НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
Кафедра Автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедрою
_____ Ролік О. І.
(підпис) (ініціали, прізвище)
«__»_____ 2019 р.

Магістерська дисертація

зі спеціальності 126 Інформаційні системи та технології

на тему: «Автоматизована система обробки даних медичних досліджень»

Виконав: студент 6-го курсу, групи ІА-82мп
(шифр групи)

Хололович Катерина Вікторівна
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент каф. АУТС, Букасов М.М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н., доцент каф. ТК, Ліхоузова Т.А.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ADaM – Analysis Data Model
ALS – Architecture Load Specification
CDISC – Clinical Data Interchange Standards Consortium
CRF – Case Report Form
CRO – Clinical Research Organization
DM – Data Management
QC – Quality Control
SAP – Statistical Analysis Plan
SAS – Statistical Analysis System
SAS EG – SAS Enterprise Guide
SDTM – Standard Data Tabulation Model
SDTM IG – SDTM Implementation Guide
TLF – Tables, Listings, Figures
БД – База даних

ВСТУП

За останні десятиліття проведення медичних досліджень зробило величезний крок вперед. Світовою спільнотою було розроблено десятки стандартів, що регулюють процес перевірки якості та ефективності ліків. Медичні дослідження можуть насправді зробити життя кращим, адже кожної хвилини проводиться розробка або аналіз препаратів, які дозволять боротися із до сих пір невиліковними хворобами.

Важливою умовою затвердження препарату є позитивні результати проведення дослідження. Лише 10% усіх препаратів, над якими проводяться медичні дослідження, стають затвердженими препаратами. Однією з причин є низька якість обробки даних медичних досліджень.

Медичні дослідження можуть обходитись фармакологічним компаніям у мільярди доларів, і штраф, який стягується з фармакологічної компанії у випадку недостовірності даних або наявності помилок у даних, може обійтись майже в ту саму вартість, що й проведення дослідження. Тому виявлення таких помилок у даних є однією із головних задач із якою стикаються спеціалісти IT у медичних дослідженнях. Ця задача не є тривіальною, адже, по-перше, обробка даних дуже стандартизована, неможливо правильно обробити дані без розуміння стандартів та конкретного протоколу дослідження. По-друге, у сирих даних можуть бути мільйони записів, сотні змінних, і зі збільшенням даних, пошук помилок стає дедалі важчим.

У CRO (Clinical Research Organization) зазвичай є спеціальний відділ Clinical Data Management (CDM), який відповідає за «чистоту» даних. Проте перевірка даних часто децентралізована, включає в себе деякі тривіальні перевірки на етапі збору даних у БД, та виправлення помилок безпосередньо у сирих даних за вимогою статистичних програмістів. Децентралізований пошук помилок є менш ефективним та створює загрозу невчасного виявлення помилок чи не виявлення помилок взагалі, що може зробити результат дослідження недійсними.

Першим етапом обробки даних медичних досліджень є створення SDTM (Standard Data Tabulation Model) датасетів – результатів первинної обробки даних. Цей процес є майже однаковим для різних досліджень, тому доцільно буде його автоматизувати, що також дозволить зменшити використання ресурсів та дозволить зосередитися на аналітичній оброці даних.

Отже, постає задача створення автоматизованої системи обробки даних, яка б включала в себе модуль пошуку помилок, виявлення суперечностей у сирих даних, та підсистеми створення SDTM датасетів. Ця система дозволить впевнитися у коректності даних, допоможе запобігти неправильній інтерпретації даних, підвищить якість та релевантність проведеного дослідження а також значно підвищить ефективність обробки даних.

Метою даної роботи є спрощення обробки даних медичних досліджень, створення датасетів та звітів згідно зі стандартами CDISC (Clinical Data Interchange Standards Consortium).

Структура дослідження: магістерська дисертація складається із вступу, 7 розділів, висновку та 8 додатків.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості сфери медичних досліджень

Медичне дослідження – наукове дослідження, що проводиться з метою оцінки ефективності та безпеки нового лікарського препарату або розширення показань до застосування вже відомого лікарського препарату. Медичні дослідження можуть також вивчати ефективність і безпеку нових методів лікування і діагностики.

Медичні дослідження в усьому світі є невід'ємним етапом розробки препаратів, який передує його реєстрації і широкому медичному застосуванню. В ході медичних досліджень новий препарат вивчається для отримання даних про його ефективність і безпеку. На підставі цих даних уповноважений орган охорони здоров'я приймає рішення про реєстрацію препарату або відмову в реєстрації. Препарат, який не пройшов медичні дослідження, не може бути зареєстрований та виведений на ринок [1].

Мета, завдання, дизайн, методологія, статистичні аспекти та організація дослідження описуються в документі, який називається протоколом медичного дослідження.

Протокол – це один із найголовніших документів у дослідженні. Лікарі-дослідники зобов'язані суворо дотримуватися протоколу – це є гарантією того, що дослідження в усіх центрах проводиться однаково. Недотримання протоколу може привести до виключення дослідника або дослідного центру з програми досліджень [1].

Результати, отримані у обмеженій вибірці пацієнтів, які брали участь у медичних дослідженнях, можна перенести на всю популяцію хворих завдяки спеціальним статистичним методам. До планування медичного дослідження завжди залучаються фахівці в області біомедичної статистики. Вони розробляють методики збору і аналізу інформації, що дозволяють зробити результати дослідження репрезентативними [2].

В ході дослідження лікарі-дослідники набирають пацієнтів відповідно до заздалегідь визначених характеристик (критеріїв відбору) і збирають інформацію про їх здоров'я під час участі в дослідженні (результати лабораторних аналізів, інформація про концентрацію препарату в крові, про наявність чи відсутність змін в стані здоров'я та ін.). Потім дослідники спрямовують зібрану інформацію в центр обробки даних, де їх опрацьовують згідно до обраних моделей даних.

Аналіз даних – заключний етап медичного дослідження, на якому отримують відповіді на поставлені запитання та підтверджується або не підтверджується справедливість статистичних гіпотез.

Статистика відіграє дуже важливу роль у будь-якому медичному випробуванні з точки зору контролю та мінімізації упереджень, змішаних факторів та вимірювання випадкових помилок. Досвід статистичних методів є фундаментальним для розуміння рандомізованих методів і результатів дослідження.

Статистичні методи дають уявлення про мінливості у реакціях пацієнтів на лікування. Використання статистичних даних у медичних випробуваннях дозволяє медичному досліднику формувати розумні та точні висновки зі зібраної інформації та обґрунтовані рішення при наявності невизначеності. Статистика допомагає запобігти помилок і упереджень у медичних дослідженнях.

Абсолютним лідером у сфері медичних досліджень та статистичного аналізу даних є система SAS. Вона широко використовується в аналізі даних медичних випробувань в фармацевтичних, біотехнологічних і медичних дослідницьких компаній.

Програмісти SAS відіграють важливу роль у медичних випробуваннях та аналізі даних. Вони впроваджують методи аналізу на зібраних даних і створюють статистичні звіти у вигляді таблиць, лістингів та графіків (TLF – Tables, Listings, Figures).

1.2 Процес обробки даних медичних досліджень

Проведення медичних досліджень є дуже регульованим та стандартизованим процесом. Команда CDISC (Clinical Data Interchange Standards Consortium) розробила ряд документів, що повністю описують вимоги до проведення медичних досліджень [3]. Такі організації як FDA (Food and Drug Administration, США) та PMDA (Pharmaceuticals and Medical Devices Agency, Японія) погоджуються приймати результати досліджень тільки якщо вони відповідають вимогам CDISC [4].

У загальному можна виділити 4 етапи обробки даних: це планування моделі досліджень, збір сирих даних, стандартизація або організація даних та аналіз даних [5] (рис. 1.1).

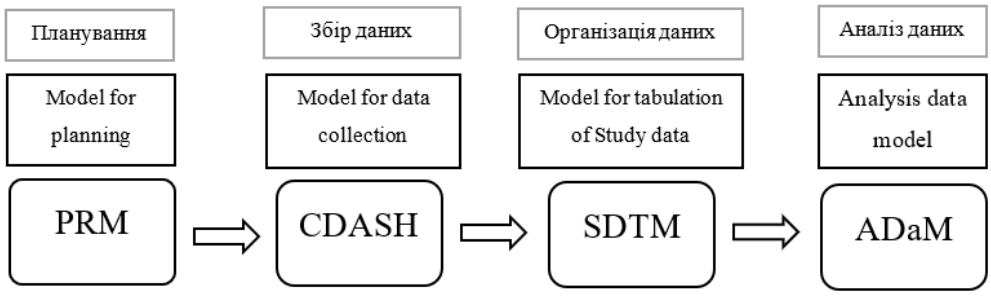


Рисунок 1.1 – Загальний процес обробки даних медичних досліджень

На етапі планування (PRM, Protocol Representation Model) створюється протокол досліджень – головний документ, який визначає весь план досліджень, первинні та вторинні цілі дослідження, кількість суб’єктів, дизайн дослідження, тривалість дослідження і т.і.

Слід зазначити, що на всіх етапах використовуються стандарти CDISC. Але на етапі планування єдиний документ що має використовуватися – Controlled Terminology (CT) – документ, який включає в себе допустиму термінологію для змінних.

На етапі збору даних визначається модель збору даних.

CDASH (Clinical Data Acquisition Standards Harmonization) встановлює стандартний спосіб для збору даних аналогічним чином у різних дослідженнях, щоб формати та структури збору даних забезпечували чітке відстеження переведення даних у модель табуляції даних досліджень (SDTM) [6].

Основним документом, який розробляться на цьому етапі є CRF – Case Report Form – документ який буде використовуватися безпосередньо на місцях збору даних – клініках, лабораторіях, та включає в себе усі змінні, що будуть збиратися у сирих даних та які є важливими для дослідження. На цьому етапі задіяно як правило 3 стандарти CDISC (рис. 1.2)

Важливо зазначити що на етапі планування та збору даних відділ SAS програмістів не задіяний.

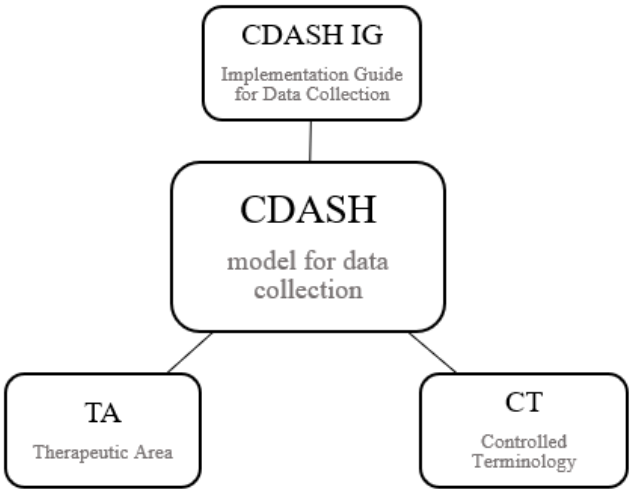


Рисунок 1.2 – Стандарти CDISC на етапі збору даних

На етапі організації даних починається програмування SDTM (Standard Data Tabulation Model) датасетів – набору датасетів, що є достатнім для включення усіх сирих даних, дає цілісне уявлення про хід дослідження та має жорстко визначену командою CDISC структурою, відхилення від якої неприпустимо – ні FDA, ні PMDA не приймає результатів які не сумісні зі CDISC стандартами. SDTM є стандартом для організації та

форматування даних для раціоналізації процесів у зборі, управлінні, аналізі та звітності [7].

Перед початком програмування створюються специфікації – обов'язковий документ, що представляє інструкції щодо мапінгу змінних, перелік CT, метадані змінних і т.і.

На цьому етапі застосовуються цілий ряд CDISC стандартів, найголовнішим з яких є SDTM IG (Implementation Guide), що містить– опис усіх дата сетів, правила застосування моделей, приклади використання (рис. 1.3). Зазвичай SDTM датасети мають мінімальний рівень обробки або трансформації даних – кількість інформації в дата сетах приблизно така сама як і в сирих даних [7].

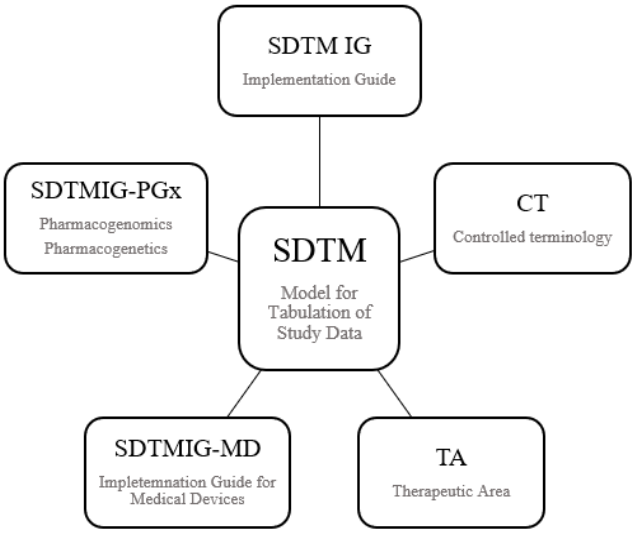


Рисунок 1.3 – Стандарти CDISC на етапі організації даних

Важливим кроком на цьому етапі є валідація SDTM датасетів – перевірка на сумісність з визначеною версією SDTM IG та CT. Зазвичай для цього використовується OpenCDISC Pinnacle 21 валідатор, який є лідером в галузі програмного забезпечення та послуг для керування дотримання CDISC, якості клінічних даних та готовності до подання FDA [8].

Останній етап – аналіз даних. На цьому етапі створюється такий документ як SAP (Statistical Analysis Plan), який визначає статистичні методи оцінювання цілей дослідження, визначає які дані у якій формі є доцільними для аналізу. Використання CDISC стандартів на цьому етапі є менш впливовим (рис. 1.4), адже аналіз більше залежить від конкретного дослідження, і його важко підігнати під певний стандарт. На цьому етапі створюється пакет ADAM датасетів, і, так само як і для SDTM, формується специфікація мапінгу змінних.

ADaM (Analysis Data Model) визначає стандарти для набору даних і метаданих, які підтримують ефективне генерування, реплікацію та аналіз статистичних аналізів медичних випробувань, а також можливість відстеження результатів аналізу в порівнянні з даними, представленими в SDTM [9].

У ADaM датасетах, порівняно із SDTM, проводиться набагато більше маніпуляцій з даними – створюються нові змінні, записи які містять у собі деяку статистичну обробку даних.

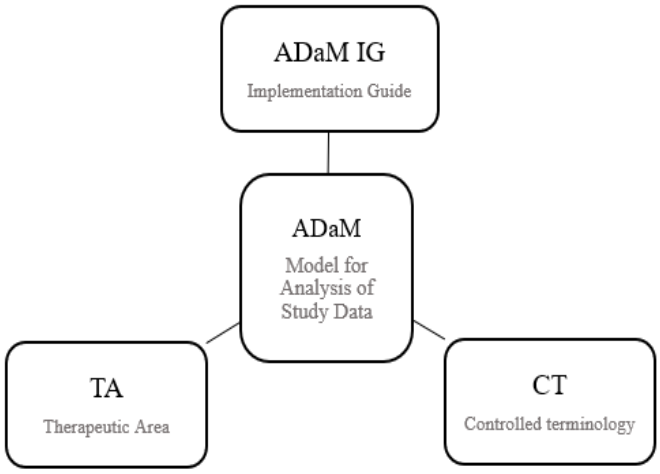


Рисунок 1.4 – Стандарти CDISC на етапі аналізу даних

Також на цьому етапі розробляється макет TLF (Tables, Listings and Figures) – статистичних звітів, та відповідно програмується узгоджений набір TLF. Саме вони є

головними репрезентативними результатами обробки даних дослідження, на основі яких буде прийматися рішення щодо ефективності препарату.

Також на етапах створення SDTM та ADaM датасетів, створюється файл Define.xml, що містить у собі список датасетів, що були створенні, усіх змінних, інформацію про атрибути змінних та правила мапінгу.

Етапи процесу обробки даних медичних досліджень представлені на рис. 1.5.

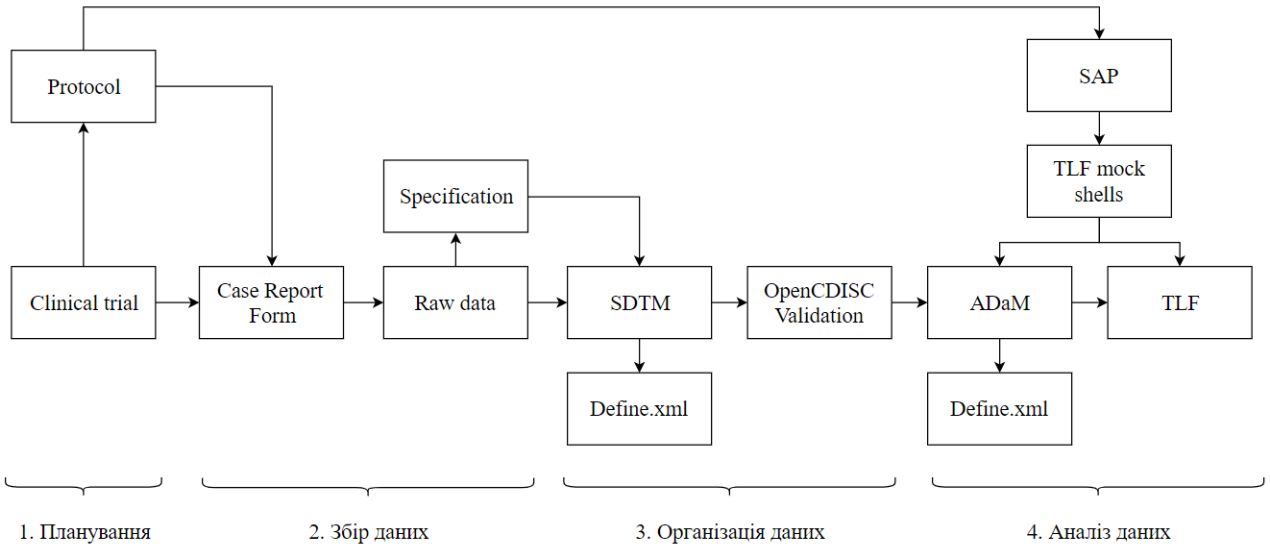


Рисунок 1.5 – Етапи обробки даних медичних досліджень

1.3 Етапи організації та аналізу даних

У цьому підрозділі розглянуто більш детально третій та четвертий етапи процесу обробки даних – етапи організації та аналізу даних, на яких програмуються датасети на основі сирих даних. Слід зазначити, що методологія розробки програм досить специфічна, що пов’язано з тим, що процес обробки даних дуже стандартизований.

Загальний процес обробки даних можна представити як на рис. 1.6.

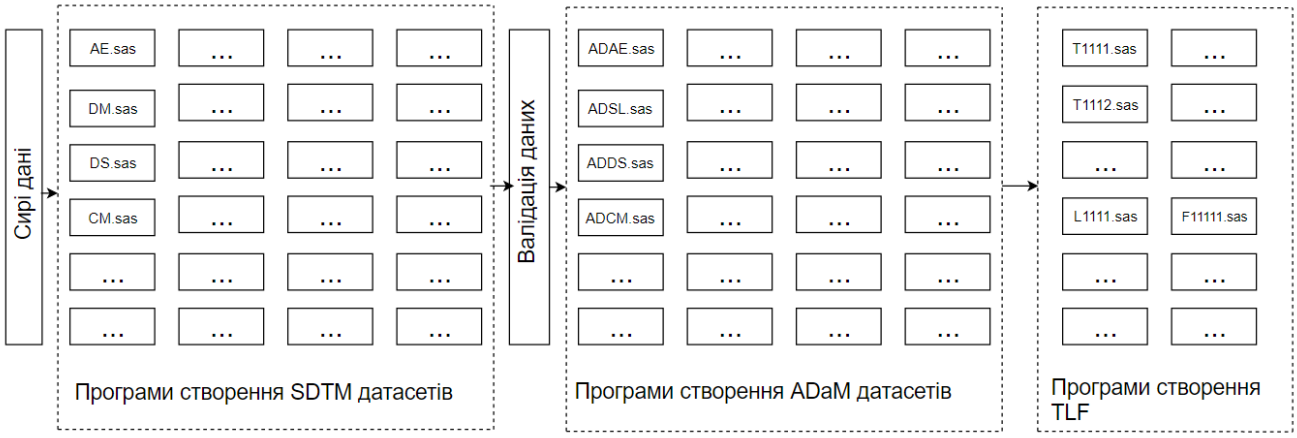


Рисунок 1.6 – Процес обробки даних у медичних дослідженнях

Сирі дані надходять до програмістів у вигляді SAS7BDAT (формат який підтримується SAS) файлів або датасетів, які відповідають тій структурі яка була визначена у CRF (Case Report Form). Сирі дані представляють собою окремі модулі, які можуть бути використані у декількох програмах. Кожен такий датасет містить у собі певну інформацію, що відповідає певній тематиці.

Процес програмування датасетів та звітів усіх типів є модульним – кожен датасет створює одна програма SAS. На одному проекті може бути задіяно десятки програмістів, та програмування датасетів є певним чином розподіленим між ними.

Важливою особливістю програмування датасетів є підхід паралельного, або подвійного програмування (double programming), коли створюється програма так званого продакшена (яка згодом і доставляється фармакомпанії), та валідатора (QC – Quality Control). На стороні QC зазвичай працює більш досвідчений програміст, який перевіряє правильність створеного датасету (також у форматі SAS7BDAT).

Процес створення SDTM датасетів є приблизно однаковим для різних досліджень, на відміну від етапу аналізу даних, адже по суті на етапі організації даних виконується первинна обробка даних, яка не включає в себе ніяку статистичну обробку даних, а полягає у форматуванні даних згідно до стандартів. Тому доцільно автоматизувати цей етап.

Також, як можна побачити, помилки в сирих даних можуть бути перенесені у SDTM, з SDTM у ADaM та у TLF. Тому дуже важливо ще на етапі програмування SDTM, коли дані містять мінімум перетворень, виявити можливі помилки у даних.

1.4 Огляд існуючих аналогів

На сьогодні не існує у відкритому доступі системи, яка б автоматизувала створення SDTM датасетів.

Щодо систем пошуку помилок у сирих даних, наведено опис найближчих аналогів.

1.4.1 Застосунок OpenRefine

Раніше відомий як Google Refine, OpenRefine це потужний інструмент для роботи з даними, для їх очищення і перетворення. Інтерфейс застосунку представлений на рис.1.7.

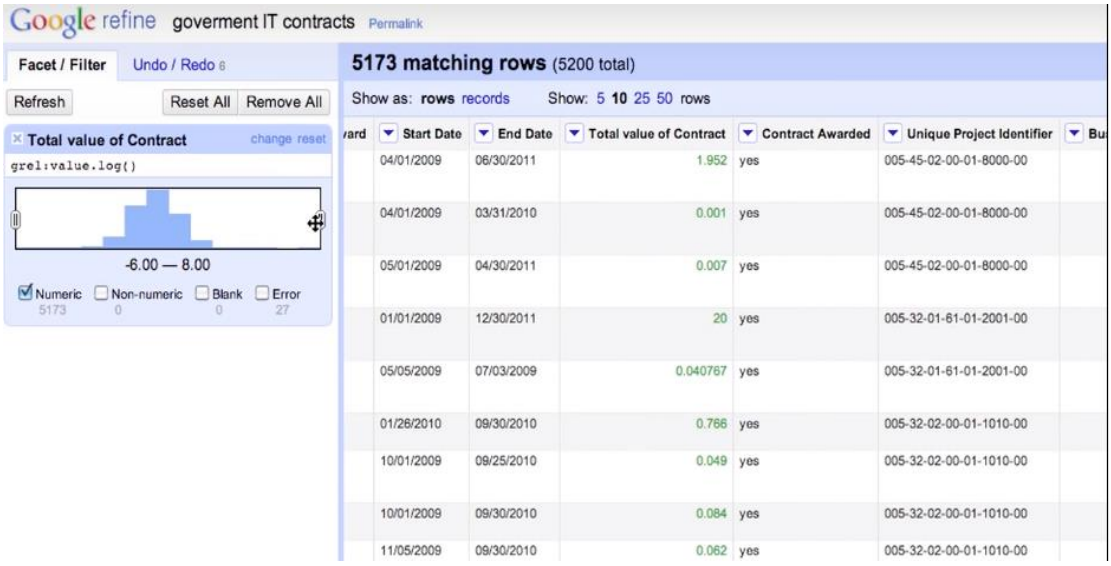


Рисунок 1.7 – Інтерфейс застосунку OpenRefine

Переваги OpenRefine:

- безкоштовний;
- з відкритим кодом;
- є можливість перетворювати дані з одного формату в інший;
- висока продуктивність на великих наборах даних.

Недоліки:

- не працює з форматом даних SAS7BDAT;
- не враховані особливості обробки даних медичних досліджень;
- неможливість інтеграції з системою обробки даних.

1.4.2 WinPure Clean & Match Data Cleansing and Matching Software

Платний застосунок для перевірки даних, включає в себе такі перевірки, як перевірка дублікатів, з виведенням рекомендації щодо того, який запис є більш ціннім; перевірка форматів змінних, наприклад дат, введення певних перевірок для конкретного набору даних. Інтерфейс застосунку представлений на рис. 1.8.

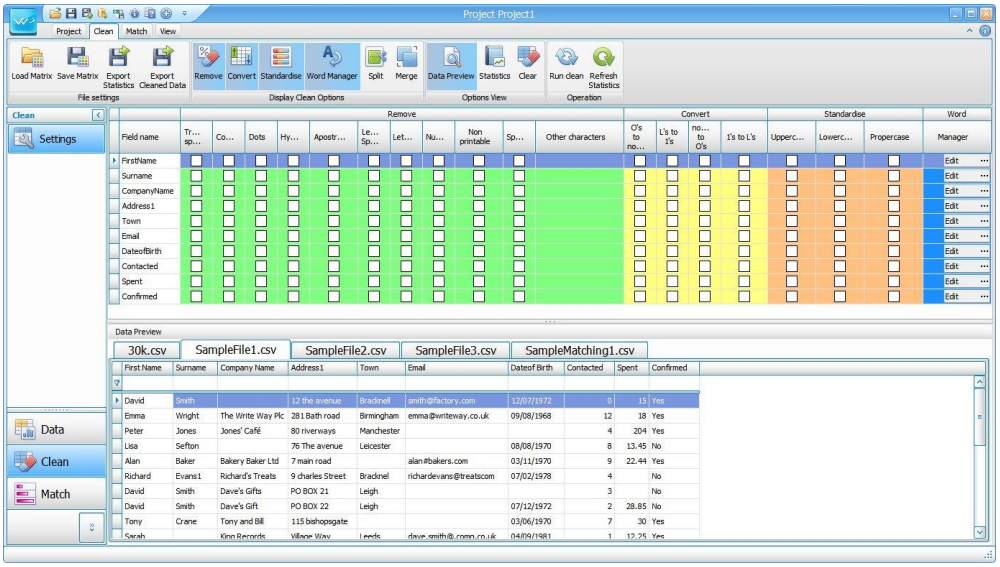


Рисунок 1.8 – Інтерфейс застосунку WinPure

Переваги даного застосунку:

- містить перевірки дублікатів, з виведенням рекомендацій щодо цінності кожного запису;
- має службу підтримки;
- є можливість налаштування певних перевірок для наборів даних.

Недоліки:

- не працює з форматом даних SAS7BDAT;
- не враховані особливості обробки даних медичних досліджень;
- неможливість інтеграції з системою обробки даних;
- висока вартість.

1.4.3 Застосунок Trifacta Wrangler

Trifacta Wrangler це безкоштовний застосунок, створений виробниками Data Wrangler, для очищення та перетворення даних. Він допомагає аналітикам даних у чищенні та підготовці безладних, різноманітних даних. Інтерфейс застосунку зображено на рис. 1.9.

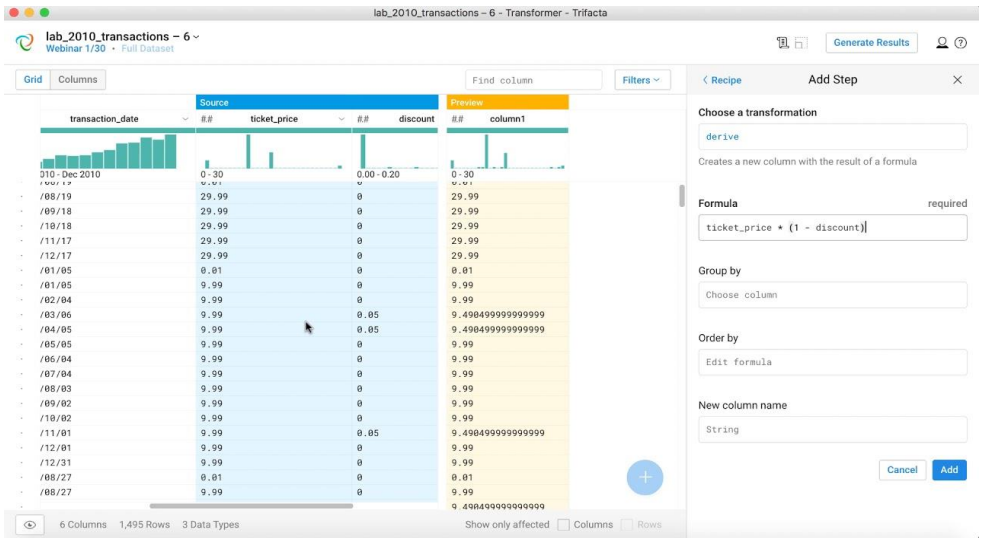


Рисунок 1.9 – Інтерфейс застосунку Trifacta Wrangler

Переваги Trifacta Wrangler:

- безкоштовний;
- алгоритми машинного навчання для агрегації та підготовки даних для аналізу;
- висока продуктивність на великих наборах даних.

Недоліки:

- не працює з форматом даних SAS7BDAT;
- не враховані особливості обробки даних медичних досліджень;
- неможливість інтеграції з системою обробки даних.

Висновки до розділу

Процес обробки даних існуючих систем можна вдосконалити шляхом автоматизації підсистеми створення SDTM датасетів та впровадження модулю пошуку помилок у сирих даних.

На сьогодні не існує програмного рішення проблеми автоматизації створення SDTM датасетів. Існуючі рішення задачі пошуку помилок у сирих даних мають ряд недоліків, з яких спільними є те, що їх неможливо інтегрувати в існуючи систему через те, що вони не підтримують формат даних, та містять досить обмежену кількість перевірок, адже мають більш загальне призначення та не враховують особливості процесу обробки даних медичних досліджень.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Функціональні вимоги до системи

Необхідно покращити існуючі автоматизовані системи обробки даних медичних досліджень шляхом розроблення підсистеми створення SDTM датасетів та модулю пошуку помилок у сирих даних. Вдосконалений процес буде мати структуру представлену на рис. 2.1.

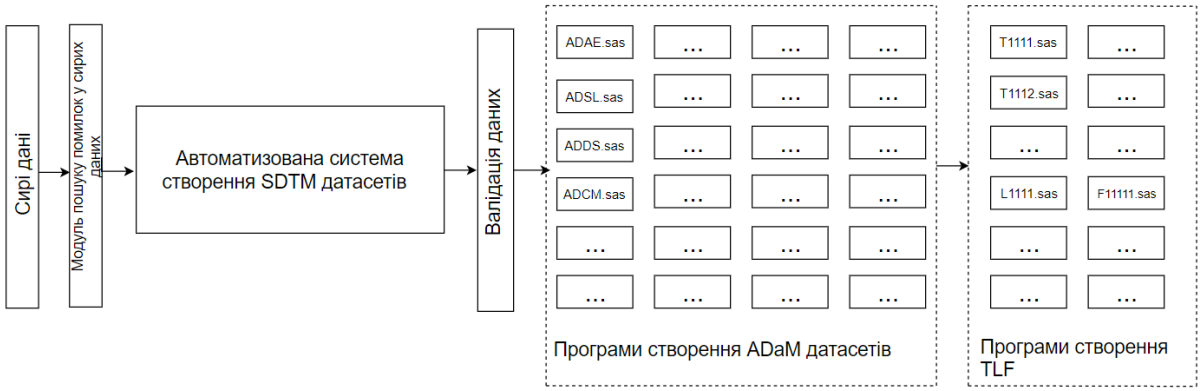


Рисунок 2.1 – Процес вдосконаленої системи

Опис процесу обробки даних:

- отримання сирих даних з бази даних виконується кожного дня о 6:00 GMT;
- після отримання даних, запускається модуль пошуку помилок у сирих даних, формується звіт, та у випадку наявності помилок відсилається електронний лист до відділу управління даними;
- після перевірки даних, починається створення SDTM датасетів:
  - створюються датасети на базі файлу специфікації;
  - виконується перевірка CT (Controlled Terminology);

    - перевіряється журнал виконання. У випадку наявності помилок або попереджень у журналі, відправляється електронний лист команді SAS програмістів;
    - якщо набір датасетів створено успішно, то надсилається електронний лист команді QC;
- валідація SDTM датасетів;
- програмування ADaM датасетів;
- валідація ADaM датасетів;
- програмування TLF звітів;
- валідація TLF звітів.

Модуль пошуку помилок у сирих даних буде інтегрований таким чином, щоб він запускався перед початком створення SDTM датасетів. Модуль пошуку помилок у сирих даних має включати в себе наступні перевірки:

- перевірки дати оновлення сирих даних – якщо дані не оновлені, то запуск модуля не відбувається;
- перевірки, що датасет не пустий;
- перевірки, що датасет дійсно був оновлений відносно датасету, який перевірявся в останній раз;
- перевірки на основі ALS (Architecture Load Specification) – документу, який представляє собою перелік усіх форм сирих даних (згідно до CRF) з мапінгом змінних сирих даних та деякими метаданими, наприклад інформацію про формат змінної, чи може змінна приймати пусті значення, словник допустимих значень;
- після визначення класу датасету, який перевіряється, та визначення ключових змінних, будуть визначені конкретні перевірки для визначеного класу датасетів (наприклад для класу датасетів, що містять результати тестів,

характерна наявність даних про візит – дати початку та кінця, які не мають пересікатися;

- деякі загальні помилки які будуть перевірятися:
  - правильний формат запису дат;
  - спеціальні символи – якщо вони присутні, то програма має їх видалити та зберегти новий датасет та надіслати повідомлення про це команді розробників;
  - перевірка дублікатів по ключовим змінним, з рекомендацією щодо того, який запис є більш інформаційним.

У результаті має бути створений звіт, у вигляді XLSX файлу, з вкладками для кожного датасету з переліком записів, що містять помилки, та вкладка з загальною інформацією про помилки. Цей звіт буде відправлений відділу управляння даними у випадку, якщо помилки знайдені.

Підсистема створення SDTM датасетів повинна виконувати наступне:

- завантажувати сирі дані;
- створювати датасети на базі правил вказаних у специфікації;
- виконувати перевірку змінних у датасетах на відповідність CT;
- проставляти атрибути на рівні датасету та на рівні кожної змінної;
- перевіряти журнал виконання, у випадку наявності помилок чи неприпустимих повідомлень (наприклад, некоректне виконання арифметичних операцій), підсистема має надіслати електронний лист команді розробників;
- у випадку успішного виконання програми (мається на увазі успішне створення повного пакету датасетів та відсутність помилок у файлі журналу), підсистема має надіслати лист команді QC.

2.2 Нефункціональні вимоги до системи

До системи висуваються наступні нефункціональні вимоги:

- апаратні та програмні вимоги (Hardware/Software Requirements) – сумісність із SAS v9.4, SAS Enterprise Guide v7.1 і пізніші. Ця вимога також включає підтримку формату даних SAS7BDAT;
- відповідність результатів роботи системи стандартам CDISC – у системі мають бути наявні способи перевірки даних на відповідність стандартам медичних досліджень;
- керування помилками – система має бути стійкою до виникнення помилок та коректно їх опрацьовувати.

Висновки до розділу

Визначено функціональні та нефункціональні вимоги до автоматизованої системи обробки даних медичних досліджень, з урахуванням недоліків існуючих аналогів для автоматизації процесів пошуку помилок у сирих даних.

Функціональні вимоги до системи, включають, зокрема, те, що модуль пошуку помилок у сирих даних інтегрований у систему таким чином, що він запускається до запуску підсистеми створення SDTM датасетів, а також враховує особливості обробки даних медичних досліджень.

Підсистема створення SDTM датасетів має повністю автоматизувати процес створення SDTM датасетів, перевірку змінних на відповідність CT, перевірку журналу виконання, та має інформувати команду QC про успішне створення SDTM датасетів.

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Сценарії використання

Найкраще продемонструє сценарії використання діаграма прецедентів, представлена на рис. 3.1 та у додатку А. Вона описує взаємодію між користувачами та системою, та відображає усі можливі дії та обов’язки користувачів у системі.

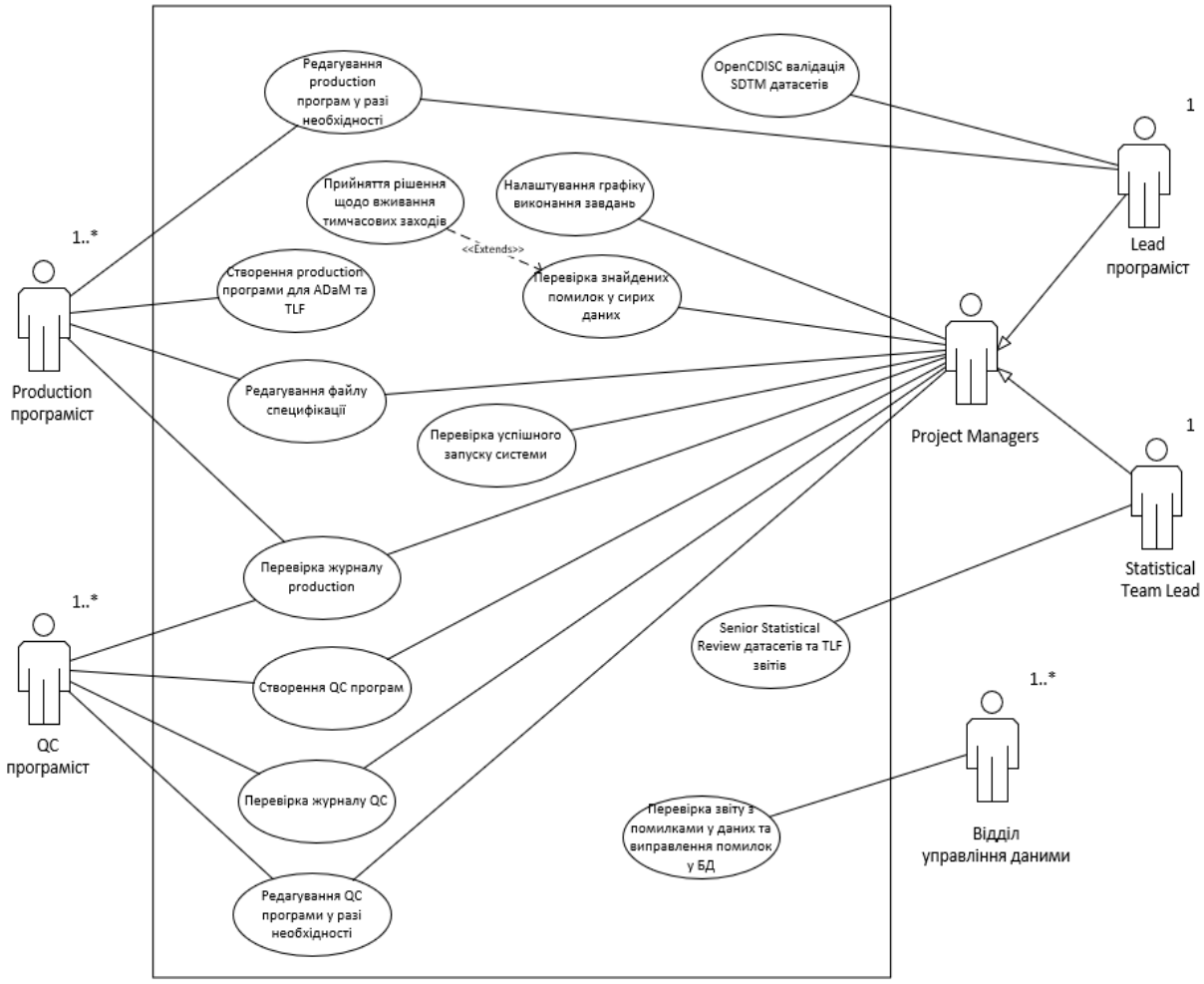


Рисунок 3.1 – Діаграма прецедентів

Передбачається, що із системою будуть взаємодіяти 5 типів користувачів: Production програміст (яких може бути декілька), QC програміст (яких також може бути

декілька), Lead програміст (зазвичай один), Statistical Team Lead (також зазвичай один) та представники відділу управління даними.

Production програміст відповідальний за створення production датасетів та звітів. До його обов'язків входить створення production програм (ADaM, TLF), внесення виправлень у production програми (у разі необхідності) та перевірка production журналу.

QC програміст відповідальний за валідацію датасетів та звітів. До його обов'язків входить перевірка production журналу, створення QC програм, внесення виправлень у програми QC у разі необхідності та перевірка журналу QC.

Користувачів Lead програміст та Statistical Team Lead узагальнено користувачем Project Managers – ці користувачі відповідальні за планування дослідження, налаштування проекту та якісне та вчасне виконання поставлених задач. При взаємодії із системою, вони відповідальні за:

- налаштування графіку виконання завдань;
- перевірку успішного запуску системи;
- перевірку знайдених помилок у сирих даних, та, у разі необхідності, прийняття рішення щодо тимчасових заходів, для забезпечення цілісності даних;
- перевірка результатів QC процесу.

Крім цього Lead програміст відповідальний за OpenCDISC валідацію SDTM та ADaM датасетів та внесення виправлень до production програм.

Statistical Team Lead також відповідальний за проведення Senior Statistical Review ADaM датасетів та TLF звітів – перевірка правильного виконання статистичного аналізу даних, відповідність створених датасетів та звітів до вимог замовника або стандартів.

Представники відділу управління даними зможуть переглядати знайдені помилки у даних у звіті, та вносити відповідні виправлення у БД.

Для ілюстрації послідовності процесу обробки даних, створено діаграму активності, яку представлено у додатку Б.

3.2 Загальна структурна схема

Розроблена загальна структурна схема представлена на рис 3.2 та у додатку В.

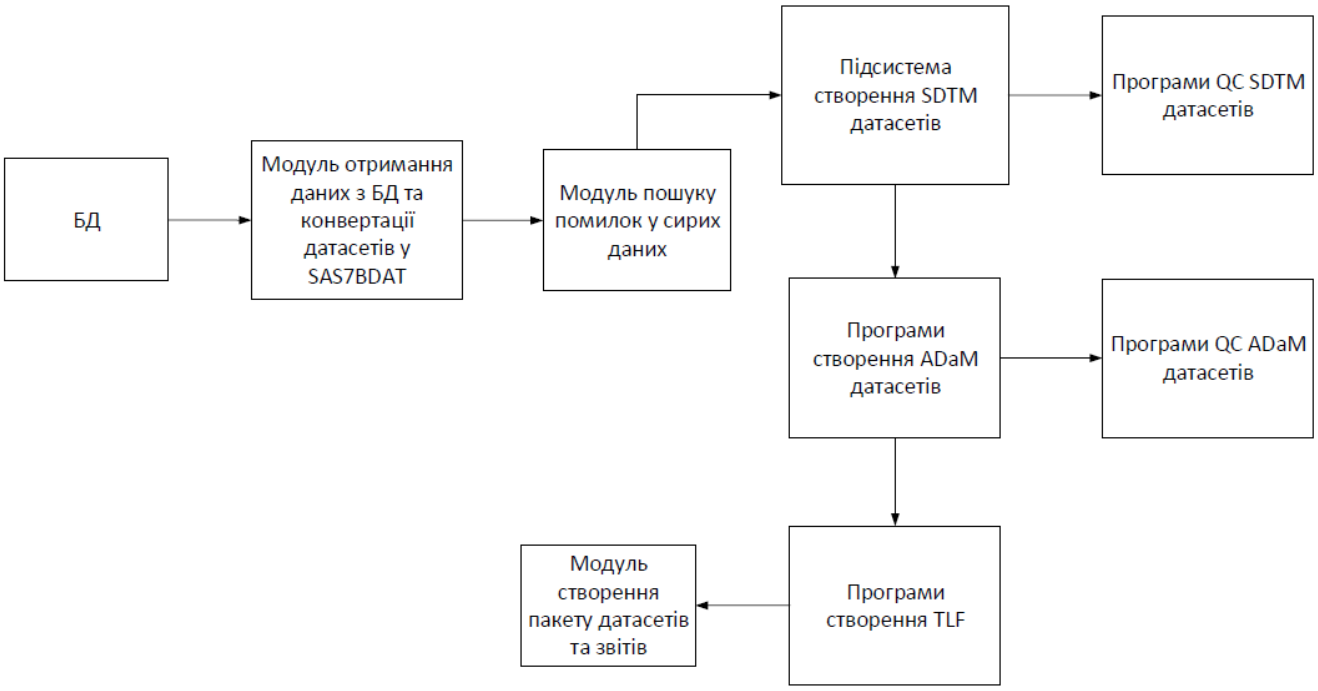


Рисунок 3.2 – Загальна структурна схема

Вона має такі компоненти:

- база даних (БД);
- модуль отримання даних з БД – виконує отримання даних у зазначений час, та конвертацію отриманих CSV або XPT датасетів у SAS7BDAT. У разі виникнення помилок на цьому етапі, надсилається електронний лист до команди розробників та відділу управління даними;
- модуль пошуку помилок у сирих даних – виконує перевірку сирих даних, отриманих з БД. Складається з декількох підмодулів (деталізована структурна схема наведена у підрозділі 3.3). У разі виникнення критичних помилок (дані

відсутні або неповні), надсилається лист до відділу управління даними, та у разі виправлення помилок, повторно робиться отримання даних з БД.

- підсистема створення SDTM датасетів – має також декілька модулів. Деталізована структурна схема цієї підсистеми наведена у підрозділі 3.4;
- програми створення QC SDTM датасетів – програми валідації SDTM датасетів.
- програми створення ADaM датасетів – набір програм SAS для створення датасетів для аналізу;
- програми створення QC ADaM датасетів – програми валідації ADaM датасетів.
- програми створення TLF звітів – набір програм SAS для створення звітів на базі ADaM датасетів. Зазвичай, кількість програм створення TLF звітів у рази перевищує кількість ADaM датасетів;
- модуль створення пакету датасетів та звітів – створення архіву з усіма датасетами, зконвертованими у формат XPT, та створення супроводжуючих DEFINE.XML файлів, за необхідності.

3.3 Структурна схема модулю пошуку помилок у сирих даних

Структурна схема модулю пошуку помилок у сирих даних має такі компоненти:

- модуль перевірки наявності датасетів – у випадку якщо датасет відсутній, або дата оновлення менша за вказану дату, або дані не були оновлені, то пошук помилок не відбувається;
- присутні окремі модулі пошуку помилок кожного типу – таким чином, є можливість обирати які помилки необхідно перевіряти;
- модуль формування звіту знайдених помилок. У разі виявлення будь-яких помилок відправляється лист відділу управління даними.

Структурна схема модулю пошуку помилок у сирих даних представлена на рис.3.3 та у додатку Г.

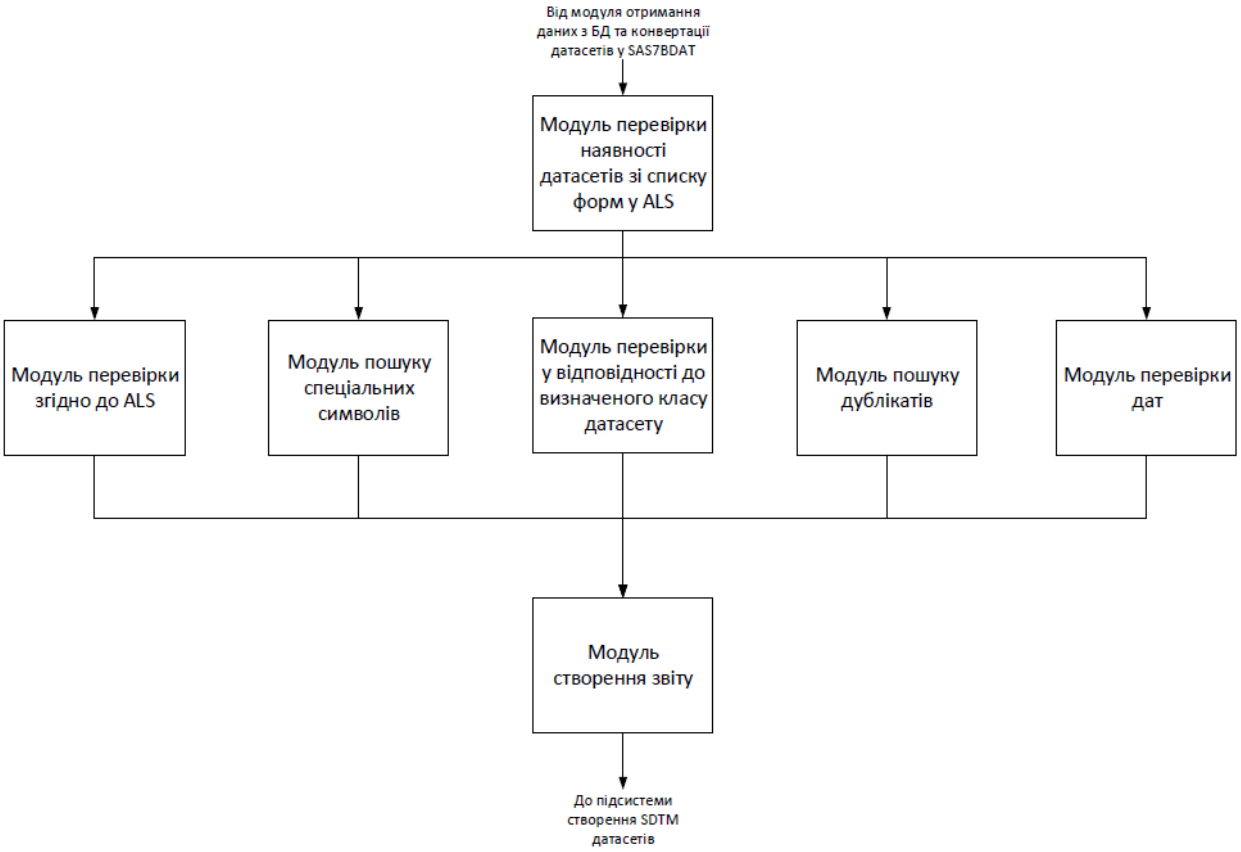


Рисунок 3.3 – Структурна схема модулю пошуку помилок у сирих даних

3.4 Структурна схема підсистеми створення SDTM датасетів

Структурна схема підсистеми створення SDTM датасетів має такі компоненти:

- модуль завантаження сирих даних – завантаження датасетів у сесію, для уникнення ризику перезапису сирих даних;
- модуль створення шаблону датасету – завантаження атрибутів змінних зі специфікації, створення форматів для змінних які будуть перевірятися на відповідність до CT. на цьому кроці також відбувається валідація специфікації

– перевірка, чи немає суперечностей у правилах або атрибутах змінних. у випадку, якщо є помилки, буде виведено повідомлення у журнал;

- модуль мапінгу змінних – модуль, в якому виконується присвоєння змінним з шаблону змінних з сирих даних у відповідному форматі або використовуючи правила зі специфікації. У цьому модулі виконується зчитування правил зі специфікації та формуються правила створення змінних;
- модуль перевірки змінних на відповідність до СТ – після створення змінних, визначаються змінні, які будуть перевірятися на відповідність до СТ, та виконується перевірка. У випадку, якщо є невідповідності, то буде виведено повідомлення у журнал;
- модуль перевірки журналу виконання – зчитування файлу журналу, та відправлення повідомлення розробникам, у випадку наявності неприпустимих повідомлень.
- у випадку, коли всі SDTM датасети створено успішно, відправляється електронний лист до команди QC.

Структурна схема підсистеми створення SDTM датасетів представлена на рис. 3.4 та у додатку Ґ.

Спроектована таким чином підсистема створення SDTM датасетів дозволить значно спростити процес створення SDTM датасетів.

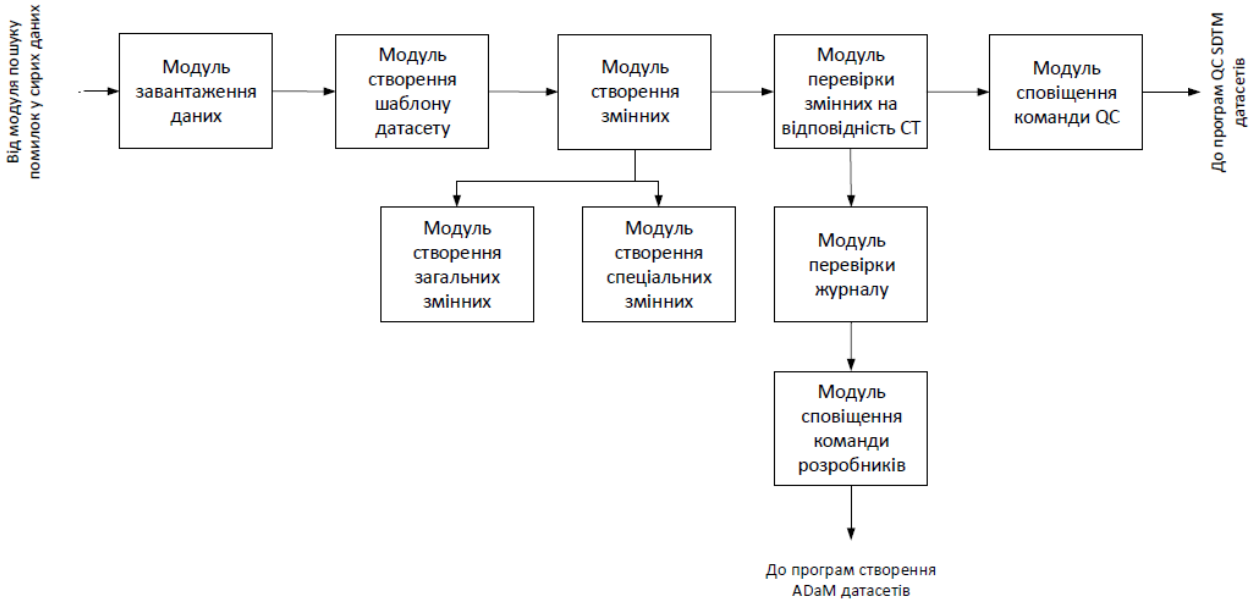


Рисунок 3.4 – Структурна схема підсистеми створення SDTM датасетів

Висновки до розділу

Визначено сценарії використання для користувачів та розроблено відповідні структурну схему автоматизованої системи та деталізовані схеми для модуля пошуку помилок у сирих даних та підсистеми створення SDTM датасетів. Спроектована таким чином система дозволить значно спростити процес обробки даних медичних досліджень.

Модуль пошуку помилок у сирих даних та підсистема створення SDTM датасетів містять декілька модулів, що забезпечить гнучкість системи та зробить систему менш залежною від конкретного типу дослідження.

4 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБЛЕННЯ

4.1 Вибір засобів розроблення

Рішення поставленої задачі розроблено з використанням мови програмування SAS версії 9.4 у середовищі розробки SAS Enterprise Guide 7.1.

Система SAS (Statistical Analysis System) – це набір статистичних програм, розроблений Інститутом SAS для розширеної аналітики, багатофакторного аналізу, управління даними та прогнозованої аналітики. SAS надає графічний інтерфейс користувача для нетехнічних користувачів та більш розширені можливості за допомогою мови програмування SAS [10].

Мова програмування SAS – це процедурна мова програмування високого рівня, що використовується для статистичного аналізу, створена Ентоні Джеймсом Барром з Державного університету Північної Кароліни. Мова SAS працює під компіляторами, які можна використовувати на Microsoft Windows, Linux та різних інших комп'ютерах UNIX та мейнфрейм.

Мова програмування SAS розвивалась майже одночасно з розвитком регулювання медичних досліджень, тому засоби розробки SAS максимально задовольняють потреби фармакомпаній у обробці даних. На сьогодні багато фармакомпаній звертають увагу на інші безкоштовні мови програмування (Python, R), але незважаючи на це, SAS залишається лідером у сфері медичних досліджень.

Переваги мови програмування SAS:

- перевірене статистичне програмне забезпечення, яке не тільки створює стандартні звіти, але й може виводити їх результати в набори даних SAS для використання в мові програмування SAS для створення будь-яких спеціалізованих звітів;

- через безпрецедентну безпеку даних, що надається програмним забезпеченням SAS, вона є лідером галузі програмного забезпечення для аналітики в секторі BFSI (Banking, Financial Services and Insurance);
- SAS надає одну з найкращих технічних підтримок;
- документація SAS дуже детальна в порівнянні з програмним забезпеченням з відкритим кодом, таким як R і Python;
- SAS може зберігати набори даних на жорсткому диску та обробляти набір даних більших розмірів, ніж розмір оперативної пам'яті;
- стабільне програмне забезпечення: усі функції та процедури попередньої версії програмного забезпечення підтримуються в нових версіях SAS. Вартість ліцензії на програмне забезпечення – це ніщо для банку або фармацевтичної компанії.
- тривалість використання – багато компаній використовували SAS протягом 20-30 років, і вони автоматизували весь процес аналізу за допомогою SAS. Для перетворення всієї стабільної системи звітності з SAS в R/Python можуть знадобитися значні витрати часу та ресурсів.

4.2 Огляд системи SAS

SAS система включає в себе декілька важливих компонентів наведених в табл.4.1.

Таблиця 4.1 – Компоненти системи SAS

Назва компоненту	Використання
Base SAS	Це основний компонент, який містить засоби управління даними та мову програмування для аналізу даних.
SAS/GRAPH	Застосовується для створення графіків, презентацій для кращого розуміння та демонстрації результату у відповідному форматі.

Продовження таблиці 4.1

SAS/STAT	Для проведення статистичного аналізу за допомогою дисперсійного аналізу, регресії, багатофакторного аналізу, аналізу виживання та психометричного аналізу, змішаного модельного аналізу.
SAS/OR	Для дослідження операцій.
SAS/ETS	Для аналізу економетрики та часових рядів.
SAS/IML	Data mining (IML – Interactive Matrix Language)
SAS/AF	Для розроблення застосунків.
SAS/QC	Для контролю якості.
SAS/INSIGHT	Data mining.
SAS/PH	Аналіз у медичних випробуваннях.
SAS/Enterprise Miner	Data mining.

Для вирішення поставленої задачі, необхідні компоненти Base SAS, SAS/STAT та SAS/GRAPH. Компонент Base SAS містить основні засоби обробки даних різних форматів, базові функції для статистичної обробки даних та компонент Macro Facility для оптимізації написання програм [12]. Засоби Base SAS будуть використовуватися протягом усього процесу розроблення системи.

Компонент SAS/STAT містить інструменти для складного статистичного аналізу, отже головним чином буде використовуватися на етапі програмування ADaM датасетів та TLF звітів.

SAS/GRAPH надає засоби створення кастомізованих звітів у форматі RTF, PDF, HTML або PPTX файлів, і отже буде використовуватися для програмування TLF звітів.

4.2.1 Особливості мови програмування SAS

Мова програмування SAS – це процедурна мова програмування високого рівня. Три компоненти будь-якої програми SAS – команди, змінні та набори даних, відповідають наведеним нижче правилам щодо синтаксису.

Набори даних SAS називаються файлами даних. Файли даних складаються з рядків і стовпців. Рядки містять спостереження, а стовпці містять назви змінних.

SAS має два типи змінних:

- числові змінні: це тип змінної за замовчуванням. Ці змінні використовуються в математичних виразах. Довжина числових змінних у наборах даних SAS становить 8 байт. У SAS під Windows тип цифрових значень даних, що мають довжину 8, має значення LONG REAL. Точність значень з плаваючою комою точна приблизно до 15 цифр. Залежно від кількості, точність може становити 16 цифр точності. Значущі цифри та найбільший цілий чисельність за довжиною для змінних SAS під Windows визначає значущі цифри та найбільші цілі значення, які можна зберігати в числових змінних SAS [13].
- символьні змінні: використовуються для значень, які не використовуються в математичних виразах. У SAS під Windows значення символів сортуються за допомогою послідовності згортання ASCII [14].

Щодо структури програми, вона має виділені «кроки» – DATA (крок даних) та PROC (крок процедури) [15].

Крок даних включає завантаження необхідного набору даних у пам'ять SAS та визначення змінних набору даних. Він також фіксує записи (їх також називають спостереженнями або суб’єктами). Синтаксис для оператора DATA наведено на рис. 4.1.

```
DATA data_set_name;         /*Інструкція створення датасету  data_set_name*/
    SET data_set_name0;     /*завантаження датасету data_set_name0*/
    LENGTH var1 $20 ;       /*оголошення нової змінної у датасеті */
    var1 = "Hello world!"; /*ініціалізація змінної*/
RUN;                        /*закінчення створення датасету data_set_name та збереження у WORK*/
```

Рисунок 4.1 – Приклад синтаксису кроку даних

Варто відзначити особливість відпрацювання кроку даних – він виконується ітеративно, зчитуючи по одному запису з датасету за одну ітерацію.

Крок процедури – використання вбудованої процедури SAS для аналізу даних.

Приклад синтаксису кроку процедури наведено на рис. 4.2.

```
PROC UNIVARIATE <options> ;
    BY variables ;
    CDFPLOT <variables> < / options> ;
    CLASS variable-1 <(v-options)> <variable-2 <(v-options)>> </ KEYLEVEL= value1 | ( value1 value2 )> ;
    FREQ variable ;
    HISTOGRAM <variables> < / options> ;
RUN;
```

Рисунок 4.2 – Приклад синтаксису кроку процедури

Як і в кроці даних, команда RUN вказує на завершення створення датасету та збереження його у тимчасовій бібліотеці.

Бібліотеки у SAS схожі на сховища даних. У SAS доступні два типи бібліотек:

- тимчасова або робоча бібліотека (WORK) – це бібліотека за замовчуванням. Усі створені програми зберігаються в цій робочій бібліотеці, якщо їм не призначено жодної іншої постійної бібліотеки. Якщо створена програма SAS не має жодної постійної бібліотеки, тоді, після завершення сеансу, всі набори даних та результати виконання процедур будуть видалені.
- постійна бібліотека – це бібліотека, якій призначено дійсний шлях до папки на диску, і тому, якщо створити програму в SAS і зберегти її в постійний бібліотеці, вона буде доступна і після завершення сесії [16].

4.2.2 Етапи обробки даних у SAS

Можна виділити два етапи обробки даних у SAS – етап компіляції та етап виконання [17].

Під час подання кроку DATA на виконання, SAS перевіряє синтаксис операторів SAS і компілює їх, тобто автоматично переводить команди в машинний код. На цьому етапі SAS ідентифікує тип та довжину кожної нової змінної та визначає, чи потрібне перетворення типу змінної для кожного наступного посилання на змінну. Під час фази компіляції SAS створює такі три елементи:

- вхідний буфер – це логічна область в пам'яті, в яку SAS зчитує кожен запис необроблених даних, коли SAS виконує оператор INPUT. Цей буфер створюється лише тоді, коли крок DATA зчитує необроблені дані – дані, створені у поточній сесії SAS). Якщо крок DATA читає існуючий набір даних SAS, SAS зчитує дані безпосередньо в ВПД.
- вектор програмних даних (ВПД) – це логічна область пам'яті, де SAS будує набір даних, по одному запису за раз. Коли програма виконується, SAS зчитує значення даних з вхідного буфера або створює їх, виконуючи оператори мови SAS. Значення даних присвоюються відповідним змінним у векторі даних програми. Звідси SAS записує значення в набір даних SAS як єдиний запис. Разом із змінними набору даних та обчисленими змінними, ПДВ містить дві автоматичні змінні, _N_ та _ERROR_. Змінна _N_ підраховує кількість ітерацій кроку даних, тобто кількість записів, які було зчитано. Змінна _ERROR_ сигналізує про помилку, викликану даними під час виконання. Значення _ERROR_ або 0 (що вказує на відсутність помилок), або 1 (що вказує на те, що сталася одна чи більше помилок). SAS не записує ці змінні у набір вихідних даних.

- описова інформація – це інформація, яку SAS створює та підтримує для кожного набору даних SAS, включаючи атрибути набору даних та атрибути змінних. Він містить, наприклад, назву набору даних та його тип, дату та час створення набору даних та кількість, імена та типи даних (символьні чи числові) змінних [18].

Після цього починається етап виконання. За замовчуванням простий крок DATA повторюється один раз для кожного створеного запису. Послідовність дій на етапі виконання простого кроку DATA можна описати так:

- крок даних починається з оператора DATA. Кожен раз, коли виконується оператор DATA, починається нова ітерація кроку DATA, а автоматична змінна _N_ збільшується на 1;
- SAS встановлює новостворені програмні змінні пустими у ВПД;
- SAS зчитує запис даних із необробленого файлу даних у вхідний буфер, або він читає спостереження з набору даних SAS безпосередньо у ВПД;
- SAS виконує будь-які наступні оператори програмування для поточного запису;
- після завершення обробки поточного запису, вихід, повернення та скидання відбуваються автоматично. SAS записує спостереження в набір даних SAS, система автоматично повертається до початку кроку DATA, а значення змінних, створених INPUT та операторами призначення, занулюються у векторі даних програми;
- SAS підраховує іншу ітерацію, читає наступний запис та виконує наступні операції програмування для поточного запису;
- Крок DATA припиняється, коли SAS доходить до кінця файлу у наборі даних SAS або у сирому файлі даних [18].

4.3 SAS Macro Facility

SAS Macro Facility – це інструмент для розширення та налаштування коду SAS. Він дозволяє зменшити кількість коду, який потрібно ввести для виконання загальних завдань, тобто дозволяє оптимізувати сам процес написання програм та зробити програми більш гнучкими.

Macro Facility має два компоненти:

- макро процесор – це частина SAS, яка виконує роботу;
- макро мова – це синтаксис, який використовується для спілкування з макропроцесором.

При зверненні до імені макро об'єкта в програмі SAS або в командному рядку, Macro Facility генерує оператори та команди SAS за потреби. Коли SAS компілює текст програми, два символи запускають макропроцесор:

- &name – посилання на макро змінну. Коли макро процесор знаходить цей символ, він підставляє замість &name значення макро змінної з простору імен;
- %name – посилання або виклик макросу.

Коли компілятор SAS знаходить один із таких символів, код надсилається до макро процесору, він його компілює та повертає згенеровані інструкції до вхідного буферу програми.

Макросом може бути колекція звичайних інструкцій SAS, операторів, макро змінних, функцій та макрофункцій, що містяться в межах %MACRO і %MEND команд. Оператор %MACRO включає ім'я макросу [20, 21].

Приклад синтаксису макросу для транспонування датасетів наведено на рис. 4.3.

```
%macro do_transpose(dsnlist=, bylist= ,varlist=, idlist=);
    %do i=1 %to %sysfunc(countw(&dsnlist));
        proc sort data = %scan(&dsnlist, &i);
            by &bylist;
        run;
        proc transpose data = %scan(&dsnlist, &i) out = tr_%scan(&dsnlist, &i);
            by &bylist;
            var &varlist;
            id &idlist;
        run;
    %end;
%mend;
```

Рисунок 4.3 – Приклад синтаксису макросу

Заміна тексту, вироблена макропроцесором, завершується перед початком виконання програми. Макроси використовують оператори та функції, що нагадують ті, які використовуються у кроці даних. Важлива різниця, однак, полягає в тому, що елементи макро мови можуть викликати лише заміну тексту, і не є присутніми у коді під час виконання програми або команди [21].

4.4 Вибір протоколу передачі даних

У якості протоколу передачі даних було обрано SSH.

SSH (Secure Shell) – це протокол криптографічної мережі рівня застосунків, який використовується для захисту мережевих з'єднань через незахищену мережу. SSH використовується для проведення віддаленого управління комп'ютером і тунелювання TCP-з'єднань (зокрема, для передачі файлів). На відміну від Telnet і rlogin, він шифрує весь трафік, а також паролі, що передаються.

Він може бути застосований для багатьох застосунків на різних платформах, включаючи UNIX та Windows [26].

Для використання SSH потрібно використовувати клієнт SSH для підключення до сервера. У якості клієнта SSH обрано безкоштовну програму PuTTY.

PuTTY спочатку розроблявся для Microsoft Windows, проте пізніше перероблений для Unix.

Вихідний код PuTTY повністю розроблений на C. PuTTY не залежить від DLL, інших застосунків, пакетів оновлень ОС. PuTTY і більшість утиліт запускаються тільки в одному потоці ОС. Програма є вільно поширюваним застосунком з відкритим вихідним кодом і випускається під Open Source з ліцензією MIT.

Деякі можливості програми:

- збереження списку і параметрів підключень для повторного використання;
- робота з ключами і версіями протоколу SSH;
- підтримка IPv6;
- підтримка аутентифікації з відкритим ключем, в тому числі і без введення пароля;
- підтримка роботи через послідовний порт (починаючи з версії 0.59);
- можливість роботи через проксі-сервер [28].

Вікно налаштування PuTTY наведено на рис. 4.4.

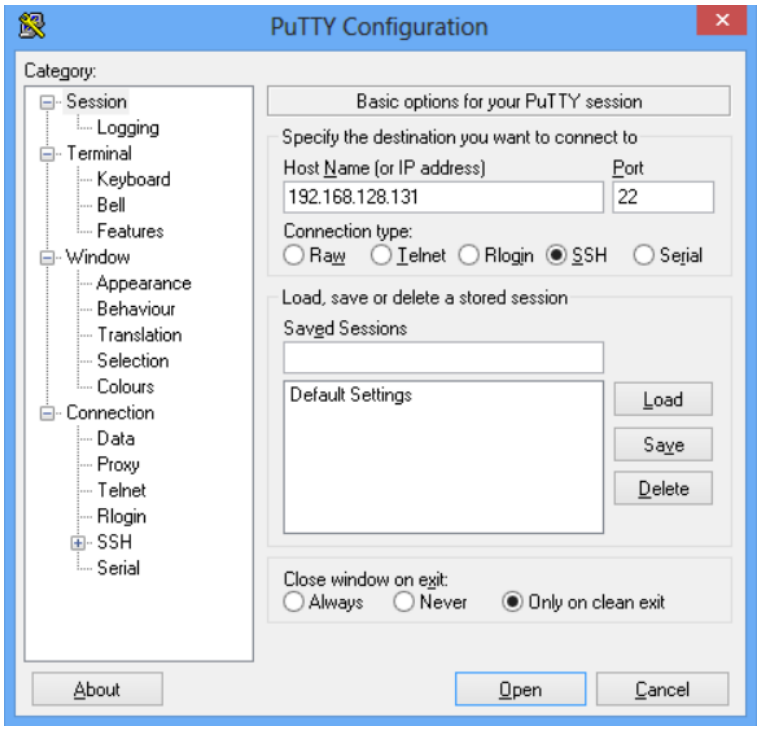


Рисунок 4.4 – Вікно налаштування PuTTY

4.5 Відомості про утиліту cron

Утиліта cron – класична утиліта в системах класу UNIX, що використовується для налаштування періодичного виконання завдань в певний час. Інструкції щодо регулярного запуску задач описуються у файлі crontab і в спеціальні каталогах. Створюючи пакетний batch (.bat) файл і пов’язуючи його з crontab, можна налаштувати виконання інструкцій із batch файлу у необхідний час [19].

Batch файл – це текстовий файл, що містить низку команд, призначених для виконання інтерпретатором команд. Коли пакетний файл запускається, програма shell зчитує файл і виконує його команди, як правило, по черзі. Пакетні файли є корисними для автоматичної запуску послідовності виконуваних файлів і часто використовується для автоматизації повторюваних або виснажливих процесів. У середовищі DOS та Windows, пакетні файли зазвичай мають розширення назви файлу .BAT.

Crontab – файл завдань, в якому вказано, в який час які програми запускати від імені певного користувача. Cron завжди викликає команди з домашнього каталогу користувача. Для редагування файлу crontab рекомендується використовувати однойменну програму crontab, що дозволяє внести зміну у файл, не перериваючи процес cron.

Деякі основні команди UNIX для їх перегляду та редагування файлу crontab наведено нижче:

- crontab -e – редагування файлу crontab або створення, якщо він ще не існує.
- crontab -l – відображення записів з файлу crontab.
- crontab -r – видалення файлу crontab.
- crontab -v – відображення дати останнього редагування [19].

Файл crontab містить 6 колонок, які розділені пробілами. У перших п'яти колонках задається час виконання (в такому порядку: хвилина, година, день, місяць, день тижня), у яких також можна задавати список, розділених комами, або діапазон чисел, розділених

дефісом. У системних файлах crontab після стовпців з часом вказується користувач, який запускає команду.

4.6 Середовище розробки SAS Enterprise Guide

SAS Enterprise Guide – це середовище розробки SAS програм з вбудованими інструментами аналізу даних [23]. Інтерфейс застосунку наведено на рис. 4.5.

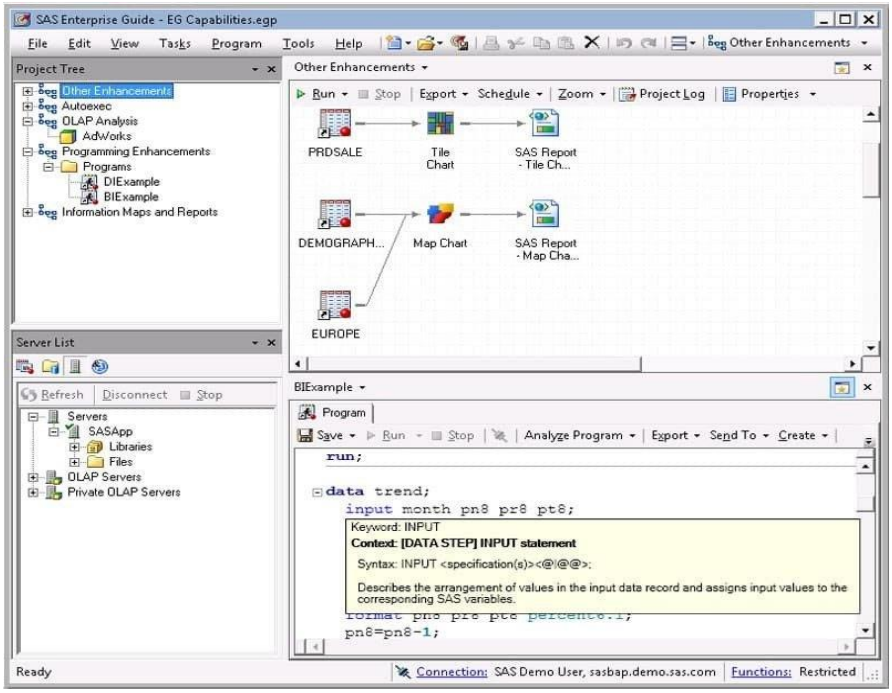


Рисунок 4.5 – Інтерфейс SAS Enterprise Guide

Основні вікна застосунку:

- Log Window – це вікно журналу виконання, де можна перевірити результати виконання програми SAS;
- Editor Window – вікно для написання коду на SAS;
- Output Window – це вікно результатів, куди завантажуються створені датасети;

- Result Window – містить у собі результати виконання усіх процедур, якщо тільки їх призначення не вказане явно (може бути RTF, PDF, HTML);
- Explore Window – тут відображаються усі існуючі в даній сесії бібліотеки. Також тут можна переглянути файли, що підтримуються системою SAS.

Переваги SAS Enterprise Guide:

- інтуїтивно зрозумілий графічний інтерфейс користувача;
- створення журналу з інформацією про результати виконання програми, включаючи примітки, попередження та помилки;
- результати можуть бути виконані у форматі HTML, RTF, PDF, SAS та текстових форматах. Результати також можна виводити у вигляді наборів даних SAS7BDAT для подальшого аналізу;
- графіки можна створити як ActiveX, Java-аплети, GIF або JPEG. ActiveX та Java аплети дозволяють безпосередньо взаємодіяти з графічними об'єкти без повторного надсилання запитів на сервер;
- інтуїтивно зрозуміла схема діаграми процесу дозволяє візуально впорядкувати та підтримувати структуру проектів;
- засоби статистичної обробки даних:
- регресійні моделі: лінійні, логістичні, нелінійні та узагальнені лінійні моделі;
- багатоваріантні моделі взаємозв'язків: кластерний аналіз, факторний аналіз, канонічний кореляційний та дискримінаційний аналіз функцій;
- аналіз виживання: таблиця життя та пропорційні небезпеки;
- дослідження операцій: числова оптимізація, мова алгебраїчного моделювання, планування ресурсів, генетичні алгоритми та програмування обмежень;
- адміністрація та безпека:
  - можна розгорнути за допомогою SAS Grid Manager для автоматизованого управління;

- високоефективні обчислення;
- дозволяє налаштувати гілки потоків процесу для паралельного запуску на різних вузлах сітки;
- аналізує програми SAS для оптимізації продуктивності коду в середовищі Grid;
- дозволяє паралельне виконання завдань на одному сервері;
- дозволяє виконувати завдання на рівні проекту або на рівні окремих завдань у SAS Grid [24].

SAS Enterprise Guide інтегрований з платформою SAS Intelligence. Архітектура платформи SAS Intelligence Platform призначена для ефективного доступу до великого обсягу даних.

Платформа SAS Intelligence Platform має багаторівневої архітектуру, яка складається з наступних 4 рівнів (рис. 4.6):

- рівень даних;
- рівень серверів SAS;
- середній рівень;
- рівень клієнтів.

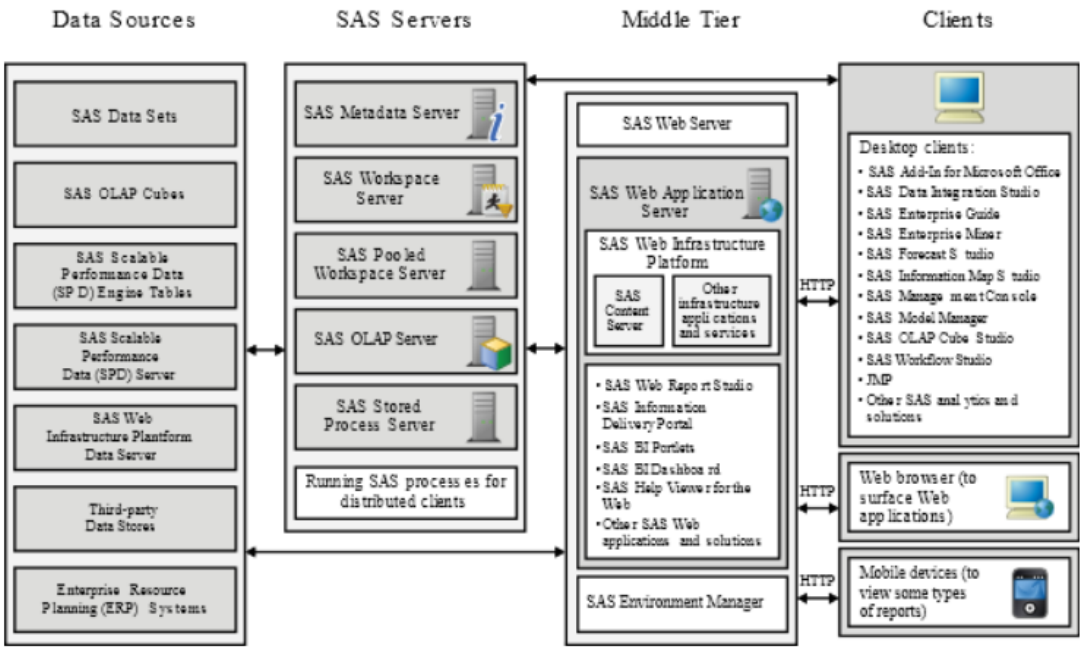


Рисунок 4.6 – Архітектура SAS Intelligence Platform

Існують різні сервери SAS, які виконують аналітичні та звітні процеси. Основними серверами на платформі SAS Intelligence Platform є:

- сервер метаданих SAS;
- сервер робочої області SAS;
- сервер об'єднаної робочої області SAS;
- сервер SAS OLAP;
- сервер збережених процесів SAS;
- сервер SAS Connect і сервер спільного доступу SAS.

У середньому рівні розміщені всі користувачі веб-застосунків SAS. Середній рівень включає наступне програмне забезпечення сторонніх виробників та програмні елементи SAS:

- сервер веб-застосунків;
- Java Development Kit;
- Веб-застосунки SAS;

- сервер SAS Content та віддалені сервіси.

Крім того, SAS Intelligence Platform надає продукти, які дозволяють отримати доступ до даних у існуючих сторонніх СУБД та ERP систем. Інтерфейси SAS/ACCESS забезпечують прямий доступ до СУБД, таких як:

- Oracle;
- DB2;
- Teradata;
- ODBC, Microsoft SQL Server та Sybase.

Клієнти платформи SAS Intelligence Platform надають інтерфейси користувача у застосунках, які працюють на Microsoft Windows:

- надбудова SAS для Microsoft Office;
- SAS Enterprise Guide;
- SAS Enterprise Miner;
- SAS Data Integration Studio;
- SAS Information Map Studio;
- SAS Management Console;
- SAS OLAP Cube Studio [26].

Таким чином, SAS Enterprise Guide обраний у якості середовища розробки, адже надає усі необхідні інструменти обробки даних.

4.7 Середовище розподіленого доступу SAS Grid

Середовище розподіленого доступу SAS Grid забезпечує спільне аналітичне обчислювальне середовище з централізованим управлінням, високу доступність та прискорює обробку даних. Він забезпечує управління навантаженням для оптимальної обробки декількох застосунків для максимальної загальної пропускної здатності. Середовище SAS Grid також забезпечує гнучкість для поступового зростання

обчислювальної інфраструктури, оскільки кількість користувачів та розмір даних з часом збільшуються, а також можливість робити постійне обслуговування та оновлення без будь-яких порушень для групи користувачів [29].

Архітектура SAS Grid представлена на рис. 4.7.

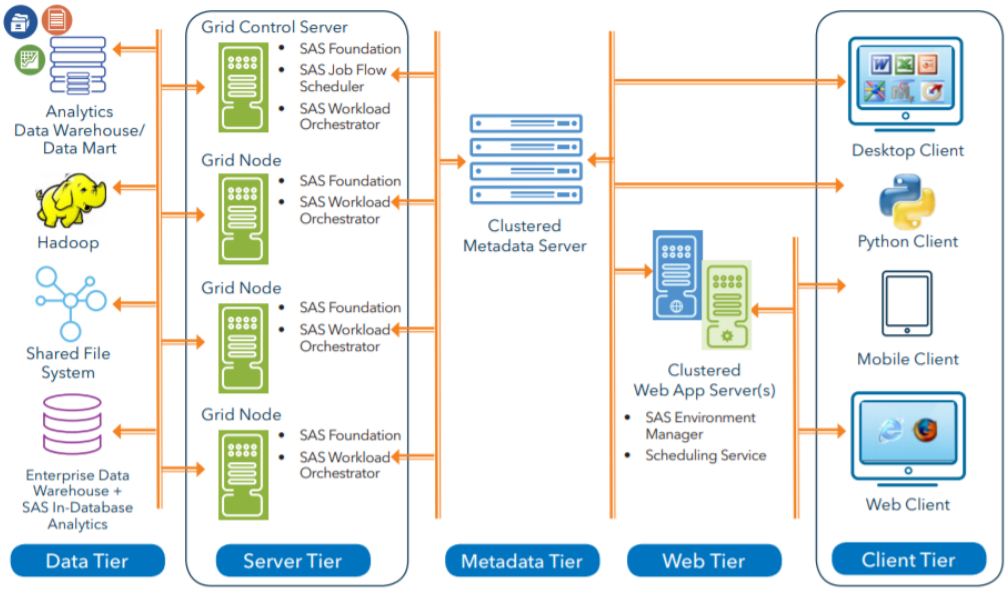


Рисунок 4.7 – Архітектура SAS Grid

Ключові особливості SAS Grid:

- обчислювальні ресурси доступні для багатьох користувачів та декількох застосунків для проведення більшого або складнішого аналізу;
- забезпечує автоматичну ідентифікацію, розподіл, управління та оптимізацію обчислювальних ресурсів та програмних потоків;
- автоматично пристосовується до робочих навантажень, що надходять до мережі;
- спрощує адміністрування середовищ SAS за допомогою централізованої політики;
- забезпечує високу доступність для критичних служб SAS, таких як SAS Metadata Server;

- виявляє апаратні та програмні збої в мережі та відновлює належним чином;
- забезпечує оптимальне виконання SAS та завдань з відкритим кодом;
- перезавантажує завдання SAS автоматично з останньої успішної контрольної точки;
- інтегрується з усіма програмами аналітики SAS;
- включає утиліту командного рядка для автоматизації batch завдань;
- включає інтеграцію з іншими стандартними планувальниками підприємства.

Таким чином, використання SAS Grid дозволяє необхідним чином надати доступ до середовища розробки усім користувачам з необхідним рівнем безпеки передачі даних. Крім того, SAS Grid забезпечує стійкість до помилок та можливість перезапуску системи з останньої успішної контрольної точки.

Висновки до розділу

У якості засобів розробки було обрано систему SAS, яка є лідером на ринку у сфері медичних досліджень, надає необхідні засоби розробки для виконання поставленої задачі, зокрема засоби обробки даних, в тому числі і для проведення статистичного аналізу даних, засоби розробки макросів, засоби розробки кастомізованих звітів TLF.

У якості протоколу взаємодії обрано SSH, так як він надає необхідний рівень безпеки. Налаштування графіку виконання задач буде розроблено за допомогою утиліти cron у клієнті SSH PuTTY.

У якості середовища розробки було обрано SAS Enterprise Guide 7.1, який містить у собі систему SAS та необхідні засоби розробки.

У якості середовища розподіленого доступу обрано SAS Grid, який дозволить надати доступ до середовища розробки усім користувачам та забезпечить необхідний рівень безпеки.

5 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

5.1 Модуль пошуку помилок у сирих даних

Модуль пошуку помилок у сирих даних, як вже зазначалося, буде містити у собі декілька підмодулів, що забезпечить його гнучкість та можливість переналаштування для потреб дослідження. Як вже зазначалось раніше, мова програмування SAS процедурна, імперативна, складається з переліку дата та процедурних кроків. У випадку коли ці кроки потрібно повторити для 100 датасетів, код може зайняти тисячі строк. Тому, для досягнення поставленої задачі, вирішено використовувати SAS Macro Facility

5.1.1 Алгоритм пошуку помилок у сирих даних

Модуль пошуку помилок складається з головного макросу, з якого починається виконання модулю, та 5 макросів для перевірок кожного типу, які будуть викликатися у головному макросі.

У головному макросі буде виконуватися перевірка наявності ALS файлу. У разі відсутності, буде виведено повідомлення у журнал, та, у випадку якщо було встановлено флаг перевірки згідно до ALS, то флаг буде занулено і перевірки будуть виконуватися для тих датасетів, які є у папці з сирими даними. Якщо ж файл ALS присутній, то перевірки будуть виконуватися для всіх датасетів, присутніх у ALS, та буде перевірятися їх наявність у папці. Далі починається цикл пошуку помилок для кожного датасету. Перевіряється, чи було обрано певний тип перевірки. якщо так, то викликається відповідний макрос. В кожному макросі створюється відповідний датасет з помилками, і після виходу з циклу, коли всі датасети було перевірено, створюється XLSX файл звіту з усіма помилками для кожного датасету та вкладкою з загальною інформацією про знайдені помилки. На рис. 5.1 та у додатку Д наведено блок-схему алгоритму основної програми модуля пошуку помилок у сирих даних.

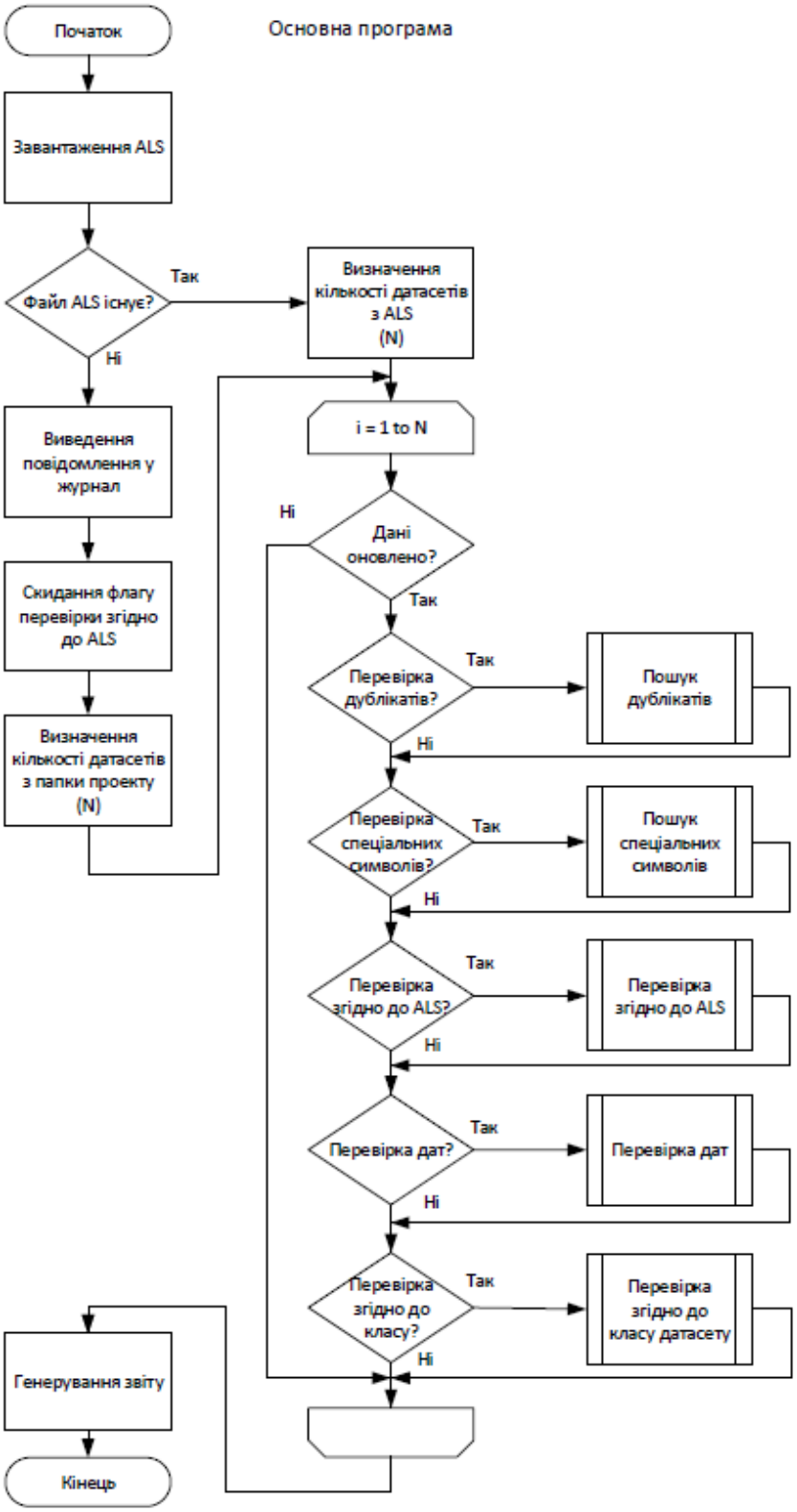


Рисунок 5.1 – Алгоритм основної програми модуля пошуку помилок у сирих даних

5.1.2 Реалізація підпрограм

Всі підмодулі розроблено у вигляді макросів.

На рис. 5.2 наведено частину лістингу коду, де визначаються параметри макросу та виконуються перевірки, якщо жоден флаг перевіруи не обран або якщо немає сирих датасетів для перевірки, то виводяться повідомлення у журнал та по мітці %exit_main відбувається перехід до завершення виконання макросу. Для перевірки наявності записів використовується автоматична макро змінна sqlobs – вона створюється лише після виконання процедурного кроку з PROC SQL, та приймає значення кількості записів які вернув запит.

```
%macro main(ALSFL=Y,
            DUPFL=Y,
            UNPRFL=Y,
            CLASSFL = Y,
            DATFL = Y);

%let N=;

/*check if raw datasets available*/
%let sqlobs = 0;
proc sql noprint;
   select *
       from sashelp.vstable
       where libname='RAW';
quit;
%if  &sqlobs=0 %then %do;
       %put WARNING: There are no raw datasets present;
       %goto  exit_main;
%end;
/*check if any data issues checks were selected*/
%if (ALSFL=N and DUPFL=N and UNPRFL=N and CLASSFL=N and DATFL=N)  %then %do;
   %put WARNING: No data issues checks were selected;
   %goto  exit_main;
%end;

```

Рисунок 5.2 – Перевірка наявності сирих даних та встановлення флагів перевірок

Далі створюються макро змінні N та DSNLIST (рис. 5.3), в залежності від того чи існує файл ALS. Якщо файлу немає, то обираються всі існуючі датасети з папки з сирими

даними. Якщо файл існує, то назви датасетів беруться з ALS. Ці макро змінні потрібні для виконання перевірок в циклі.

```
%let sqlobs = 0;
proc sql noprint;
   select *
       from sashelp.vcolumn
       where libname='ALS' and memname="%upcase(&alsversion)";
quit;
%if sqlobs=0 %then %do;
   %let ALSFL=N;
   %put WARNING: ALS file is absent;
   proc sql noprint;
       select count(distinct memname) into :N
       from sashelp.vstable
       where libname='RAW';

       select distinct memname into :dsnlist separated by ' '
       from sashelp.vstable
       where libname='RAW';
   quit;
%end;

%else %if sqlobs^=0 %then %do;
proc sql noprint;
       select count(distinct formid) into :N
       from als.forms;

       select distinct formid into :dsnlist separated by ' '
       from als.forms;
quit;
%end;
```

Рисунок 5.3 – Приклад лістингу коду

Потім у циклі для кожного датасету перевіряється розмір та дата оновлення датасету. Якщо датасет пустий то відбувається вихід з циклу. Цикл починає виконуватися тільки при умові, що дата модифікації дорівнює поточній даті. Частина лістингу коду представлена на рис. 5.4.

```
%do i=1 %to &N;
       * Take raw data filename into 'item' macro variable ;
       %let item = %scan(&dsnlist, &i, ' ');

       %local rc fileid fidc;
       %local Bytes ModifyDT;

       * Store information about opened file ;
       %let rc=%sysfunc(filename(onefile,&raw./&item));
       %let fileid=%sysfunc(fopen(&onefile));
       %let Bytes=%sysfunc(finfo(&fileid,File Size (bytes)));
       %let ModifyDT=%qsysfunc(finfo(&fileid,Last Modified));
       %let fidc=%sysfunc(fclose(&fileid));
       %let rc=%sysfunc(filename(onefile));

   %if &Bytes eq 0 %then %goto quit_cycle;

   %if %eval(&ModifyDT=&sysdate) %then %do;
       %if ALSFL=Y %then %do;
               %ALSCHECK(dsname=%scan(&dsnlist,&i));
       %end;
       %if DUPFL=Y %then %do;
               %DUPCHECK(dsname=%scan(&dsnlist,&i));
       %end;
       %if UNPRFL=Y %then %do;
               %UNPRCHECK(dsname=%scan(&dsnlist,&i));
       %end;
       %if CLASSFL=Y %then %do;
               %CLASSCHECK(dsname=%scan(&dsnlist,&i));
       %end;
       %if DATFL=Y %then %do;
               %DATCHECK(dsname=%scan(&dsnlist,&i));
       %end;

    %end;
%end;
%quit_cycle:
```

Рисунок 5.4 – Приклад лістингу коду

Макрос ALSCHECK(dsname=) – макрос перевірки датасетів на відповідність до ALS. Він включає у себе такі перевірки, як наявність вказаного датасету, наявність усіх змінних вказаних у ALS, відповідність форматів, типів даних, довжин, словників до ALS. Він також визначає клас датасету, ключові змінні та створює відповідні глобальні макро змінні &CLASS та &KEY_VARS. Ці змінні потім використовуються у макросах %DUPCHECK та %CLASSCHECK. На виході цього макросу створюється датасет <NAME>_ALS, де <NAME> – це назва сирого датасету.

Макрос %DUPCHECK(dsname=) виконує пошук дублікатів та створює рекомендації, у випадку якщо якісь із записів є більш цінним для аналізу. У такому випадку, Project Top Management може прийняти рішення щодо тимчасового використання цього запису, доки проблема з дублікатами не буде вирішена. На виході створюється датасет <NAME>_DUP, який містить і дублюючі записи, і рекомендації.

Блок схема алгоритму макросу %DUPCHECK наведено на рис. 5.5.

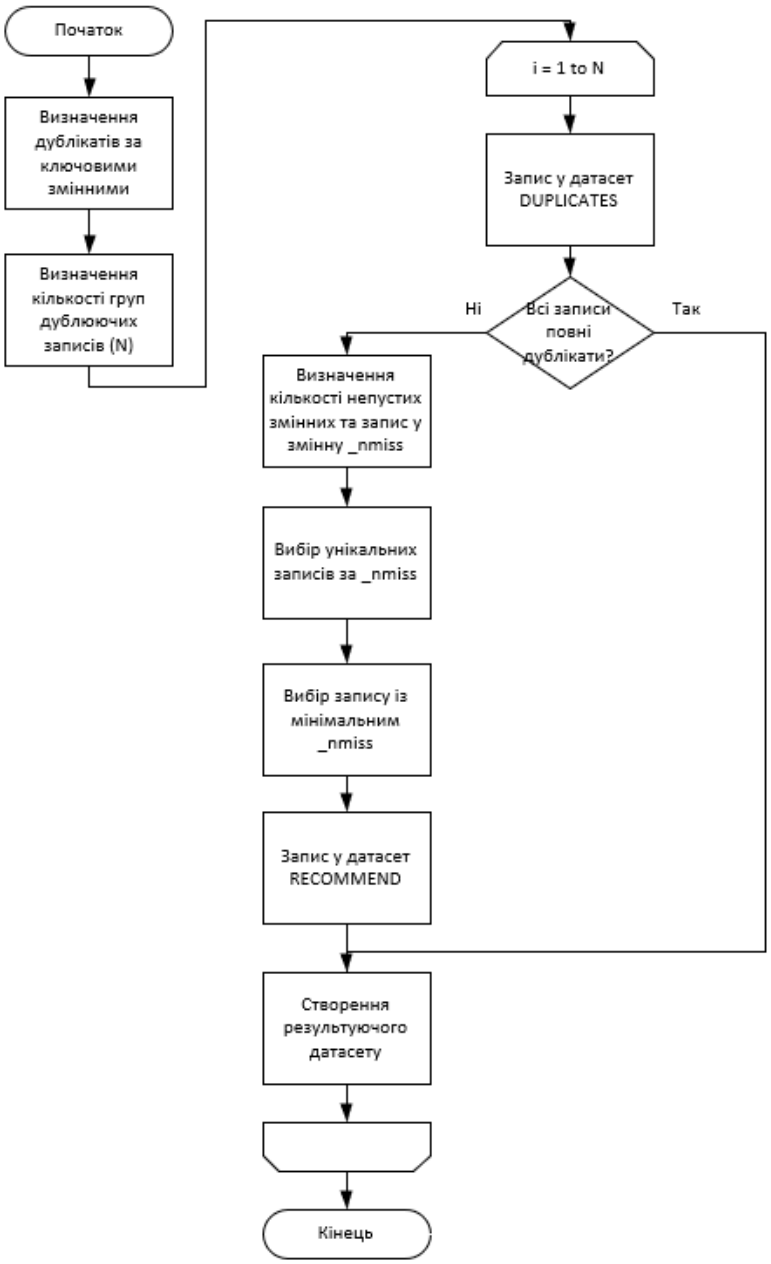


Рисунок 5.5 – Блок-схема алгоритму пошуку дублікатів.

Макрос %UNPRCHECK(dsname=) – цей макрос робить перевірку усіх символьних змінних та шукає недруковані символи, для уникнення проблем у випадку, якщо проект та данні мають різні налаштування послідовності кодування. На виході створюється датасет <NAME>_UNP.

Макрос %CLASSCHECK(dsname=) – макрос перевірки правил, характерних для певних класів датасетів. Наприклад, для датасетів класу FINDINGS, характерним буде наявність змінних візитів, тому необхідно перевірити, що дати візитів не пересікаються. На виході створюється датасет <NAME>_CLS_.

Макрос %DATCHECK(dsname=) – макрос перевірки змінних які містять дати. На виході створюється датасет <NAME>_DAT.

Потім усі створені датасети об’єднуються, створюються датасети REPORT_<NAME>, який містить усі знайдені помилки у кожному наборі сирих даних. Далі рахується загальної кількості помилок, кількість помилок кожного типу, створюється датасет SUMMARY, та створюється XLSX звіт.

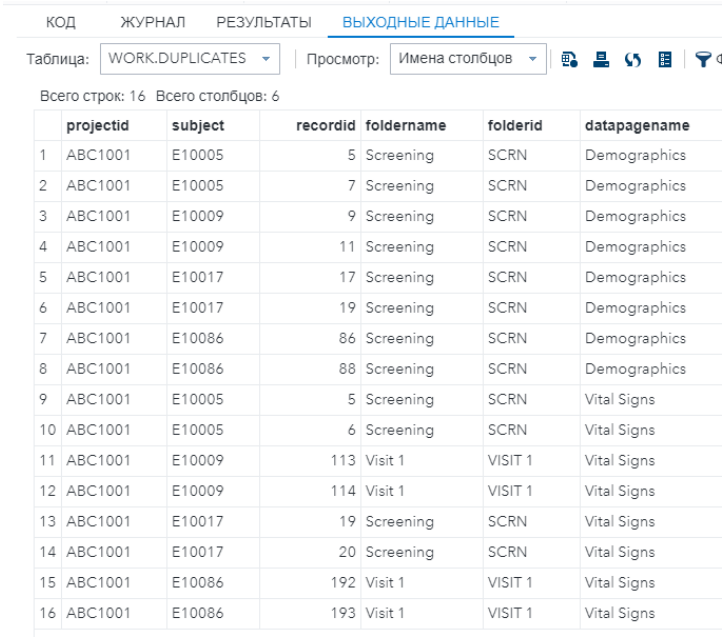
На рис. 5.6 наведено частину коду створення XLSX файлу звіту з усіма помилками для кожного датасету та вкладкою SUMMARY з загальною інформацією про знайдені помилки. Цей звіт автоматично відправлено до відділу управління даними.

```
 /*Generating XLSX report with all issues*/
libname report xlsx "&project/docs/outputs/DM_report.xlsx";
data _null_;
    set sashelp.vstable( where =(memname=:'REPORT_' or memname=:'SUMMARY')) end=end;
    if memname=:'REPORT_' then call execute('data report.'||strip(scan(memname,2,'_'))||';
                    set '||strip(memname)||';
                    run;');
    else if memname=:'SUMMARY' then call execute('data report.'||strip(memname)||';
                    set '||strip(memname)||';
                    run;');
run;

data _null_;
    FILE MailBox TO=('DataManagement@example.com>'
                    'NameSurname1@example.com>')
                     CC=(PL 'NameSurname2@example.com'
                     FROM='Biostat <Biostat@example.nl>'
                     SUBJECT="Data issues found on &sysdate"
                     ATTACH="&project/docs/outputs/DM_report.xlsx";
    put "Hi team, there were data issues found on &sysdate, please see file attached. Best regards, Biostat dept";
run;
```

Рисунок 5.6 – Приклад лістингу коду

Приклад виконання пошуку помилок на тестових даних наведено на рис. 5.7. В результаті виконання програми, було знайдено у дублікати у датасетах DEMOG (Demographics) та VS (Vital Signs). Було створено датасети з дублюючими записами з сирих даних (датасет DUPLICATES (рис. 5.7)), та датасет з рекомендаціями щодо того які записи мають більше інформації (датасет RECOMMEND (рис. 5.8)).



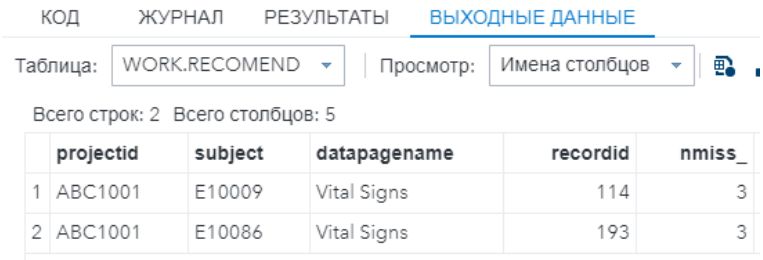
КОД ЖУРНАЛ РЕЗУЛЬТАТЫ ВЫХОДНЫЕ ДАННЫЕ

Таблица: WORK.DUPLICATES Просмотр: Имена столбцов

Всего строк: 16 Всего столбцов: 6

	projectid	subject	recordid	foldername	folderid	datapagename
1	ABC1001	E10005	5	Screening	SCRN	Demographics
2	ABC1001	E10005	7	Screening	SCRN	Demographics
3	ABC1001	E10009	9	Screening	SCRN	Demographics
4	ABC1001	E10009	11	Screening	SCRN	Demographics
5	ABC1001	E10017	17	Screening	SCRN	Demographics
6	ABC1001	E10017	19	Screening	SCRN	Demographics
7	ABC1001	E10086	86	Screening	SCRN	Demographics
8	ABC1001	E10086	88	Screening	SCRN	Demographics
9	ABC1001	E10005	5	Screening	SCRN	Vital Signs
10	ABC1001	E10005	6	Screening	SCRN	Vital Signs
11	ABC1001	E10009	113	Visit 1	VISIT 1	Vital Signs
12	ABC1001	E10009	114	Visit 1	VISIT 1	Vital Signs
13	ABC1001	E10017	19	Screening	SCRN	Vital Signs
14	ABC1001	E10017	20	Screening	SCRN	Vital Signs
15	ABC1001	E10086	192	Visit 1	VISIT 1	Vital Signs
16	ABC1001	E10086	193	Visit 1	VISIT 1	Vital Signs

Рисунок 5.7 – Датасет DUPLICATES



КОД ЖУРНАЛ РЕЗУЛЬТАТЫ ВЫХОДНЫЕ ДАННЫЕ

Таблица: WORK.RECOMEND Просмотр: Имена столбцов

Всего строк: 2 Всего столбцов: 5

	projectid	subject	datapagename	recordid	nmiss_
1	ABC1001	E10009	Vital Signs	114	3
2	ABC1001	E10086	Vital Signs	193	3

Рисунок 5.8 – Датасет RECOMEND

Після виконання всіх перевірок, було створено XLSX звіт REPORT, який містить окремі вкладки з загальною інформацією про знайдені помилки, та окрему вкладку для кожного датасету (рис. 5.9-рис. 5.11).

	Issue type	Data pages involved	Date issue found	Number of records with issues	Severity
2	Duplicates	DEMOG VS	01.12.2019	16	Serious
3	Unprinted characters	AE	01.12.2019	1	Mild
4	Missing key variables	AE CM	01.12.2019	4	Serious
5	Visits overlapping	LB PE	01.12.2019	1	Serious
6	Functional variables issues (ALS)	CM PE	01.12.2019	6	Mild
7					
8					
9					
10					

Summary | AE | CM | DEMOG | LB | PE | VS

Рисунок 5.9 – Загальні відомості про знайдені помилки у сирих даних

	Issue type	PROJECTID	SUBJECT	RECORDID	FOLDERNAME	FOLDERID	Number of missing variables
2	Duplicates	ABC1001	E10005	5	Screening	SCRN	
3	Duplicates	ABC1001	E10005	6	Screening	SCRN	
4	Duplicates	ABC1001	E10009	113	Visit 1	VISIT 1	
5	Duplicates	ABC1001	E10009	114	Visit 1	VISIT 1	
6	Duplicates	ABC1001	E10017	19	Screening	SCRN	
7	Duplicates	ABC1001	E10017	20	Screening	SCRN	
8	Duplicates	ABC1001	E10086	192	Visit 1	VISIT 1	
9	Duplicates	ABC1001	E10086	193	Visit 1	VISIT 1	
10	Recommended valuable records	ABC1001	E10009	114	Visit 1	VISIT 1	3
11	Recommended valuable records	ABC1001	E10086	193	Visit 1	VISIT 1	3
12							
13							
14							

Summary | AE | CM | DEMOG | EOS | EOT | VS

Рисунок 5.10 – Приклад знайдених помилок у датасеті VS

	Issue type	PROJECTID	SUBJECT	RECORDID	FOLDERNAME	FOLDERID	AETERM	REASACN	Comment
2	Missing key variables	ABC1001	E10005	5	Adverse Events	AE	...		Missing key variable AETER
3	Missing key variables	ABC1001	E10005	7	Adverse Events	AE	...		Missing key variable AETER
4	Missing key variables	ABC1001	E10009	9	Adverse Events	AE	...		Missing key variable AETER
5	Unprinted characters	ABC1001	E10009	1	Adverse Events	AE	HEADACHE	ANS ≥3.1	

Рисунок 5.11 – Приклад знайдених помилок у датасеті AE

5.2 Підсистема створення SDTM датасетів

5.2.1 Алгоритм підсистеми створення SDTM датасетів

Підсистема створення SDTM датасетів буде містити у собі головний макрос, з якого починається процес створення SDTM датасетів та який містить виклик та 5 макросів:

- модуль створення шаблону та завантаження атрибутів;
- модуль мапінгу загальних змінних;
- модуль мапінгу спеціальних змінних;
- модуль перевірки змінних на відповідність до CT;
- модуль перевірки журналу програми.

У головному макросі буде виконуватися перевірка наявності файлу специфікації, наявності у файлі специфікації відповідної вкладки з правилами мапінгу змінних та вкладки зі списком датасетів файлу. У разі їх відсутності, буде виведено повідомлення у журнал та буде завершено виконання програми. Якщо ж необхідні вкладки присутні, то буде по черзі викликатися кожен макрос у циклі для кожного SDTM датасет, який вказаний у вкладці датасетів у специфікації.

На рис. 5.12 та у додатку Е зображено блок-схему алгоритму основної програми підсистеми створення SDTM датасетів.

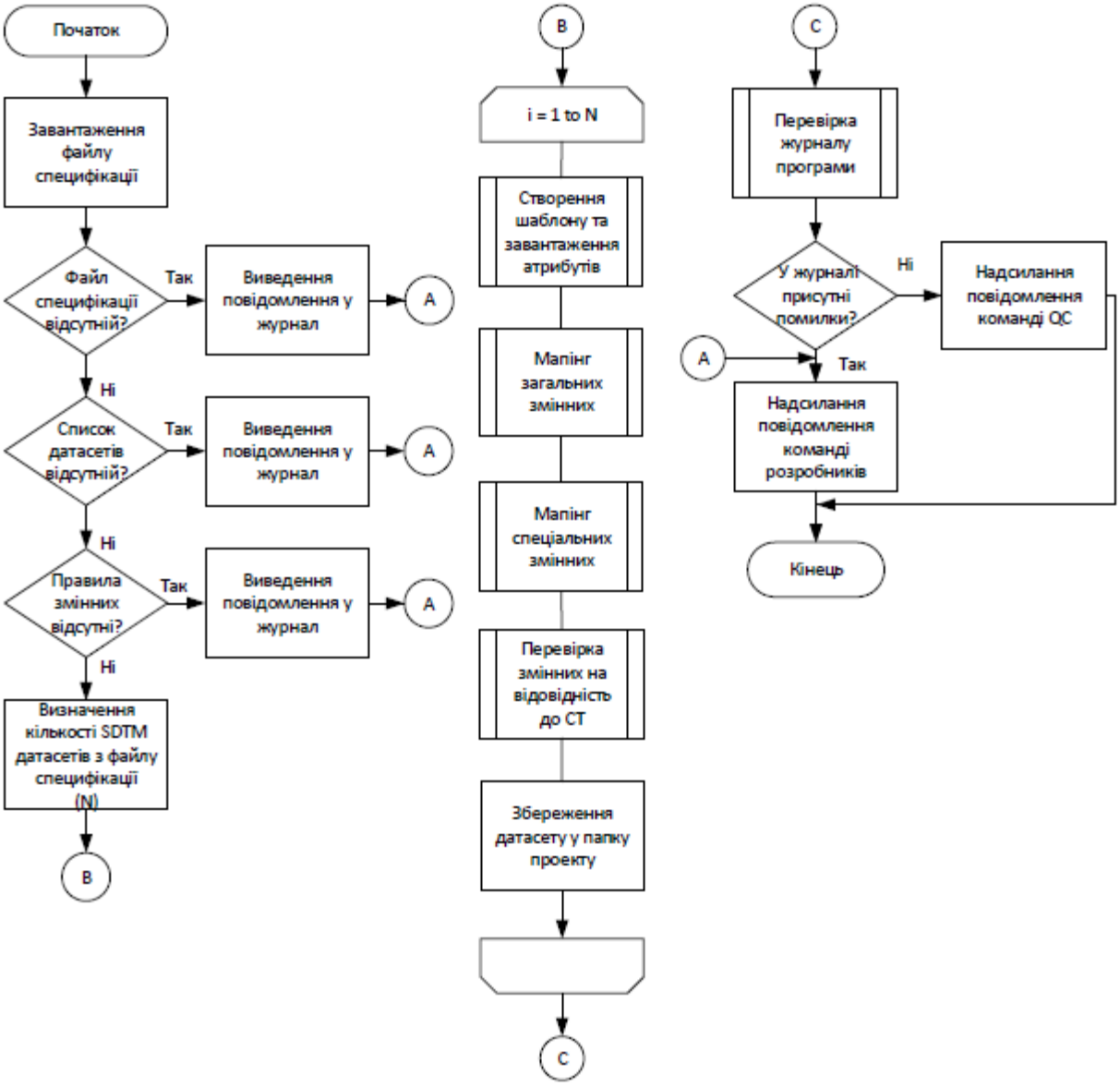


Рисунок 5.12 – Алгоритм основної програми підсистеми створення SDTM датасетів

5.2.2 Реалізація модулів підсистеми

Аналогічним чином, як і для модулю пошуку помилок у сирих даних, всі модулі у підсистемі створення SDTM датасетів розроблено у вигляді макросів.

На рис. 5.13 наведено частину лістингу коду, де відбувається перевірка наявності необхідних файлу специфікації та вкладок з датасетами та змінними. Файл специфікації є обов'язковим документом для кожного дослідження, а наявність вкладок з датасетами і змінними необхідна для створення define.xml файлу у застосунку Pinnacle21. У випадку якщо не існує специфікації або однієї з двох вкладок, виводяться повідомлення у журнал та по мітці %exit_main відбувається перехід до завершення виконання макросу.

```
libname spec xlsx "&project/docs/SDTM/Specifications/&specname";
/*check if spec file available*/
%let sqlobs = 0;
proc sql noprint;
   select *
       from sashelp.vstable
       where libname='SPEC';
quit;
%if  &sqlobs=0 %then %do;
       %put WARNING: Specification file was not found;
       %goto  exit_main;
%end;
/*check if Datasets sheet in spec is available*/
%let sqlobs = 0;
proc sql noprint;
   select *
       from sashelp.vstable
       where libname='SPEC' and memname = 'DATASETS';
quit;
%if  &sqlobs=0 %then %do;
       %put WARNING: Datasets sheet in specification file is not found;
       %goto  exit_main;
%end;
/*check if Variables sheet in spec is available*/
%let sqlobs = 0;
proc sql noprint;
   select *
       from sashelp.vstable
       where libname='SPEC' and memname = 'VARIABLES';
quit;
%if  &sqlobs=0 %then %do;
       %put WARNING: Variables sheet in specification file is not found;
       %goto  exit_main;
%end;
```

Рисунок 5.13 – Приклад лістингу коду

На рис. 5.13 представлено блок головного макросу, де відбувається виклик макросів у циклі для кожного датасету.

Файл специфікації є XLSX файлом, і він містить колонки (що у SAS стають змінними), які містять пробіли. На різних системах такі змінні можуть по різному відображатися у SAS, наприклад з нижнім підкресленням замість кожного пробілу, або злітно. Використання команда options validvarname=any; скидає налаштування щодо зчитування назв змінних XLSX файлів (за замовчуванням V7), та тепер у датасеті назви змінних такі самі як і у файлі специфікації, і в цьому випадку можна використати форму звернення до змінної як ‘Dataset Name’n.

Далі формуються макро змінні N – кількість циклів, dsnlist – список датасетів, який парситься у макросі за допомогою макро функції %scan, і починається виконання макро циклу для кожного датасету та викликаються відповідні макроси:

- %Attrib(dsname=) – макрос створення шаблону датасету, з усіма атрибутами, та створює формати для змінних на основі значень із вкладки Codelist (CL). Він використовує файл специфікації для отримання метаданих про датасет та за допомогою CALL EXECUTE макро рутини генерує код створення шаблону. В цьому макросі також присутні деякі перевірки файлу специфікації: перевірка наявності усіх атрибутів для усіх змінних, для датасетів. У випадку коли у специфікації знайдені помилки, то виведеться повідомлення у журнал.
- %CommonVar(dsname=) створює загальні змінні. Він також використовує файл специфікації та правила для змінних, походження яких або CRF , або Assigned, або Protocol. Правила для таких змінних представляють собою копіювання змінних із сирих даних, або присвоєння їм якогось постійного значення, як наприклад для змінної STUDYID значення завжди однакове. Також у разі необхідності, до змінної буде застосовано формат, вказаний у відповідній колонці.

- %DerivVar(dsname=) – макрос створює спеціальні змінні. Він ділить правила створення змінних на декілька підгруп, наприклад, змінні які будуть прийматизначаення максимальної або мінімальноїзначення з певного датасету, або змінні-категорії, значення яких залежить від значення в певній змінній у датасеті. Після цього починає виконувати у циклі створення змінної за визначеним правилом. Він також застосовує CALL EXECUTE макро рутину, яка дозволяє згенерувати код в рамках одного кроку DATA, таким чином, замість створення окремих макросів для кожної підгрупи змінних, використовуються лише кроки DATA.
- %CT_Check(dsname=) – макрос перевірки змінних на відповідність до CT. Він завантажує усі можливі значення із CL, які використовуються для змінних у поточному датасеті, та робить перевірку чи мають змінні значення, які неприсутні у CL.

Блок-схему алгоритму перевірки змінних на відповідність до CT наведено на рис. 5.14 та у додатку Е.

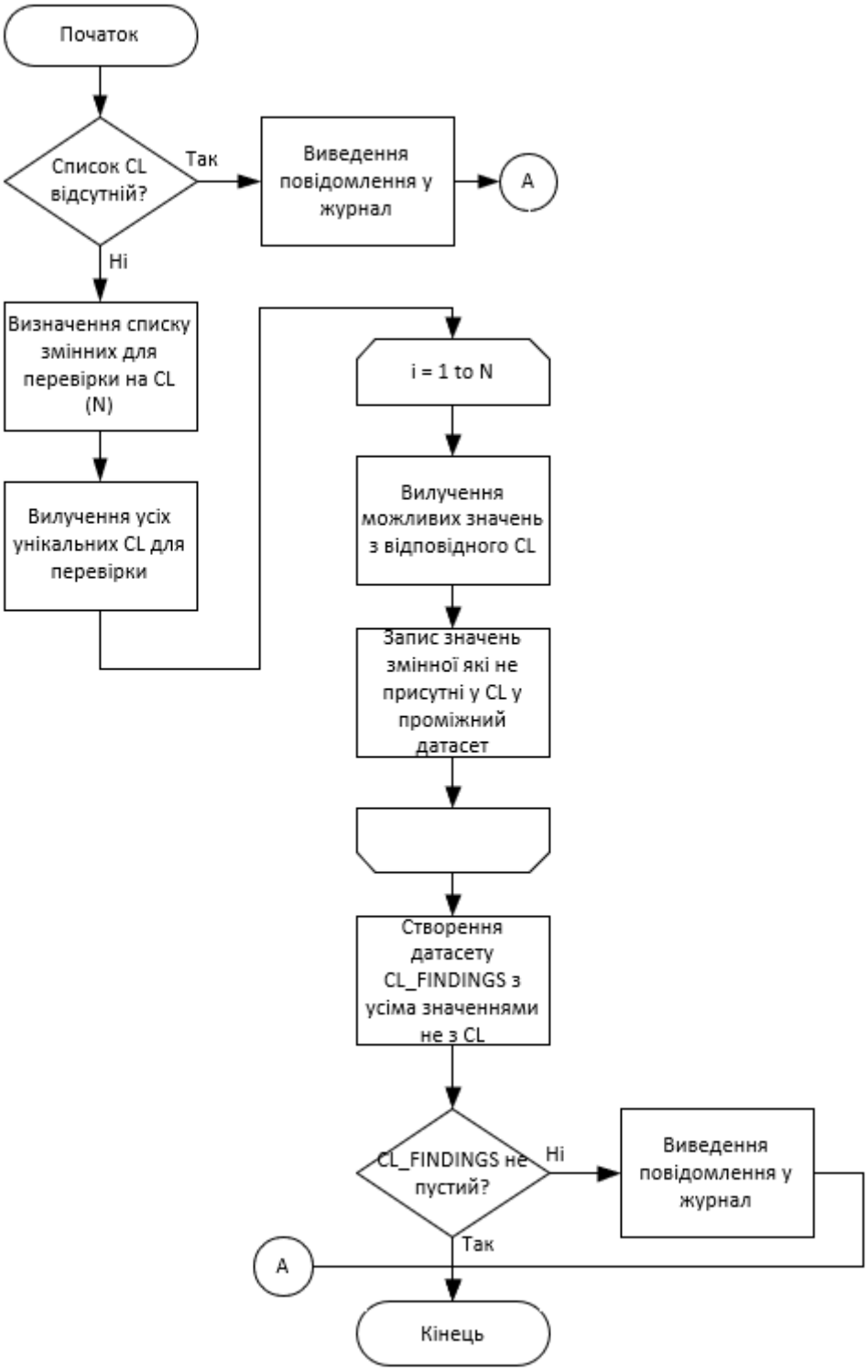


Рисунок 5.14 – Алгоритм перевірки змінних на відповідність до СТ

Після перевірки змінних, файл датасету зберігається у постійну бібліотеку, яка має посилання out (рис. 5.15).

```
/*Start of derivation*/
options validvarname=any;
proc sql noprint;
        select count(distinct 'Dataset Name'n) into :N
        from spec.datasets;

        select distinct 'Dataset Name'n into :dsnlist separated by ' '
        from spec.datasets;
 quit;

 %do i=1 %to &N;
       %Attrib(dsname=%scan(&dsnlist,&i));
       %CommonVar(dsname=%scan(&dsnlist,&i));
       %DerivVar(dsname=%scan(&dsnlist,&i));
       %CT_Check(dsname=%scan(&dsnlist,&i));
       /*Saving dataset to permanent directory*/
       data out.%scan(&dsnlist,&i);
            set final%scan(&dsnlist,&i);
       run;

       %quit_cycle:
 %end;
```

Рисунок 5.15 – Приклад лістингу коду

Після завершення циклу викликається макрос %CheckLog(), який перевіряє файли журналу для кожного датасету, та створює датасет findings_log. Якщо датасет містить записи, то відправляється лист до команди програмістів. Якщо журнал чистий, то надсилається лист до команді QC (рис. 5.16).

```
 /*Checking LOG file*/
 %CheckLog();

/*If no issues in LOG, then email is sent to QC team*/
%let sqlobs = 0;
proc sql noprint;
    select *
        from findings_log;
quit;
%if  &sqlobs=0 %then %do;
data _null_;
    FILE MailBox TO=('QCTeam@example.com>'
                    'NameSurname1@example.com>')
                     CC=(PL 'NameSurname2@example.com'
                     FROM='Biostat <Biostat@example.nl>'
                     SUBJECT="SDTMs ready for QC/ EOM";
run;
%end;
%else %if &sqlobs^=0 %then %do;
data _null_;
    FILE MailBox TO=('ProjectTeam@example.com>'
                     'NameSurname1@example.com>'
                     'PL <NameSurname2@example.com>')
                     FROM='Biostat <Biostat@example.nl>'
                     ATTACH="&project/sdtm/prod/outputs/findings_log.lst";
                     SUBJECT="Findings in LOG file. QC is not started./ EOM";
run;
%end;
```

Рисунок 5.16 – Приклад лістингу коду

Приклад виконання створення датасетів за файлом специфікації. Для прикладу приведено створення датасету DM (Demographics). Вкладка специфікації з правилами мапінгу змінних має вигляд представлений на рис. 5.17.

Order	Domain	Variable name	Label	Type	Length	Codelist	Format	Origin	Pages	Comme	Derivation Rule
1	DM	STUDYID	Study Identificator	char	10			CRF			DEMOG.PROJECTID
2	DM	USUBJID	Unique Subject Identificator	char	20			Derived			DEMOG.PROJECTID||'-'||DEMOG.SUBJECT
3	DM	SUBJECT	Subject Identificator	char	10			CRF			DEMOG.SUBJECT
4	DM	BRTHDTC	Birth date	char	10		ISO 8601	CRF			DEMOG.BRTHDAT_RAW
5	DM	ARM	Planned Arm	char	15	ARM		Derived			Part '||DEMOG.PART
6	DM	ARMCD	Planned Arm Code	char	2	ARM		CRF			DEMOG.PART
7	DM	ACTARM	Actual Arm	char	15	ARM		Derived			Part '||DEMOG.PART
8	DM	ACTARMCD	Actual Arm Code	char	2	ARM		CRF			DEMOG.PART
9	DM	RFSTDTC	Reference study start date	char	10		ISO 8601	Derived			IC.DSSTDAT@FIRST.SUBJECT
10	DM	RFXSTDTC	Reference treatment start date	char	10		ISO 8601	Derived			EXL.EXSTDAT@FIRST.SUBJECT
11	DM	RFXENDTC	Reference treatment end date	char	10		ISO 8601	Derived			MAX(EXL.EXSTDAT@LAST.SUBJECT, EXL.EXENDAT@LAST.SUBJECT)
12	DM	RFENDTC	Reference study end date	char	10		ISO 8601	Derived			MAX(EOT.DSSTDAT@LAST.SUBJECT, EOS.DSSTDAT@LAST.SUBJECT)
13	DM	RFICDT	Reference IC date	char	10		ISO 8601	Derived			IC.DSSTDAT@first.subject
14	DM	AGE	Age	num	8			CRF			DEMOG.AGE

Рисунок 5.17 – Приклад файлу специфікації

Процес мапінгу змінних розбито на два етапи – мапінг загальних змінних (у яких колонка Origin має значення CRF, Assigned або Protocol) та спеціальних змінних (у яких Origin має значення Derived). У першому випадку мапінг змінних складається з простого копіювання змінних з сирих даних, та застосування необхідного формату (якщо колонка Format не пуста). Якщо ж змінна спеціальна, та правила мапінгу діляться на декілька підтипів, логіка яких прописана у макросі %DerivVar(dsname=). Приклад створеного датасету DM зображено на рис. 5.18.

Всего строк: 99 Всего столбцов: 30 Строки 1-99

STUDYID	USUBJID	SUBJECT	BRTHDTC	ARM	ARMCD	ACTARM	ACTARMCD	RFSTDTC	RFXSTDTC	RFXENDTC	RFENDTC	RFICDT	AGE
ABC1001	ABC1001-E10001	E10001	1996-11-09	Part A1	A1	Part A1	A1	2002-05-02	2002-05-05	2002-11-05	2002-11-18	2002-05-02	.
ABC1001	ABC1001-E10002	E10002	1983-05-12	Part A0	A0	Part A0	A0	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	.
ABC1001	ABC1001-E10003	E10003	1983-05-12	Part A1	A1	Part A1	A1	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	10
ABC1001	ABC1001-E10004	E10004	1983-05-12	Part A0	A0	Part A0	A0	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	20
ABC1001	ABC1001-E10005	E10005	1983-05-12	Part A1	A1	Part A1	A1	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	30
ABC1001	ABC1001-E10006	E10006	1996-11-09	Part A0	A0	Part A0	A0	2002-05-02	2002-05-05	2002-11-05	2002-11-18	2002-05-02	40
ABC1001	ABC1001-E10007	E10007	1983-05-12	Part A1	A1	Part A1	A1	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	50
ABC1001	ABC1001-E10008	E10008	1983-05-12	Part A0	A0	Part A0	A0	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	60
ABC1001	ABC1001-E10009	E10009	1996-11-09	Part A1	A1	Part A1	A1	2002-05-02	2002-05-05	2002-11-05	2002-11-18	2002-05-02	70
ABC1001	ABC1001-E10010	E10010	1976-06-22	Part A0	A0	Part A0	A0	1981-12-13	1981-12-16	1982-06-18	1982-07-01	1981-12-13	70
ABC1001	ABC1001-E10011	E10011	1983-05-12	Part A1	A1	Part A1	A1	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	41
ABC1001	ABC1001-E10012	E10012	1996-11-09	Part A0	A0	Part A0	A0	2002-05-02	2002-05-05	2002-11-05	2002-11-18	2002-05-02	42
ABC1001	ABC1001-E10013	E10013	1983-05-12	Part A1	A1	Part A1	A1	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	43
ABC1001	ABC1001-E10014	E10014	1983-05-12	Part A0	A0	Part A0	A0	1988-11-01	1988-11-04	1989-05-07	1989-05-20	1988-11-01	44
ABC1001	ABC1001-E10015	E10015	1996-11-09	Part A1	A1	Part A1	A1	2002-05-02	2002-05-05	2002-11-05	2002-11-18	2002-05-02	45

Рисунок 5.18 – Приклад створеного датасету DM

На рис. 5.19 представлено вміст датасету SASHELP.VCOLUMN – системного файлу, що містить інформацію про всі атрибути у всіх датасетах у всіх бібліотеках.

memname	memtype	name	type	length	npos	varnum ▲	label
DM	DATA	STUDYID	char	10	40	1	Study Identificator
DM	DATA	USUBJID	char	20	50	2	Unique Subject Identificator
DM	DATA	SUBJECT	char	10	70	3	Subject Identificator
DM	DATA	BRTHDTC	char	10	80	4	Birth date
DM	DATA	ARM	char	15	90	5	Planned Arm
DM	DATA	ARMCD	char	2	105	6	Planned Arm Code
DM	DATA	ACTARM	char	15	107	7	Actual Arm
DM	DATA	ACTARMCD	char	2	122	8	Actual Arm Code
DM	DATA	RFSTDTC	char	10	124	9	Reference study start date
DM	DATA	RFXSTDTC	char	10	134	10	Reference treatment start date
DM	DATA	RFXENDTC	char	10	144	11	Reference treatment end date
DM	DATA	RFENDTC	char	10	154	12	Reference study end date
DM	DATA	RFICDT	char	10	164	13	Reference IC date
DM	DATA	AGE	num	8	0	14	Age

Рисунок 5.19 – Вміст SASHELP.VCOLUMN

Як видно з рис. 5.19, атрибути для датасету DM завантажилися коректно, згідно до правил специфікації, отже %Attrib(dsname=) макрос працює правильно.

Повідомлення у журналі про виконання макросу наведено на рис. 5.20.

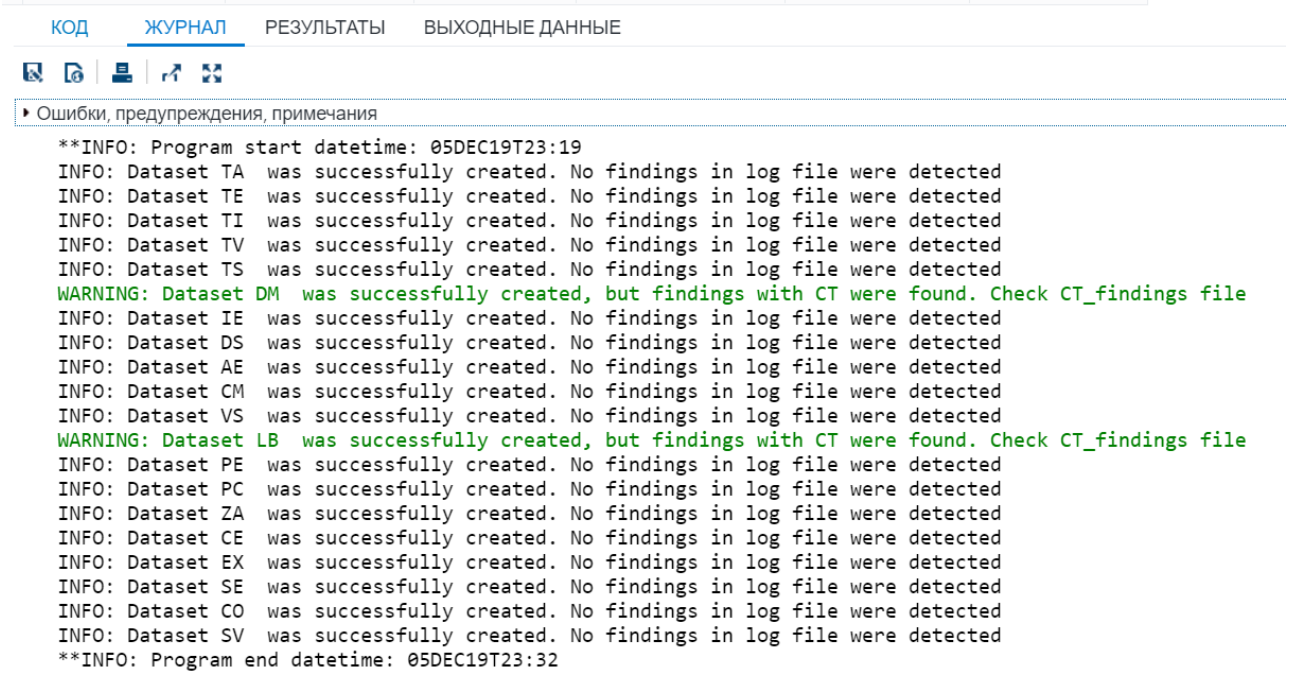


Рисунок 5.20 – Повідомлення, виведені у журнал програми

Як видно з рис 5.20, усі датасети створено успішно, та виведено повідомлення про те, що потрібно оновити Codelist для датасетів LB та DM.

5.3 Приклад налаштування batch файлів та графіку виконання програм

Для прикладу приведено налаштування отримання даних з БД. Відповідний batch файл містить інструкції щодо переходу у режим виконання SAS та запуску на виконання програми отримання даних, та збереження їх у CRF відповідних формах у форматі SAS7BDAT:

```
sasb -sv extract_dat.sas
```

Sasb – це команда виконання програми SAS у Putty у режимі batch.

Далі потрібно прив’язати цей файл до crontab:

```
crontab -e
00 6 * * * /sasdata/projectpath/common/extract_dat.bat
```

Таким чином, отримання даних з БД буде виконуватися кожного дня о 6.00.

Далі налаштуємо запуск на виконання модулю пошуку помилок. Цей модуль виконується, якщо дата оновлення даних співпадає з поточною датою, тому можемо вказати час виконання модулю, наприклад 8.00:

```
crontab -e
00 8 * * * /sasdata/projectpath/common/check_data.bat
```

Крім того за допомогою команди SAS X, можна в тексті check_data.sas (програма макроса main) вказати команду початку виконання sdtm_auto.sas – програми створення SDTM датасетів.

```
x ‘sasb -sv sdtm_auto.sas’
```

Ця команда буде виконуватися останньою, тому гарантовано процес створення SDTM датасетів почнеться після виконання модулю пошуку помилок.

Далі потрібно налаштувати запуск на виконання створення ADaM датасетів. Вони повинні почати створюватися після створення SDTM, навіть якщо процес валідації незавершений, адже це окремий процес. Тому аналогічно, ми можемо додати до main макросу підсистеми створення SDTM команду:

x ‘run_all_adam.bat’

run_all_adam.bat містить інструкції щодо запуску усіх програм ADaM датасетів.

Приклад вмісту run_all_adam.bat на рис. 5.21.

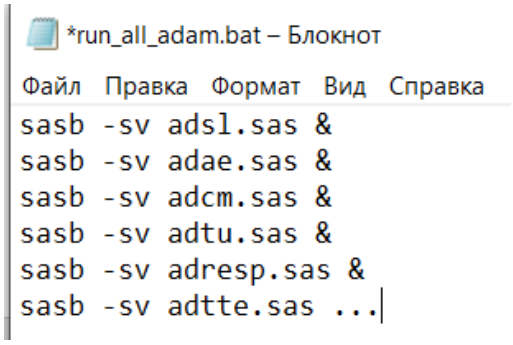


Рисунок 5.21 – Приклад run_all_adam.bat

Та, аналогічним чином, потрібно додати у run_all_adam.bat виклик batch файлу створення TLF:

x ‘run_all_tlf.bat’

Програми валідації SDTM будуть запускатися у разі успішного створення SDTM – команді QC буде відправлено електронний лист з інформацією. Так само для ADaM датасетів процес QC повинен бути виконаний тільки у разі успішного створення ADaM та чистого файлу журналу. Тому у run_all_adam.bat та run_all_ tlf.bat повинні бути додані аналогічні інструкції щодо надсилання електронного листа команді QC.

Таким чином, налаштовано графік отримання даних з БД, запуск модуля пошуку помилок у сирих даних та послідовний запуск підсистеми створення SDTM, ADaM та TLF звітів.

5.4 Розгортання системи

Розгортання системи було обрано на основі Grid Computing. Цей підхід надає наступні переваги:

- стандартизація та підтримка декількох середовищ;
- спрощена адміністративна підтримка;
- кращі засоби для аналізу та обробки даних;
- централізоване управління;
- масштабованість.

Діаграма розгортання представлена на рис. 5.22 та у додатку Є.

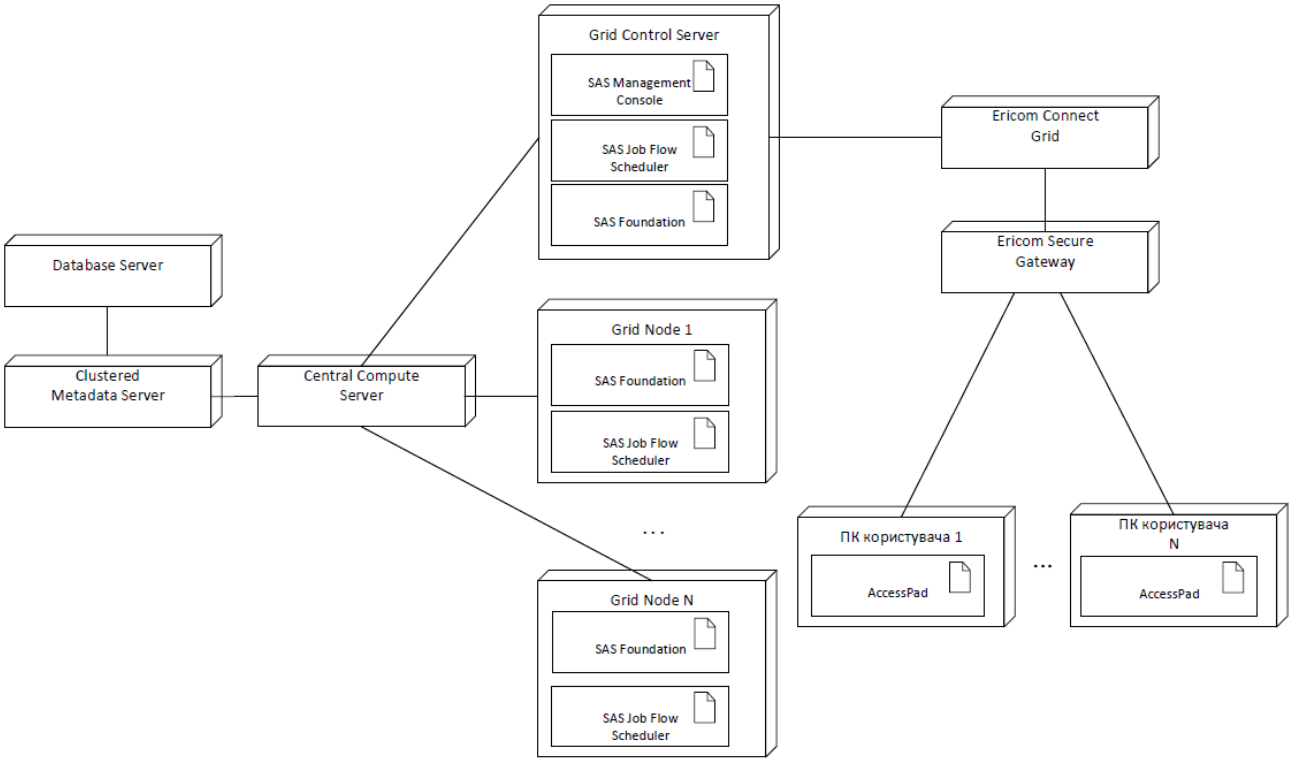


Рисунок 5.22 – Діаграма розгортання

SAS 9.4 надає можливість реалізації кластера серверів метаданих. Ця функція забезпечує високу доступність сервера метаданих, що є основним компонентом

інфраструктури SAS. Кластеризація забезпечує те, що сервер продовжує працювати, якщо сервер хост-сервера вийшов з ладу.

Кластер серверів метаданих – це група з трьох або більше машин (вузлів) хостів, які були налаштовані як однакові сервери метаданих. Кожен вузол запускає власний серверний процес і має власну інформацію про конфігурацію сервера, журнальний файл та копію наборів даних сховища. Крім того, кожен вузол зберігає повну копію пам'яті сховища метаданих. Вузли працюють разом, ніби вони були єдиним сервером метаданих [30].

Клієнтські програми та користувачі взаємодіють із кластером так само, як і з сервером метаданих, який не кластеризований. Процес збалансування навантаження автоматично розподіляє роботу між вузлами. Якщо вузол перестає працювати, сервер продовжує бути доступним, використовуючи інші вузли.

SAS Grid Server служить мостом між програмами SAS та середовищем grid. Це дозволяє програмі розпізнавати grid та надсилати їй завдання (jobs).

Grid сервер насправді є логічним сервером, який є компонентом SAS Application Server. Grid складається з таких вузлів:

- Grid Control Server – машина, яка розподіляє завдання машинам в мережі. Grid Control Server також може виконувати роботу, виділену для мережі;
- один або декілька вузлів grid – машина або машини, які виконують частину роботи, виділену на grid [31].

Grid Control Server містить такі компоненти:

- SAS Management Console;
- SAS Job Flow Scheduler;
- SAS Foundation (включає в себе Base SAS, DATA Step Batch Server та SAS / CONNECT) [32].

SAS Management Console дозволяє керувати визначеннями серверів, бібліотек, користувачів, груп, ролей, контролем доступу до ресурсів, сховищем метаданих та графіками завдань.

SAS Management Console працює, створюючи та підтримуючи визначення метаданих для кожного обчислювального ресурсу чи елемента керування. Ці визначення метаданих зберігаються у сховищі на сервері метаданих SAS, де вони доступні для використання іншими програмами [33].

SAS Job Flow Scheduler використовує три основні компоненти: службу SAS Job Flow Scheduler, SAS Job Flow Scheduler Orchestrator та SAS Job Flow Scheduler Trigger.

Коли потік подається на виконання, служба SAS Job Flow Scheduler обробляє запит щодо планування потоку. Служба використовує параметри конфігурації та параметри потоку, щоб визначити, яку інформацію надсилати до SAS Job Flow Scheduler Orchestrator та SAS Job Flow Scheduler Trigger.

Для запуску потоку послуга SAS Job Flow Scheduler Trigger використовує послуги в операційній системі. У UNIX він використовує crontab. У Windows використовується Планувальник завдань Windows.

Після запуску потоку SAS Job Flow Scheduler Orchestrator обробляє виконання завдань у потоці. Він надсилає завдання до grid, якщо grid доступна, або в операційну систему, якщо grid недоступна. Потім він повертає результати виконаних завдань [34].

Вузли Grid містить такі компоненти:

- SAS Job Flow Scheduler;
- SAS Foundation.

SAS Compute Server дозволяє користувачам подавати програми SAS і збережені процедури у вигляді завдань для обробки з використанням мови SAS. Для кожного завдання, яке обробляється, обчислювальний сервер записує повідомлення в журнал SAS. Завдання дає результати, коли створюється вихід [35].

Ericom Connect – це рішення віддаленого доступу/програм, вироблене компанією Ericom Software, що забезпечує безпечний, централізований доступ до програм, що працюють на системах Microsoft Windows та Linux. Він містить компоненти Ericom Connect Grid та Ericom Secure Gateway – шлюз SSL, який забезпечує доступ до внутрішніх ресурсів, що працюють на хостах RDP (Remote Desktop Protocol) [36].

У якості клієнта для ПК Ericom Connect використовує AccessPad.

Висновки до розділу

Розроблене рішення дозволило покращити процес обробки даних медичних досліджень, а саме, було налаштовано графік отримання даних з БД, розроблено модуль пошуку помилок у сирих даних та підсистема створення SDTM датасетів. Таким чином, втручання користувачів у процес значно зменшилося – програмісти залучені до розробки аналітичних датасетів та створення статистичних звітів, а система сама робить вилучення даних (а отже програмісти гарантовано працюють з оновленими даними), пошук помилок у сирих даних та створює SDTM.

Розроблені модулі виконанні у вигляді SAS макросів, що дозволило зробити їх більш гнучкими та стійкими до виникнення помилок. Налаштування графіку виконання програм було виконано за допомогою утиліти cron. Створено batch файли для отримання даних з БД та модулю пошуку помилок у сирих даних, з прив’язкою до crontab. З огляду на сувору послідовність виконання кожного етапу обробки даних, налаштування послідовності створення виконано безпосередньо у файлах відповідних програм. Також створено окремі batch файли для створення ADaM датасетів та TLF звітів. Так як процес QC має відбуватися окремо від процесу створення production датасетів, то відповідно автоматизувати запуск QC програм недоцільно.

6 ТЕСТУВАННЯ СИСТЕМИ

6.1 Огляд фреймворку для модульного тестування SASUnit

Для виконання модульного тестування було обрано фреймворк SASUnit.

SASUnit – це фреймворк тестування для програм SAS і макросів SAS, і сам реалізований як SAS макро. Він контролює виконання тестових сценаріїв та створює тестову документацію в форматі HTML. Тестові сценарії – це програми SAS. Вони використовують набір SAS макросів для контролю та виконання тестів. Статичні об’єкти даних (набори даних SAS або зовнішні файли) можуть знаходитися в спеціальній папці в межах тестового середовища. Вони також можуть бути сформовані як частина тестового сценарію із заявами програмування SAS (кроки даних, PROC SQL, %LET заяви тощо).

Модулями, що перевіряються, є макроси SAS (або звичайні SAS програми), які параметризуються та виконуються як частина кожного тестового випадку. Набір %assert-макросів дозволяє перевірити макро змінні значення, вміст набору даних, наявність ODS звітів та наявність або відсутність повідомлень у журналі.

Всі необхідні сценарії тестування тестового набору виконуються у макросі %runSASUnit. Кожен тестовий сценарій працює у своєму власному сеансі SAS, щоб уникнути побічних ефектів [37].

Результати тестів записуються у тестове сховище, з яких потім можна створити тестові звіти.

Приклад організації модульного тестування у SASUnit наведено на рис. 6.1.

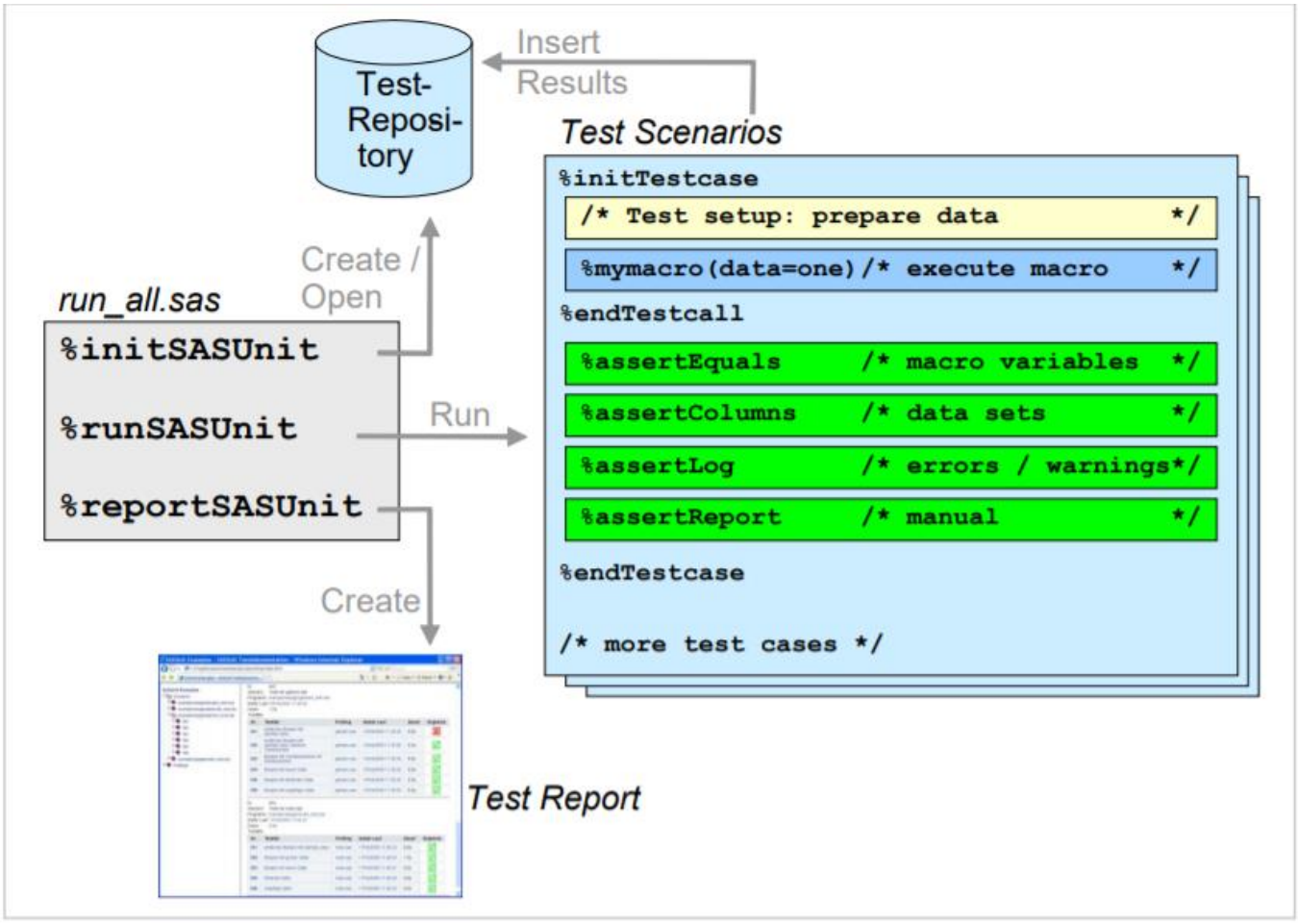


Рисунок 6.1 – Приклад організації модульного тестування у SASUnit

SASUnit контролюється програмою, яку зазвичай називають run_all.sas. Вона містить виклик трьох макросів:

- %initSASUnit створює або відкриває тестове сховище;
- %runSASUnit запускає кожен тестовий сценарій у своєму сеансі SAS;
- %reportSASUnit створює звіт із тестового сховища.

Макрос %initSASUnit відкриває тестове сховище або створює його, якщо воно ще не існує. Тестове сховище складається з набору SAS датасетів з метаданими для тестів, сценаріїв тестування, а також журнали SAS та різні результати виконань тестів.

Макрос %runSASUnit отримує специфікацію файлу як параметр, який може бути єдиним тестовим сценарієм або групою сценаріїв. Для кожного виконаного сценарію загальний результат консолідується з результатів тестового випадку.

Макрос %reportSASUnit створює звіт про тест із вмісту тестового сховища. Звіт про випробування структурується за тестовими сценаріями, тестовими прикладами, assert-макросами та модулі, що перевіряються (рис. 6.2).

```
/* open test repository or create when needed */
%initSASUnit(
   i_root       = c:\projekte\sasunit /* root path, all other paths can then be
                                         relative paths */
  ,io_target    = example\doc\sasunit /* Output of SASUnit: test repository, logs,
                                         results, reports */
  ,i_overwrite  = 0                   /* set to 1 to force all test scenarios to be run,
                                         else only changed scenarios or scenarios with
                                         changed unit under test will be run*/
  ,i_project    = SASUnit Examples    /* Name of project, for report */
  ,i_sasunit    = saspgm\sasunit      /* SASUnit macro library */
  ,i_sasautos   = example\saspgm      /* Search for units under test here */
  ,i_testdata   = example\dat         /* test data, libref testdata */
  ,i_refdata    = example\dat         /* reference data, libref refdata */
)

/* Run specified test scenarios. There can be more than one call to runSASUnit */
%runSASUnit(i_source = example\saspgm\*_test.sas)

/* Create or recreate HTML pages for report where needed */
%reportSASUnit()
```

Рисунок 6.2 – Приклад організації модульного тестування у SASUnit

Кожен тестовий сценарій містить ряд тестових випадків, структурованих таким чином (рис. 6.3):

- %initTestcase ініціалізує тестовий випадок. Назва модулю, що перевіряється (i_object), та i_desc (опис тесту) використовується в тестовому звіті. У цьому макросі журнал SAS перенаправляється в окремий файл. Запускається таймер для перевірки працездатності. Тестова установка готує дані тесту за необхідності. Викликається блок, що перевіряється.
- %endTestcall – необов'язковий макрос, який закінчує виклик тестового блоку, перенаправляє журнал SAS і записує час виконання в тестове сховище.

- assert-макроси перевіряють правильність функціональності та записують результати у тестовий сховище;
- %endTestcase необов'язковий макрос, що закінчує тестовий випадок і консолідує результати виконання тестів від assert-макросів для тестового випадку.

```
%initTestcase(i_object=nobs.sas,
              i_desc=simple example with sashelp.class)
%let nobs=%nobs(sashelp.class);
%endTestcall()
%assertEquals(i_actual=&nobs, i_expected=19,
              i_desc=number of observations in sashelp.class)
%endTestcase()
```

Рисунок 6.3 – Послідовність виклику макросів у %runSASUnit

Assert-макроси завжди порівнюють очікувану величину з фактичною величиною, отриманою при виклику одиниці, що перевіряється. Для кожного assert-макросу у звіті існує окремий рядок з інформацією про результат, де порівнюється очікуване значення та фактичне значення, а результат оцінюється як Passed чи Failed.

Основні assert-макроси, що використовуються у SASUnit:

- %assertEquals перевіряє значення макро змінної;
- %assertColumns перевіряє набори даних SAS або підмножину стовпців у датасетах SAS. Цей макрос базується на процедурі PROC COMPARE , та містить деякі допоміжні параметри для порівнянна датасетів;
- %assertLibrary порівнює всі набори даних або підмножину наборів даних у бібліотеках SAS. Звіт містить списки каталогів та бібліотек, що перевіряються;
- %assertReport перевіряє, чи був створений або оновлений зовнішній файл (як правило, генерований ODS). Звіт про випробування містить значення 1 у колонці "Actual" якщо файл існує;

- %assertLog сканує журнал SAS на наявність помилок та повідомлень. Зазвичай він використовується для того, щоб підтвердити, що немає жодних і жодних попереджень, але він не виводить інформацію про деякі важливі повідомлення, які є неприпустимі для медичних досліджень, тому для перевірки усіх файлів журналу його не доцільно;
- %assertLogMsg є похідним від %assertLog, і сканує журнал SAS на наявність чи відсутність певного повідомлення.

6.2 Результати модульного тестування

Приклад проведення модульного тестування наведено для модулю пошуку помилок у сирих даних (%check_data макрос) та підсистеми створення SDTM (%sdtm_auto макрос).

На рис. 6.4 предоставлено організацію проведення модульного тестування для %check_data.

```
%initTestCase(i_object=check_data.sas,
                i_desc = testing check_data m acro);
%endTestCall();
/*Start of test cases*/
%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=N, DUPFL=Y, UNPRFL=N, CLASSFL = N, DATFL = N),
                i_expected=8, i_desc = check number of duplicates in raw data);
%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=N, DUPFL=Y, UNPRFL=N, CLASSFL = N, DATFL = N),
                i_expected=2, i_desc = check number of recommendations on valuable duplicate records in raw data);

%assertEquals(i_actual = &nobs,  paramsset=%str(ALSFL=N, DUPFL=N, UNPRFL=N, CLASSFL = Y, DATFL = N),
                i_expected=1, i_desc = check number of records with visits overlapping);
%assertEquals(i_actual = &nobs,  paramsset=%str(ALSFL=N, DUPFL=N, UNPRFL=N, CLASSFL = Y, DATFL = N),
                i_expected=0, i_desc = check number of records with result units not as per standard);

%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=N, DUPFL=N, UNPRFL=Y, CLASSFL = N, DATFL = N),
                i_expected=1, i_desc = check special characters in variables);

%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=Y, DUPFL=N, UNPRFL=N, CLASSFL=Y, DATFL = N),
                i_expected=4, i_desc = check key variables missing);
%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=Y, DUPFL=N, UNPRFL=N, CLASSFL=N, DATFL = N),
                i_expected=0, i_desc = check datasets with some variables having type different from ALS);
%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=Y, DUPFL=N, UNPRFL=N, CLASSFL=N, DATFL = N),
                i_expected=0, i_desc = check datasets with some variables absent);

%assertEquals(i_actual = &nobs, paramsset=%str(ALSFL=N, DUPFL=N, UNPRFL=N, CLASSFL=N, DATFL = Y),
                i_expected=0, i_desc = check datasets with incorrect date variables);
%assertReport(i_actual=%str('&project/doc/reports/report.xlsx'),
              i_expected==%str('&project/doc/reports/report.xlsx'),
            i_desc = check if REPORT file created successfully);
/*End of test cases*/
%endTestcase();
```

Рисунок 6.4 – Організація модульного тестування %check_data

%initTestcase ініціалізує тестовий випадок – запускає макрос та завантажує результати (датасети та звіти) до поточної сесії. %endTestcall потрібний для перенаправлення журналу програми SAS, для подальшого генерування звіту. далі починається виклик assert-макросів для перевірки різних тестових випадків. %endTestcase об'єднає результати усіх тестових випадків, для подальшого генерування звіту. Вміст тестового звіту наведено на рис. 7.7.

Аналогічним чином організовано проведення модульного тестування для %sdtm_auto (рис. 6.5).

```
%initTestCase(i_object=sdtm_auto.sas,
                i_desc = testing sdtm_auto m acro);
%endTestCall();
/*Start of test cases*/
%assertEquals(i_actual = &nsets,
                i_expected=19, i_desc = check number datasets created);
%assertEquals(i_actual = &nobs,
                i_expected=3, i_desc = check number datasets having issues in LOG);
%assertEquals(i_actual = &nobs,
                i_expected=2, i_desc = check number datasets having issues with CT);
%do i = 1 %to %sysfunc(countw(&dsnlist));
%assertReport(i_actual=%str('&project/prod/data/%scan(&dsnlist, &i).SAS7BDAT'),
              i_expected=%str('&project/prod/data/%scan(&dsnlist, &i).SAS7BDAT'),
              i_desc = check if %scan(&dsnlist, &i) SDTM dataset was created successfully and saved in SAS7BDAT) format);
%end;
%assertReport(i_actual=%str('&project/prod/log/chklog.log'),
              i_expected=%str('&project/prod/log/chklog.log'),
              i_desc = check if LOG file created successfully);
%endTestcase();
```

Рисунок 6.5 – Організація модульного тестування для %sdtm_auto

Вміст тестового звіту для %sdtm_auto наведено на рис. 6.8.

Згенерований звіт містить загальну інформацію про пройдені тестові сценарії (рис. 6.6). Номер сценарію та текст сценарію є гіперпосиланнями до звітів цих тестових сценаріїв.

Test scenario overview

No.	Test scenario	Program	Last run date	Duration	Result
001	Test for check_data macro	&project/prod/program/check_data.sas	02DEC2019T13:25	7.0s	*Passed*
002	Test for sdtm_auto macro	&project/prod/program/sdtm_auto.sas	02DEC2019T13:25	12.3s	*Passed*

Рисунок 6.6 – Загальна інформація про виконані сценарії тестування

Results for testing check_data macro

No.	Assertion	Description	Expected	Actual	Result
001	assertEquals	check number of duplicates in raw data	8	8	*Passed*
002	assertEquals	check number of recommendations on valuable duplicate records in raw data	2	2	*Passed*
003	assertEquals	check number of records with visits overlapping	1	1	*Passed*
004	assertEquals	check number of records with result units not as per standard	0	0	*Passed*
005	assertEquals	check special characters in variables	1	1	*Passed*
006	assertEquals	check key variables missing	4	4	*Passed*
007	assertEquals	check datasets with some variables having type different from ALS	0	0	*Passed*
008	assertEquals	check datasets with some variables absent	0	0	*Passed*
009	assertEquals	check datasets with incorrect date variables	0	0	*Passed*
010	assertReport	check if REPORT file created successfully	1	1	*Passed*

Рисунок 6.7 – Результати виконання модульного тестування %check_data

Results for testing sdtm_auto macro

No.	Assertion	Description	Expected	Actual	Result
001	assertEquals	check number datasets created	19	19	*Passed*
002	assertEquals	check number datasets having issues in LOG	3	3	*Passed*
003	assertEquals	check number datasets having issues with CT	2	2	*Passed*
004	assertReport	check if TA SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
005	assertReport	check if TE SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
006	assertReport	check if TI SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
007	assertReport	check if TV SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
008	assertReport	check if TS SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
009	assertReport	check if DM SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
010	assertReport	check if IE SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
011	assertReport	check if DS SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
012	assertReport	check if AE SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
013	assertReport	check if CM SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
014	assertReport	check if VS SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
015	assertReport	check if LB SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
016	assertReport	check if PE SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
017	assertReport	check if PC SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
018	assertReport	check if ZA SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
019	assertReport	check if CE SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
020	assertReport	check if EX SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
021	assertReport	check if SE SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
022	assertReport	check if CO SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
023	assertReport	check if SV SDTM dataset was created successfully and saved in SAS7BDAT format	1	1	*Passed*
024	assertReport	check if LOG file created successfully	1	1	*Passed*

Рисунок 6.8 – Результати виконання модульного тестування %sdtm_auto

Усі тестові випадки були пройдені успішно, отже модуль пошуку помилок у сирих даних (%check_data макрос) та підсистема створення SDTM (%sdtm_auto макрос) працюють правильно.

Тестування проведено на тестових даних та тестовій версії специфікації та файлу ALS. Для перевірки стійкості системи до неправильно організованих файлу специфікації та ALS, відсутності даних, також були проведені тестові сценарії з модифікованими файлами.

В результаті проведеного тестування, було отримано покриття тестами на 89%. всього було виконано 2 тестових сценаріїв для модулю пошуку помилок у сирих даних (%check_data макрос) та підсистеми створення SDTM (%sdtm_auto макрос), при цьому були по різному організовані файли специфікації та ALS (відсутні файли, відсутні необхідн вкладки у файлах). Загалом було пройдено 34 тестових випадки, та виконано 204 assert-макроси. Результати тестування проекту представлені на рис. 6.9.

SASUnit Test Documentation

Properties of the test suite

Name of project	&project	SASUnit results
Root directory	&project_path	C:\SAS\SASproject
Path to test repository	&target	C:\SAS\SASproject\doc\SASUnit
Program libraries (macro autocall paths)	&sasautos	C:\SAS\SASproject\macro
Folder for test data	&testdata	C:\SAS\SASproject\data
Folder for reference data	&refdata	C:\SAS\SASproject\doc
Path to SASUnit macros	&sasunit_m	C:\SAS\SASproject\doc\SASUnit\macro
SAS log of reporting job	&log_path	C:\SAS\SASproject\doc\SASUnit\run_all.log
Coverage		89%
Number of test scenarios		2
Number of test cases		34
Number of assertions		204

Рисунок 6.9 – Загальна інформація про проходження тестів

Висновки до розділу

Проведено модульне тестування модулю пошуку помилок у сирих даних та підсистеми створення SDTM датасетів за допомогою фреймворку SASUnit. Цей фреймворк призначений для тестування SAS макросів, та містить набір необхідний набір вбудованих макросів для ініціалізації тестових сценаріїв, тестових випадків та оформлення тестової документації.

Результати тестів довели коректність роботи системи, зокрема правильність виконання макросів пошуку помилок у сирих даних, створення SDTM датасетів, також було перевірено час виконання програм для оцінки продуктивності. У результаті тестування було покрито 89% програми. Цей відсоток означає що більшість шляхів виконання програми були протестовані, але залишилися шляхи які не були перевірені. З огляду на особливість синтаксису програм на SAS (ітеративне виконання DATA кроків та наявність внутрішніх шляхів) та на велику кількість модулів у системі, можна стверджувати, що відсоток покриття тестами задовільний. Отримані результати надають можливість стверджувати, що система виконує поставлену задачу.

Загалом було пройдено 2 сценарії тестування, 34 тестових випадки, та виконано 204 assert-макроси.

7 СТАРТАП-ПРОЕКТ

Метою даного розділу є проведення маркетингового аналізу стартап-проекту для визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

7.1 Опис ідеї проекту

Опис ідеї стартап-проекту наведено у таблиці 7.1.

Таблиця 7.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. Обробка даних медичних досліджень	Дозволяє автоматизувати процес обробки даних медичних досліджень, а саме отримання даних з БД, пошук помилок у сирих даних та процес створення SDTM датасетів,
	2. Обробка даних неклінічних даних медичних досліджень	Дозволяє автоматизувати процес обробки даних медичних досліджень, а саме отримання даних з БД та пошук помилок у сирих даних

У таблиці 7.2 наведено сильні, слабкі та нейтральні характеристики ідеї проекту.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Порівняння потенційних товарів/концепцій конкурентів				W	N	S
		МП	К1	К2	К3			
1	Форма використання	Веб	Веб	Веб + мобільний клієнт	Веб + мобільний клієнт			+
2	Собівартість	Низька	Низька	Низька	Висока		+	
3	Крос-платформеність	-	+	-	+	+		

МП – мій проект; К1, К2, К3 – конкурент №1, №2, №3; W – слабкі сторони; N – нейтральні сторони; S – сильні сторони;

7.2 Технологічний аудит ідеї проекту

Технологічну здійсненність ідей проекту наведено у таблиці 7.3.

Таблиця 7.3 – Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
		SAS	Наявна	Безкоштовна, доступна
		Python	Наявна	Безкоштовна, доступна
		R	Наявна	Безкоштовна, доступна
		SSH	Наявна	Безкоштовна, доступна
		Telnet	Наявна	Безкоштовна, доступна
Обрана технологія реалізації ідеї проекту: Розробка системи – SAS тому що члени команди мають досвід роботи а також через поширеність технології у сфері медичних досліджень, протокол взаємодії – SSH, бо надає вищий рівень безпеки				

7.3 Аналіз ринкових можливостей запуску стартап-проекту

Попередню характеристику потенційного ринку стартап-проекту наведено у таблиці 7.4.

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	5000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Характеристику потенційних клієнтів стартап-проекту наведено у таблиці 7.5.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проекту.

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Необхідність спрощення процесу обробки даних медичних досліджень	Фармакологічні компанії, які підтримують подання звітів до FDA/PMDA, які прагнуть оптимізувати процес обробки даних медичних досліджень	Різні види досліджень, різні набори даних	Можливість інтеграції з SAS Enterprise Guide, підтримка форматів даних SAS7BDAT, можливість застосування до різних типів досліджень

Фактори загроз наведено у таблиці 7.6.

Таблиця 7.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція	Вихід на ринок великої компанії	1) Вихід з ринку 2) Запропонувати великій компанії поглинути себе 3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачі в	Користувачам необхідне програмне забезпечення з іншим функціоналом	1) Передбачити можливість додавання нового функціоналу до створюваного ПЗ

Фактори можливостей наведено у таблиці 7.7.

Таблиця 7.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зростання можливостей потенційних покупців	Ріст зацікавленості до продукту серед інших груп користувачів (наприклад фармакологічних компаній, які не підтримують стандарти CDISC)	Розширити функціонал системи, для задоволення потреб нової групи користувачів
2.	Зниження довіри до конкурента 2	У ПЗ конкурента №2 нещодавно була знайдена вразливість, яка впливає на якість пошуку помилок у великих даних	При виході на ринок звертати увагу покупців на якість пошуку помилок у сирих даних

У таблиці 7.8 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкуренто-спроможною)
1. Вказати тип конкуренції - досконала	Існує 3 фірми-конкурентки на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби - міжнародний	Компанії з інших країн	Додати документацію до ПЗ на інших мовах майбутньому вийти на міжнародний ринок
3. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме – програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
4. За характером конкурентних переваг - нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти

Продовження таблиці 7.8

5. За інтенсивністю - марочна	Бренди присутні	Активна реклама, яка вказує на переваги саме даного рішення, натя-каючи на недоліки конкурентів

У таблиці 7.9 наведено аналіз конкуренції в галузі за М.Портером.

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачаль-ники	Клієнти	Товари-замінники
	Навести перелік прями конкурентів	Визначити бар’єри входження на ринок	Визначит и фактори сили постача-льників	Визначити фактори сили споживачів	Фактори загроз з боку замін-ників

Продовження таблиці 7.9

Висновки:	Існує 3 конкуренти на ринку. Найбільш схожим за функціоналом є конкурент №1, так як його рішення також містить певні перевірки сирих даних	Є конкуренти, є можливість виходу на ринок	Постачальників багато, тому можна вважати, що вони не диктують умови на ринку	Важливим для користувача є можливість пошуку помилок у сирих даних та врахування особливостей галузі, а також можливість інтеграції з SAS Enterprise Guide 7.1	Товари замінники можуть використовувати більш дешеві технології створення ПЗ, та зменшити собівартість товару

У таблиці 7.10 наведено обґрунтування факторів конкурентоспроможності.

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Можливість інтеграції з SAS Enterprise Guide 7.1	SAS Enterprise Guide 7.1 є лідером на ринку медичних досліджень, і можливість інтеграції значно спрощує процес обробки даних
2.	Враховані особливості обробки даних медичних досліджень	Конкурентні проекти мають більш загальне призначення, та не враховують особливості обробки даних медичних досліджень

У таблиці 7.11 наведено порівняльний аналіз сильних та слабких сторін.

Таблиця 7.11 – Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			-3	-2	-1	0	1	2	3
1.	Можливість інтеграції з SAS Enterprise Guide 7.1	20				+			
2.	Враховані особливості обробки даних медичних досліджень	15			+				

У таблиці 7.12 наведено SWOT-аналіз стартап-проекту.

Таблиця 7.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Можливість інтеграції з SAS Enterprise Guide 7.1, враховані особливості обробки даних медичних досліджень	Слабкі сторони: Неможливість інтеграції з дослідженнями, які використовують інші технології обробки даних (Python, R)
Можливості: ріст зацікавленості до продукту серед інших груп користувачів (наприклад фармакологічних компаній, які не підтримують стандарти CDISC), у ПЗ конкурента №2 нещодавно була знайдена вразливість, яка впливає на якість пошуку помилок у великих даних	Загрози: конкуренція, зміна потреб користувачів

У таблиці 7.13 наведено альтернативи ринкового впровадження стартап-проекту.

Таблиця 7.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Використання протоколу взаємодії SSH	80%	1 місяць
2.	Використання протоколу взаємодії Telnet	60%	2 місяці

Обираємо альтернативу 1, тому що вона має більшу ймовірність отримання ресурсів та менший час реалізації.

7.4 Розроблення ринкової стратегії проекту

У таблиці 7.14 наведено вибір цільових груп потенційних споживачів.

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційн их клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивні сть конкуренц ії в сегменті	Простота входу у сегмент
1.	Фармакологіч ні компанії які підтримують звітність до FDA	Критичним є інтеграція з іншими сервісами	Автоматизац ія процесу пошуку помилок у сирих даних та створення SDTM датасетів	Існує 3 конкуренти , які надають схожі рішення. Але вони не враховують особливості медичних досліджень	У сегмент увійти просто, необхідно лише надати можливість до самостійної зміни деяких компонентів

Продовження таблиці 7.14

2.	Фармаколо-гічні компанії які підтримують звітність до FDA	Критичним є інтеграція з іншими сервісами та гнучкість системи до перенала-штування	Автома-тизація процесу пошуку помилок у сирих даних		У сегмент увійти просто, необхідно лише надати можливі-сть до самостій-ної зміни деяких компонен-тів

Обрано цільові групи1 та 2.

У таблиці 7.15 наведено визначення базової стратегії розвитку.

Таблиця 7.15 – Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспро-можні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.				

У таблиці 7.16 наведено визначення базової стратегії конкурентної поведінки.

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Ні	Так		

У таблиці 7.17 наведено визначення стратегії позиціонування.

Таблиця 7.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспро-можні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Можливість інтеграції з SAS Enterprise Guide, можливість застосування до різних типів досліджень	Диферен-ціації	Можливість інтеграції з SAS Enterprise Guide, врахування особливостей обробки даних медичних досліджень	Кастомізація, доцільність, гнучкість

7.5 Розроблення маркетингової програми стартап-проекту

У таблиці 7.18 наведено визначення ключових переваг концепції потенційного товару.

Таблиця 7.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Можливість інтеграції з SAS Enterprise Guide	Можливість інтеграції з SAS Enterprise Guide	Можливість роботи з SAS Enterprise Guide – лідером на ринку медичних досліджень
2.	Врахування особливостей обробки даних медичних досліджень	Доцільність використання ПЗ для медичних досліджень	Система враховує особливості процесу обробки даних медичних досліджень

У таблиці 7.19 наведено опис трьох рівнів моделі товару.

Таблиця 7.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
І. Товар за задумом	Об’єкт дозволяє вимоги користувачів щодо обробки даних медичних досліджень		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Інтеграція з іншими сервісами 2. Врахування особливостей обробки даних медичних досліджень	-	-
	Якість: згідно до стандартів CDISC, буде проведено перевірку валідності результатів		
	Відсутня підтримка до продажу		
	Постійна підтримка для користувачів після продажу		
За рахунок чого потенційний товар буде захищено від копіювання: ноу-хау.			

У таблиці 7.20 наведено визначення меж встановлення ціни.

Таблиця 7.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	6000	7000	200000	5000

У таблиці 7.21 наведено формування системи збуту.

Таблиця 7.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Купують пристрої, ПЗ йде у комплекті	Продаж	0 (напряму, 1 (через одного посередника)	Власна та через посередників

У таблиці 7.22 наведено концепція маркетингових комунікацій.

Таблиця 7.22 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими Користую-ться цільові клієнти	Ключові позиції, обрані для позиціонуван-ня	Завдання рекламного повідомлення	Концепція рекла-много звернення
1.	Замовлення через інтернет	Інтернет	Можливість кастомізацій, доцільність використання у галузі медичних досліджень	Показати переваги ПЗ, у тому числі і перед конкурентами	Демо-ролик із використання

Висновки до розділу

В результаті проведених досліджень було встановлено, що:

- існує можливість ринкової комерціалізації проекту;
- існують перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження не є високими, проект має дві значні переваги перед конкурентами;
- необхідно реалізувати систему з використанням протоколу передачі даних SSH ;
- подальша імплементація є доцільною.

ВИСНОВКИ

В результаті виконання магістерської дисертації розроблено автоматизовану систему обробки даних медичних досліджень, яка включала в себе модуль пошуку помилок у сирих даних та підсистеми створення SDTM датасетів.

У якості засобів розробки було обрано систему SAS, яка надає необхідні засоби розробки для виконання поставленої задачі, зокрема засоби обробки даних, в тому числі і для проведення статистичного аналізу даних, засоби розробки макросів, засоби розробки кастомізованих звітів TLF.

У якості середовища розробки було обрано SAS Enterprise Guide 7.1, який містить у собі систему SAS та необхідні засоби розробки.

Рішення було розгорнуто на базі середовища розподіленого доступу SAS Grid, яке дозволить надати доступ до середовища розробки усім користувачам, забезпечить необхідний рівень безпеки та відновлюваності системи та дозволить збалансувати навантаження на сервери.

Розроблене рішення дозволило автоматизувати процес обробки даних медичних досліджень, а саме, було налаштовано графік отримання даних з БД, розроблено модуль пошуку помилок у сирих даних та підсистема створення SDTM датасетів. Таким чином, втручання користувачів у процес значно зменшилося – програмісти найбільше залучені до розробки аналітичних датасетів та створення статистичних звітів, а система сама робить отримання даних, пошук помилок у сирих даних та створює SDTM датасети для подальшого аналізу.

Для розроблення модулі було використано SAS Macro Facility, що дозволило зробити їх більш гнучкими та стійкими до виникнення помилок.

Налаштування графіку виконання програм було виконано за допомогою утиліти cron та створення відповідних batch файлів. З огляду на сувору послідовність виконання кожного етапу обробки даних, налаштування послідовності створення виконано безпосередньо у файлах відповідних програм. Так як процес QC має відбуватися окремо від процесу створення production датасетів, то відповідно автоматизувати запуск QC програм недоцільно.

Проведено модульне тестування програм пошуку помилок у сирих даних та підсистеми створення SDTM датасетів, в результаті якого було доведено коректність роботи системи, зокрема правильність виконання макросів пошуку помилок у сирих даних, створення SDTM датасетів. У результаті тестування було покрито 89% програм. Цей відсоток означає, що більшість шляхів виконання програм були протестовані, але залишилися шляхи які не були перевірені. З огляду на особливість синтаксису програм на SAS (ітеративне виконання DATA кроків та наявність внутрішніх шляхів) та на велику кількість модулів у системі, можна стверджувати, що відсоток покриття тестами задовільний. Отримані результати надають можливість стверджувати, що система виконує поставлену задачу.

Також було проведено маркетинговий аналіз стартап-проекту та було оцінено можливість ринкової комерціалізації проекту, обрано альтернативу розроблення ПЗ, визначено перспективи впровадження з огляду на потенційні групи клієнтів. Проведений аналіз дає підстави стверджувати, що подальша імплементація є доцільною.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://en.wikipedia.org/wiki/Clinical_trial (дата звернення 11.10.2019)
2. Wang D. Clinical Trials. A Practical Guide to Design, Analysis, and Reporting / D. Wang, A. Bakhai. – London: Remedica, 2006. – 498 с.
3. Stemplinger R. T. Considerations for CDISC Implementation [Електронний ресурс] / R. T. Stemplinger, J. Lane // PhUSE. – 2007. – Режим доступу до ресурсу: https://www.phusewiki.org/docs/2007/PAPERS/_RA10.pdf.
4. https://en.wikipedia.org/wiki/Clinical_Data_Interchange_Standards_Consortium (дата звернення 11.10.2019)
5. https://www.cdisc.org/standards (дата звернення 11.10.2019)
6. https://www.cdisc.org/standards/foundational/cdash (дата звернення 11.10.2019)
7. https://www.cdisc.org/standards/foundational/sdtm (дата звернення 11.10.2019)
8. https://www.pinnacle21.com/news/opencdisc-validator-15-available-now (дата звернення 11.10.2019)
9. https://www.cdisc.org/standards/foundational/adam (дата звернення 13.10.2019)
10. https://en.wikipedia.org/wiki/SAS_(software) (дата звернення 14.10.2019)
11. https://www.sas.com/ru_ua/solutions/analytics.html (дата звернення 19.10.2019)
12. https://www.tutorialspoint.com/sas/sas_overview.htm (дата звернення 25.10.2019)
13. http://support.sas.com/documentation/cdl/en/basess/58133/HTML/default/viewer.htm#a001334669.htm (дата звернення 25.10.2019)

14. http://support.sas.com/documentation/cdl/en/hostwin/69955/HTML/default/viewer.htm#p1dvw5bf92luyjn12rd2q9r2h2oh.htm (дата звернення 25.10.2019)

15. https://www.tutorialspoint.com/sas/sas_program_structure.htm (дата звернення 26.10.2019)

16. https://support.sas.com/software/products/universityedition/faq/SAS_libname.htm (дата звернення 26.10.2019)

17. https://www.sas.com/content/dam/SAS/en_ca/User%20Group%20Presentations/TASS/Mehatab-DataStepPDV.pdf (дата звернення 29.10.2019)

18. https://documentation.sas.com/?docsetId=lrcon&docsetTarget=p08a4x7h9mkwqvn16jg3xqwfxful.htm&docsetVersion=9.4&locale=en (дата звернення 01.11.2019)

19. https://ru.wikipedia.org/wiki/Cron (дата звернення 10.11.2019)

20. https://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_91/base_macro_6997.pdf (дата звернення 20.11.2019)

21. https://www.lexjansen.com/nesug/nesug03/bt/bt009.pdf (дата звернення 20.11.2019)

22. https://medium.com/using-specialist-business-databases/introduction-to-data-analysis-with-sas-enterprise-guide-63cad33fc72b (дата звернення 20.11.2019)

23. https://www.sas.com/content/dam/SAS/en_us/doc/factsheet/sas-enterprise-guide-101431.pdf (дата звернення 20.11.2019)

24. http://www.torsas.ca/attachments/File/12142012/Mehatab92PlatformArchitecture.pdf (дата звернення 26.11.2019)

25. http://support.sas.com/documentation/cdl/en/emag/66802/HTML/default/viewer.htm#n0bz3z0b0ejz01n16li1ct03yxkg.htm#n1pdcpbrf6xkkhn1btocp0onzk44 (дата звернення 26.11.2019)

26. https://en.wikipedia.org/wiki/Secure_Shell (дата звернення 30.11.2019)

27. https://en.wikipedia.org/wiki/PuTTY (дата звернення 30.11.2019)

28. https://support.sas.com/rnd/scalability/grid/ (дата звернення 30.11.2019)

29. https://documentation.sas.com/?docsetId=bisag&docsetTarget=p0yh730iuqhpp4n1ki3h7k6juu9h.htm&docsetVersion=9.4&locale=en (дата звернення 02.12.2019)

30. https://documentation.sas.com/?docsetId=biasag&docsetTarget=n06001intelplatform00srvradm.htm&docsetVersion=9.4&locale=en (дата звернення 02.12.2019)

31. http://support.sas.com/documentation/cdl/en/gridref/67371/HTML/default/viewer.htm#p0fubu0wuont8bn17g6zsqkjd1un.htm (дата звернення 02.12.2019)

32. https://documentation.sas.com/?docsetId=bisag&docsetTarget=n11i3mqkmnhgu0n1bhm1qmtiep0u.htm&docsetVersion=9.4&locale=en (дата звернення 03.12.2019)

33. https://documentation.sas.com/?docsetId=scheduleug&docsetTarget=p0k2v9j1tpfp8in1l9fl6o180c3v.htm&docsetVersion=9.4&locale=en (дата звернення 04.12.2019)

34. https://documentation.sas.com/?docsetId=calsrvpgm&docsetTarget=n00001viyaprgmsrvs00000admin.htm&docsetVersion=3.3&locale=en#n05000viyaprgmsrvs00000admin (дата звернення 04.12.2019)

35. https://en.wikipedia.org/wiki/Ericom_Connect (дата звернення 04.12.2019)

36. Mangold A. Automatic Unit Testing of SAS Programs with SASUnit [Електронний ресурс] / A. Mangold, P. Warnat // PhUSE. – 2008. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/8e46/3612ae497426316edf74e1306de73597a9cd.pdf.

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність 126 Інформаційні технології та системи
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік О. І.
(підпис) (ініціали, прізвище)
«_» _____ 2019 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Хололович Катерині Вікторівні
(прізвище, ім'я, по батькові)

1. Тема дисертації «Автоматизована система обробки даних медичних досліджень»

науковий керівник дисертації Букасов Максим Михайлович,
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
к.т.н., доцент кафедри АУТС

затверджені наказом по університету від «___» _____ 2019 р. №___

2. Строк подання студентом дисертації _____

3. Вихідні дані до роботи розроблена автоматизована система має бути сумісна з системою SAS v9.4 та SAS Enterprise Guide v 7.1; має виконувати пошук помилок у сирих даних з урахуванням особливостей проведення медичних досліджень; у системі має бути автоматизовано створення SDTM датасетів

4. Перелік завдання, які потрібно розробити: огляд існуючих рішень, розроблення алгоритмів пошуку помилок у даних та створення SDTM датасетів, реалізація системи

5. Орієнтовний перелік ілюстративного (графічного) матеріалу: діаграма прецедентів, діаграма діяльності, схема структурна, схема структурна модулю пошуку помилок у даних, схема структурна підсистеми створення SDTM датасетів, блок-схеми алгоритмів, діаграма розгортання

6. Орієнтовний перелік публікацій: «Winter InfoCom Advanced Solutions 2019»,

7. Консультанти розділів дисертації

8. Дата видачі завдання – 02.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
1.	Огляд існуючих рішень	2.09.2019 р.	
2.	Визначення функціональних та нефункціональних вимог до системи	14.09.2019 р.	
3.	Визначення сценаріїв використання	28.09.2019 р.	
4.	Вибір засобів розроблення	1.10.2019 р.	
5.	Розроблення системи	15.10.2019 р.	
6.	Тестування системи	31.10.2019 р.	
7.	Розроблення схеми структурної	5.11.2019 р.	
8.	Розроблення діаграми розгортання	10.11.2019 р.	
9.	Розробка стартап – проекту	15.11.2019 р.	
10	Оформлення текстової документації	28.11.2019 р.	

Студент _____ Хололович К. В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Букасов М.М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Хололович К.В. Автоматизована система обробки даних медичних досліджень. КПІ ім. Ігоря Сікорського, Київ, 2019.

Дисертація містить 108 сторінок, 48 рисунків, 20 таблиць, 36 літературних джерел та 8 додатків.

Управління медичними даними є важливим процесом у медичних дослідженнях, що призводить до формування високоякісних, достовірних і статистично обґрунтованих даних медичних досліджень. Керування та підтримка цілісності даних є важливим елементом при зборі інформації. Проведення медичних досліджень коштують мільярди доларів. Штраф який стягується з фармакологічної компанії у випадку недостовірності даних або наявності помилок у даних може обійтись майже в ту саму вартість що й проведення дослідження. Тому виявлення таких помилок у даних є однією із головних задач із якою стикаються спеціалісти IT у медичних дослідженнях.

Об’єктом магістерської дисертації є автоматизація процесу обробки даних медичних досліджень. Предметом магістерської дисертації є алгоритм пошуку помилок у сирих даних та створення SDTM датасетів.

Метою магістерської дисертації є спрощення процесу обробки даних медичних досліджень, створення датасетів та звітів згідно зі стандартами CDISC (Clinical Data Interchange Standards Consortium). Було проаналізовано специфіку процесу обробки даних медичних досліджень та розроблено модуль пошуку помилок у сирих даних та підсистему створення SDTM датасетів. В якості технологій розробки було використано систему SAS 9.4, середовище розробки SAS Enterprise Guide 7.1.

Результати дослідження автоматизації пошуку помилок у сирих даних та створення SDTM датасетів були представлені на VIII міжнародній науково-практичній конференції «Winter InfoCom Advanced Solutions 2019», м. Київ, 2-3 грудня 2019 року.

Ключові слова: обробка даних медичних досліджень, CDISC, SDTM, SAS, SAS Enterprise Guide 7.1.

ABSTRACT

Khololovich K.V. Automated system for processing data of medical research. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2019.

The dissertation contains 108 pages, 48 drawings, 20 tables, 36 literary sources and 8 appendixes.

Medical data management is a process conducted in medical research that maintains high quality, reliable and statistically validated data of health care providers. Providing and maintaining data integrity is one of the most important components of processing data. Conducting medical research can worth billions of dollars. A fine that is applied to a pharmacological company when data is used inaccurately, or there is a data error, can be as expensive as the research itself. Therefore, searching for data errors is one of the most important challenges that IT professionals face in medical research.

The master's dissertation object is the automation of processing data of medical research.

The Master's dissertation subject is the algorithm of a search for errors in raw data and SDTM datasets creation.

The Master's Thesis purpose is to improve process of processing data of medical research, create datasets and reports compliant with CDISC (Consortium of Clinical Data Exchange Standards) standards. Peculiarities of medical research were analyzed and a module for searching for data errors and SDTM datasets creation subsystem were developed. SAS 9.4 system was used to develop the specified programs, and the SAS Enterprise Guide 7.1 was used as an integrated development environment.

The results of automated search for errors in raw data and SDTM datasets creation were presented at the VIII International Winter InfoCom Advanced Solutions 2019 Scientific and Practical Conference, Kyiv, December 2-3, 2019.

Keywords: medical data processing, CDISC, SDTM, SAS, SAS Enterprise Guide 7.1.