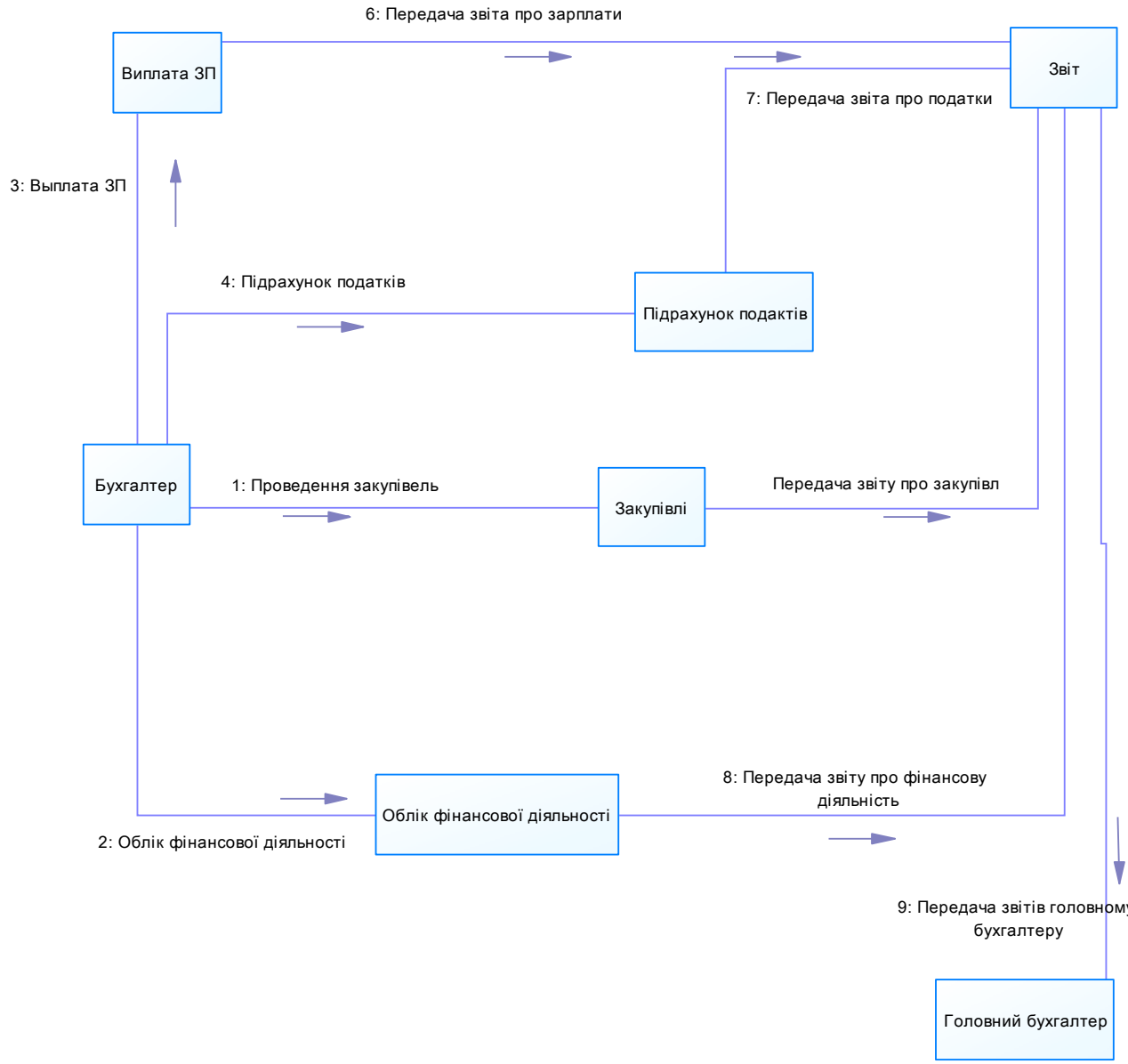

Рис. 15 діаграма кооперації для бухгалтерії

На складі зберігаються усі матеріальні цінності закуплені та необхідні для виконання проектних робіт технічного відділу. Спочатку майстер замовляє комірнику матеріальні цінності, після чого майстер перевіряє наявність товару на складі і у випадку наявності видає документ про видачу матеріальних цінностей, якщо товару немає в наявності, то комірник повідомляє бухгалтера про необхідність придбання товару. Після замовлення товар потрапляє на склад, де його перевіряє комірник і у випадку відповідності товару заявленій якості, товар передається майстру.

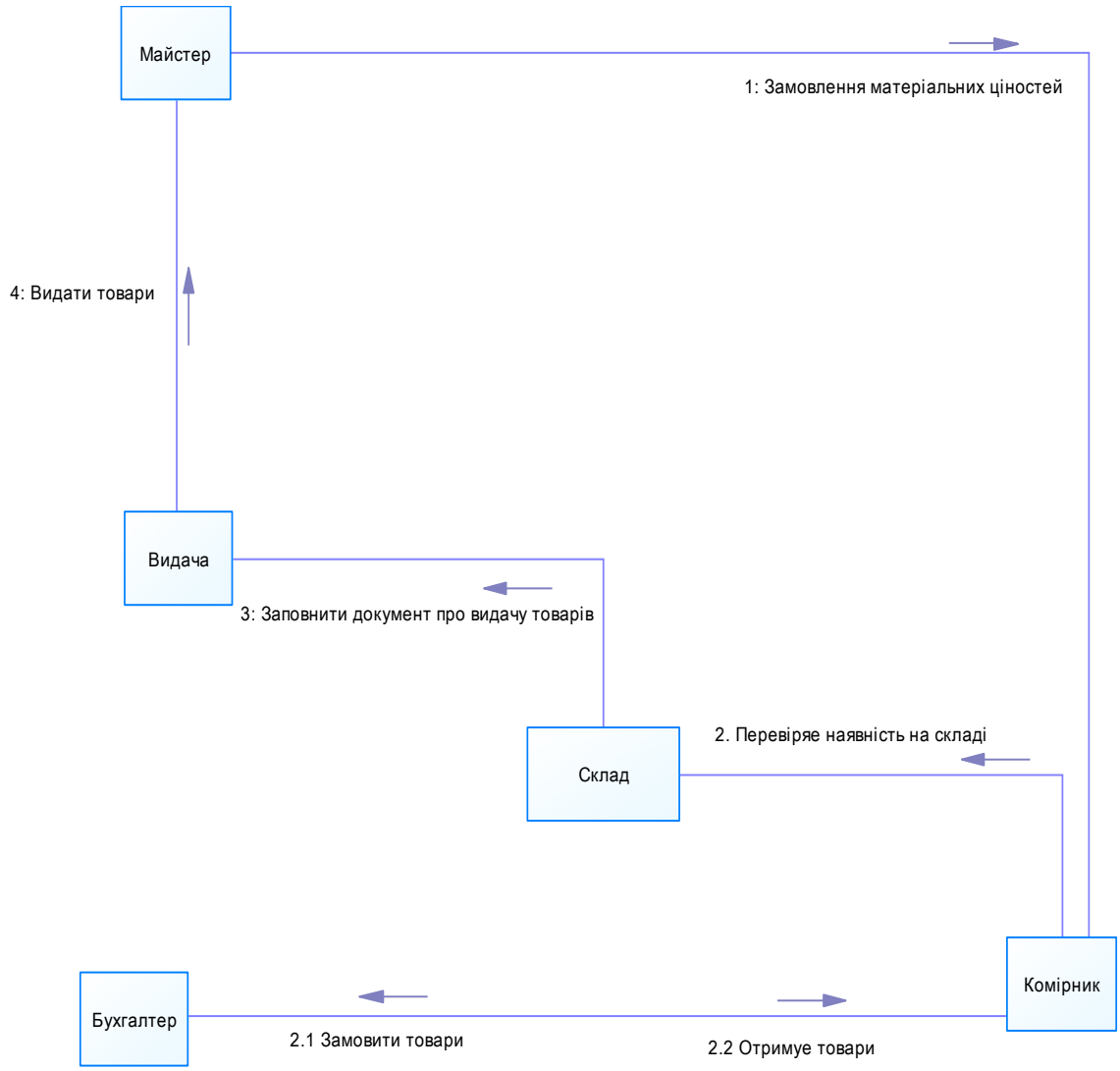

Рис. 16 діаграма кооперацій для складу

Тех. відділ займається розробкою документації і виконанням проектів. Інженер разом з майстром розробляє документацію, після чого вона затверджується головним інженером. Після затвердження – документація передається майстру, який відповідає за виконання проекту. Майстер ознайомлює бригадира з документацією і підписує договір про виконання проекта. Після бригада приступає до виконання проекту і здається акт виконаних робіт майстра, який оцінює і підписує виконаний проект.

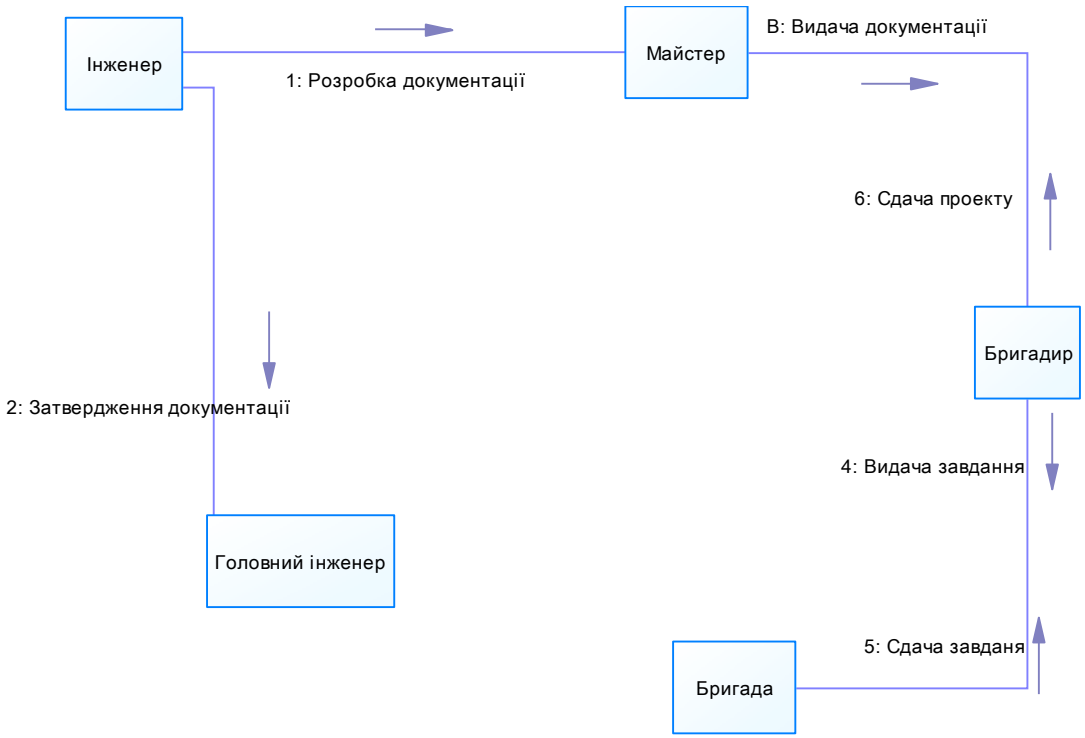

Рис. 17 Діаграма кооперацій для тех. відділу

3.4 Діаграма логічної моделі даних

На початку роботи із середовищем PowerDesigner, варто визначитися з типом моделі даних

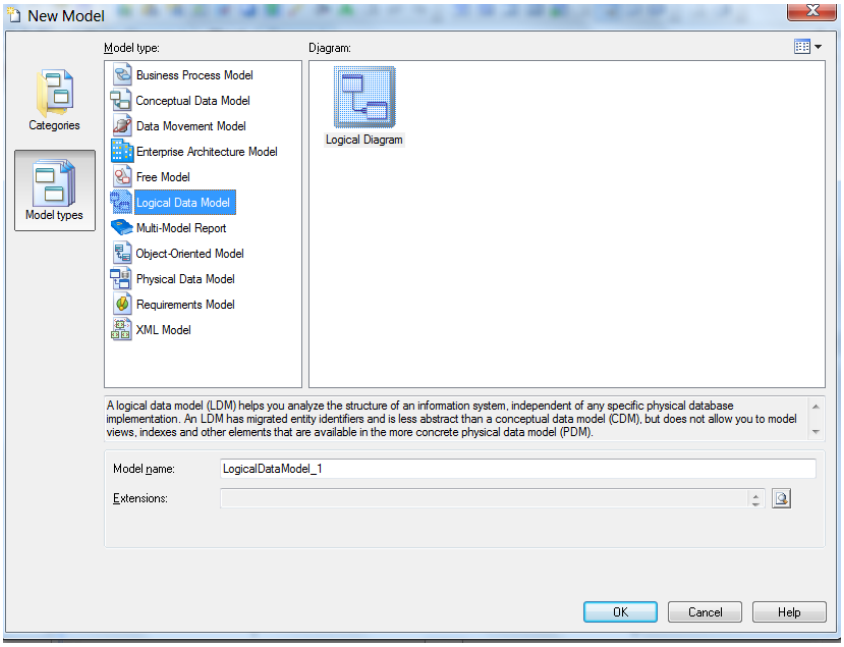

Рис. 18 Вибір моделі даних

При реалізації логічної моделі даних, був використаний тип моделі Logical Data Model. Логічна модель відображає логічні зв'язки між елементами даних незалежно від їх змісту і середовища зберігання.

Логічна модель даних може бути реляційної, ієрархічної або мережевий. Користувачам виділяються підмножини цієї логічної моделі, які називаються зовнішніми моделями (в деяких джерелах їх також називають підсхема), що відображають їхнє представлення про предметну область. Зовнішня модель відповідає уявленням, які користувачі отримують на основі логічної моделі, в той час як концептуальні вимоги відбивають уявлення, які користувачі спочатку бажали мати і які лягли в основу розробки.

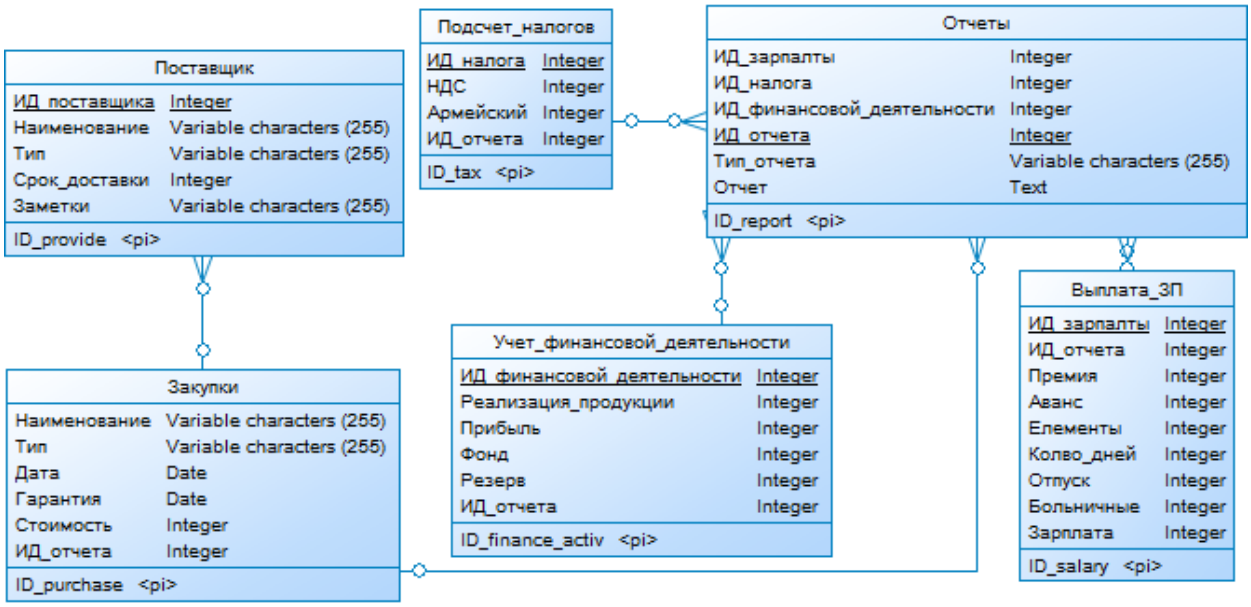

Рис. 19 Логічна модель даних для бухгалтерії

На рис.19 зображена логічна модель даних для бухгалтерії. Для того, щоб зберігати дані про бухгалтерську службу необхідно створити таблиці, які будуть містити інформацію про:

1. Постачальники.
2. Звіти.
3. Підрахунок податків.
4. Звіт фінансової діяльності.
5. Закупівлі.

6. Виплата зарплати.

Таблиця «постачальники» містить інформацію про найменування, тип, строк доставки та замітки.

Таблиця «звіт» містить інформацію про всі звіти які складають бухгалтери.

Таблиця «підрахунок податків» містить інформацію про ПДВ та військовий податок.

Таблиця «звіт фінансової діяльності» містить інформацію про реалізацію продукції, прибуток, фонд, резерв.

Таблиця «закупівлі» містить інформацію про найменування, тип, дату, гарантію, вартість.

Таблиця «виплата зарплати» містить інформацію про премії, аванси, елементи, кількість днів, відпустка, лікарняний.

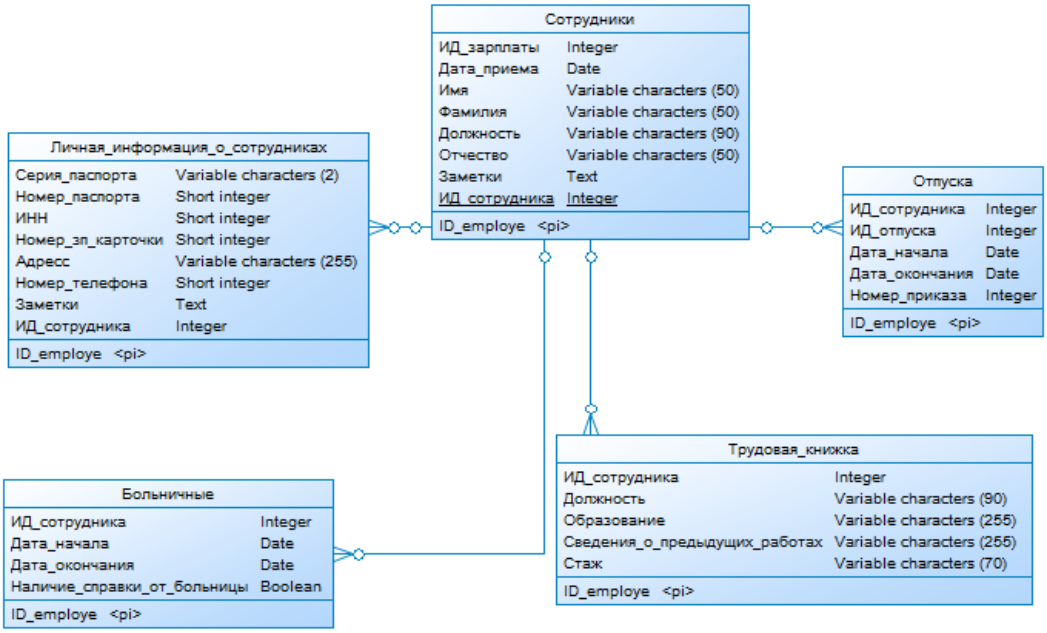

Рис. 20 Логічна модель даних для відділу кадрів

Логічна модель містить інформації про працівників підприємства. Для цього було створено наступні таблиці:

1. Інформація про працівників.
2. Працівники.
3. Трудова книжка.

4. Лікарняний.
5. Відпустки.

Таблиця «інформація про працівників» містить інформацію про серію та номер паспорта, ІПН, номер зарплатної карточки, адреса, номер телефону, замітки.

Таблиця «працівники» містить інформацію про дату прийому, ім’я, прізвище, посаду, по-батькові, замітки.

Таблиця «трудова книжка» містить інформацію про посаду, освіту, відомості про попередні роботи, стаж.

Таблиця лікарняний містить інформацію про дату початку лікарняного та дату завершення, і наявність довідки з лікарні.

Таблиця «Відпустки» містить інформацію про дату початку, дату завершення, номер приказу.

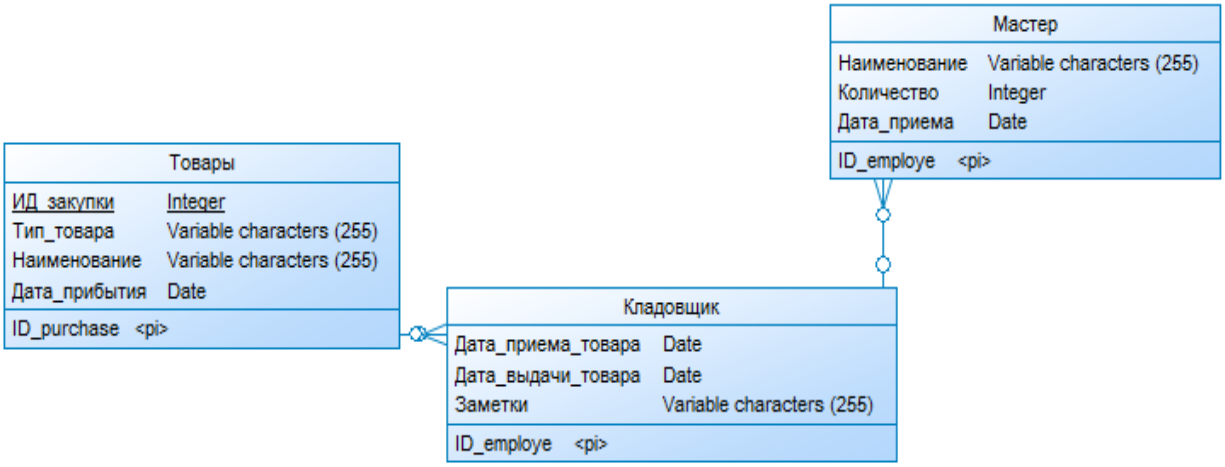

Рис. 21 Логічна модель даних для складу

Для того щоб зберігати інформацію про товар на складі та вести його облік, було створено наступні таблиці:

1. Склад.
2. Комірник.
3. Майстер.

Таблиця «склад» містить інформацію про найменування товару та дату прибуття.

Таблиця «комірник» містить інформацію містить інформацію про дату прийому товару та дату видачі.

Таблиця «майстер» містить інформацію про найменування товару та кількість.

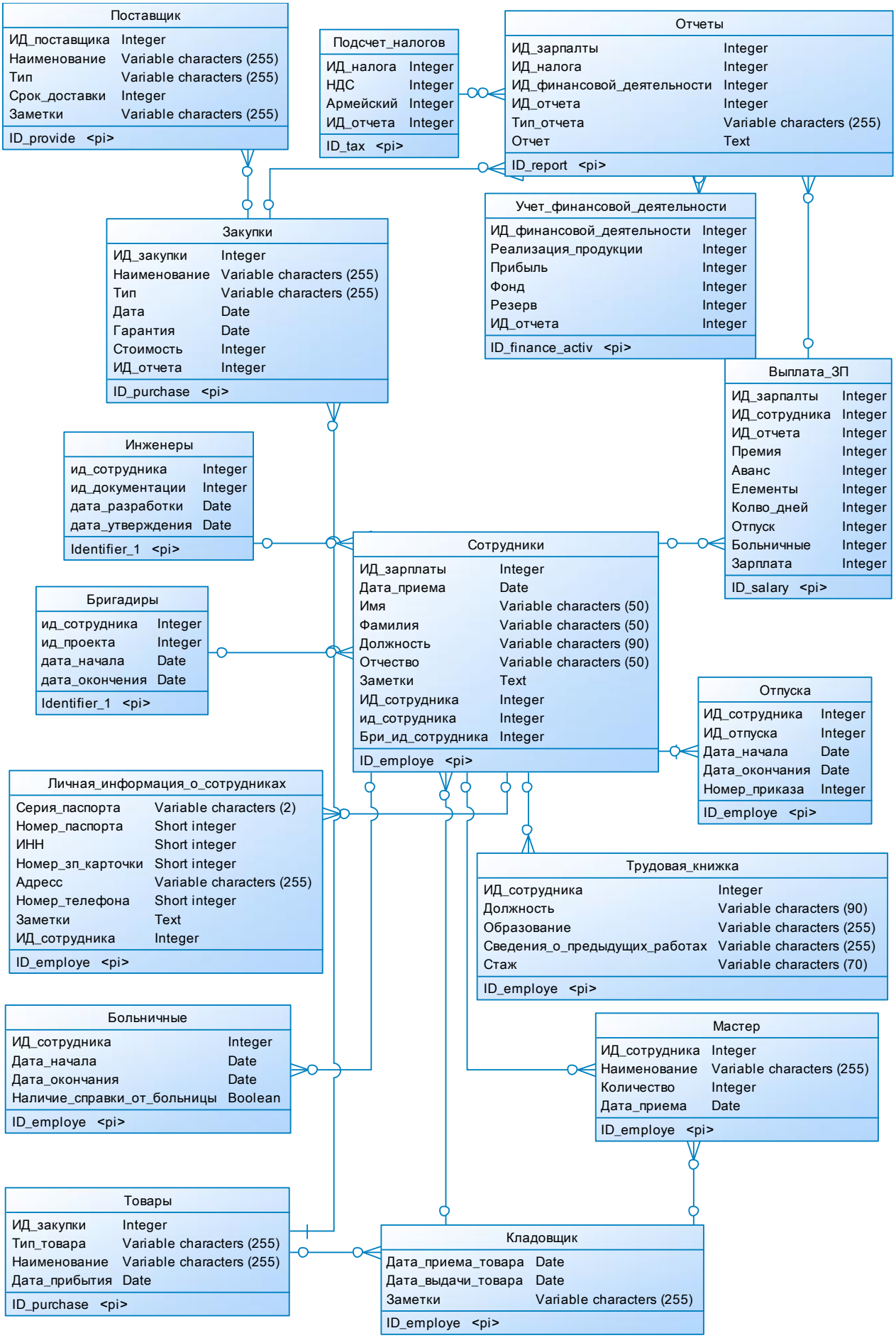

Рис. 22 Кінцева логічна модель даних

3.5 Розробка фізичної діаграми

Фізичне проектування бази даних - це процес підготовки опису й реалізації бази даних у зовнішній пам'яті; опис повинне включати основні відносини, файлову організацію, індекси, що забезпечують ефективний доступ до даних, а також всі відповідні обмеження цілісності й засобу захисту. Етапи проектування фізичної моделі даних:

I. Перенос глобальної логічної моделі даних у середовище цільовий СУБД. 2. Проектування основних відносин.

3. Реалізація обмежень предметної області.

4. Проектування фізичного подання бази даних.

Стадії фізичного проектування включає розробку основних відносин і реалізацію обмежень предметної області з використанням доступних функціональних засобів цільовий СУБД. На цьому етапі повинне бути також схвалені рішення щодо вибору способів одержання похідних даних, які включені в модель даних. На наступному етапі відбувається вибір файлової організації й індексів для основних відносин. Як правило, СУБД для персональних комп'ютерів мають фіксовану структуру зовнішньої пам'яті, а інші СУБД надають кілька альтернативних варіантів файлової організації для зберігання даних. З погляду користувача організація внутрішньої структури зберігання відносин повинна бути зовсім прозорої - користувач повинен мати можливість одержувати доступ до будь-якого відношення й до окремих його рядків без обліку способу зберігання даних. Це означає, що СУБД повинна забезпечувати повну незалежність фізичного зберігання даних від їхньої логічної організації. Тільки в цьому випадку внесення змін у фізичну організацію бази даних не зробить ніякого впливу на роботу користувачів.

Відповідність між логічною моделлю даних і фізичною моделлю даних визначаться внутрішньою схемою бази даних. Розроблювач повинен надати докладні фізичні проекти бази даних з обліком застосовуваної СУБД і операційної системи ц проекті реалізації бази даних у СУБД розроблювач повинен визначити структури файлів, які будуть використовуватися для подання кожного

відношення. У проекті реалізації бази даних в операційній системі розроблювач повинен указати розташування окремих файлів і забезпечити необхідний їхній захист. На наступному етапі здійснюється проектування засобів захисту, необхідних для запобігання несанкціонованого доступу до даних, включаючи керування доступом до основних відносин. Потім аналізується також необхідність зниження рівня вимог нормалізації даних у логічній моделі, що може сприяти підвищенню загальної продуктивності системи. Однак ці дії варто вживати тільки у випадку реальної необхідності, оскільки введення в базу даних надмірності неминуче викличе поява проблем з підтримкою цілісності даних.

Налаштування PowerDesigner на режим створення фізичної моделі. Найпершим завданням на етапі фізичного проектування баз даних є перетворення відносин, створених на основі глобальної логічної моделі даних, у таку форму, що може бути реалізована в середовищі цільовий СУБД. Перша частина цього процесу передбачає перевірку інформації, зібраної на етапі логічного проектування бази данин і поміщеної в словник даних. Друга частина процесу полягає у використанні цієї інформації для розробки проекту основних відносин. Цей процес вимагає наявності глибоких знань про функціональні можливості, надаваних цільовий СУБД.

Для реалізації фізичної моделі, конвертуємо нашу логічну модель у фізичну, за допомогою убудованої можливості PowerDesigner. Для того, щоб конвертувати логічну модель у фізичну, треба в меню Tools/ вибрати Generate Physical Date Model.

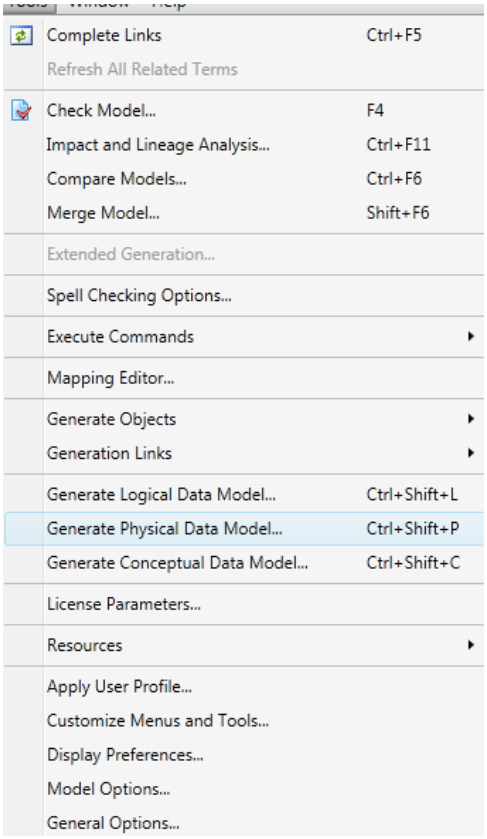

Рис. 23 Генерація фізичної моделі

У вікні, що відкрилося у полі DBMS вибираємо з існуючого списку sql-серверів – MySQL 5.0. У полі name указуємо ім’я майбутньої моделі даних.

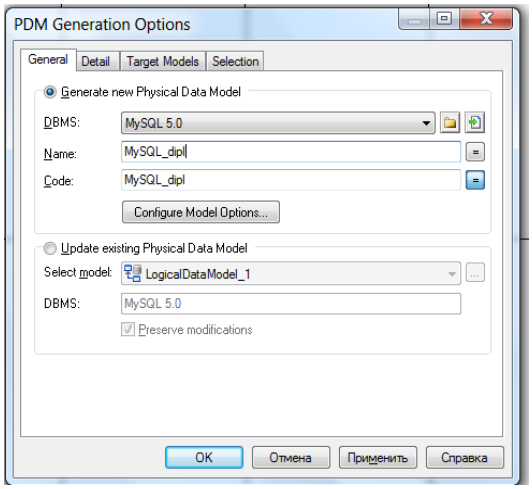

Рис. 24 Генерація фізичної моделі

На цьому етапі процедури розробки баз даних виконуються наступні дії.

1. Проектування основних відносин.

2. Розробка способів одержання похідних даних.

3. Реалізація обмежень предметної області.

Приступаючи до фізичного проектування, насамперед необхідно проаналізувати й добре засвоїти інформацію про відносини, зібрану на етапі побудови логічної моделі бази даних. Ця інформація може втримуватися в словнику даних і у визначеннях відносин. Кожні виділені в глобальній логічній моделі дані відносини включає наступні елементи:

1. ім'я відносини;
2. список простих атрибутів, ув`язнений у круглі дужки;
3. визначення первинного ключа й(якщо такі існують) і зовнішніх (FK) ключів;
4. список похідних атрибутів і опис способів їхнього обчислення;
5. визначення вимог посилальної цілісності для будь-яких зовнішніх ключів.

Для кожного атрибута в словнику даних повинна бути присутнім наступна інформація:

1. визначення його домена, що включає вказівка типу даних, розмірність
2. внутрішнього подання атрибута й будь-які необхідні обмеження на припустимі значення;
3. прийняте за замовчуванням значення атрибута (необов'язково),
4. допустимість значення NULL для даного атрибута.

Після генерації фізичної моделі з логічної, були визначені автоматично зв'язки, ключі, з урахуванням установлених настроювань на логічній моделі й зазначеного перед генерацією sql-сервера. Як видно на отриманій діаграмі, відбулася також міграція полів між таблицями. Міграція полів відбувається в тому випадку, якщо: На логічній моделі був зазначений зв'язок між таблицями одні до багатьох. Тоді на фізичній моделі, у таблицю в якої кратність зв'язку була «один», стрибне первинний ключ із таблиці, у якої кратність зв'язку була «багато».

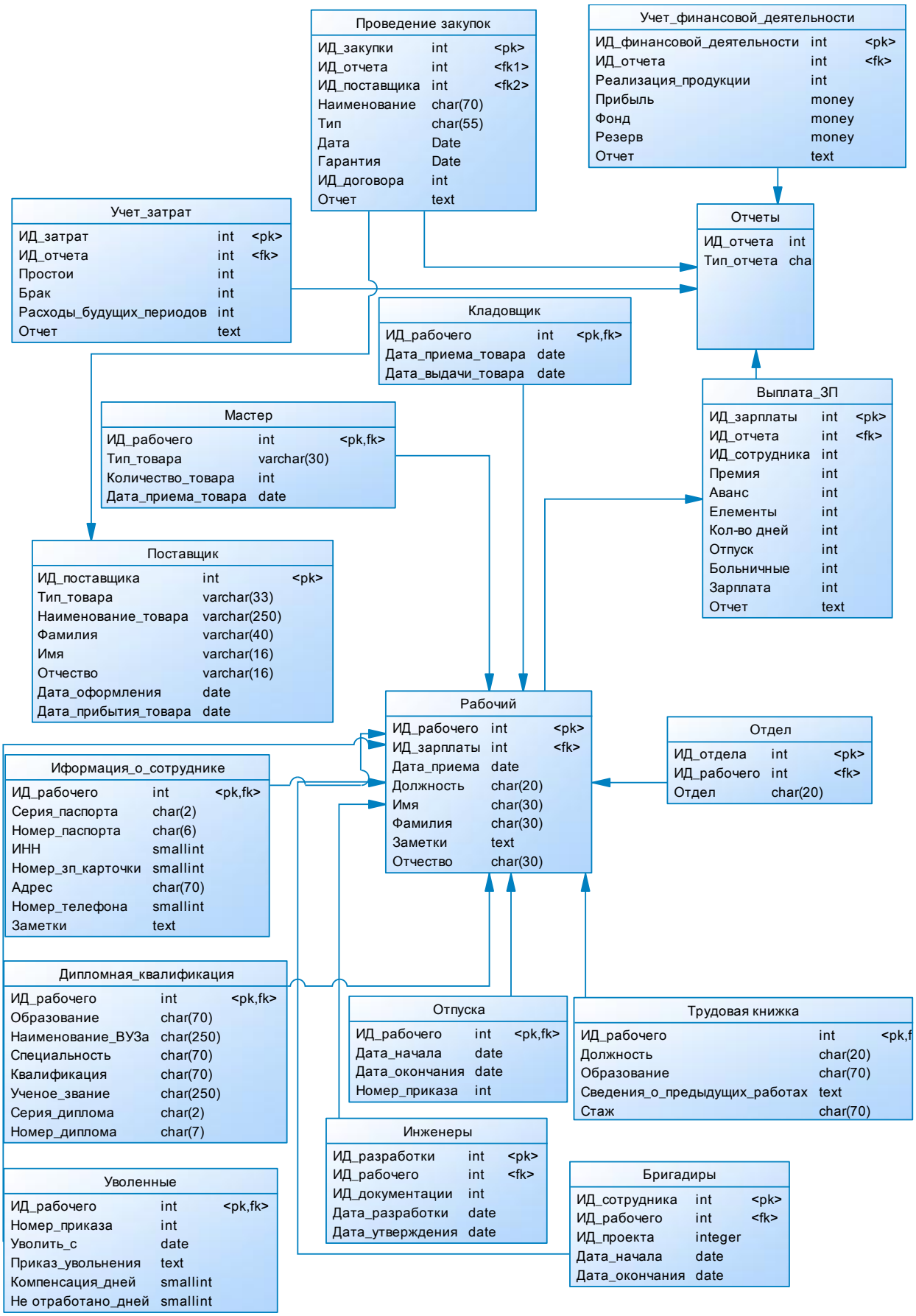

Рис. 25 Фізична модель даних

3.6 Генерація SQL коду

Для генерації MySQL на вкладці Database в програмі PowerDesigner, треба вибрати “Generate Database”.

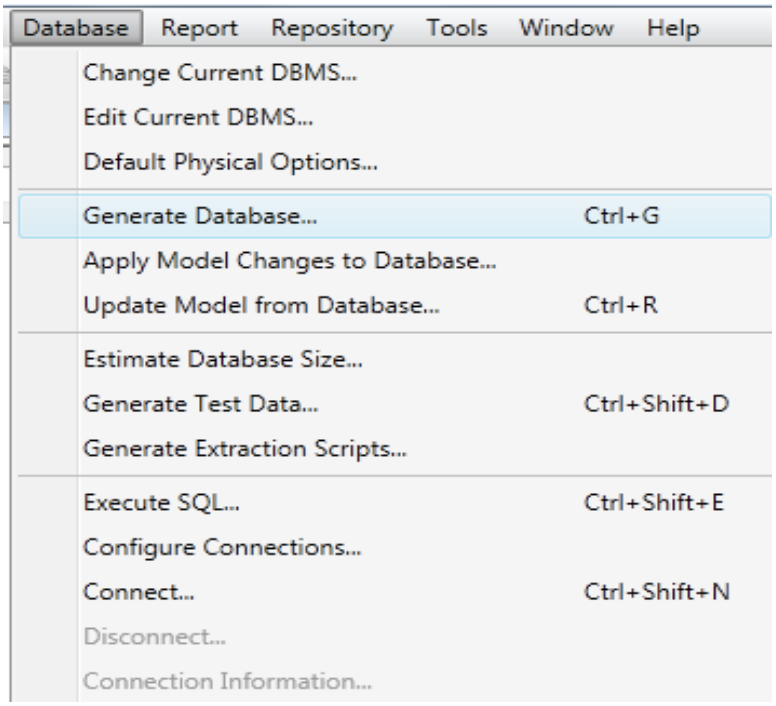

Рис. 26 – Генерація MySQL-коду

У вікні “Database Generation” необхідно обрати шлях сгенерованого SQL файлу, як зображено на рисунку 26.

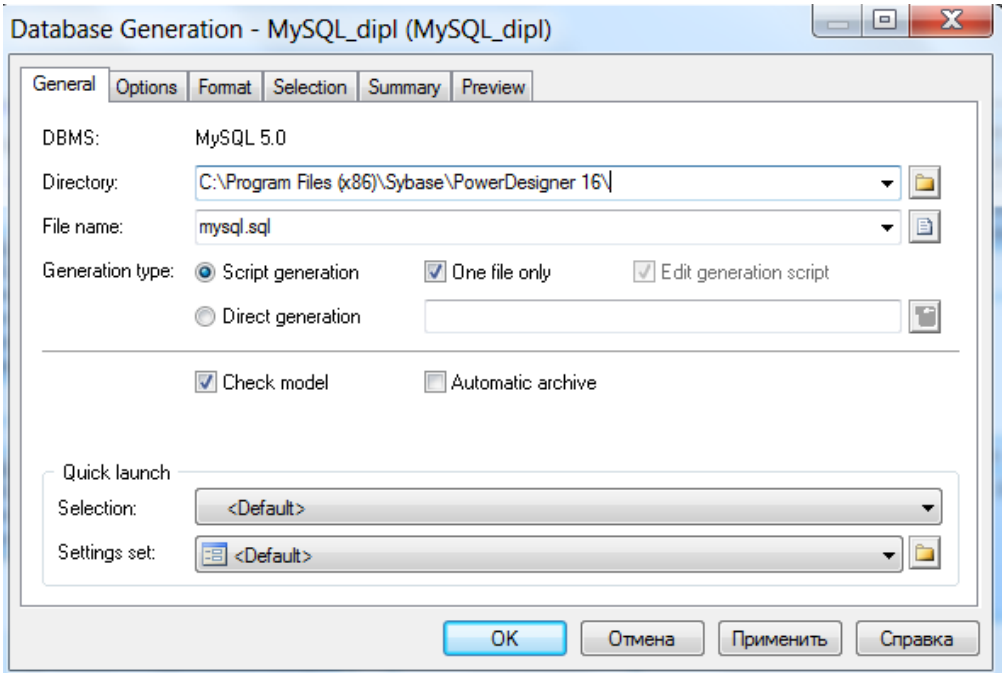

Рис. 27 Генерація MySQL-коду

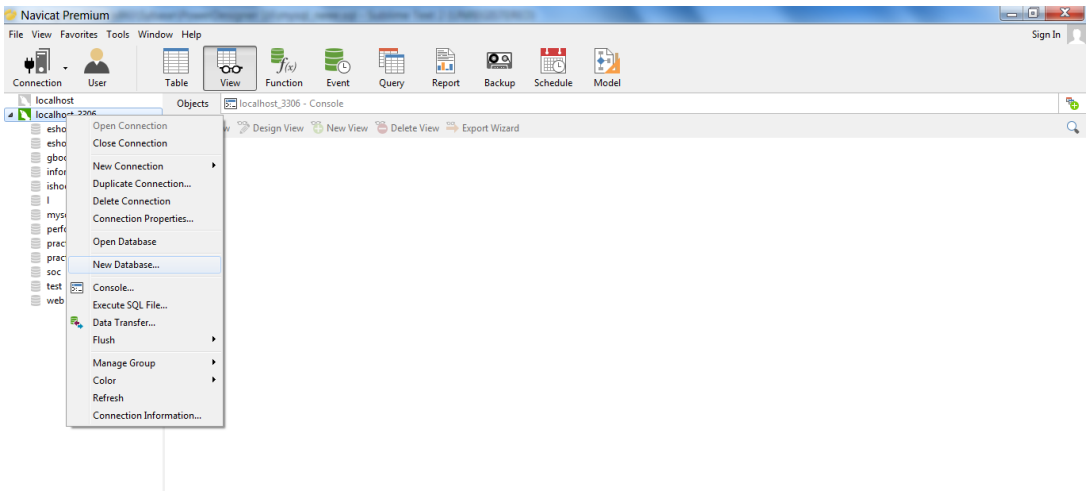

Рис. 28 Створення бази даних

На сервері необхідно створити нову базу даних, в яку загрузити MySQL код.

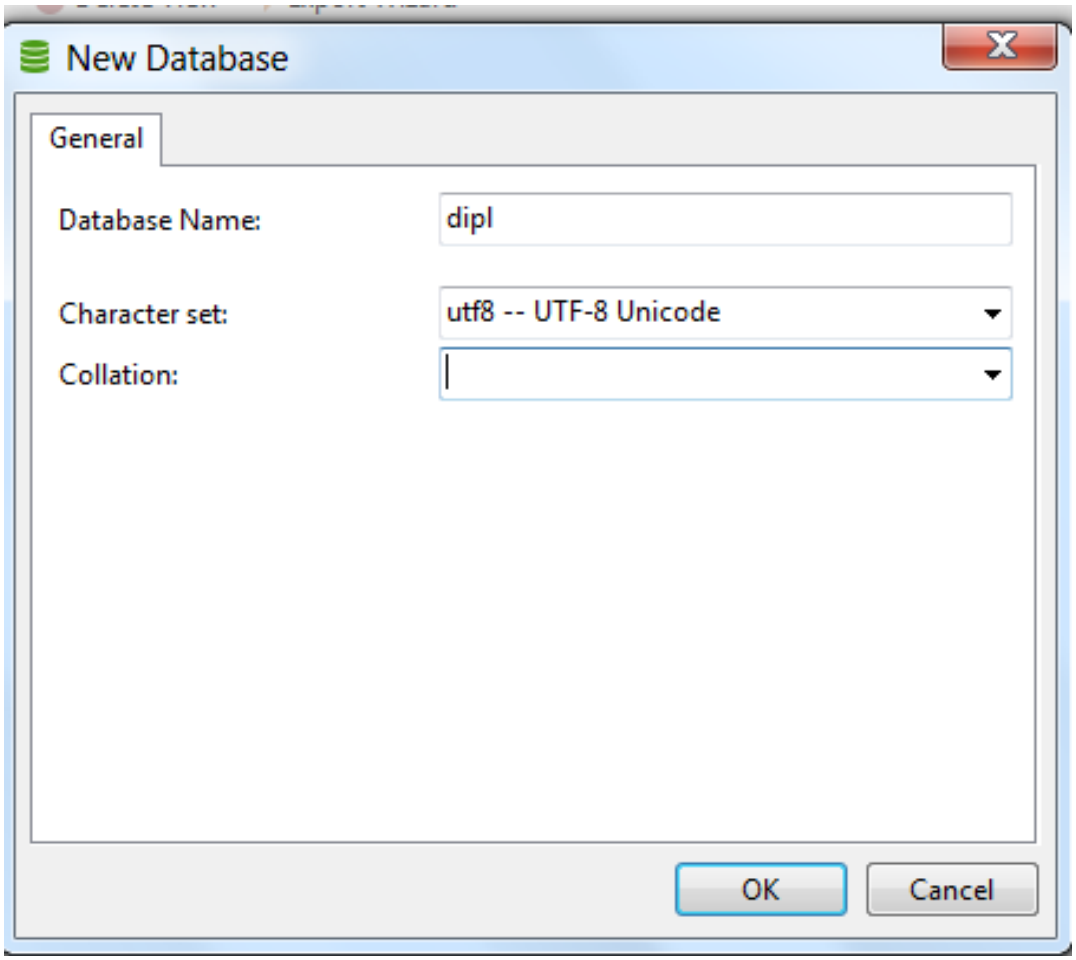

Рис. 29 – Створення бази даних

В обрану базу даних треба загрузити SQL код, для цього необхідно у вкладці Query вставити сгенерований SQL код, як зображено на рисунку 2.6.5.

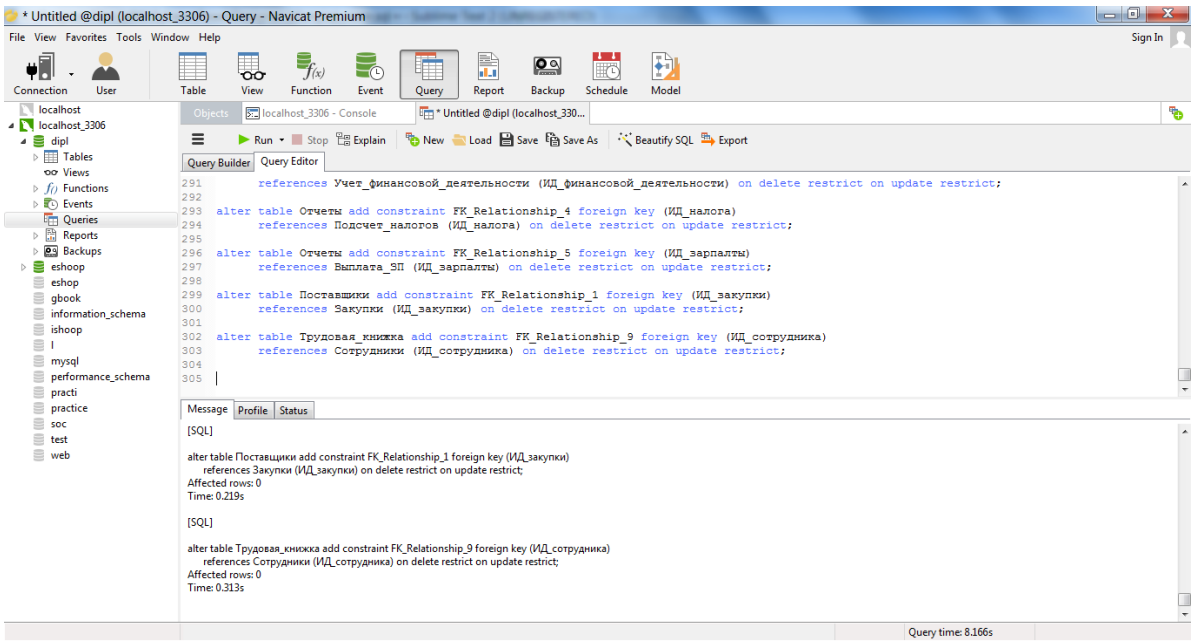

Рис. 30 – Загрузка MySQL коду

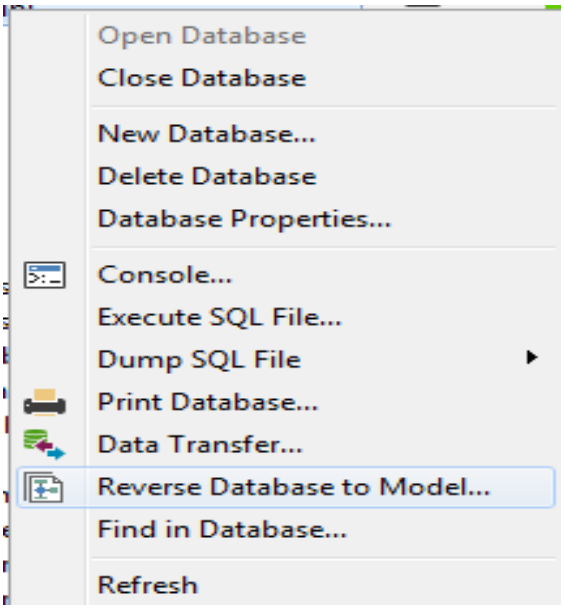

Рис. 31 Генерація схеми даних

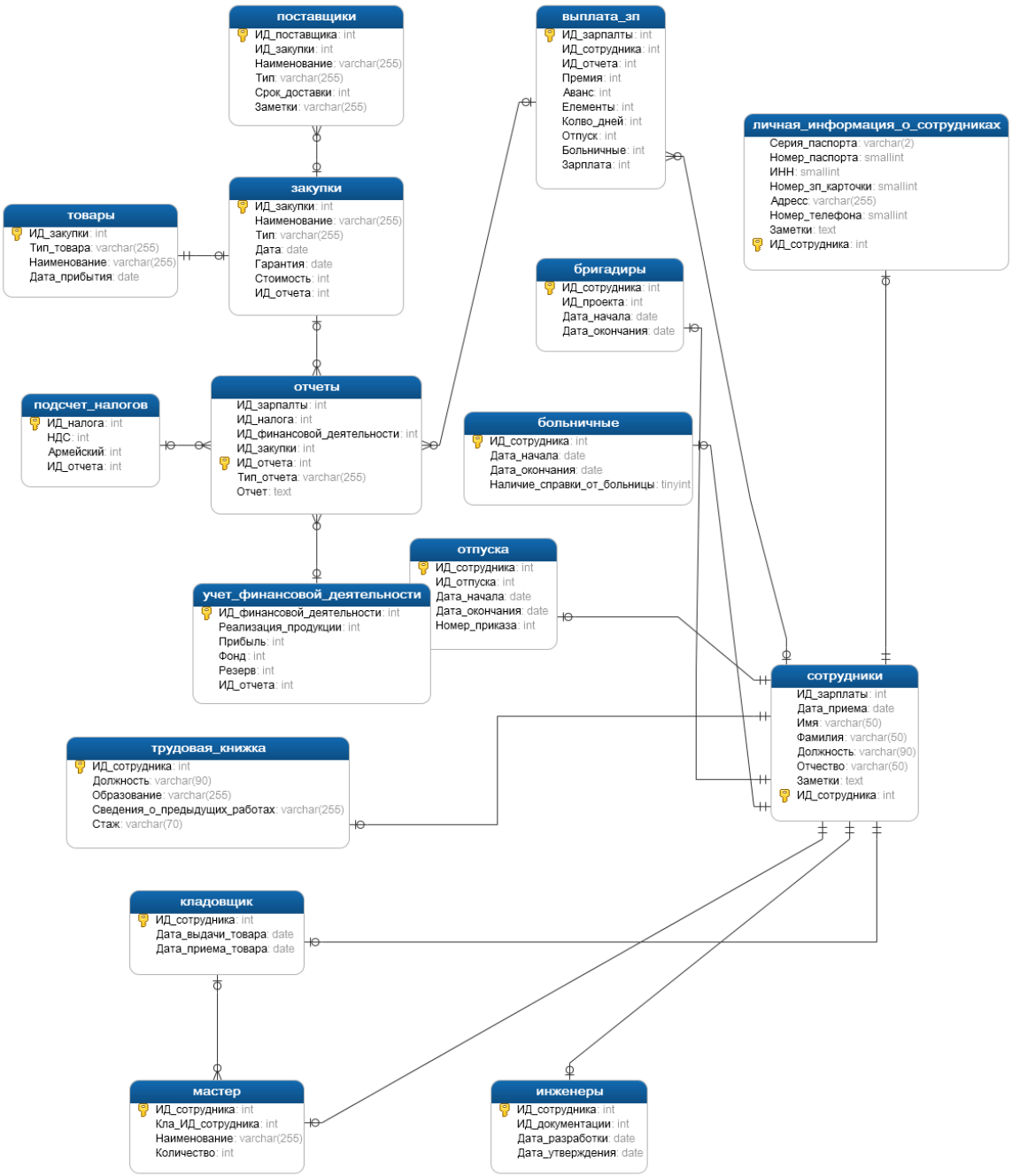

Рис. 32 схема даних

РОЗІЛ 4.

4.1 Маркетингові дослідження ринку збуту розробленого програмного продукту

Успішними і конкурентоспроможними на сучасному ринку є компанії, які використовують всі резерви для підвищення ефективності своєї діяльності і зменшення неефективних витрат. Типовими факторами, що створюють умови для непродуктивних витрат, є:

1. відсутність систем для автоматизації діловодства, а як наслідок - належного контролю і виконавської дисципліни, зрив термінів виконання завдань і проектів;
2. «Непрозорість» того, що відбувається в компанії - руху інформаційних потоків, ходу виконання рішень, відсутність аналітики;
3. відсутність інструментів для колективної роботи і управління знаннями;
4. зайві витрати робочого часу на узгодження документів, пересилання листів, пошук інформації, обробку даних
5. Оптимізацію бізнес-процесів можна почати з впровадження автоматизації, це дозволить вирішити ці проблеми.

Під бізнес-процесом слід розуміти певну сукупність дій (кроків, етапів, функцій), які здійснюються в певному порядку для досягнення кінцевих цілей компанії. Автоматизувавши і стандартизувати ці дії, ви зможете налагодити ефективну комунікацію з клієнтами і забезпечити безперервний діалог між відділами вашої компанії. І, як результат, зберегти клієнтів і збільшити прибуток свого бізнесу.

CRM-система bpm'online, заснована на концепції BPM (Business process management) - зручний і функціональний інструмент для побудови та управління всіма бізнес-процесами підприємства.

У bpm'online реалізована можливість моделювання бізнес-процесів будь-якої складності онлайн.

За допомогою зручного візуального інструменту можна відстежити якість ведення даних про угоди, контролювати повноту наповнення профілів клієнтів в базі, оцінити ефективність роботи співробітників і хід виконання бізнес-завдань.

Менеджери отримують зручний і багатофункціональний менеджер завдань. Вони можуть вказати відповідальних та учасників задач по процесу, розраховувати витрачений на виконання час і планувати на основі цього подальші дії.

Система завжди нагадає про дедлайни за завданнями, повідомить про знаменні події співробітників і клієнтів. А інтуїтивно зрозумілий інтерфейс зробить роботу в системі простий і мене витратною за часом.

Важливою перевагою CRM-системи є можливість впорядкувати та ефективно управляти всім документообігом компанії - договорами, рахунками, кореспонденцією, наказами та протоколами. В системі реалізовані інструменти для формування документів з використанням вбудованих шаблонів. Для швидкого доступу до друкованих оригіналів документів можна прикріпити скан-копії. Вся документація консолідована в електронному сховищі. Відшукати потрібний документ можна швидко, використовуючи зручний пошук. Розширивши права доступу, можна налагодити групову роботу з документацією, використовувати процес візування для узгодження комерційних документів. Хоча перші кроки полягали в автоматизації різних облікових і розрахункових завдань, наприклад: ведення журналу операцій, розрахунок прибутків і збитків, заробітної плати, складських запасів, і т.д. Правильніше було б називати це автоматизацією окремих бізнес-завдань, або навіть комплексу завдань.

А ось автоматизація саме бізнес-процесів вийшла на перший план порівняно недавно. Для того, щоб відрізняти свою діяльність і свої продукти від продуктів попередньої хвилі, розробники змушене підкреслюють їх виразами «процессная автоматизація», «BPM (Business Process Management) автоматизація», «workflow».

У чому ж насправді суть справжньої «автоматизації бізнес-процесів», на відміну від званої цими словами автоматизації комплексу облікових, розрахункових, і інших бізнес-завдань?

Основним об'єктом автоматизації в цьому випадку є бізнес-процес як спеціально організована, актуальна послідовність дій (завдань, операцій, подій), спрямована на створення будь-якого продукту або послуги.

Як правило, основна частина дій - це дії співробітників, що беруть участь в бізнес-процесі. Багато свої дії вони виконують безпосередньо в інформаційно-комунікаційних системах, або, як мінімум, фіксують в них результати виконання своїх дій. Ряд дій виконуються ІКТ-системами автоматично. А головне - ІКТ-системи виконують організуючу, сполучну, яка контролює і навіть керуючу роль у всьому процесі, що автоматизується бізнес-процесі.

Спеціально організована послідовність дій, що утворюють бізнес-процес, як правило, видається в такій системі автоматизації у вигляді графічної схеми, яку можна конструювати за допомогою візуальних засобів (конструктор бізнес-процесів). При цьому можна налаштовувати як параметри бізнес-процесу в цілому, так і його окремих елементів: завдань і нагадувань для співробітників, кроків обміну даними з зовнішніми системами (поштою, Порталом, ERP, ...), обчислення правил розгалуження маршруту, тощо.

Все це налаштовується в системі автоматизації бізнес-процесів у вигляді шаблону процесу, за яким багаторазово стартують і запускаються екземпляри процесів. Старт виконання примірника може початися або за вказівкою співробітника, або автоматично по події, виявленому системою (наприклад, при надходженні листа від клієнта).

Важливою властивістю такої системи є те, що вона активна по відношенню до співробітника, на відміну від інших засобів автоматизації. Раніше співробітник сам повинен був проявляти ініціативу і кмітливість з приводу того, що і коли йому потрібно зробити, і застосувати потрібний інструмент. У нашому ж випадку сама система автоматизації бізнес-процесів дає сигнал співробітнику про те, що і коли він повинен зробити, і надає йому для цього всю необхідну інформацію і інтерфейс для виконання дій. Вона ж зафіксує, коли співробітник виконав дію, і якщо він порушив термін - нагадає йому.

Вся інформація про виконання бізнес-процесів фіксується в системі. Це дає безцінну можливість управляти бізнес-процесами підприємства. Менеджер в будь-який момент знає, який із них в якому стані знаходиться, може аналізувати метрики процесів для виявлення і «розшивки» вузьких місць, для організації своїх дій, що управляють, що дозволяють поліпшити показники бізнес-процесів.

Вперше технології та системи автоматизації бізнес-процесів масово почали застосовуватися в розвинених системах електронного документообігу, для автоматизації процесів обробки документів відповідно до визначених регламентами. Але це лише окремий випадок. У загальному випадку не документ, а саме бізнес-процес є ключовим об'єктом. А вже при ньому виникають і обробляються документи. Їх може бути кілька, що з'являються на різних етапах процесу.

Так, найбільш просунуті системи електронного документообігу розширили межі свого застосування за межі канцелярії і стали успішно еволюціонувати в бік систем автоматизації бізнес-процесів.

4.2. Витрати на створення програмного забезпечення

Витрати на створення ПО ($К_{по}$) включають витрати на заробітну плату розробників програми ($З_з$ / п), яка визначається множенням сумарної трудомісткості розробки ПЗ (t) на середню заробітну плату програміста з нарахуваннями та вартості машинного часу на налагодження.

$$К_{ПО} = З_{ЗП} + З_{МВ} \qquad (4.11)$$

Заробітна плата розробників визначається за формулою:

$$З_{зп} = tС_{пр}, \qquad (4.12)$$

де t – загальна трудомісткість, люд.-годин.

$С_{пр}$ – середня годинна заробітна плата програміста, грн / год.

$С_{пр}$ = 131 грн. / Год.

$$З_{зп} = 1864,5 * 31 = 244184 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} C_{мч}, \quad (4.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, г.

$C_{Мч}$ – вартість машинного часу ЕОМ грн / год.

$$З_{мв} = 988,4 * 5 = 4942 \text{ грн.}$$

Витрати на створення програмного забезпечення складуть:

$$К_{по} = 244184 + 4942 = 249126 \text{ грн.}$$

Визначені таким чином витрати на створення програмного забезпечення є одноразовими капітальними витратами на створення АС.

Очікуваний період створення ПО:

$$Т = \frac{t}{В_к F_р} \text{ міс}, \quad (4.14)$$

де $В_к$ – число розробників;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні $F_р$ = 176 годин).

$$Т = \frac{186465}{1*176} = 3 \text{ міс.}$$

Висновок:

Таким чином, очікувана тривалість проведення бізнес-аналізу та розробки моделі даних становить приблизно, 3 місяців, а витрати – 249 126 грн.

4.3. Оцінка економічної ефективності впровадження ПЗ

Розрахувати економічну ефективність розробленого продукту не є можливим, так як дуже складно порахувати кількість часу на пошук необхідної інформації працівникам ПВТП, що власне і допомагає розроблена архітектура БД.

Розроблені бізнес-процеси організації та архітектура БД на їх основі уявляють собою гнучку систему для розробки ПО для зберігання документів в електронній формі на базі WEB-застосунків, завдяки чому працівники підприємства зможуть дуже швидко знаходити, зберігати, оновлювати необхідну інформацію.

Систему зберігання даних в електронній формі є показником конкурентоздатності підприємства, та має ряд переваг які складно пора.

ВИСНОВКИ

В ході виконання дипломного проекту на тему "Дослідження функціонування моделі даних за стандартом UML із застосуванням технології Liquibase" було досліджено функціонування моделі даних на прикладі ПВТП «Укрпромавтоматика» та розроблено:

1. Сценарій підприємства "Укрпромавтоматика".
2. Діаграми використання.
3. Діаграми кооперації.
4. Логічна модель даних.
5. Фізична модель даних.
6. Згенерований MySQL код.
7. Відтворена схема даних у программі на сервері.

В результаті виконання дипломної роботи була створена архітектура бази даних, завдяки методології “Model Driven Architecture” яка передбачає повний цикл розробки програмного забезпечення починаючи з вироблення і погодження вимог до введення в експлуатацію. Архітектура БД є гнучкою завдяки фреймворку Liquibase. У випадку змін та доповнень бізнес-процесів підприємства – можливо швидко вносити зміни в БД, що збереже час та кошти на підтримку програмного забезпечення.

СПИСОК ПОСИЛАНЬ ТА ДЖЕРЕЛ

1. Object Management Group. OMG Model Driven Architecture. http://www.omg.org/mda/
2. "OMG Unified Modeling Language Specification", Version 1.5, OMG group, March 2003. http://www.omg.org/cgi-bin/doc?formal/03-03-01
3. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо. СПб.: Питер, 2007. 544 с.
4. В. Ю. Арьков, М. Н. Мамчур. Модельно-ориентированный подход к проектированию информационной системы управления филиалом вуза. Управление в социальных и экономических системах / Уфа: Угату, 2007. Т. 9, №7 (25). С. 53–59
5. Мамчур, М.Н. Применение технологии MDA к разработке АИС «Учебный план» / М. Н. Мамчур, Л. Р. Рустамханова // Матер. Всерос. науч. - практ. конф. Нефтекамск, 2005. С. 139–144.
6. Грибачев, К. Г. Delphi и Model Driven Architecture. Разработка приложений баз данных / К. Г. Грибачев. СПб.: Питер, 2004. 348 с.
7. IBM. An introduction to Model Driven Architecture. http://www.ibm.com/developerworks/rational/library/3100.html
8. Model-driven architecture Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Model-driven_architecture
9. Anneke Kleppe. MDA Explained // The Model Driven Architecture: Practice and Promise. Addison-Wesley. ISBN 0-321-19442-X
10. Meghan Kiffer The MDA Journal // Model Driven Architecture Straight From The Masters. ISBN 0-929652-25-8
11. David S. Frankel. Model Driven Architecture // Applying MDA to Enterprise Computing. John Wiley & Sons, ISBN 0-471-31920-1
12. Украинское сообщество программистов. Очень краткое введение в Архитектуру, управляемую моделью (Model Driven Architecture MDA). http://www.developers.org.ua/lenta/articles/mda-introduction/
13. Martin Fowler. Model Driven Architecture: from the beginning.

http://www.martinfowler.com/bliki/ModelDrivenArchitecture.html

14. A BPT Column. MDA Journal. http://www.bptrends.com/publicationfiles/01-04%20COL%20Dom%20Spec%20Modeling%20Frankel-Cook.pdf
15. Free software development resources on programming, software testing and project management. Understanding the Model Driven Architecture (MDA). http://www.methodsandtools.com/archive/archive.php?id=5
16. Shakhgeldyan, C. Integration of university information resources into the unified information environment / C. Shakhgeldyan, V. Kryukov // Proc. of the 10th Int. Conf. of European University Information Systems (ENUS 2004). Slovenia, 2004. P. 321–327.
17. Буч Г., Рамбо Г., Якобсон И. UML. Руководство пользователя. М.: ДМК, 2000, 358 с.
18. Mellor S., Balcer M. Executable UML. A Foundation for Model-Driven Architecture. MA: Addison-Wesley. 2002. 368 p.
19. Шалыто А.А., Туккель Н.И. SWITCH-технология автоматный подход к созданию программного обеспечения "реактивных" систем //Программирование. 2001. №5, с. 45-62. http://is.ifmo.ru/ (раздел «Статьи»).
20. Кондратьев А. М. CASE-средство и объектные базы данных // Объектно-ориентированное визуальное моделирование / Под ред. Терехова А. Н. СПб.: Изд-во С.-Петерб. ун-та, 1999. С. 57–78.
21. SINTEF. Tools. http://modelbased.net/wp/tools/
22. Jason Dokken, Process Driven Modernization in Insurance. http://www.omg.org/cgi-bin/doc?omg/10-07-01.pdf
23. Executable UML: A Foundation for Model-Driven Architecture, Mellor and Balcer, Addison-Wesley, 2003.
24. MDA Distilled: Principles of Model-Driven Development, Mellor, Scott, Uhl and Weise, Addison-Wesley, 2004.
25. INTERFACE Ltd. BOLD инструмент реализации MDA в Delphi. http://www.interface.ru/fset.asp?Url=/misc/bold1.htm
26. Viewpoint. Bold for Delphi Resource Center.

http://www.viewpointsa.com/bold_resources/

27. Леоненков А. Самоучитель UML - СПб.: BHV, 2001.
28. Грибачев К. Тонкие базы данных и инструменты для их разработки в Delphi и C++Builder. - КомпьютерПресс, 2003, № 7, 8.
29. Object Management Group, Inc. Object Management Group. http://www.omg.org
30. INTUIT.ru::Интернет-Университет Информационных Технологий - дистанционное образование. Сервис-ориентированная архитектура (SOA) и архитектура, управляемая моделями (MDA). http://www.intuit.ru/department/itmngt/entarc/7/7.html
31. Кулябов Д.С., Королькова А.В. Введение в формальные методы описания бизнес-процессов
32. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования = Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. — 3-е изд. — М.: Вильямс, 2006. — 736 с. — ISBN 0-13-148906-2.
33. Джозеф Шмуллер. Освой самостоятельно UML 2 за 24 часа. Практическое руководство = Sams Teach Yourself UML in 24 Hours, Complete Starter Kit. — М.: Вильямс, 2005. — 416 с. — ISBN 0-672-32640-X.
34. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя = The Unified Modeling Language user guide. — 2-е изд. — М., СПб.: ДМК Пресс, Питер, 2004. — 432 с. — ISBN 5-94074-260-2.
35. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS / С. Орлов. — 2-е изд.. — СПб.: Питер, 2006. — 736 с. — ISBN 5-46900-599-2.
36. Яргер, Р.Дж.; Риз, Дж.; Кинг, Т. MySQL и mSQL: Базы данных для небольших предприятий и Интернета; СПб: Символ-Плюс, **2013**. - 560 с.
37. Методические указания по выполнению экономического раздела в дипломних проектах студентов специальности “Компьютерные системы ” / Составители А.Г. Вагонова, Нікітіна А.Б. Н.Н. Романюк - Днепропетровск: Национальный горный университет. - 2013.

38. Бойко В.В. Экономика предприятий Украины. Основной курс: Учебник для вузов. - Д.: Пороги, 1997. - 312 с.
39. Андрусенко Г.О. Основы маркетинга. - Е.: НМК ВО, 1992. - 143 с.
40. Баркан Д.И. Маркетинг для всех. - Л.: «Культинформпрес», 1991. - 256 с.
41. Бусыгин А.В. Предпринимательство. Основной курс: Учебник для вузов. - М.: ИНФРА-М, 1997. - 608 с.
42. Завялов П.С., Демидов В.Е. Формула успеха - маркетинг. - М.:
43. Междунар. Отношения, 1991.
44. Котлер Ф. Основы маркетинга: Пер. с англ. / Общ. Ред. и вступ. ст.
45. Е.М. Пеньковой - М.: Прогресс, 1990. - 636с.
46. Скворцов Н.Н. Бизнес-план предприятия. - К.: Вища школа, 1995. - 187 с.
47. Мескон М.Х., Альберт М., Хедоури Ф. Основы менеджмента. - М.: Дело, 1992. - 702 с.
48. Мете А.Ф., Штец К.А., Бельгольский Б.П. и др. Организация и планирование предприятий. - М.: Металлургия, 1986. - 560 с.
49. Основы инновационного менеджмента: Теория и практика: Учеб. пособие / Под ред. П.Н. Завлина и др. - М: ОАО Издательство «Экономика». 2000. - 475 с.

46 Савчук В.П., Прилипко С.И., Величко Е.Г. Анализ и разработка инвестиционных проектов. - Учебное пособие. - Киев: Абсолют-В. Эльга. 1999.- 304 с.

50. Лутц М. Изучаем Python 4-е издание. -М.:Орели, 2010. -1280 с.
51. Бизли Д. Python, подробный справочник. –М.: Орели, 2012. -864с.

Додаток А

```
/*==============================================================*/
/* DBMS name:      MySQL 5.0                                    */
/*==============================================================*/

drop table if exists Больничные;
drop table if exists Бригадиры;
drop table if exists Выплата_ЗП;
drop table if exists Закупки;
drop table if exists Инженеры;
drop table if exists Кладовщик;
drop table if exists Личная_информация_о_сотрудниках;
drop table if exists Мастер;
drop table if exists Отпуска;
drop table if exists Отчеты;
drop table if exists Подсчет_налогов;
drop table if exists Поставщики;
drop table if exists Сотрудники;
drop table if exists Товары;
drop table if exists Трудовая_книжка;
drop table if exists Учет_финансовой_деятельности;

/*==============================================================*/
/* Table: Больничные                                            */
/*==============================================================*/
create table Больничные
(
   ИД_сотрудника        int not null,
   Дата_начала          date,
```

```
  Дата_окончания       date,
  Наличие_справки_от_больницы bool,
  primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Бригадиры                                             */
/*==============================================================*/
create table Бригадиры
(
  ИД_сотрудника       int not null,
  ИД_проекта          int,
  Дата_начала         date,
  Дата_окончания      date,
  primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Выплата_ЗП                                            */
/*==============================================================*/
create table Выплата_ЗП
(
  ИД_зарпалты         int not null,
  ИД_сотрудника       int,
  ИД_отчета           int,
  Премия              int,
  Аванс               int,
  Елементы            int,
  Колво_дней          int,
  Отпуск              int,
```

```
  Больничные          int,
  Зарплата            int,
  primary key (ИД_зарпалты)
);

/*==============================================================*/
/* Table: Закупки                                               */
/*==============================================================*/
create table Закупки
(
  ИД_закупки          int not null,
  Наименование        varchar(255),
  Тип                 varchar(255),
  Дата                date,
  Гарантия            date,
  Стоимость           int,
  ИД_отчета           int,
  primary key (ИД_закупки)
);

/*==============================================================*/
/* Table: Инженеры                                              */
/*==============================================================*/
create table Инженеры
(
  ИД_сотрудника       int not null,
  ИД_документации     int,
  Дата_разработки     date,
  Дата_утверждения    date,
  primary key (ИД_сотрудника)
```

```
);

/*==============================================================*/
/* Table: Кладовщик                                             */
/*==============================================================*/
create table Кладовщик
(
   ИД_сотрудника        int not null,
   Дата_выдачи_товара   date,
   Дата_приема_товара   date,
   primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Личная_информация_о_сотрудниках                       */
/*==============================================================*/
create table Личная_информация_о_сотрудниках
(
   Серия_паспорта       varchar(2),
   Номер_паспорта       smallint,
   ИНН                  smallint,
   Номер_зп_карточки    smallint,
   Адресс               varchar(255),
   Номер_телефона       smallint,
   Заметки              text,
   ИД_сотрудника        int not null,
   primary key (ИД_сотрудника)
);

/*==============================================================*/
```

```
/* Table: Мастер                                */
/*==============================================================*/
create table Мастер
(
   ИД_сотрудника        int not null,
   Кла_ИД_сотрудника    int,
   Наименование         varchar(255),
   Количество           int,
   primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Отпуска                               */
/*==============================================================*/
create table Отпуска
(
   ИД_сотрудника        int not null,
   ИД_отпуска           int,
   Дата_начала          date,
   Дата_окончания       date,
   Номер_приказа        int,
   primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Отчеты                                */
/*==============================================================*/
create table Отчеты
(
   ИД_зарпалты          int,
```

```
  ИД_налога           int,
  ИД_финансовой_деятельности int,
  ИД_закупки          int,
  ИД_отчета           int not null,
  Тип_отчета          varchar(255),
  Отчет               text,
  primary key (ИД_отчета)
);

/*==============================================================*/
/* Table: Подсчет_налогов                                       */
/*==============================================================*/
create table Подсчет_налогов
(
  ИД_налога           int not null,
  НДС                 int,
  Армейский           int,
  ИД_отчета           int,
  primary key (ИД_налога)
);

/*==============================================================*/
/* Table: Поставщики                                            */
/*==============================================================*/
create table Поставщики
(
  ИД_поставщика       int not null,
  ИД_закупки          int,
  Наименование        varchar(255),
  Тип                 varchar(255),
```

```
  Срок_доставки       int,
  Заметки             varchar(255),
  primary key (ИД_поставщика)
);

/*==============================================================*/
/* Table: Сотрудники                                            */
/*==============================================================*/
create table Сотрудники
(
  ИД_зарплаты         int,
  Дата_приема         date,
  Имя                 varchar(50),
  Фамилия             varchar(50),
  Должность           varchar(90),
  Отчество            varchar(50),
  Заметки             text,
  ИД_сотрудника       int not null,
  primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Товары                                                */
/*==============================================================*/
create table Товары
(
  ИД_закупки          int not null,
  Тип_товара          varchar(255),
  Наименование        varchar(255),
  Дата_прибытия       date,
```

```
  primary key (ИД_закупки)
);

/*==============================================================*/
/* Table: Трудовая_книжка                                       */
/*==============================================================*/
create table Трудовая_книжка
(
  ИД_сотрудника       int not null,
  Должность           varchar(90),
  Образование         varchar(255),
  Сведения_о_предыдущих_работах varchar(255),
  Стаж                varchar(70),
  primary key (ИД_сотрудника)
);

/*==============================================================*/
/* Table: Учет_финансовой_деятельности                          */
/*==============================================================*/
create table Учет_финансовой_деятельности
(
  ИД_финансовой_деятельности int not null,
  Реализация_продукции int,
  Прибыль             int,
  Фонд                int,
  Резерв              int,
  ИД_отчета           int,
  primary key (ИД_финансовой_деятельности)
);
```

```
alter table Больничные add constraint FK_Relationship_8 foreign key
(ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Бригадиры add constraint FK_Reference_17 foreign key (ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Выплата_ЗП add constraint FK_Relationship_10 foreign key
(ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Закупки add constraint FK_Relationship_11 foreign key (ИД_закупки)
    references Товары (ИД_закупки) on delete restrict on update restrict;
alter table Инженеры add constraint FK_Reference_16 foreign key (ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Кладовщик add constraint FK_Relationship_12 foreign key
(ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Личная_информация_о_сотрудниках add constraint FK_Relationship_6
foreign key (ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Мастер add constraint FK_Relationship_13 foreign key (ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Мастер add constraint FK_Relationship_15 foreign key
(Кла_ИД_сотрудника)
    references Кладовщик (ИД_сотрудника) on delete restrict on update restrict;
alter table Отпуска add constraint FK_Relationship_7 foreign key (ИД_сотрудника)
    references Сотрудники (ИД_сотрудника) on delete restrict on update restrict;
alter table Отчеты add constraint FK_Relationship_2 foreign key (ИД_закупки)
    references Закупки (ИД_закупки) on delete restrict on update restrict;
alter table Отчеты add constraint FK_Relationship_3 foreign key
(ИД_финансовой_деятельности)
```

references Учет_финансовой_деятельности (ИД_финансовой_деятельности) on delete restrict on update restrict;

alter table Отчеты add constraint FK_Relationship_4 foreign key (ИД_налога)
references Подсчет_налогов (ИД_налога) on delete restrict on update restrict;

alter table Отчеты add constraint FK_Relationship_5 foreign key (ИД_зарпалты)
references Выплата_ЗП (ИД_зарпалты) on delete restrict on update restrict;

alter table Поставщики add constraint FK_Relationship_1 foreign key (ИД_закупки)
references Закупки (ИД_закупки) on delete restrict on update restrict;

alter table Трудовая_книжка add constraint FK_Relationship_9 foreign key (ИД_сотрудника)
references Сотрудники (ИД_сотрудника) on delete restrict on update restrict

ДОДАТОК Б

ВІДГУК

на дипломну роботу магістра на тему:
«Розробка моделі даних за стандартом UML і дослідження її функціонування із застосуванням технології Liquibase»

студента групи 122м-16-1 Шарапат Владислава Євгеновича

1. Метою дипломної роботи магістра є проектування інформаційної системи за допомогою впровадження технології UML та розробка гнучкої архітектури БД.
2. Тема дипломної роботи безпосередньо пов'язана з об'єктом діяльності магістра спеціальності «Комп'ютерні науки» напряму «Інженерія програмного забезпечення» - створення, дослідження і реалізація моделей і програмних засобів.
3. Наукова новизна результатів, які очікуються, полягає у проектуванні інформаційної систем з використанням технології проектування моделі даних UML і відстеження змін моделі на основі бізнес-процесів організацій.
4. Оригінальність технічних рішень при розробці програмного засобу полягає в використанні зв'язки таких інструментальних засобів як UML, PowerDesigner, Liquibase.
5. Практична цінність дослідження полягає в розробці програмних модулів, програмного продукту, які дозволяють оцінити переваги проектування інформаційних систем, розробки програмного забезпечення за допомогою використання технології UML.
6. Оформлення дипломної роботи магістра виконано на сучасному рівні і відповідає вимогам, що пред'являються до робіт даної кваліфікації. Ступінь самостійності виконання досить висока.
7. Дипломна робота магістра в цілому заслуговує оцінки «відмінно», а сам автор - присвоєння кваліфікації «інженер програмного забезпечення».

Керівник дипломної
роботи магістра, к.т.н.,
доц. кафедри ПЗКС Куваев В.М.

ДОДАТОК В

Рецензія

на дипломний проект магістра на тему:
«Розробка моделі даних за стандартом UML і дослідження її функціонування із застосуванням технології Liquibase»

студента групи 122М-16-1 Шарапат Владислав Євгеновича

Дипломна робота присвячена розробці моделі даних підприємства. В результаті проведеної роботи було розглянуто теоретичні положення різноманітних моделей даних в UML з використанням технології Liquibase, досліджено бізнес процеси підприємства та їх функціонування,були розроблені діаграми UML, які в повному обсязі відображають бізнес-процеси на ПВТП та на їх основі створена архітектура БД на системі управляння базами даних MySQL.

Також були розроблені логічна та фізична моделі даних, які відображають логічні зв’язки. На основі фізичної моделі даних був згенерований MySQL код . Розроблена модель даних допоможе зберігати інформацію про діяльність підприємства.

Студент Шарапат В.Є. добре розібрався в специфіці розробки моделі даних за стандартом UML та застосуванням технології Liquibase.

З огляду на вищевикладене, можна зробити висновок, що даний проект цілком відповідає вимогам, що пред'являються до кваліфікаційних робіт рівня магістр.

Ступінь опрацювання компонентів даного проекту, дозволяє оцінити роботу на «добре» і рекомендувати присвоїти студенту Шарапат В.Є. кваліфікацію «інженер програмного забезпечення».

Рецензент,