Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмний засіб для реалізації процесу міграції даних

Виконав: студент IV курсу, групи СНзс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Дегятрук І.Ю.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль - 2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Дегтяруку Івану Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмний засіб для реалізації процесу міграції даних

Керівник роботи Дмитроца Леся Павлівна, к.т.н., доц. каф. КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «02» 03 2021 року № 4/7-170

2. Термін подання студентом завершеної роботи 14.06.2021р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Теоретична частина. 1.1 Опис процесу міграції даних. 1.2. Основні визначення та поняття міграції даних. 1.3. Аналіз окремих представлених на ринку продуктів міграції даних 2. Проектна частина. 2.1 Огляд системи з точки зору бізнес-процесів. 2.2. Загальна ідея побудови послідовності обходу схеми БД. 2.3. Альтернативний алгоритм формування послідовності обходу. 2.4.Порівняльний аналіз алгоритмів формування послідовності обходу 3. Опис програмної реалізації розробки. 3.1. Бібліотека sequence-traversal-database. 3.2. Пакет data-migration. 4. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність, мета. 3. Співвідношення між основними поняттями реляційних БД. 4. Визначення міграції даних і контекст використання терміну. 5. Варіанти міграції Даних. 6. Діаграма бізнес-процесу «Міграція даних». 7. Приклади обходу даних в ВБД. 8. ПЗ, яке використовувалося в КРБ. 9. Діаграма пакетів структури розробленого додатку. 10. Діаграма класів бібліотеки sequence-traversal-database. 11. Діаграма основних класів пакета data-migration. 12. Опис основних класів. 13. Висновки по роботі.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	02.03 – 05.03	*Виконано*
2.	Підбір джерел про процес міграції даних	06.03 – 25.03	*Виконано*
3.	Опрацювання джерел про особливості окремих		
	представлених на ринку продуктів міграції даних	26.03 – 12.04	*Виконано*
4.	Виконання дослідження щодо програмного засобу для		
	реалізації процесу міграції даних	13.04 – 29.04	*Виконано*
5	Розроблення програмного коду	30.04 – 15.05	*Виконано*
6.	Оформлення розділу «Теоретична частина»	15.05 – 20.05	*Виконано*
7.	Оформлення розділу «Проектна частина»	21.05 – 28.05	*Виконано*
8.	Оформлення розділу «Опис програмної реалізації		
	розробки»	29.05 – 06.06	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	12.05 – 22.05	*Виконано*
10.	Оформлення кваліфікаційної роботи	28.05 – 10.06	*Виконано*
11.	Нормоконтроль	11.06 – 14.06	*Виконано*
12.	Перевірка на плагіат	11.06 – 14.06	*Виконано*
13.	Попередній захист кваліфікаційної роботи	08.06 – 10.06	*Виконано*
14.	Захист кваліфікаційної роботи	14.06	

Студент _____ Дегтярук І.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмний засіб для реалізації процесу міграції даних // Кваліфікаційна робота бакалавра // Дегтярук Іван Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СНзс–42 //Тернопіль, 2021 // С. – 53, рис. – 19, табл. – 1, слайдів – 13, бібліогр. – 19.

Ключові слова: JDBC, АЛГОРИТМ, БАЗА ДАНИХ, ЗАПИС, МІГРАЦІЯ ДАНИХ, ТАБЛИЦЯ

Кваліфікаційна робота присвячена проектуванню та розробці програмного інтсрументу для розв'язання питання міграції даних. Виконано аналіз основних елементів реляційної бази даних. Наведено оособливості міграції даних, описано головні визначення та поняття цього процесу. Зроблено огляд деяких програмних інструментів для здійснення міграції. Докладно проаналізовано основні стадії міграції даних, виконано грунтовний опис міграції як бізнес-процесу. Представлено загальний спосіб формування порядку обходу існуючої схеми бази даних, внесено головний та допоміжний алгоритми створення порядку обходу, здійснено порівняння їх основних елементів.

Докладно розглянуту алгоритм розвитку міграції даних Представлено і описано реалізацію програмного засобу із використанням мови Java із докладним тлумаченням алгоритму формування порядку обходу схеми вихідної бази даних та імплементації методів для здійснення міграції даних.

Розроблений програмний продукт дозволяє переглядати та переносити усі записи з вихідної бази даних з урахуванням існуючих залежностей, а також забезпечує їх цілісність. Будь-який запис можна перемістити тоді, коли він є незалежним, чи ті записи, від яких власне залежать дані, були вже переміщені в цільову базу даних.

ANNOTATION

Data migration implementation software application // Dehtiaruk Ivan // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2021 // P. - 53, Fig. - 19, Table - 1, Slide - 13, References - 19.

Keywords: JDBC, ALGORITHM, DATABASE, DATA MIGRATION, TABLE, RECORD

Thesis deals with the the design and development of software tools for the implementation of the data migration process. The basic concepts of the relational database are analyzed, the process of data migration is analyzed, its main definitions and concepts are given. An overview of the peculiarities of the use of some software tools for migration. The main stages of data migration are analyzed in detail, a thorough description of migration as a business process is performed.

The general way of forming the order of bypass of the existing scheme of the database is presented, the main and auxiliary algorithms of creation of the order of bypass are brought, the comparison of their basic elements is carried out. The algorithm of data migration development is considered in detail.

The developed software product allows to view and transfer all records from the source database taking into account existing dependencies, and also provides their integrity. Any record can be moved when it is independent, or the records on which the data actually depends have already been moved to the target database.

ЗМІСТ

ВСТУП

Зараз фактично всюди людство застосовує комп’ютерну техніку. Використовується спеціальне ПЗ для використання у тій чи іншій області. Зокрема для зберігання даних давно і достатньо часто застосовують РБД. Тут виникає проблема у проведенні грамотного процесу переносу даних з ВБД в ЦБД, оскільки їх схеми можуть бути різними відрізнятися. В даний час не має єдиного трактування вирішення проблеми МД. Компанія-розробник ПЗ не постійно надає можливість підтримки збереження напрацьованих даних при переході на новішу версію ПЗ. За проханням клієнта ця проблема може вирішитися індивідуально, і як правило, послуга є достатньо коштовною. Виробники повинні регулярно проводити аналіз усіх можливих змін та кожного разу розробляти оновлені засоби міграції стосовно кожного конкретного випадку при зміні схеми БД. Тут варто зауважити, що при виконані цих дій кожна фірма притримується свого алгоритму та механізму розв’язання такої задачі, котрі, як правило, не підлягають розголошенню. Як наслідок постає проблема забезпечення коректної МД при допомозі якісного ПЗ [1].

Мета роботи полягає у розробці такого програмного засобу, який забезпечить проведення процесу МД з гарантією перенесення всіх записів з ВБД, цілісність таких даних, а також необхідно забезпечити те, що перенесення записів відбуватиметься із збереженням всіх існуючих залежностей.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Складові елементи РБД

Складовими РБД, які найчастіше використовуються, є тип даних, домен, атрибут, кортеж, первинний ключ, відношення [2]. Взаємозалежності між ними продемонстровані на рисунку 1.1

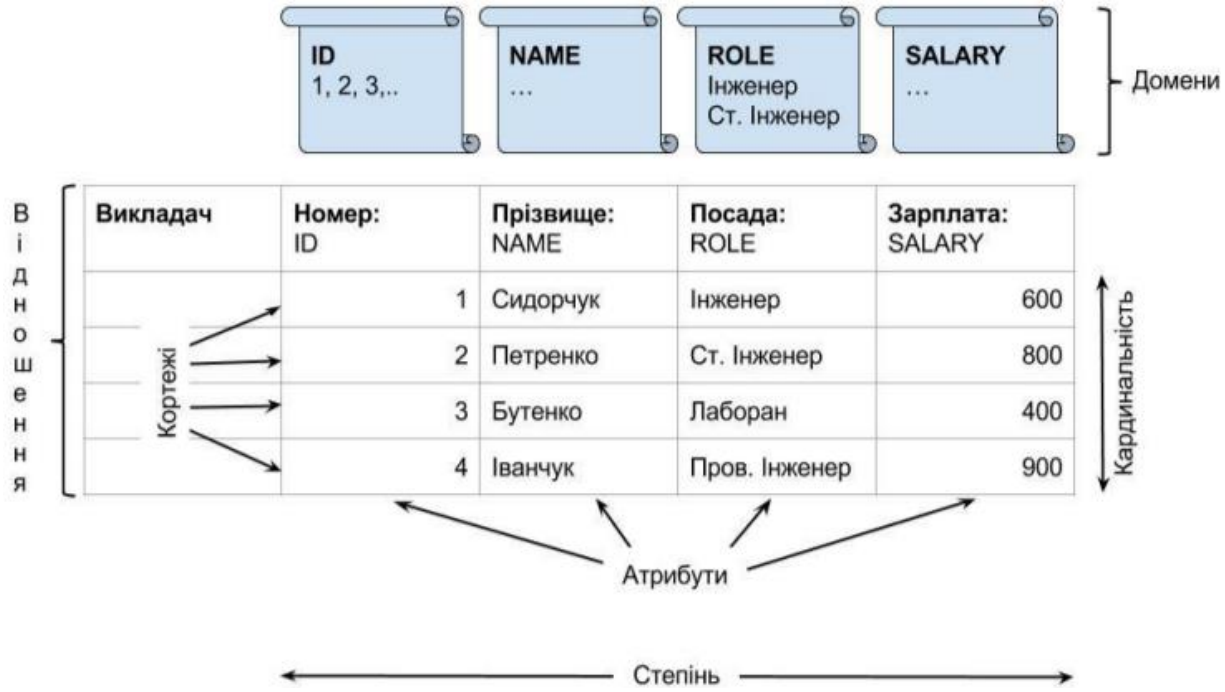


Рисунок 1.1 – Взаємозалежності між основними складовими РБД

Дальше по тексту розглянемо їх кожне окремо.

Відношення є визначальним у реляційній моделі, котре визначається набором своїх всіх елементів та їх значень. З точки зору математики його можна визначити як множину таких кортежів, котрі будуть підмножиною декартового добутку визначеної кількості областей D_i. Ці області носять назву доменів. Вони, в свою чергу, є допустимими ймовірними множинами значень визначеного типу і мають такі властивості:

– унікальна назва;

– визначені на іншому домені чи типі даних;

– певне сенсове навантаження.

Самі типи даних в розумінні РБД співставляються зі стандартними типами у синтаксисі мов програмування. Серед іншого: числові; символьні; часові; рисунки та медіа-об’єкти.

Саме ж відношення графічно можна показати як таблицю, чиї стовпці мають назву атрибутів і відповідають попаданням доменів у відношення.

Кількість атрибутів відношення – його степінь.

Кардинальність – це кількість кортежів.

Множина пар (*A, D*), де *A* – назва атрибуту, *D* – відповідний домен, носить назву схеми (заголовку) відношення.

Рядки таблиці носять назву кортежів і є впорядкованими наборами з *n* значень (*A, D, v*), де *v* – відповідне значенням атрибуту.

Тілом відношення називається невпорядкована множина різних кортежів.

Встановлений зв'язок між окремими таблицями визначає відношення власне між конкретними значеннями, які спадають, в полях, які є ключовими. Для майже всіх варіантів ключовому полю однієї з таблиць (головної), котре для окремого запису вважається унікальним варіантом ідентифікатора, у іншій таблиці (підпорядкованій) ставиться у відповідність (зв’язується) визначений зовнішній ключ.

Самі ж зв'язки між таблицями є конкретними відношеннями, котрі визначені між визначеними стовпцями (інша назва - полями) для обох таблиць.

Власне в моделі «Сутність-зв'язок» застосовуються бінарні зв'язки трьох типів: 1:1 (читається як «один до одного»); 1:M (читається як «один до багатьох»); M:M (читається як «багато до багатьох.

Функціонально встановлюються такі обмеження на всі таблиці, при тому, що кожна з них є графічним еквівалентом відношень у теорії РБД:

– кортежі-дублікати не повинні бути;

– атомарність значень параметрів;

– довільність послідовності кортежів відношення;

– обов’язкова унікальність імен параметрів відношення.

Необхідно зауважити, що поняття реляційного ключа використовується для унікальної ідентифікації будь-якого кортежу відношення.

1.2 Опис процесу МД

Термін МД має багато значень. Пов'язано це, перш за все, з використанням цього терміну в різних контекстах. У таблиці 1.1 наведено деякі визначення МД з контекстом їх використання.

Таблиця 1.1 – Визначення МД і контекст використання терміну

Визначення терміну	Контекст використання терміну	Література
Процес створення точної копії поточних даних організації з одного пристрою на іншому пристрої та перенаправлення всіх операцій введення/ виведення на новий пристрій	Заміна або оновлення технологій сервера або СД, консолідація серверів або СД, переміщення центру збору і обробки даних (data center), обслуговування апаратури серверів і СД	3
Перенесення даних між типами пристроїв зберігання даних, форматами даних і комп'ютерними системами	Зміна комп'ютерних систем, перенесення даних з одного пристрою зберігання на інший, оновлення систем, впровадження нових і консолідація існуючих додатків	4
Процес перенесення даних з однієї системи в іншу при зміні пристрою зберігання, БД або додатки	Оновлення апаратного забезпечення, перехід на нову систему, оновлення БД, злиття даних декількох систем	1
Процес переміщення даних з одного пристрою зберігання на інший з перенаправленням введення/виведення на новий пристрій	Балансування навантаження, оновлення технологій, консолідація серверів і СД, переміщення центру збору і обробки даних і ін.	5

В загальному, МД є процесом, при якому переносяться дані з ВБД в ЦБД зі схемами, які можуть бути різними.

Слід відрізняти МД від інтеграції даних. Інтеграція, на відміну від міграції - це постійна частина архітектури ІТ, і відповідальна за потоки даних між різноманітними системами і СД, і є процесом, а не діяльністю по здійсненню проекту.

Застосування МД для досягнення різних задач визначило окремі види МД за числом тих сценаріїв, котрі виконуються за певну часову одиницю [5] :

– послідовна;

– паралельна.

Подальші тлумачення згаданих видів міграції показано на прикладі РБД.

При послідовній міграції проходить виконання одного сценарію за визначену часову одиницю. Власне черговість побудови сценаріїв базується на чіткій залежності між відношеннями. При паралельній МД проходить детальніший аналіз залежностей між відношеннями. Тут виконання трапляється тільки тоді, коли дані всіх необхідних відношень вже були перенесеними.

Властиво причин може бути досить багато, через які компанії і змушені почати роботу з МД: від потреби у оновленні різних додатків і впровадженні нових корпоративних систем до повної реструктуризації через злиття фірм.

Ситуація стає складнішою, оскільки все ще застосовуються давні СУБД, зокрема dbase-III, FoxPro або Clipper. Результатом будуть противники впровадження нової версії ПЗ.

Найтяжчі наслідки буде мати рішення, при якому потрібно застосувати комплект модулів, які беруться зі старої системи. Саме тут зустрічаємо проблему МД в «реалі». Буде мати місце факт, коли накладні витрати, які стосуються підтримки роботи всіх окремих модулів взагальному ймовірно перевищуватимуть витрати, які матиме розробка нового ПЗ. Відповідно настане такий момент, при якому нова система вже буде написана, і постане питання здійсення перетворення і МД в оновлену систему.

Після аналізу МД для значної кількості ІС виділяється типова структура

МД, у складі:

– докладного аналізу існуючих форматів даних структури ВБД і ЦБД та підготовки чіткого плану міграції і перетворення реальних даних;

– визначення взаємозв'язків між таблицями (ієрархії об'єктів);

– визначення послідовності перенесення даних відповідно до встановленої ієрархії залежностей. Часом можна не враховувати взаємозв'язки, а просто відключити всі зовнішні ключі перед виконанням міграції і перебудувати їх після завершення всіх маніпуляцій з даними. Вийнятком може бути, наприклад, якщо нова версія БД вже експлуатується;

– виконання встановленого сценарію зі зміни об'єктів в новій версії БД;

– безпосереднє перенесення даних з необхідними перетвореннями «на льоту»;

– виконання сценарію для відновлення раніше відключених індексів, різних додаткових перетворень і т.д. після завершення процедури МД.

Найбільш простим варіантом виконання міграції є використання якоїсь проміжної програми, котра має зв'язуватися з ВБД і ЦБД та проводити потрібні перетворення. Рисунок 1.2 ілюструє це твердження.

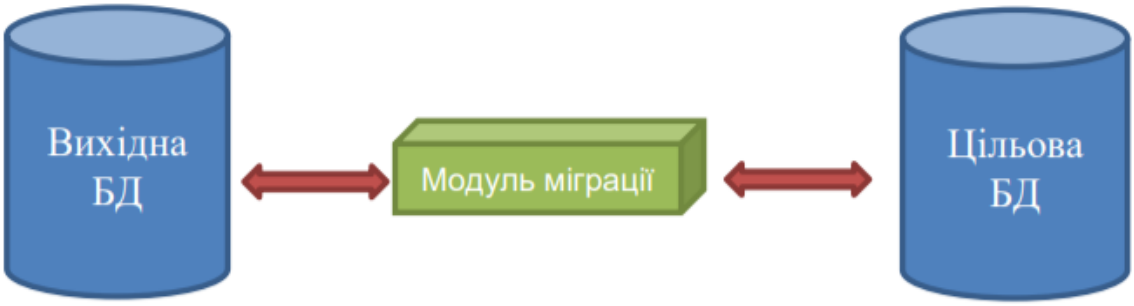


Рисунок 1.2 – Випадок проведення МД із застосуванням проміжної програми

В цьому випадку значним недоліком може бути навантаження на мережу при виконані передачі даних. За наявності великого обсягу даних мережевий обмін може мати сильний вплив на продуктивність.

Значно вигіднішим рішенням є те, при якому старі дані попереднє

завантажуються в перехідні таблиці свіжої СУБД (див. рисунок 1.3). Варто згадати, що теперішні СУБД (Oracle, як варіант), як правило, мають спеціалізовані утиліти, котрі виконують завантаження зовнішніх різноформатних даних надзвичайно швидко. Сам модуль МД створюється спеціальними мовами, які вбудовані в СУБД (PL / SQL або Java для прикладу). Також можна значно збільшити швидкість проведення процесу міграції використовуючи той факт, що вбудовані мови програмування працюють в "рідному" середовищі, оптимізовані під цільову СУБД і немає накладних витрат на обмін даними по мережі.

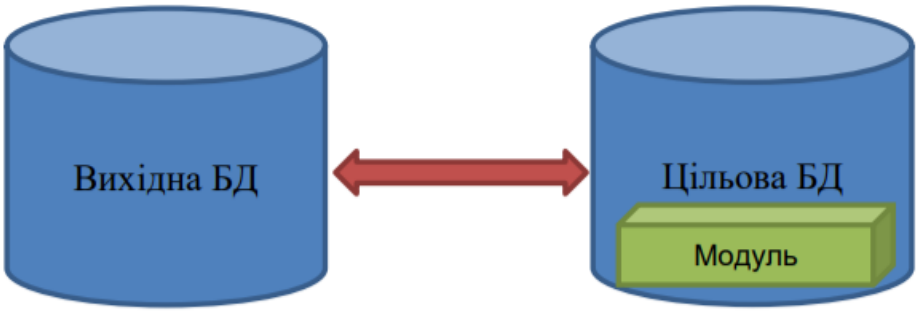


Рисунок 1.3 – Випадок виконання процесу МД засобами СУБД

Варто скористатися також пакетними SQL методами, які мають підтримку більшості СУБД і мов програмування. Так робота одного оператора INSERT або UPDATE для масиву даних разом по 100 - 200 записів даватиме значну перевагу у продуктивності, ніж робота того ж оператора по черзі з використанням циклу для кожного окремого запису.

1.3 Основні складові МД

Для розуміння суті предметної області поставленого завдання необхідно визначити такі поняття [6,7]:

– залежна таблиця – має зовнішні ключі з інших таблиць;

– конфігураційний файл – XML–документ, який передається на етап перенесення даних. У ньому міститься сформована ПОС ВБД і потрібн

інформація для проведення механізму МД (факти про таблиці і зв'язки між ними, які вже встановлені);

– історичні системи – ті БД замовника, котрі необхідно замінити при впровадженні нової розробки;

– модель – об’єднання класів, що представляють схему БД в пам’яті в уніфікованому вигляді і дозволяють працювати з її вмістом;

– незалежна стрічка даних – стрічка даних з таблиці, яка не має зовнішніх ключів, або стрічка з таблиці з зовнішніми ключами, але значення цих зовнішніх ключів для цього рядка не визначені (NULL);

– параметри з’єднання шлях до БД, ім’я користувача і його пароль;

– ПОС ВБД – послідовність таблиць ВБД, дані якої необхідно перенести в ЦБД, що визначає порядок МД;

– таблиця з «петлею», яка містить один або більше зовнішніх ключів сама з собою;

– трансформація, конвертація – технологічний процес перетворення якихось вихідних даних в дані, необхідні для завантаження. Процес трансформації проходить згідно з шаблонами для завантаження. Власне вже в результатом проведення трансформації будуть дані, необхідні для завантаження.

– успішна МД – МД, що завершилася без вийняткових ситуацій, в т.ч. неможливість перенесення через обмеження на дані в схемі БД і т.д.

1.4 Огляд окремих інструментів МД

1.4.1 Migration Architect

Є одним з найбільш популярних інтрументів МД. Розробником є Evoke software, http://www.information-management.com.

Цей застосунок використовується для побудови плану та виконання таких операцій:

– формування інформаційних СД і вітрин даних;

– МД в готові програмні рішення;

– МД додатків, яка викликана переходом на новий варіант СУБД;

– МД при застосуванні зміни схеми БД.

Властиво потрібно вказати можливості Migration Architect:

– якісний аналіз реальних даних та метаданих з метою точнішого задання області значень параметрів і контролю існування первинних ключів;

– знайдення функціональних залежностей між параметрами і знайдення надмірності даних (відношень між таблицями);

– застосування простої моделі для зберігання і знайдення відображень між параметрами ВБі ЦБД.

1.4.2 TRUmigrate

Застосунок, який дає можливість переносити дані. Офіційний сайт – http://www.valiancepartners.com.

Варто навести можливості програми:

– задання правил;

– відображення вказуються за допомогою графічного інтерфейсу і знаходяться в XML-файлі;

– добування вмісту та мета-інформації з одного СД чи кількох і можливість перенесення добутих даних в проміжне СД;

– дозволяє доповнити даними з додаткових джерел і об’єднати їх із ттими, які отрималися раніше;

– використання стандартних правил перетворень для добутих даних;

– поміщення вмісту об’єктів у потрібний додаток та ініціалізація з атрибутами налаштування.

1.4.3 Data Moving Tool

Виробник http://www.sersoftware.com/prod/data-moving-tool.

Особливості програмного засобу:

– дозволяє здійснювати експорт даних з будь-якого джерела даних (з форматами CSV, HTML, текст тощо);

– завантаження даних в будь-яке джерело даних;

– додає логіку до експорту та завантаження, виконуючи їх, лише якщо виконані певні передумови;

– може запускати зовнішні процеси до і після експорту та завантаження: SQL запити, командні рядки, FTP, відправляти електронні листи;

– працює практично з будь-якою БД (Oracle, SQL Server, Sybase, IBM DB2 / Informix, MySQL, MS Access та багато іншого, включаючи старі FoxPro, Dbase тощо).

2 ПРОЕКТНА ЧАСТИНА

2.1 Міграція як бізнес-процес

На рисунку 2.1 представлена діаграма основного бізнес-процесу – МД.

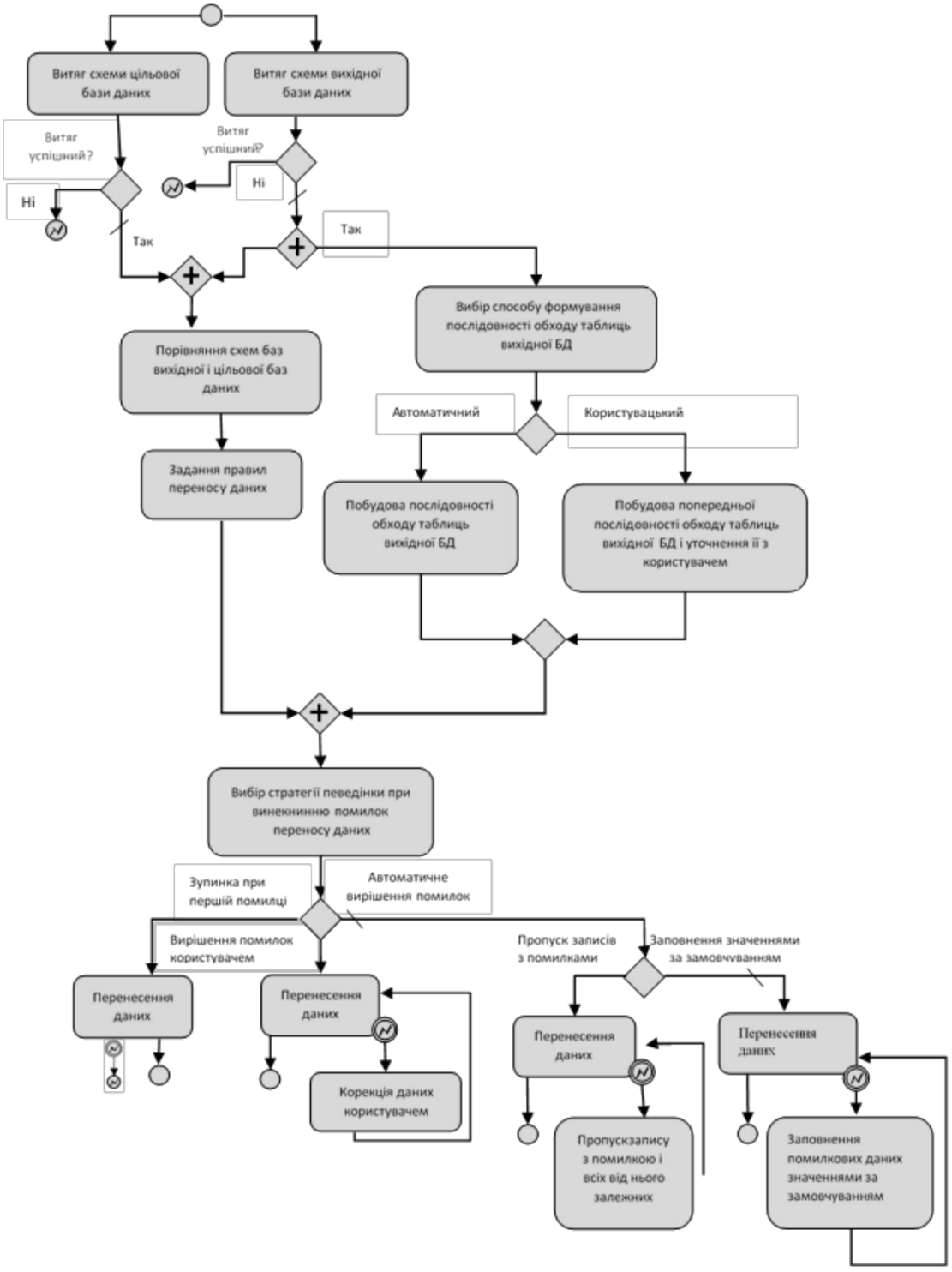


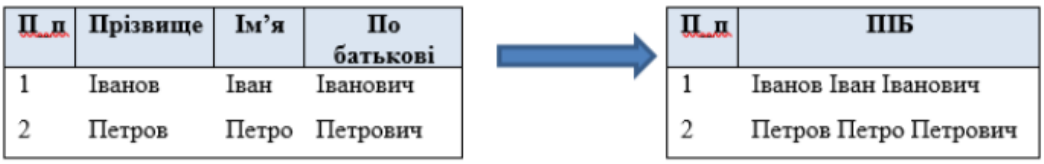
Рисунок 2.1 – Діаграма бізнес-процесу «МД»

Проаналізувавши цей бізнес процес можна визначити деякі основні принципи МД [9, 10]. Є РБД, заповнена даними (ВБД). Ці дані потрібно перенести в іншу РБД – ЦБД, але вже з можливістю зміни схеми. Для здійснення МД необхідно, в першу чергу, отримати метаінформацію про схеми ВБД і ЦБД. Для цього була розроблена відповідна модель, яка реалізується як засіб для подання витягнутої інформації про БД в пам'яті і роботи з її вмістом. Засіб дозволяє не тільки отримувати інформацію з самої БД, але і з XML-файлу, попередньо сформованого при попередньому зверненні до БД. Також були враховані виняткові ситуації. Інформація про помилки зберігається у вигляді протоколу в окремому файлі, ім'я та шлях до якого задаються до початку роботи з БД.

Так як необхідність МД може виникнути через еволюції схеми БД, то в загальному випадку вважається, що схема ВБД і схема ЦБД розрізняються. Для автоматизації визначення ступеня подібності схем БД був запропонований наступний підхід. Для кожної таблиці зі схеми ВБД обчислюється ступінь подібності з кожною таблицею зі схеми ЦБД. Ступінь подібності таблиці розраховується на підставі подібності стовпців і обмежень цілісності, а також ідентичності імен таблиць. Дані обчислення проводяться в блоці «Порівняння схем вихідної і цільової БД».

Користувачеві тут пропонується подивитися звіт, наскільки вибрана ним таблиця з ВБД схожа на таблицю1, на таблицю2, ..., на таблицюN. N – кількість таблиць в ЦБД. Ступінь подібності вимірюється в межах від 0 до 1. Також користувач може подивитися більш детальний звіт, який включатиме методи подібності стовпців і обмежень цілісності.

Далі користувачеві буде запропоновано задати правила МД: блок «Задання правил перенесення даних». Це можна проілюструвати наступним рисунком (рисунок 2.2).



П_п	Прізвище	Ім'я	По батькові
1	Іванов	Іван	Іванович
2	Петров	Петро	Петрович

П_п	ПІБ
1	Іванов Іван Іванович
2	Петров Петро Петрович

Рисунок 2.2 – Приклад задання правил перетворення

За цим прикладом видно, що відбулося злиття декількох стовпців і дані кожного рядка з стовпців Прізвище, Ім'я, По батькові були з'єднані за допомогою пробілів і в такому вигляді перенесені в таблицю ЦБД. Користувач в даному випадку повинен буде написати сценарій перетворення, враховуючи зміни схеми і використовуючи алгоритми, що здійснюють модифікацію даних. Далі ці сценарії використовуються безпосередньо алгоритмом міграції при перенесенні даних, так звані, перетворення «на льоту».

При міграції важливо знати, в якому порядку переносити дані. Тому в рамках даного блоку відбувається діалог з користувачем і вибір типу формування ПОС ВБД: автоматичний; призначений для користувача.

При автоматичному типі побудови послідовності кінцевий результат вибирається як перший зі списку можливих альтернатив. При виборі користувацького задання послідовності система формує все можливі альтернативні послідовності, одна з яких вибирається користувачем в якості кінцевої. Під час МД можливе виникнення помилок, таких як неможливість перенесення через невідповідність типів даних, невідповідності обмеженням цілісності і інші.

При виникненні помилок такого роду користувачеві пропонується вибрати конкретну стратегію дій із запропонованих варіантів:

– зупинка міграції при виникненні першої помилки;
– введення коректних значень користувачем;
– пропуск записів з помилками і всіх залежних записів;
– заповнення значенням за замовчуванням.

Спосіб переносу даних («Блок перенос даних») крім сценаріїв перетворення застосовує ще два типи сценаріїв. Сценарії першого типу служать

для відімкнення тригерів і певних обмежень при перенесенні даних, а другого – для їх включення після МД.

2.2 Загальна ідея побудови ПОС БД

Є РБД, заповнена даними. Ці дані потрібно перенести в іншу РБД (це окремий випадок; в загальному випадку – неважливо, куди переносити дані), але вже, можливо, зі зміненою схемою [8].

В першу чергу необхідно визначити, в якій послідовності переносити дані. Є декілька випадків, які розглядаються нижче. Найпростіший варіант – є одна таблиця. В цьому випадку можна запропонувати тільки один підхід: просто перенести все записи з даної таблиці в ЦБД.

Більш складний варіант схеми БД (рисунок 2.3).

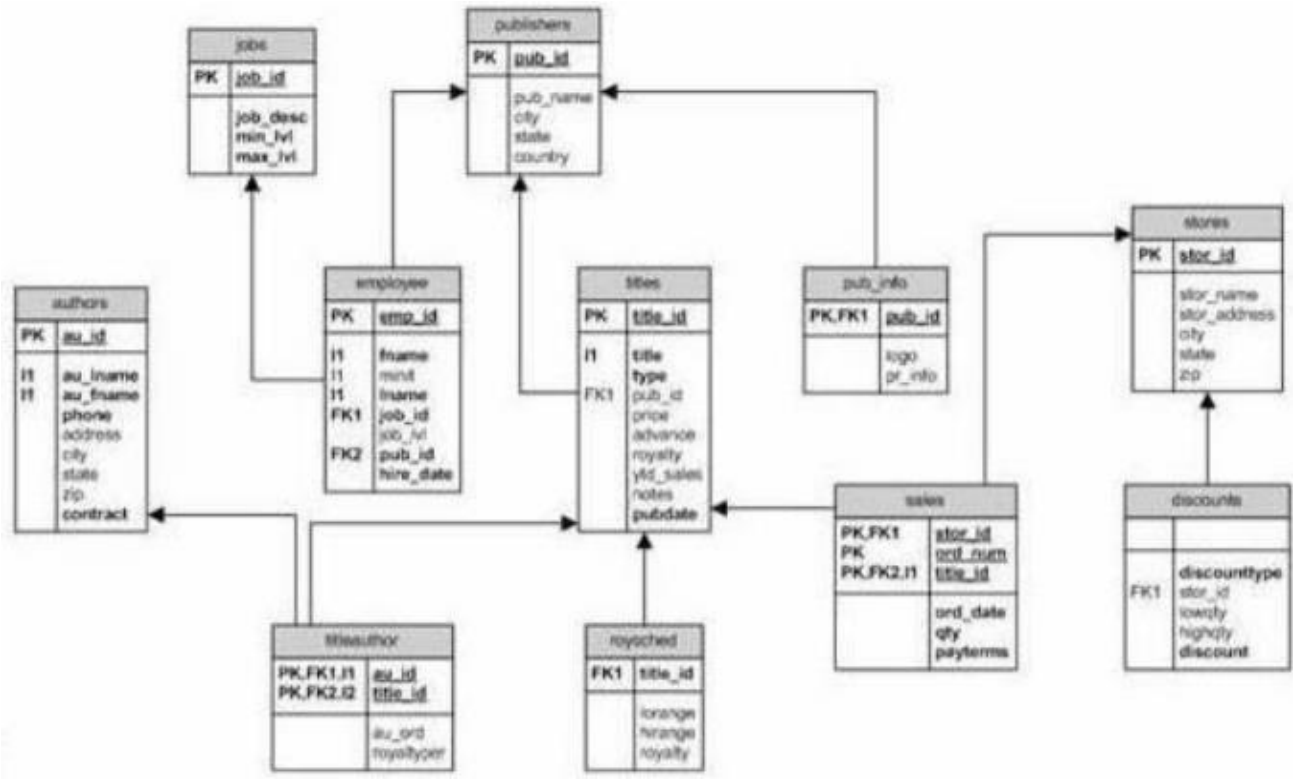


Рисунок 2.3 – Більш складна схема БД

В даному випадку складно з першого погляду визначити, в якому порядку переносити дані. Тому визначення порядку, згідно з яким буде відбуватися МД, є

важливим завданням. Таким чином, необхідно виявити правила, що встановлюють порядок на множини таблиць схеми БД, і запропонувати новий алгоритм побудови ПОС.

Спосіб МД має забезпечити огляд і переміщення всіх записів з ВБД, забезпечити цілісність даних та що переміщення записів проходитиме з урзі всіма залежностями. Будь який запис можна перемістити тільки тоді, коли він є незалежним, чи ті записи, від яких, власне, і залежать дані, переміщені в ЦБД до того. Тому основною нормою, котра і визначає форму огляду та переміщення, є залежності між записами.

Спосіб МД імплементується як процедура, котру спершу викликають для використання з незалежними записами, а далі роблять те ж саме рекурсивно вже і для залежних записів. Ось чому порядок переходу між ними властиво повинен стартувати з таблиці з незалежними записами. Саме в цьому випадку головна послідовність, котра визначена на множині таблиць схеми БД, це є, властиво, залежність між таблицями.

Такий підхід дозволяє подати схему БД у вигляді орієнтованого графа (див. рисунок 2.4).

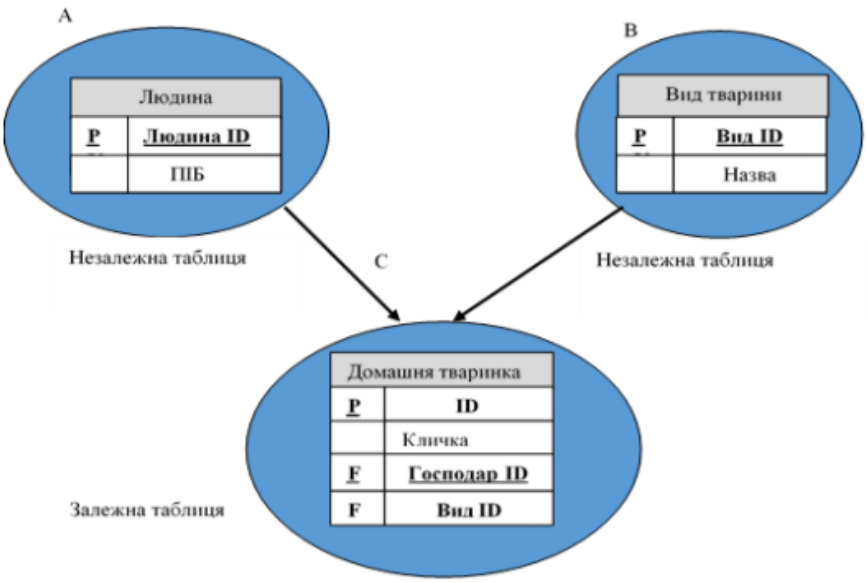


Рисунок 2.4 – Відображення схеми БД як орієнтованого графа

Вершинами є таблиці, а дугами – залежності між таблицями, причім дуга

починається з незалежної таблиці і завершується в залежній таблиці. Окреме дерево формується для кожної незалежної вершини із застосуванням алгоритму перегляду вглиб. Таким чином сумістивши дерева перегляду для всіх незалежних вершин можна отримати загальне дерево перегляду. Воно і є бажаним порядок переміщення, причім механізм МД можна розпочинати з будь-якої незалежної вершини [11].

Така сформована послідовність, а також ще необхідні факти про схему ВБД для проведення механізму МД (про таблиці та зв'язки між ними) записується у конфігураційний файл. Вподальшому він піддається аналізу аналізується і буде застосований при переміщенні даних.

2.3 Розробка алгоритму побудови ПОС

2.3.1 Опис

За основу була взята ідея алгоритму пошуку вглиб. Алгоритм був значно модифікований. Введено додаткові обмеження, перевірки, обробка виняткових ситуацій. На першому етапі проводиться перегляд всіх таблиць і встановлення зв'язку між таблицями типу «Залежна від конкретної таблиці», «Незалежна таблиця». Таблиці з «петлями» також вважаються незалежними. Отже, на першому етапі для кожної таблиці формується список залежних від неї таблиць.

Для кожної таблиці визначено поле Counter (Лічильник), значення якого показує число входжень даної таблиці в ПОС.

Якщо значення цього поля дорівнює 0, то вважається, що дана таблиця вже включена в послідовність необхідну кількість разів. На першому етапі у всіх незалежних таблиць полю Лічильник присвоюється значення «1». Далі, переглядом графа вглиб, для всіх залежних таблиць полю Лічильник присвоюється значення, яке дорівнює загальній кількості значень полів Лічильник всіх таблиць, від яких поточна таблиця залежить.

З незалежних таблиць можна починати перегляд для формування ПОС (визначається за значенням поля Лічильник). Якщо ж таких таблиць не виявиться, то пріоритет матимуть таблиці, що містять зовнішній ключ з однієї

таблиці (визначається за значенням поля Лічильник). Тут також більш пріоритетними будуть вважатися таблиці з «петлями».

Рекурсивно спускаючись від незалежних вершин до термінальних вершин (вершини, в яких закінчується рекурсія – вершини, які не мають залежних вершин), формується послідовність. Причому після включення таблиці (вершини) в послідовність значення поля Лічильник зменшується на одиницю. Алгоритм завершується тоді, коли не залишиться таких таблиць, у яких значення поля Лічильник не дорівнюватиме 0. Далі послідовності перегляду для всіх незалежних вершин об’єднуються в одну загальну, яка і буде кінцевою ПОС БД.

2.3.2 Результати роботи

Можна проаналізувати результати роботи алгоритму для найбільш частих випадків. Відношення між таблицями 1:M, 1:1.Для роботи алгоритму не має значення, яке відношення – 1:1 або 1:M, важливо лише напрямок залежності. Тому, результуюча послідовність має вигляд AB. Так як спочатку необхідно перенести незалежні записи (вони зберігаються в таблиці A), а потім і всі залежні від них (залежні записи зберігаються в таблиці B).

Відношення між таблицями M:N в схемах БД представляються наступним чином (рисунок 2.5).

Рисунок 2.5 – Подання в схемах БД відношення M:N

Після проходження попереднього етапу алгоритму є наступні значення поля Лічильник для таблиць:

A.Counter = 1;

B.Counter = 1;

C.Counter = 2.

Отже, результуюча послідовність має вигляд ACBC або BCAC.

Відношення типу «петля» (рисунок 2.6).

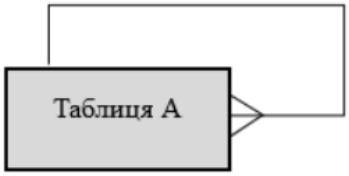


Рисунок 2.6 – Відношення типу «петля»

При такому варіанті таблиця буде включена один раз в послідовність, але при цьому властивість таблиці boolean Loop встановлюється рівною true. Результуюча послідовність – A. A.Loop = true. При виконанні алгоритму МД, якщо виявляється таблиця, у якої поле Loop приймає істинне значення, то в першу чергу необхідно перевіряти кожен із записів на незалежність всередині таблиці.

Якщо запис є залежним, то необхідно виконати пошук незалежного запису всередині таблиці, від якої даний запис залежить, і тільки потім відбувається виклик рекурсивної процедури МД для даного незалежного запису.

Однак може зустрітися випадок виникнення орієнтованого циклу на рівні записів. На даному етапі розвитку проекту ще не прийнято однозначне правило поведінки в даному випадку. Також складність представляє випадок виникнення орієнтованого циклу на рівні таблиць (рисунок 2.7). У цьому випадку також можлива присутність орієнтованих циклів на рівні даних.

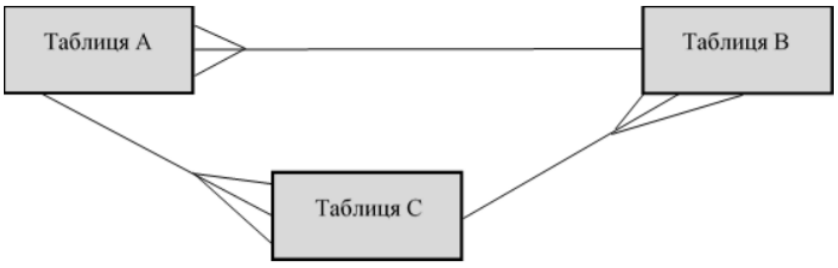


Рисунок 2.7 – Орієнтований цикл

Для вирішення даних проблемних ситуацій передбачаються додаткові теоретичні дослідження і удосконалення алгоритму складання ПОС БД.

2.4 Альтернативний алгоритм побудови ПОС

Для аналізу ефективності розробленого алгоритму було прийняте рішення розробити альтернативний алгоритм, а потім провести їх порівняння. Схема БД, як і в попередньому алгоритмі, представляється у вигляді орієнтованого графа.

За основу була взята ідея алгоритму топологічного сортування вершин графа. Однак сам алгоритм був значно модифікований – крім ярусної нумерації введена нумерація таблиць, які стосуються одного ярусу. Також введені додаткові обмеження і обробка виняткових ситуацій, що виникають через специфіку вирішуваної проблеми.

Окрема ітерація матиме такий вид.

Крок 0: Номер ярусу L=0.

Крок 1: Знаходяться всі незалежні таблиці. Якщо таких таблиць немає, то перехід на Крок 4. Якщо є, то відносимо ці таблиці до ярусу L. Перехід на Крок 2.

Крок 2: (Нумерація таблиць на одному ярусі). Всім незалежним таблицями приписується номер ярусу L. Оскільки таблиці, мають зовнішній ключ з самих себе, також вважаються незалежними, то при нумерації таблиць на одному рівні необхідно це враховувати. Отже, таблиці, які не мають зовнішнього ключа, навіть з самих себе, будуть вважатися більш пріоритетними, тому послідовно нумеруються ці таблиці. І тільки після цього нумеруються таблиці, що мають зовнішні ключі з самих себе, за наступного правила: таблиця, що має меншу кількість зовнішніх ключів, має найбільший пріоритет і, очевидно, найменший номер в послідовності.

Крок 3: Для кожної пронумерованої на попередньому кроці таблиці (Table_1) переглядається список залежних від неї таблиць. У всіх залежних таблиць видаляється зовнішній ключ, що посилається на Table_1. Видаляється з тимчасової схеми таблиця Table_1, що переглядалася. Збільшується номер ярусу:

L++.

Крок 4: Якщо всі таблиці пронумеровані, то перехід на Крок 6. Якщо ні, то це означає, що в схемі БД є цикли. В цьому випадку перехід на Крок 5.

Крок 5: (Обробка ситуації виявлення циклів). Перегляд всіх таблиць, які утворюють цикл. Вибір таблиці для внесення в послідовність перегляду за наступними критеріями, розглянутими в порядку важливості: кількість зовнішніх ключів з різних таблиць (зовнішні ключі з однієї таблиці будуть вважатися за один ключ). Вибір тієї таблиці, яка має менше число зовнішніх ключів з різних таблиць; кількість залежних таблиць. Вибір тієї таблиці, яка має більшу– кількість залежних таблиць. Далі приписується даній таблиці ярус L і відповідний номер на ярусі, після чого збільшується номер ярусу: L++. У таблиці переглядається список залежних від неї таблиць, видаляючи при цьому зовнішній ключ, що посилається на дану пронумеровану таблицю. Далі перехід на Крок 1.

Крок 6: завершення алгоритму.

2.5 Порівняльний аналіз алгоритмів побудови ПОС

Для вирішення поставленої проблеми було розроблено два алгоритми – на основі ідеї пошуку вглиб (Алгоритм1) і топологічного сортування вершин графа (Алгоритм2). Далі наведені переваги і недоліки цих алгоритмів.

Перевагою другого алгоритму є швидка збіжність при побудові послідовності.

Недоліки Алгоритму2: для алгоритму МД необхідно виявлення таблиці (розташована на рівні L2), яка залежить від таблиці, що переглядається (розташована на рівні L1, причому L1<L2). Для цього на кожному кроці алгоритму міграції доводиться аналізувати всі таблиці рівня L2, що робить алгоритм досить заплутаним і трудомістким.

Переваги Алгоритму1:

– швидка збіжність;

– зручне представлення результату (у вигляді дерева рішення);

– простий аналіз результату при виконанні алгоритму МД;

– обробка виняткових ситуацій.

На основі даного порівняльного аналізу було прийнято рішення про практичну реалізацію Алгоритму1.

2.6 Алгоритм процесу МД

Перед тим як описати механізм МД, слід узагальнити викладене в попередньому розділі. На даному етапі є:

– модель, яка уніфікує доступ до даних, що зберігаються в БД, а також дозволяє отримувати необхідну метаінформацію про схему БД;

– сформована ПОС ВБД, збережена в файлі конфігурації.

Підготовчий етап: Відновлення порядку пересування між схеми ВБД з конфігураційного файлу. Перехід на Етап1 (i=0).

Етап 1 (виділення таблиці з послідовності): виділяється з послідовності таблиця з індексом i. Якщо вона не існує, перехід на етап 7. Інакше для i-ої таблиці перехід на етап2 (n=0).

Етап 2 (ітерація для таблиці): виділяється n-ий запис з таблиці. Якщо такий запис вже існує, тоді для нього відбувається перехід на етап 4. Якщо індекс n більший, ніж кількість записів в таблиці, значить, всі записи з цієї таблиці вже були розглянуті. Перехід на етап 1 (i++).

Етап 3 (ітерація для набору записів): виділяється k-ий запис з набору. якщо такий запис існує, то перехід для нього на етап 4.

Якщо індекс k більший ніж кількість записів в наборі, значить, всізаписи з нього вже були розглянуті. Перехід на етап 2 n++.

Етап 4 (окрема ітерація): перевірка, чи був перенесений запис раніше. Якщо так: n++; етап 2. Інакше: перевірка запису на незалежність (перевірка переглядом зовнішніх ключів). Якщо запис є незалежним, або записи, від яких

залежить він, були перенесені раніше, то даний запис можна перенести. Перехід на етап 6. Інакше: не можна перенести даний запис; n++; етап 2.

Етап 5 (формування списку залежних записів): Для цього запису формується список всіх залежних від нього записів (набір записів). Отриманий RecordSet і k=0 передається на етап3.

Етап 6 (міграція): запис переноситься в нову схему БД, а перетворення «на льоту» здійснюються за допомогою сценаріїв перетворення. У разі успішного перенесення даних з'являється запис як перенесено і перехід на етап 5. При виникненні помилки подальшу поведінку системи вибирається виходячи з стратегії МД.

Етап 7 (завершення міграції): припинення процедури МД. Зауваження: для кожної таблиці (набору записів) параметр n (k) свій.

2.6.1 Варіанти позначення перенесених записів

В алгоритмі МД потрібно розрізняти вже перенесені записи в ЦБД і ще не перенесені записи.

У процесі розробки програми були запропоновані наступні рішення:

1. Розширення всіх таблиць в самій БД шляхом додавання до кожної таблиці додаткового стовпчика. При цьому якщо запис перенесений, то значення комірки цього запису – істина, інакше – хибність (рисунок 2.8).

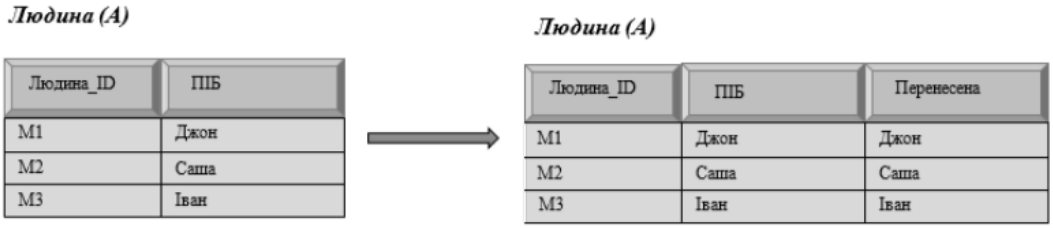


Рисунок 2.8 – Приклад розширення таблиці в ВБД для позначення перенесених записів

Переваги: не потрібно додаткових інструментів, що дозволяють визначати, перенесений даний запис чи ні; всю необхідну інформацію можна отримати за

допомогою SQLзапитів.

Недоліки: велика кількість звернень до БД, внаслідок чого процес перевірки виходить трудомістким, і як результат, трудомісткий механізм МД; так як відбувається зміна схеми ВБД, виникає складність з установкою відображень елементів схеми ВБД в елементи схеми ЦБД; через зміни схеми ВБД механізм міграції неможливо– провести без зупинки сервісу БД.

2. Використання курсору (рисунок 2.9).

Переваги: простота у використанні.

Недоліки:

– обмеження на кількість одночасно відкритих курсорів. А при такому підході необхідна кількість курсорів збігається з кількістю таблиць з схемою БД;

– обхід схеми буде строго послідовним (через властивості курсора), отже, немає гнучкості при зверненні до записів.

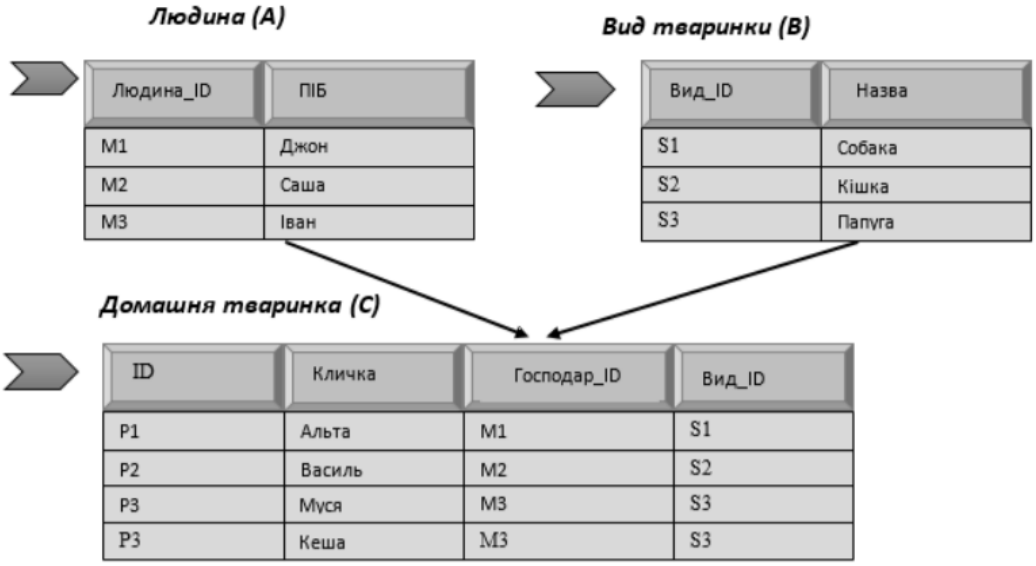


Рисунок 2.9 – Приклад обходу даних в ВБД з використання курсорів

3. Застосування вбудованої БД.

Переваги:

– гнучкість в поданні даних;

– гнучкість в управлінні даними;

– швидкість і простота звернень до даних.

Аналізуючи запропоновані варіанти, було прийнято рішення використовувати третій підхід. Як вбудована БД використовується Berkeley DB Java Edition. Вона є бібліотекою, високопродуктивною вбудованою БД. BDB - це нереляційна БД. Варто відмітити, що вона може обслуговувати велике число процесів або потоків, що одночасно контролюють БД об’ємом в 256 терабайт та різноманітне обладнання під різними ОС, в т.ч. більшість UNIX-подібних і Windows, а також на ОС реального часу. BDB володіє функціями БД, зокрема ACID-транзакції, детальні блокування, інтерфейс XA, гарячі бекапи і реплікацію.

Можна застосовувати як інструмент для формування збережених індексів, так і як СД. Власне кажучи, BDB - СУБД-невидимка, якою користуються мільйони людей, не підозрюючи про її існування, так як вона являє собою "движок" - ядро СУБД, над яким програмісти - розробники створюють свої СУБД, ОС і інші додатки, які оперують великими масивами даних. Постачається з джерельним кодом, засобами збірки, тестування та комплектом документів. Якісний код і практичність вкупі з вільною ліцензією дозволило використовувати BDB в багатьох програмах.

BDB пропонується в трьох різних варіантах: DB - власне бібліотека мовою C; DB Java - бібліотека, переписана на Java (підтримка Google Android, Apache Maven); DB XML - бібліотека на C, що реалізує XML-СУБД на основі BDB із засобами роботи з XML (Xerces, XPath, XQuery, XQilla).

BDB входить до складу більшості дистрибутивів Linux. Існують інтерфейсні засоби для роботи з Berkeley DB на Perl, Python і іншими мовами.

BDB відрізняється своєю простою архітектурою в порівнянні з іншими системами БД, такими як, наприклад Microsoft SQL Server і Oracle Database. Наприклад, в ній відсутній мережевий доступ - програми використовують БД через виклики внутрішньопроцесного API. Вона підтримує SQL в якості одного з інтерфейсів, починаючи з версії 5.0, хоча і не підтримує стовпці в таблицях в традиційному розумінні на рівні внутрішньої архітектури.

Berkeley DB дозволяє програмісту, який використовує бібліотеку, зручно оперувати БД і записами в них. У трактуванні деяких термінів Berkeley DB відрізняється від високорівневих СУБД, в більшості ж випадків термінологія

аналогічна. Так, запис Berkeley DB, що є парою, утвореної ключем-ідентифікатором запису і даними записи, в деякому сенсі ідентичний запису в двоколонковій таблиці в термінах РБД. Але так як і дані, і навіть ключ в Berkeley DB можуть самі бути наборами даних (наприклад, структурами мови С), то цілком припустимі аналогії і з багатоколонковими таблицями. Поняття БД в Berkeley DB, навпаки, радикально відрізняється від терміну високорівневих аналогів - тут це множина записів з однаковими типами ключа і даних, що формується на підставі обраного програмістом механізму доступу (в високорівневих СУБД, в свою чергу, останній зазвичай повністю прихований).

БД передбачає роботу з парами ключ-значення, де ключ і значення можуть мати фіксовану або змінну довжину, а функція порівняння ключів може бути написана і призначена прикладним програмістом. Програма, яка використовує БД, сама вирішує, як дані зберігаються в запису; БД не накладає обмежень на дані, що зберігаються в записах.

BDB можу бути застосовувати як інструментом для побудови збережених індексів, так і СД. Як ключ виступає ідентифікатор запису в таблиці (передбачена робота як з сурогатними, так і з природними ключами), а в якості значення – структура: переглянута/не переглянута, перенесена/не перенесена, код помилки (запис не можливо перенести через помилку).

У лістингу 2.1 показано приклад опису класу для зберігання даних.

Лістинг 2.1 – Клас для зберігання даних в Berkeley DB

```
@Entity
public class RecordBD
{
@PrimaryKey
private Map id;
private boolean read;
private boolean moved;
private String error;
private Map fKVal;

…
}
```

Значення зовнішніх ключів використовуються для: визначення, чи є запис залежним або незалежним;− пошуку залежних записів. Пошук залежних записів ефективніше− проводити у вбудованих БД, а не в ВБД, тому що звернення до ВБД є досить трудомісткою операцією.

2.6.2 Перетворення «на льоту»

Здійснюються запуском відповідних сценаріїв, написаних на етапі задання правил переміщення даних.

Застосовується для відімкнення тригерів і деяких обмежень. Імплементація сценаріїв такого роду проходить до переміщення даних. Після цього необхідно назад включити раніше відключені тригери і обмеження. Для цих цілей використовуються спеціальні сценарії, виконання яких відбувається після завершення механізму МД.

Переміщення даних починається з перегляду сформованої раніше ПОС ВБД. Для чергової таблиці, що переглядається при наявності відповідного сценарію відбувається його виконання.

Залежно від того, які правила перетворення задані, сценарії перетворень (виконуються під час перенесення даних) слід розділити на наступні групи:

- операція з одним кортежем;
- операція з набором кортежів.

Далі слід провести аналіз перетворень для кожної групи сценаріїв. Операція з одним кортежем (рисунок 2.10).

Людина (вихідна база даних)

Людина_ID	Прізвище	Ім'я	По батькові
1	Іванов	Іван	Іванович
2	Петрів	Петро	Петрович

Людина (цільова база даних)

Людина_ID	ПІБ
1	Іванов Іван Іванович
2	Петрів Петро Петрович

Рисунок 2.10 – Перетворення, в рамках якого здійснюються операції з одним кортежем

Операція з набором кортежів (рисунок 2.11).

В даному випадку схеми ВБД і ЦБД однакові, проте, для здійснення переносу даних виробляється перетворення декількох кортежів (або одного) з ВБД в один кортеж ЦБД.

Навантаження викладача (вихідна база даних)

ID	Предмет	Вид	Викладач_ID	Кількість годин	Семестр
1	математика	Лекції	1	25	1
2	математика	Практика	2	20	1
3	математика	Лекції	1	20	1
4	математика	Лекції	2	35	1

Навантаження викладача (цільова база даних)

ID	Предмет	Вид	Викладач_ID	Кількість годин	Семестр
1	математика	Лекції	1	45	1
2	математика	Практика	1	20	1
4	математика	Лекції	2	35	1

Рисунок 2.11 – Перетворення, в рамках якого здійснюються операції з набором кортежів

2.6.3 Помилки на рівні даних

У процесі безпосереднього перенесення даних з ВБД в ЦБД можуть виникнути помилки на рівні даних.

Нижче наведена найпростіша класифікація змін, які можуть привести до появи помилок на рівні даних:

– зміна типу даних: поле стрічкового типу змінилося на поле з цілочисельним значенням;

– зміна максимальної довжини поля: поле стрічкового типу мало максимальну довжину, рівну 100 символів, а при зміні схеми БД максимальна довжина зменшилася до 50 символів;

– зміна формату даних поля: дата раніше зберігалася в форматі ДДММ-РРРР, а тепер зберігається в форматі ДД-ММ-РРРР;

– раніше необов’язкове поле (можливо NULL) змінило свої–властивості на поле, обов’язкове для заповнення (NOT NULL).

Найчастіше цих помилок можна уникнути при коректному написанні сценаріїв, що передбачають подібні зміни, або ж при виборі відповідної стратегії обробки виключень в разі виникнення помилок.

Залежно від обраної стратегії обробки виключень в разі виникнення помилок на рівні даних система буде виконувати наступні дії:

– зупинка механізму МД при виявленні першої помилки;

– при виникненні помилки перенесення запису і всіх залежних від нього записів не відбувається, записи позначаються як помилкові. Інформація про виниклі помилки фіксується в лог-файлі. МД при цій стратегії триває;

– поля, в яких виникла помилка, заповнюються значенням за замовчуванням. Фіксація інформації в лог-файлі. Перенесення вважається успішним, отже, МД триває;

– при кожній помилці користувачеві надається змога ввести коректне значення. МД при цьому триває.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБУ МД

Засіб створено в NetBeans IDE 6.8 на мові Java [12-14]. При реалізації були використані наступні бібліотеки: mysql- connector-Java-5.1.12-bin, log4j-1.2.16 і je-4.0.92. Для роботи зі схемою БД використовується модель, відома в літературі як model-metadata-database-2.0.0.

Дана модель представлена бібліотекою класів, а її концептуальне уявлення наведено на рис. 3.1.

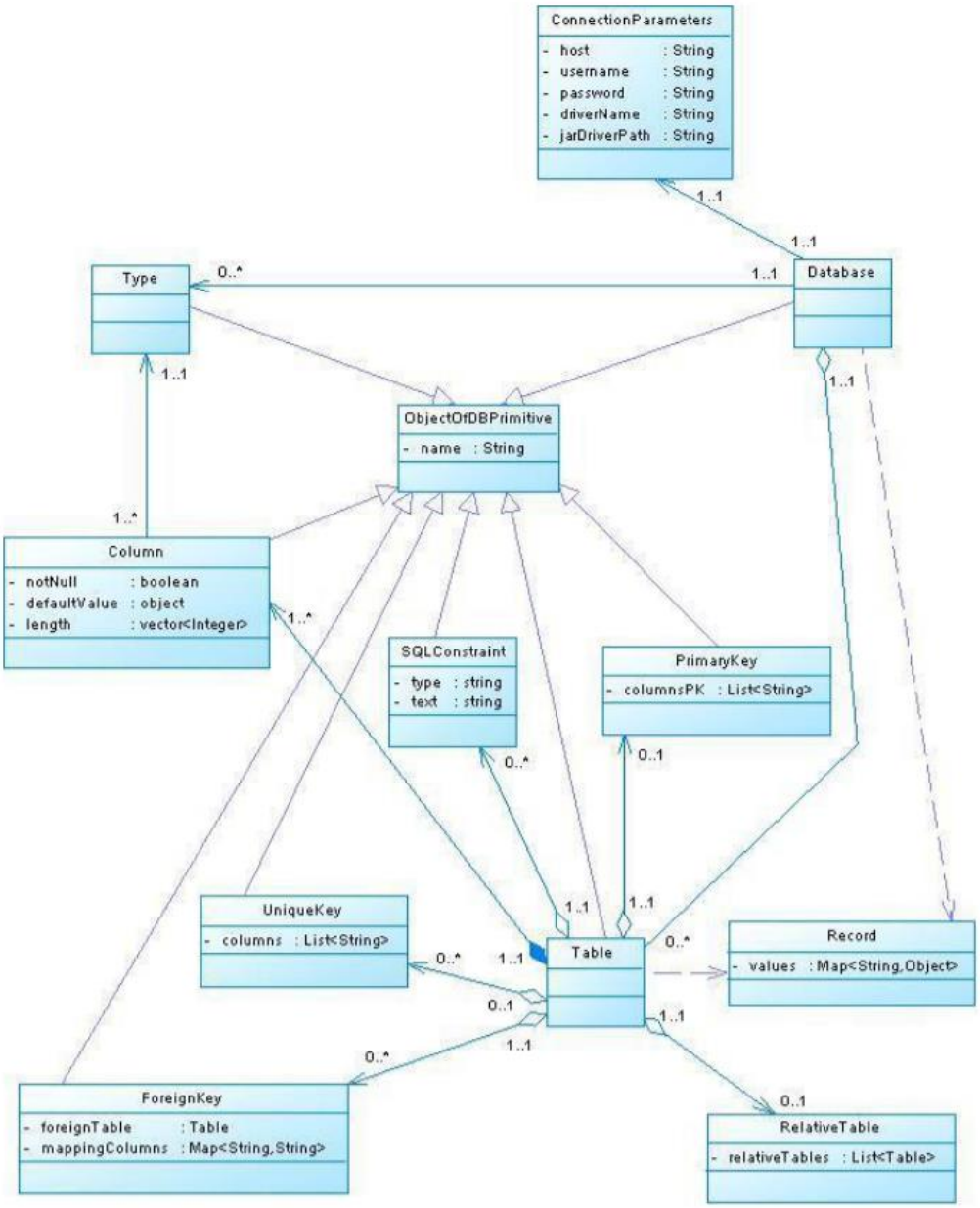


Рисунок 3.1 – Модель для представлення схеми БД в оперативній пам’яті

На рисунку 3.2 наведена діаграма пакетів, що ілюструє структуру програмного додатку.

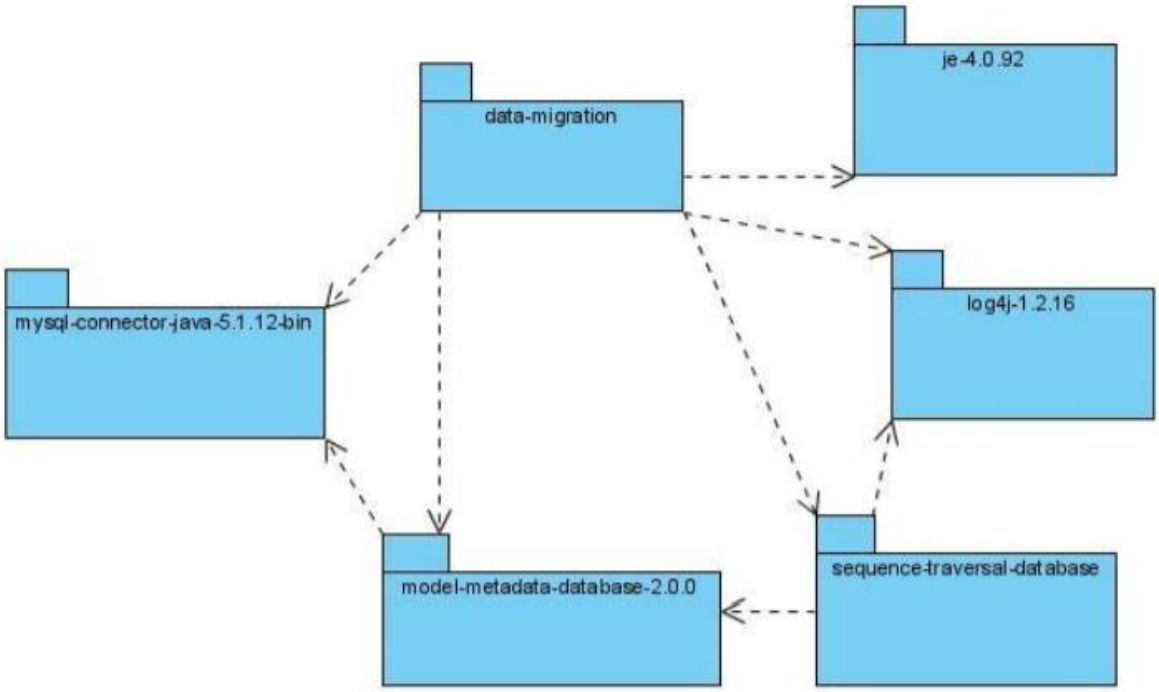


Рисунок 3.2 – Діаграма пакетів

Основними елементами діаграми є:

– mysql-connector-Java-5.1.12-bin – бібліотека, що містить необхідний інструментарій для роботи з БД MySQL;

– log4j-1.2.16 – бібліотека, що надає засоби для протоколювання помилок, попереджень, що виникають при виняткових ситуаціях;

– je-4.0.92 – бібліотека, яка містить необхідний інструментарій для роботи з Berkeley DB Java Edition;

– model-metadata-database-2.0.0 – бібліотека, яка реалізує модель для швидкого і уніфікованого доступу до схеми БД, а також засоби для порівняння схем ВБД і ЦБД;

– sequence-traversal-database – бібліотека, яка підтримує реалізацію алгоритму формування ПОС ВБД, а також методи, збереження і читання конфігураційного файлу.

– конфігураційний файл зберігається у форматі XML. Формат XML файлу представлений в лістингах 3.1 та 3.2.

– data_migration – пакет, в якому реалізовані необхідні методи для здійснення МД.

Лістинг 3.1 – DTD-схема, що відповідає конфігураційному файлу

```
<!ELEMENT Tables (table*)>
<!ELEMENT table (table.Name, column)>
<!ELEMENT column (column.ColumnName, IsInPrimaryKey?, NotNull?,
Type)>
<!ELEMENT table.Name (#PCDATA) >
<!ELEMENT column.ColumnName (#PCDATA) >
<!ELEMENT IsInPrimaryKey (#PCDATA) >
<!ELEMENT NotNull (#PCDATA) >
<!ELEMENT Type (#PCDATA) >
```

Лістинг 3.2 – Приклад XML-документу

```
<!DOCTYPE Tables SYSTEM "example.dtd">
<?xml version="1.0" encoding="UTF-8" ?>
<Tables>
<table Name="table1">
<column ColumnName="table1_id" IsInPrimaryKey="true"
NotNull="true" Type="INT UNSIGNED" />
<column ColumnName="name" IsInPrimaryKey="false"
NotNull="false" Type="CHAR" />
<column ColumnName="year_of_birth" IsInPrimaryKey="false"
NotNull="false" Type="INT" />
</table>
<table Name="table2">
<column ColumnName="table2_id" IsInPrimaryKey="true"
NotNull="true" Type="INT UNSIGNED" />
<column ColumnName="table1_id" IsInPrimaryKey="true"
NotNull="true" Type="INT" />
</table>
</Tables>
```

3.1 Бібліотека sequence-traversal-database

Опишемо докладніше бібліотеку, яка підтримує реалізацію алгоритму формування ПОС ВБД, а також методи, збереження і читання конфігураційного файлу [15, 16].

Діаграма основних класів бібліотеки sequence-traversal-database приведена на рисунку 3.3.

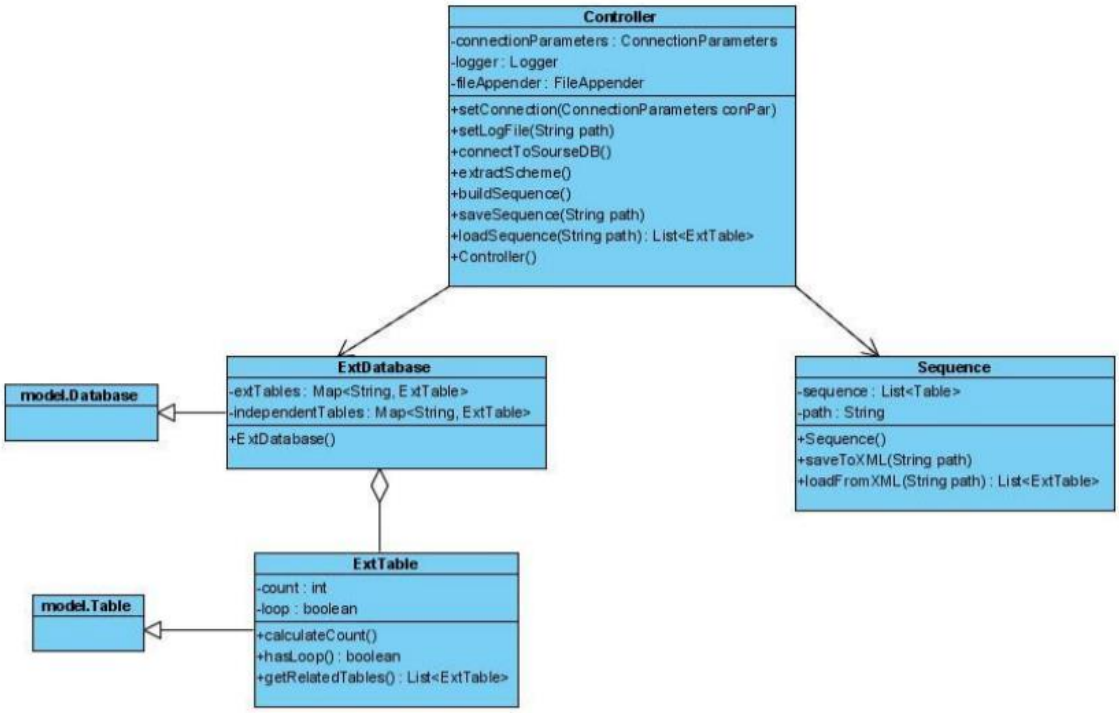


Рисунок 3.3– Діаграма класів бібліотеки sequence-traversal-database

Бібліотека sequence-traversal-database реалізована в середовищі NetBeans 6.8 на мові Java. При реалізації були використані наступні бібліотеки: mysqlconnector-Java-5.1.12-bin і log4j-1.2.16. Для роботи зі схемою БД використовується модель model- metadata-database2.0.0.

До основних файлів реалізації бібліотеки відноситься:

– Controller.Java;
– ExtTable.Java;
– ExtDatabase.Java;
– Sequence.Java.

Controller – керуючий клас, що забезпечує взаємодію між класами ExtDatabase і Sequence. Методи даного класу дозволяють будувати ПОС БД, зберігати її і завантажувати в конфігураційний файл, а також встановлювати з’єднання з початковою БД і витягувати схему ВБД.

ExtDatabase – клас, який зберігає всю необхідну інформацію про БД для формування ПОС ВБД. Даний клас успадковує клас model.Database з моделі рис. 3.3.

ExtTable – клас, який успадковує клас model.Table з моделі, представленої

на рис. 3.3. Функціональність даного класу розширена для зручного застосування при побудові ПОС ВБД.

Sequence – клас для зберігання, побудови і завантаження з XML-файла ПОС БД.

Основний клас Controller – клас, що задає керуючу логіку програми. Основні поля класу:

– connectionParameters – містить необхідну інформацію для встановлення з'єднання з початковою БД;

– logger – використовується для запису помилок, що виникають в процесі роботи, в лог-файл;

– fileAppender – файл, в який записуються помилки.

Основні методи класу:

– Controller () – конструктор, створення екземпляра класу Controller;

– setConnection (ConnectionParameters) – задання параметрів з'єднання з початковою БД, таких як шлях до БД, ім'я користувача, пароль;

– setLogFile (String path) – метод, що задає шлях до файлу, в якому будуть міститися відомості про помилки;

– connectToSourseDB () – встановлення з'єднання з початковою БД. Вхідний параметр – екземпляр класу ConnectionParameters, який містить шлях до БД, ім'я користувача і його пароль;

– extractScheme () – витяг метаінформації з БД і створення примірника класу Database;

– buildSequence () – формування ПОС БД і створення екземпляра класу Sequence;

– saveSequence (String path) – збереження ПОС БД. Вхідний параметр path – шлях до конфігураційного файлу. Даний метод викликає метод saveToXML (String path) класу Sequence;

– loadSequence (String path) – аналіз конфігураційного файлу і завантаження сформованої послідовності з файлу в пам'ять. Даний метод викликає метод loadFromXML (String path) класу Sequence.

Клас ExtDatabase – клас, який зберігає всю необхідну інформацію про БД для формування ПОС ВБД. Крім полів з успадкованого класу model.Database додані наступні основні поля класу:

– extTables – множина таблиць схеми БД. Як таблиці– тут маються на увазі екземпляри класу ExtTable;

– independentTables – список незалежних таблиць. Список– формується під час попереднього етапу побудови ПОС ВБД.

Основним методом класу є ExtDatabase () – конструктор, створення екземпляра класу.

Клас ExtTable – клас, який успадковує клас model.Table з моделі, представленої на рис. 3.6. Крім полів успадкованого класу додані наступні основні поля:

– count – поле лічильник (застосування даного поля докладно пояснюється в описі алгоритму побудови ПОС БД);

– loop – поле логічного типу, яке вказує на наявність «петлі» в таблиці.

Основні методи класу:

– calculateCount () – обчислює для таблиці значення поля count;

– hasLoop () – визначає наявність «петлі» в таблиці.– Визначає значення поля loop;

– getRelatedTables () – повертає список залежних– таблиць для даної таблиці.

Клас Sequence – клас для зберігання, побудови і завантаження з XMLфайла ПОС БД.

– Sequence () – конструктор, створення екземпляра класу Sequence;

– saveToXML (String path) – збереження послідовності в XML-файл;

– loadFromXML (String path) – аналіз конфігураційного файлу і завантаження сформованої послідовності з файлу в пам’ять.

3.2 Пакет data_migration

Діаграма основних класів пакета data_migration приведена на рисунку 3.4.

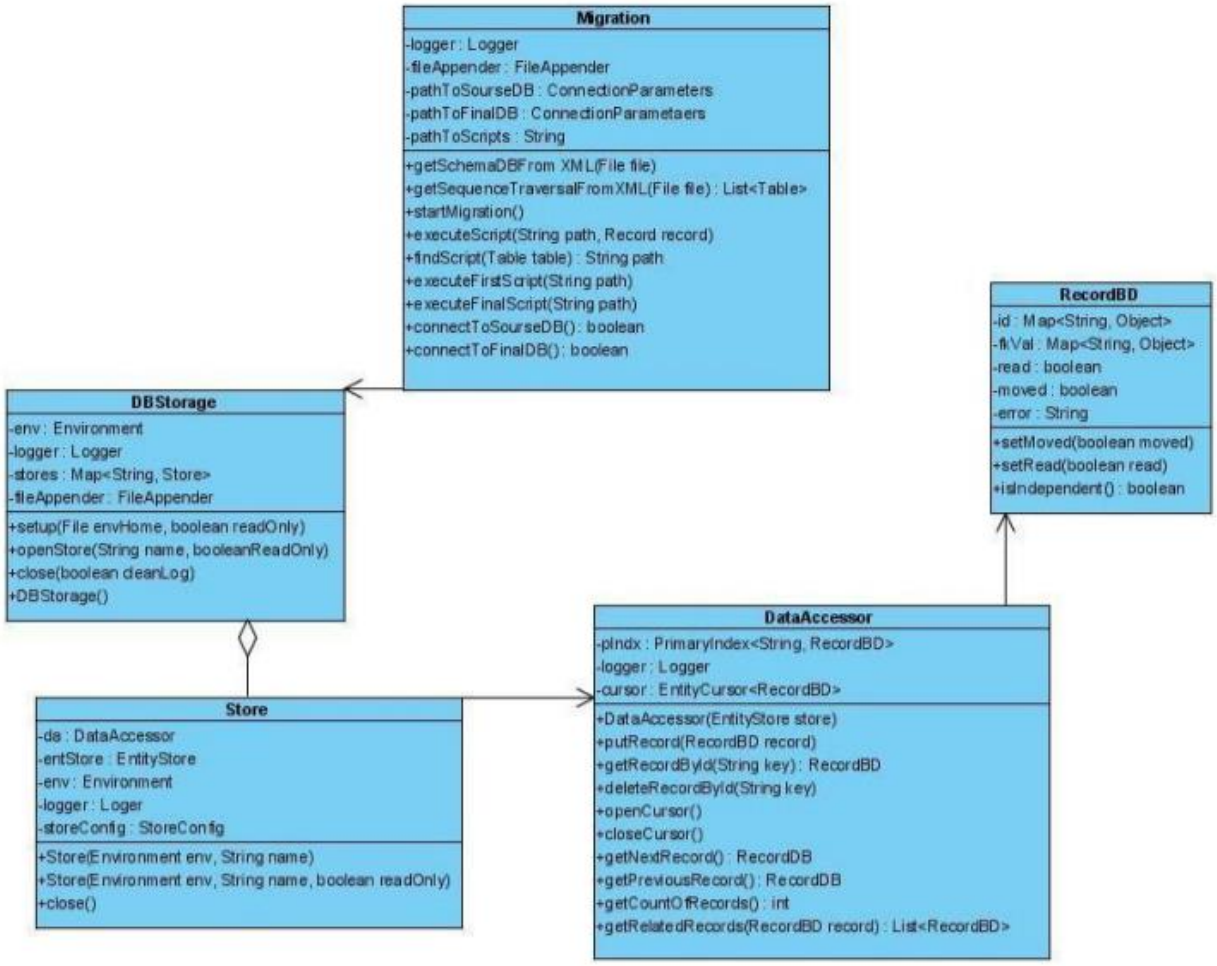


Рисунок 3.4 – Діаграма основних класів пакета data_migration

Діаграму класів можна розділити на дві логічні частини:

– організація СД для пошуку залежних записів, а також для зберігання інформації про те, чи був перенесений запис;

– клас, який реалізує алгоритм перенесення даних.

До першої частини входять такі класи:

– DBStorage – СД. В даному класі задаються загальні– параметри доступу і зберігання даних (налаштування оточення). А також метод закриття оточення і звільнення пам’яті;

– Store – окреме СД для таблиці;

– DataAccessor – визначає правила доступу до даних, що зберігаються в Store, а також надає різні методи роботи з даними, збереженими в Store;

– RecordDB – клас, який визначає структуру зберігання записів– таблиці в нотації Berkeley DB.

Клас, який реалізує алгоритм перенесення даних – Migration.

Реалізований в середовищі NetBeans 6.8 на мові Java. При реалізації були використані наступні бібліотеки:

– mysql-connector-Java-5.1.12-bin,

– log4j-1.2.16;

– je-4.0.92.

Для роботи зі схемою БД використовується модель model-metadata-database-2.0.0. Для побудови ПОС ВБД використовується описана вище бібліотека sequence-traversal-database.

Основні файли реалізації:

– Migration.Java;

– DBStorage.Java;

– Store.Java;

– DataAccessor.Java;

– RecordBD.Java.

Основний клас Migration – даний клас реалізує алгоритм перенесення даних.

Основні поля класу:

– logger – використовується для запису помилок, що виникають в процесі роботи, в лог-файл;

– fileAppender – файл, в який записуються помилки;

– pathToSourseDB – містить необхідну інформацію для встановлення з’єднання з початковою БД, такі як шлях до БД, ім’я користувача і його пароль і т.д.;

– pathToFinalDB – містить необхідну інформацію для встановлення з’єднання з ЦБД;

– pathToScripts – шлях до директорії, де зберігаються сценарії

перетворення.

Основні методи:

– getSchemaDBFrom XML (File file) – отримання метаінформації про схему ВБД з конфігураційного файлу;

– getSequenceTraversalFromXML (File) – відновлення ПОС ВБД з конфігураційного файлу;

– startMigration () – основний метод МД;

– executeScript (String path, Record record) – виконує сценарій перетворення для конкретного запису record. Path – шлях до директорії, де зберігається сценарій;

– executeFirstScript (String path) – виконання сценарію, який відключає тригери і додаткові обмеження до початку механізму МД;

– executeFinalScript (String path) – виконання сценарію, що− відновлює раніше відключені тригери і обмеження;

– connectToSourseD connectToFinalDB () – встановлення з’єднання з ЦБД. В () – встановлення з’єднання з початковою БД;

– connectToFinalDB () – встановлення з’єднання з ЦБД.

Клас DBStorage – СД. В даному класі задаються загальні параметри доступу і зберігання даних (налаштування оточення).

Основні методи:

– setup (File envHome, boolean readOnly) – створення оточення для зберігання даних, а також налаштування доступу і управління;

– openStore (String name, booleanReadOnly) – створення в оточенні СД для конкретної таблиці, також налаштування доступу до СД;

– close (boolean cleanLog) – закриває оточення, видаляючи дані з диска зберігання. Також підтримується можливість очищення логфайлів.

Клас Store – представляє окреме СД для таблиці.

Основні поля класу:

– da – екземпляр класу DataAccessor. Визначає правила доступу до записів конкретної таблиці;

– env – посилання на оточення, в якому знаходиться дана таблиця.

Основні методи:

– Store (Environment env, String name) – конструктор, створення екземпляра класу з заданням необхідних параметрів;

– close () – закриває СД і очищає дані і пам’ять.

Клас DataAccessor – визначає правила доступу до даних, що зберігаються в Store, а також надає різні методи роботи з даними, збереженими в Store. Основні поля класу:

– pIndx – визначає метод доступу до даних, що зберігаються в Store, шляхом задання первинного ключа (визначає ставлення ключ-значення);

– cursor – курсор для послідовного читання даних, що зберігаються в Store.

Основні методи:

– putRecord (RecordBD record) – додавання запису в таблицю (екземпляр класу Store) згідно з правилом, що задане за допомогою pIndx;

– getRecordById (String key) – повертає запис за ідентифікатором;

– deleteRecordById (String key) – видалення запису зі СД за ідентифікатором;

– openCursor () – створення курсора для послідовного читання даних, збережених в Store. При неможливості створення курсора інформація зберігається в лог-файлі;

– closeCursor () – закриття курсору. При виникненні помилок інформація зберігається в лог-файлі;

– getNextRecord () – повертає наступний запис щодо становища курсору;

– getPreviousRecord () –повертає попередній запис щодо положення курсора;

– getCountOfRecords () – повертає кількість записів, збережених в Store;

– getRelatedRecords (RecordBD record) – повертає, в загальному випадку, список записів, які залежать від поточного запису (record). Якщо залежних записів немає, то метод повертає порожній список.

Клас RecordDB – клас, який визначає структуру зберігання записів таблиці в нотації Berkeley DB.

Основні поля класу:

– id – ідентифікатор запису в таблиці (первинний ключ);

– read – покажчик на те, переглянутий запис чи ні;

– moved – покажчик на те, перенесений запис чи ні;

– error – код помилки;

– fkVal – значення зовнішніх ключів. Використовуються для визначення залежностей між записами.

Основні методи:

– setMoved (boolean moved) – дозволяє відзначити запис як перенесений;

– setRead (boolean read) – дозволяє відзначити запис як переглянутий;

– isIndependent () – дозволяє перевірити чи є запис залежним чи ні. Визначає можливість перенесення запису.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вимоги ергономіки до організації робочого місця оператора ПК

Робоче місце — це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників [17].

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпуса, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні положення тіла. Проведені дослідження показують, що при раціональній організації робочих місць продуктивність праці зростає на 15–25% [18].

Організація робочого місця користувача ПК має відповідати ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги», ДСан ПіН 3.3.2.007-98, характеру та особливостям трудової діяльності.

Площа одного робочого місця користувача ПК повинна складати не менше 6 м2, а об’єм – не менше 20 м3. Конструкція робочого місця користувача ПК повинна відповідати сучасним вимогам ергономіки, характеру виконуваної роботи і забезпечити оптимальне розміщення на робочій поверхні документів та обладнання ПК (монітора, системного блоку, клавіатури, мишки та інших периферійних пристроїв. Монітор на робочому місці встановлюється так, щоб верхній край екрана знаходився на рівні очей.

Розташування монітора ПК має забезпечувати:

– безпечність роботи в цілому;

– зручність та ефективність зорової роботи з екраном в вертикальній площині під кутом 300 від лінії зору, площина екрана при цьому має бути

перпендикулярною нормальній лінії зору користувача.

Клавіатура розміщується на поверхні столу або висувній полиці на відстані 100-300мм від краю, ближчого до користувача. Кут нахилу клавіатури має бути в межах 5-150. Поверхня клавіатури повинна бути матовою з коефіцієнтом відбиття 0,4. клавіші клавіатури мають бути зручними в роботі і м’якими при натисканні (хід всіх клавіш має бути одинаковим з мінімальним опором натискання 0,25Н та максимальним – не більше 1,5Н) [19].

При розміщенні робочих місць з ПК слід дотримуватися вимог, зазначених в ДНАОП 0.00-1.31-99:

– робочі місця розміщуються на відстані не менше 1м від стін з світловими прорізями; – відстань між бічними поверхнями моніторів ПК має бути не менше 1,2м; – відстань між тильною поверхнею монітора одного ПК та екраном монітора іншого ПК має бути не меншою 2,5м.

Вимоги двох останніх пунктів враховуються також при розміщенні робочих місць з ПК в суміжних приміщеннях з урахуванням конструктивних особливостей стін та перегородок.

Загальні принципи організації робочого місця:

– на робочому місці не повинно бути нічого зайвого. Усі необхідні для роботи предмети мають бути поряд із працівником, але не заважати йому;

– ті предмети, якими користуються частіше, розташовуються ближче, ніж ті предмети, якими користуються рідше;

– предмети, які беруть лівою рукою, повинні бути зліва, а ті предмети, які беруть правою рукою – справа;

– якщо використовують обидві руки, то місце розташування пристосувань вибирається з урахуванням зручності захоплювання його двома руками;

– робоче місце не повинно бути захаращене;

– організація робочого місця повинна забезпечувати необхідну оглядовість.

Статичні напруження працівника в процесі праці пов'язані з підтриманням у нерухомому стані предметів і знарядь праці, а також підтриманням робочої пози.

Робоча поза – це основне положення працівника у просторі: зручна робоча поза має забезпечувати стійкість положення корпуса, ніг, рук, голови працівника під час роботи, мінімальні затрати енергії та максимальну результативність праці. Неправильна сидяча поза може викликати застій крові в ногах, а якщо виконується великий обсяг роботи для пальців рук – запалення суглобів.

Організація робочого місця користувача комп'ютера повинна забезпечувати відповідність усіх елементів робочого місця та їх взаємного розташування ергономічним вимогам (рисунок 4.1).

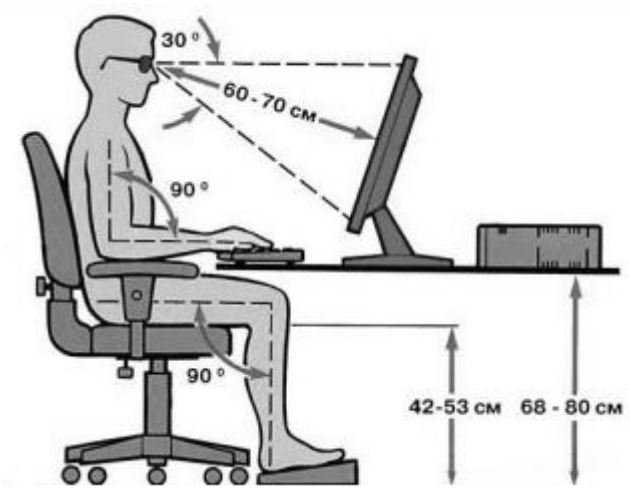


Рисунок 4.1 – Робоче місце і робоча поза користувача ПК

Найпоширенішими у процесі праці є пози сидячи і стоячи. Проектуючи робоче місце, потрібно враховувати, що при виконанні роботи з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи.

Робоча поза стоячи втомлює людину більше, ніж сидяча. Вона вимагає на 10% більше енергії, спричиняє підвищення артеріального і венозного тиску крові, розширення вен на ногах, пошкодження ступень, викривлення хребта [19].

4.2 Заходи захисту від випромінювань оптичного діапазону

До випромінювання оптичного діапазону відносяться інфрачервоні й ультрафіолетові хвилі, видиме світло, лазерне випромінювання.

По фізичній природі інфрачервоні промені мають хвильові (довжина хвилі

0,78-540 мкм) і квантові властивості. Генератором випромінювання є будь-яке тіло, температура якого вище абсолютного нуля. За законом Стефана-Больцмана інтегральна густина випромінювання, Вт/м^2, абсолютно чорного тіла пропорційна четвертому ступеню його абсолютної температури [17].

З підвищенням температури тіла змінюється спектральний склад його випромінювання. Чим вища температура тіла, тим коротша довжина хвилі, максимального випромінювання.

Інфрачервона енергія, яка потрапляє на тіло людини, діє передусім на незахищені його частини (лице, руки, шию, груди), причому конвективне тепло впливає на зовнішній шкіряний покрив, тоді як інфрачервоне випромінювання може проникнути на деяку глибину в тканину. При довготривалому перебуванні людини в зоні інфрачервоного випромінювання, як і при систематичній високій температурі наступає різке порушення теплового балансу в організмі. Для вимірювання густини потоку випромінювання на робочому місці застосовують актинометр – прилад, який дозволяє вимірювати густину потоку інфрачервоного випромінювання у діапазоні від 0 до 14кВт/м^2 . Основні види захисту від інфрачервоного випромінювання–захист часом, захист віддалю, усунення джерела тепловиділення, теплоізоляція, охолодження гарячої поверхні, забезпечення тепловіддачі тіла людини та індивідуальні засоби захисту. Потужність випромінювання можна знизити за рахунок конструкторських і технологічних рішень (зміною нагрівання виробів у нагрівальних пічках індукційним нагріванням та ін.) і за рахунок покриття поверхні, яка нагрівається, тепло ізолювальним матеріалом. Для захисту очей застосовують світлофільтри зі спеціального жовто-зеленого або синього скла.

Ультрафіолетове випромінювання змінює склад виробничої атмосфери. Утворюється озон, оксиди азоту і пероксид водню. Короткохвильове випромінювання іонізує повітря, утворює в атмосфері ядра конденсації, які зменшують освітленість робочих місць і призводять до утворення туманів.

Основні засоби захисту. Першочергові заходи – це конструкторські і технологічні рішення, які виключають генерацію або понижують інтенсивність випромінювання. Спеціальні засоби захисту (екранування джерел

випромінювання, фарбування стін у світлі кольори) попереджують розповсюдження і знижують інтенсивність цих випромінювань у виробничих приміщеннях. Очі захищають окулярами або щитками зі склом – світлофільтром. Для захисту шкіри використовують мазі з речовинами – світлофільтрами для цих променів (салол, саліцилово-метиловий ефір та ін.), а також спецодяг з бавовняних тканин і грубововняного сукна. Руки захищають рукавицями [18].

Діапазон довжин хвиль які випромінюють оптичні квантові генератори (ОКГ) – лазери, охоплює видимий спектр і розповсюджується в інфрачервоній і ультрафіолетовій областях.

Найбільш чутливими до дії випромінювання ОКГ є очі. Випромінювання викликають опіки і пошкодження сітківки ока, це може призвести до сліпоти. Небезпечне не тільки пряме випромінювання, але й відбите від стін, обладнання.

Існують спеціальні норми, до яких ввійшли організаційні та інженерно-технічні заходи, які можуть забезпечити зменшення густини потоків енергії (потужності) на робочих місцях до величин, значно менших від допустимих. ОКГ розміщують в окремих або відгороджених приміщеннях. Саме приміщення і обладнання не повинні мати дзеркальної поверхні. Стіни, стелі, обладнання й інші предмети фарбують матовою фарбою з малою сорбційною здатністю. Приміщення повинно мати високу освітленість, а також припливно-витяжну вентиляцію. При розміщенні в одному приміщенні декількох ОКГ їх огороджують ширмами, шторами або екранами, що не пропускають випромінювання. Надійним захистом від випадкового попадання випромінювання на людину є світловод, який екранує промінь на усьому шляху його дії (від ОКГ до мішені) [18].

ВИСНОВКИ

Причин, з яких організації починають проекти з МД, може бути досить багато: від оновлень додатків і впровадження нових корпоративних систем до повномасштабної реструктуризації внаслідок злиття компаній.

На даний момент не існує єдиного підходу до вирішення проблеми МД. Фірма, яка розробляє ПЗ, не завжди підтримує можливість збереження накопичених даних при переході на більш пізню версію програмного продукту. Розробникам доводиться аналізувати всі зміни та кожен раз створювати нові процедури перенесення даних для конкретного випадку зміни схеми БД. Клієнти, які вирішили перейти на нову версію ПЗ зі зміненою схемою БД, часто стикаються з проблемою перенесення даних, накопичених в процесі роботи зі старою версією ПЗ. Для таких випадків необхідно спеціальне ПЗ для проведення якісної і коректної МД.

Розроблений програмний засіб гарантує перегляд і перенесення всіх записів з ВБД, цілісність даних, а також, що перенесення записів буде здійснюватися з урахуванням всіх залежностей. Запис може бути перенесений тільки в тому випадку, якщо він є незалежним, або записи, від яких залежать дані, вже перенесені в ЦБД.

Відповідно до поставленої мети та завдань було досягнуто наступних результатів:

– проведено грунтовні теоретичний аналіз змін схем БД;

– результатом проведених досліджень топологій схем БД став створений алгоритм виконання послідовності обходу схеми БД для МД;

– запропоновано алгоритм МД, який базується на порядковому перенесенні даних;

– програмно реалізовані алгоритми з використанням мови Java;

– описано програмну реалізацію додатку для здійснення МД.

У плани подальшої роботи входять:

– автоматичний аналіз змін схем БД;

– створення графічного інтерфейсу користувача (GUI);

– розширення списку підтримуваних СУБД;

– реалізація алгоритмів для випадку виникнення орієнтованих циклів за таблицями і циклів за даними.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Data migration [Електронний ресурс] // Data Integration Info. - Режим доступу до ресурсу: http://www.dataintegration.info/data_migration - (дата звертання: 01.11.2018).

2. Перевозчикова О. Інформаційні системи і структури даних. – К.: Києво-Могилянська академія, 2007. – 288 с.

3. Best practices for data migration [Електронний ресурс] // IBM Global Services. - Режим доступу до ресурсу: http://www-935.ibm.com/services/us/gts/pdf/softek-best-practices-datamigration.pdf - (дата звертання: 07.10.2018).

4. Chester B. Data migration 101 // AIIM E-DOC. - 2006. - Vol. 20, Issue 1. - P. 10.

5. Data migration best practices [Електронний ресурс] // NetApp. / Режим доступу до ресурсу: http://partners.netapp.com/go/techontap/ NGS_migration.pdf (reference date: 26.09.2019).

6. Fishman D. Database migration: the key to unlocking your business assets [Електронний ресурс] // Narayana Vyas Kondreddi's website. - Режим доступу до ресурсу: http://vyaskn.tripod.com/database_migration_white_paper.htm - (дата звертання: 28.09.2018).

7. McLaughlin B. XML and Java technology: XML persistence in three flavors [Електронний ресурс] / B. McLaughlin – Режим доступу до ресурсу: http:// www.ibm.com/developerworks/xml/library/x-xjavaforum5/x-xjavaforum5-pdf.pdf - (дата звертання: 15.09.2018).

8. Чаушник А. В. Обработка, анализ и преобразование данных при миграции ПО // Научный вестник МИРЭА. - 2008. - № 1(4). - С. 72-75.

9. Matthes F. Testing & quality assurance in data migration projects / F. Matthes, C. Schulz, K. Haller // Proceedings of 27th IEEE International Conference on Software Maintenance (ICSM'11). Williamsburg, VA, USA, September 25-30, 2011. - IEEE, 2011. - P. 438-447.

10. Extracting database metadata [Електронний ресурс] / Режим доступу до ресурсу: https://dzone.com/articles/jdbc-tutorial-extracting-database-metadata-via-jdb - (дата звертання: 05.10.2018).

11. Гасфилд Д. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология / Д. Гасфилд; пер. с англ. И. В. Романовский. - Спб.: Невский Диалект; БХВ-Петербург, 2003. - 654 с.

12. Эккель Б. Философия Java /Б. Эккель; пер. с англ. Е. Матвеева – СПб.: Питер, 2009 – 640 с. –4-е изд.

13. Эмблер С. В. Рефакторинг баз данных: эволюционное проектирование / С. Эмблер, П. Садаладж ; пер. с англ. К. А. Птицына. - М.:Вильямс, 2007. – 672 с.

14. Barclay K. Groovy programming: an introduction for Java developers/ K. Barclay, J. Savage – Morgan Kaufmann Publishers, Inc., 2007. – 496 p.

15. Java xml/xml and java a powerful combination [Електронний ресурс] / Режим доступу до ресурсу: https://www.javaworld.com/article/2076453/java-xml/xml-and-java--a-powerful-combination.html - (дата звертання: 05.10.2018).

16. Tutorial: XML and Java for Scientists/Engineershttps [Електронний ресурс] / Режим доступу до ресурсу: http://isr.umd.edu/~austin/ence489c.d/xml.html – (reference date 05.10.2018).

17. Толок А.О. Крюковська О.А. Безпека життєдіяльності: Навч. посібник. – 2011. – 215 с.

18. Яремко З. М. Безпека життєдіяльності: Навч. посіб. — Львів., 2005. – 301 с.

19. Желібо Є. П. Заверуха Н.М., Зацарний В.В. Безпека життєдіяльності. Навчальний посібник. / Є. Желібо Є.П., Н.М. Заверуха П., В.В. Зацарний. – К.; Каравела, 2004. -328 с.