МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Практикум №1

з курсу «Аналіз даних в інформаційних системах»

на тему: «Створення сховища даних»

Викладач:

Олійник Ю.О.

Виконав:

студентка 2 курсу

групи ІП-13 ФІОТ

Макарчук Л. О.

Київ-2023

Мета роботи: ознайомитись з підходами до створення сховищ даних.

Предметною областю лабораторної роботи є стихійні лиха та їх наслідки. У даному випадку розглядаються лише 3 із них, а саме: виверження вулканів, землетруси та цунамі.

1. Для виконання роботи було обрано 3 джерела даних на сайті https://www.kaggle.com// та 1 джерело на сайті https://www.ngdc.noaa.gov/ngdc.html
   1.1. Інформація про континенти, субконтиненти та країни: https://www.kaggle.com/datasets/andradaolteanu/country-mapping-iso-continent-region
   1.2. Землетруси починаючи від часів до н. е. та закінчуючи сьогоденням: https://www.kaggle.com/datasets/mohitkr05/global-significant-earthquake-database-from-2150bc
   1.3. Цунамі: https://www.ngdc.noaa.gov/hazel/view/hazards/tsunami/event-data
   1.4. Виверження вулканів: https://www.kaggle.com/datasets/elizabethdgroot/volcanoevents02021

Крім цього, були використані 3 допоміжні json файли, де міститься опис наслідків: https://www.ngdc.noaa.gov/hazel/view/swagger (графа Effect Description).

2. Проектування моделі Stage зони для ETL процесів

   Скрипти створення Stage зони знаходяться у додатку А

   На рисунку 2.1 зображена модель Stage зони.

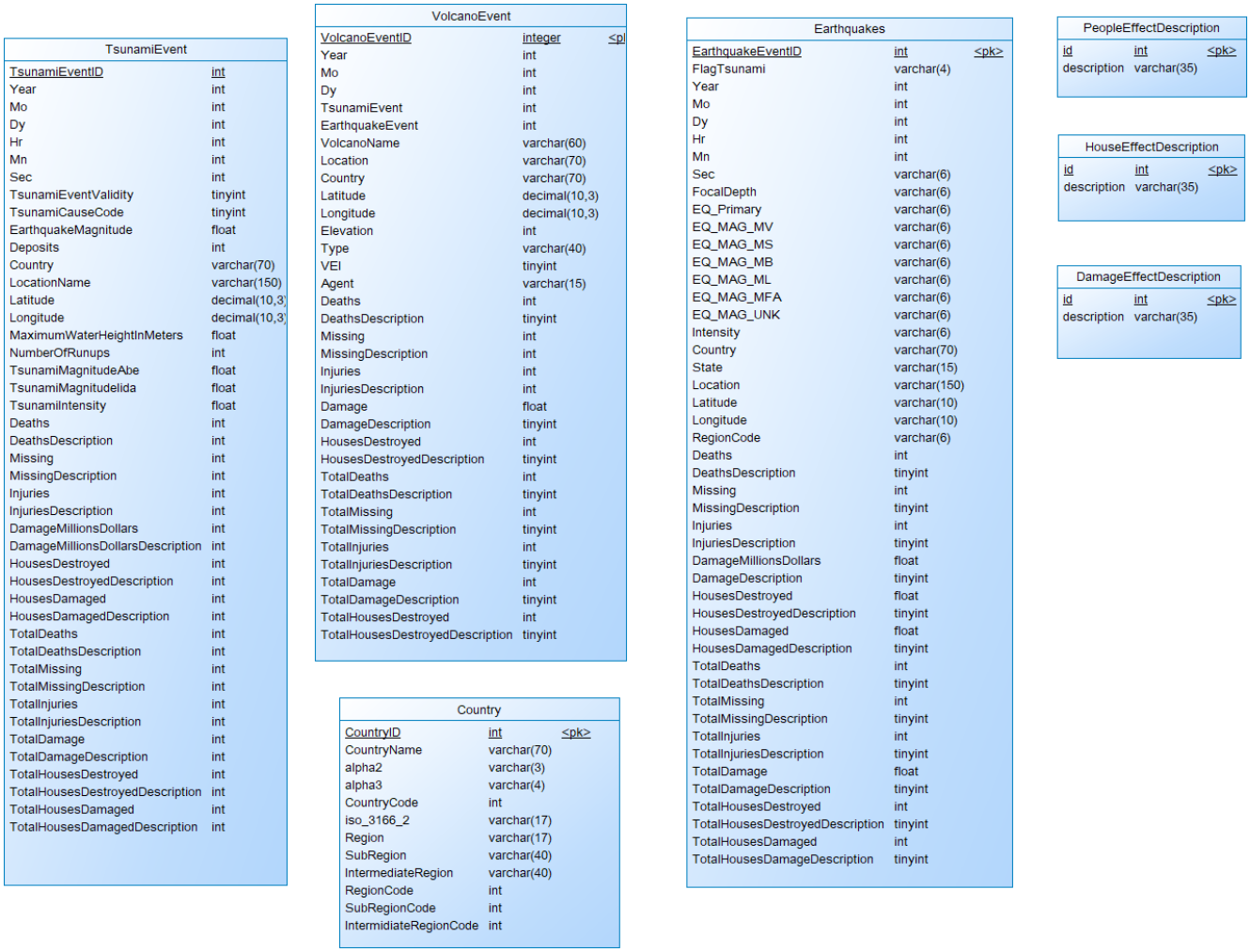

Рисунок 2.1-Модель Stage зони

Опис таблиць Stage зони:

Country – містить інформацію про континенти, субконтиненти, країни, їхні коди.

Earthquakes – зберігає дані про час та місце певного землетрусу, його основні характеристики, кількість загиблих, зниклих, поранених; збитки, задані будівлям та економіці.

VolcanoEvent - зберігає дані про час та місце виверження вулкану, його назву та основні характеристики; кількість загиблих, зниклих, поранених; збитки, задані будівлям та економіці.

TsunamiEvent - зберігає дані про час та місце виникнення цунамі, його основні характеристики, кількість загиблих, зниклих, поранених; збитки, задані будівлям та економіці.

PeopleEffectDescription – зберігає дані про приблизну оцінку загиблих, зниклих чи поранених, яка знаходиться у певних межах та має певний статус.

HousesEffectDescription – зберігає дані про приблизну оцінку пошкоджених чи повністю зруйнованих будинків, яка знаходиться у певних межах та має певний статус.

DamageEffectDescription – зберігає дані про приблизну оцінку збитків у мільйонах доларів, завданих явищем. Оцінка яка знаходиться у певних межах та має певний статус.

3. Проектування моделі основного сховища за типом “сніжинка”

Скрипти створення сховища знаходяться у додатку Б

На рисунку 2.2 зображена модель сховища даних.

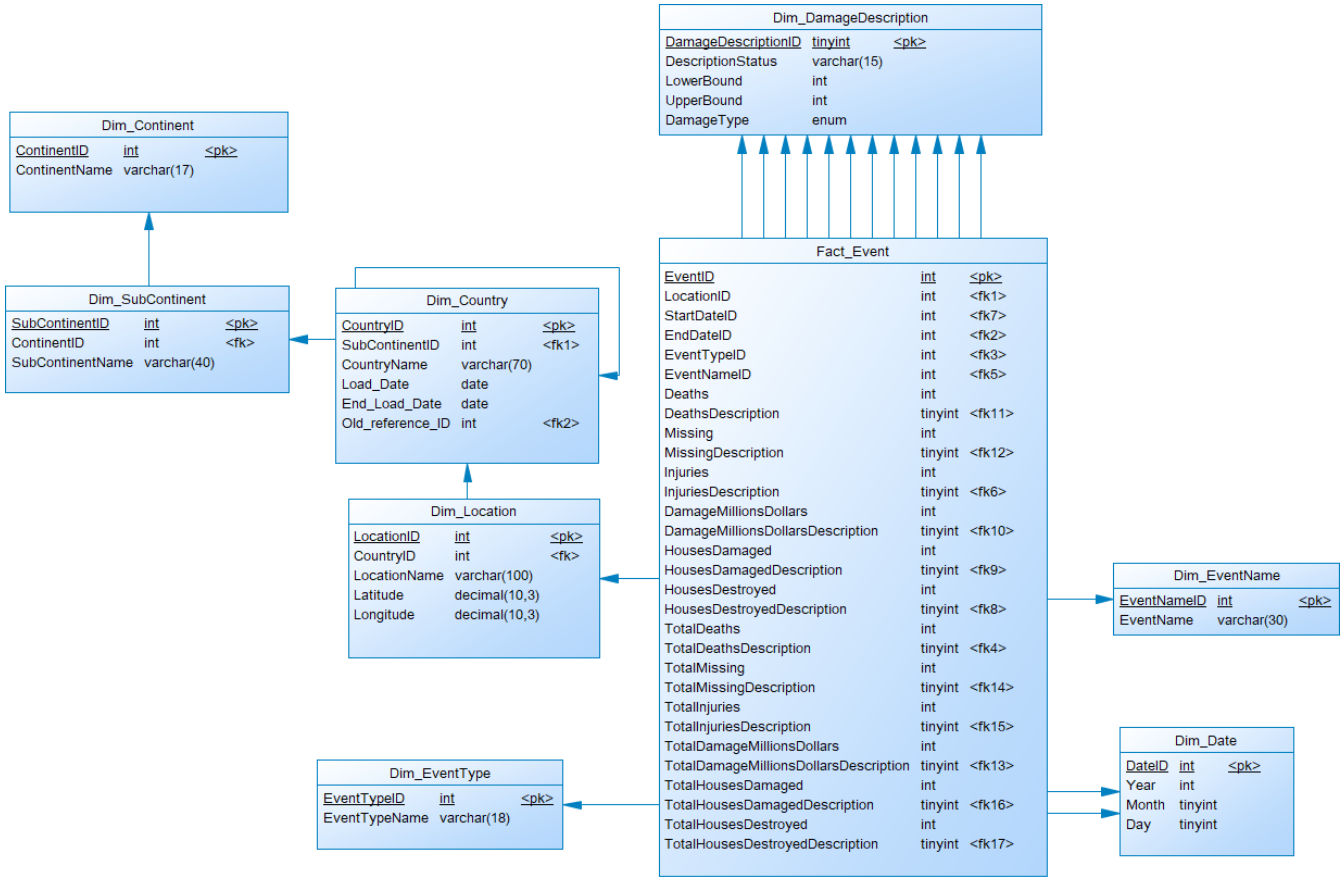

Рисунок 2.2 - Модель сховища даних

Сховище складається із 8 таблиць-вимірів та однієї фактової таблиці.

Опис таблиць сховища даних:

Таблиці-виміри:

Dim_Continent – містить назви континентів

Таблиця 3.1 – Опис полів таблиці Dim_Continent

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
ContinentID	ідентифікатор	int	PK	+
ContinentName	Назва континенту	varchar(17)		+

Dim_Subcontinent – містить назви субконтинентів

Таблиця 3.2 – Опис полів таблиці Dim_Subcontinent

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
SubContinentID	ідентифікатор	int	PK	+
ContinentID	ідентифікатор континенту	int	FK	+
SubContinentName	Назва субконтиненту	varchar(40)		+

Dim_Country – містить інформацію про країни. Також для даної таблиці реалізовано 2-й тип slowly changing dimensions.

Таблиця 3.3 – Опис полів таблиці Dim_Country

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
CountryID	ідентифікатор	int	PK	+
SubContinentID	ідентифікатор субконтиненту	int	FK	+
CountryName	Назва країни	varchar(70)		+
Load_Date	Дата, починаючи з якої даний запис є дійсним	date		+
End_Load_Date	Дата, до якої даний запис є дійсним	date		+
Old_reference_ID	Посилання на запис, який був дійсним перед даним	int	FK	-

Dim_Location – зберігає інформацію про назву локації події, її широту та довготу.

Таблиця 3.4 – Опис полів таблиці Dim_Location

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
LocationID	ідентифікатор	int	PK	+
CountryID	ідентифікатор країни	int	FK	-
LocationName	Назва локації	varchar(100)		+
Latitude	Географічна широта місця події	decimal(10,3)		+
Longitude	Географічна довгота місця події	decimal(10,3)		+

Dim_EventName – якщо події присвоєна назва, то вона зберігається у даній табличці. Наприклад, для виверження вулканів це буде назва вулкану, а для землетрусів та цунамі дана таблиця нічого не міститиме, адже ці події, зазвичай, ідентифікуються виключно локацією та часом.

Таблиця 3.5 – Опис полів таблиці Dim_EventName

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
EventNameID	ідентифікатор	int	PK	+
EventName	Назва події	varchar(30)		+

Dim_Date – зберігає рік, місяць та день події

Таблиця 3.6 – Опис полів таблиці Dim_Date

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
DateID	ідентифікатор	int	PK	+
Year	Рік події	int		+
Month	Місяць події	tinyint		-
Day	День події	tinyint		-

Dim_DamageDescription – зберігає опис збитків, а саме: статус, верхню та нижню межі оцінки; предмет, для якого дана інформація є релевантною (кількість людей, будинків та мільйонів доларів).

Таблиця 3.7 – Опис полів таблиці Dim_DamageDescription

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
DamageDescriptionID	ідентифікатор	int	PK	+
DescriptionStatus	Статус наслідків	int		+
LowerBound	Нижня межа оцінки	tinyint		-
UpperBound	Верхня межа оцінки	tinyint		-
DescriptionType	Тип наслідку	Enum(people, houses, damageMillionDollars)		+

Dim_EventType – тип події (землетрус, виверження вулкану, або ж цунамі)

Таблиця 3.8 – Опис полів таблиці Dim_EventType

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
EventTypeID	ідентифікатор	int	PK	+
EventTypeName	Назва типу події	varchar(18)		+

Фактова таблиця:

Fact_Event – зберігає наступну інформацію по кожній події: час настання та завершення, локація, тип та назва події, завдані збитки або їх приблизна оцінка.

Таблиця 3.9 – Опис полів таблиці Dim_Event

Назва атрибуту	**Опис атрибуту**	**Тип даних**	**Ключ**	**Обов'язкове поле**
EventID	ідентифікатор	int	PK	+
LocationID	Ідентифікатор місця події	int	FK	+

StartDateID	Ідентифікатор часу початку	int	FK	+
EndDateID	Ідентифікатор часу закінчення	int	FK	-
Deaths	Точна к-сть загиблих внаслідок події	int		-
DeathsDescription	Ідентифікатор приблизної к-сті загиблих внаслідок події	tinyint	FK	-
Missing	Точна к-сть зниклих внаслідок події	int		-
MissingDescription	Ідентифікатор приблизної к-сті зниклих внаслідок події	tinyint	FK	-
Injuries	Точна к-сть поранених внаслідок події	int		-
InjuriesDescription	Ідентифікатор приблизної к-сті поранених внаслідок події	tinyint	FK	-
DamageMillionsDollars	Точна оцінка збитків у мільйонах доларів	int		-
DamageMillionsDollarsDescription	Ідентифікатор приблизної оцінки збитків у мільйонах доларів внаслідок події	tinyint	FK	-
HousesDamaged	Точна к-сть пошкоджених будинків внаслідок події	int		-
HousesDamagedDescription	Ідентифікатор приблизної к-сті пошкоджених будинків внаслідок події	tinyint	FK	-
HousesDestroyed	Точна к-сть зруйнованих будинків внаслідок події	int		-
HousesDestroyedDescription	Ідентифікатор приблизної к-сті зруйнованих будинків внаслідок події	tinyint	FK	-
TotalDeaths	Точна к-сть загиблих внаслідок події та спричинених побічних ефектів	int		-
TotalDeathsDescription	Ідентифікатор приблизної к-сті загиблих внаслідок події та спричинених побічних ефектів	tinyint	FK	-
TotalMissing	Точна к-сть зниклих внаслідок події та спричинених побічних ефектів	int		-

TotalMissingDescription	Ідентифікатор приблизної к-сті зниклих внаслідок події та спричинених побічних ефектів	tinyint	FK	-
TotalInjuries	Точна к-сть поранених внаслідок події та спричинених побічних ефектів	Int		-
TotalInjuriesDescription	Ідентифікатор приблизної к-сті поранених внаслідок події та спричинених побічних ефектів	tinyint	FK	-
TotalDamageMillionsDollars	Точна оцінка збитків у мільйонах доларів внаслідок події та спричинених побічних ефектів	int		-
TotalDamageMillionsDollarsDescription	Ідентифікатор приблизної оцінки збитків у мільйонах доларів внаслідок події та спричинених побічних ефектів	tinyint	FK	-
TotalHousesDamaged	Точна к-сть пошкоджених будинків внаслідок події та спричинених побічних ефектів	Int		-
TotalHousesDamagedDescription	Ідентифікатор приблизної к-сті пошкоджених будинків внаслідок події та спричинених побічних ефектів	Tinyint	FK	-
TotalHousesDestroyed	Точна к-сть зруйнованих будинків внаслідок події та спричинених побічних ефектів	int		-
TotalHousesDestroyedDescription	Ідентифікатор приблизної к-сті зруйнованих будинків внаслідок події та спричинених побічних ефектів	tinyint	FK	-

4. Створення ETL засобів

ETL засоби були реалізовані за допомогою мови програмування Python та MySQL. Дані були витягнуті з таких типів файлів як csv, tsv, json. Текст коду ETL засобів наведено в додатках В та Г. Засоби для інкрементного завантаження та реалізація 2 типу scd приведені у додатку Д.

4.1.Алгоритм «витягування» даних та наповнення Stage зони:

1. ПОЧАТОК

2. Зчитати дані з файлів у окремі датафрейми.
3. Для кожного датафрейму:
   3.1. Видалити непотрібні колонки
   3.2. Замінити пропущені значення на ‘None’
   3.3. Видалити порожні рядки
   3.4. Вставити дані з датафрейму у відповідну табличку бази даних
4. КІНЕЦЬ

4.2.Алгоритм обробки даних та їх первинного завантаження до сховища:

1. ПОЧАТОК
2. Трансформувати назви деяких країн
   2.1. Замінити всі абревіатури країн на їх повні назви (наприклад, U.S. -> United States)
   2.2. Замінити скорочення на повні слова (наприклад, Is. -> Islands)
3. Завантажити дані до таблиць-вимірів
   3.1. Із таблиці Country (Stage зона) витягти всі назви континентів без повторень та вставити до таблиці Dim_Continent (сховище даних)
   3.2. Із таблиці Country (Stage зона) витягти всі назви субконтинентів без повторень. Витягти відповідні їм id з Dim_Continent. Вставити з’єднані дані до таблиці Dim_SubContinent
   3.3. Із таблиці Country (Stage зона) витягти всі назви країн без повторень. Витягти відповідні їм id з Dim_SubContinent. Вставити з’єднані дані до таблиці Dim_Country
   3.4. Із таблиць Earthquakes, Volcanoes, Tsunami витягти назву локації, широту та довготу без повторень. Знайти відповідні їм назви країн (якщо такі є) у таблиці Dim_Country. Витягти з даної таблиці необхідні id, вставити з’єднані дані до таблиці Dim_Location
   3.5. Із таблиць Earthquakes, VolcanoEvent, TsunamiEvent витягти унікальні комбінації року, місяця та дня події. Видалити дублікати. Вставити отримані дані до таблиці Dim_Date

3.6.3 таблиці VolcanoEvent вибрати унікальні назви вулканів. Додати дані до таблиці Dim_EventName

3.7. До таблиці Dim_EventType додати всі типи подій, а саме: ‘volcano’, ‘tsunami’, ‘earthquake’.

3.8. Заповнення таблиці Dim_DamageDescription:

3.8.1. Із таблиць PeopleEffectDescription, HousesEffectDescription, DamageEffectDescription вибрати опис ефектів та тип (відповідно до назви таблиці).

3.8.2. Зробити парсинг опису ефектів, у результаті чого отримати статус ефекту, нижню та верхню межі оцінки.

3.8.3. Вставити отримані результати до таблиці Dim_DamageDescription

4. Завантажити дані до фактових таблиць

4.1. Із таблиць Dim_Location, Dim_Date, Dim_DamageDescription, Dim_EventName, Dim_EventType витягти ідентифікатори, що відповідають записам таблиць Earthquakes, Volcanoes, Tsunami. З’єднати дані таблиць. Вставити отримані дані до таблиці Fact_Event

5. КІНЕЦЬ

4.3. Алгоритм додаткового завантаження даних до сховища:

1. ПОЧАТОК
2. ЯКЩО дані стосуються стихійного лиха, яке ще не було додане ТО додати його назву до таблиці Dim_EventType.
3. Доповнення таблиці Dim_Location

3.1. Витягти інформацію про локацію, географічну широту та довготу місця події зі Stage зони.

3.2. ЯКЩО таблиця Dim_Location не містить певного запису, додати його до цієї таблиці

4. Доповнення таблиці Dim_Date

4.1. Витягти унікальні комбінації року, місяця та дня зі Stage зони

4.2. ЯКЩО таблиця Dim_Date не містить певного запису, додати його до цієї таблиці

5. ЯКЩО для даного типу стихійних лих передбачені назви, ТО
   5.1. Витягти унікальні назви зі Stage зони
   5.2. ЯКЩО таблиця Dim_EventName не містить певного запису, додати його до цієї таблиці
6. Доповнити фактову таблицю, використовуючи дані зі Stage зони та нещодавно додані дані до таблиць вимірів.
7. КІНЕЦЬ

4.4. Алгоритм оновлення даних з використанням 2-го типу slowly changing dimensions:

1. ПОЧАТОК
2. Додавання нового запису, Load_Date якого дорівнює поточній даті, а End_Load_Date максимальній можливій.
3. Додати посилання нового запису на старий
4. Зміна End_Load_Date старого запису на поточну
5. Заміна id у всіх таблицях, де було посилання на старий запис
6. КІНЕЦЬ

5. Завантаження даних до основного сховища

Завантаження даних відбувається за допомогою розроблених ETL засобів. Приклад завантажених даних можна побачити на наступних рисунках (5.1 – 5.9). Приклад роботи процедури, що забезпечує SCD type 2 можна побачити на рисунках 5.10-5.14

ContinentID	ContinentName
1	Asia
2	Europe
3	Africa
4	Oceania
5	Americas

Рисунок 5.1 - Дані таблиці dim_Continent

SubCont...	Co...	SubContinentName
1	1	Southern Asia
2	1	Western Asia
3	1	South-eastern Asia
4	1	Eastern Asia
5	1	Central Asia
6	2	Northern Europe
7	2	Southern Europe
8	2	Western Europe
9	2	Eastern Europe
10	3	Northern Africa
11	3	Sub-Saharan Africa
12	4	Polynesia
13	4	Australia and New Zealand
14	4	Melanesia
15	4	Micronesia
16	5	Latin America and the Caribbean
17	5	Northern America

Рисунок 5.2 - Дані таблиці dim_SubContinent

Cou...	SubCo...	Country...	Load_Date	End_Loa...	Old_ref...
1	1	Afghanistan	1000-01-01	9999-12-31	<null>
2	1	Bangladesh	1000-01-01	9999-12-31	<null>
3	1	Bhutan	1000-01-01	9999-12-31	<null>
4	1	India	1000-01-01	9999-12-31	<null>
5	1	Iran	1000-01-01	9999-12-31	<null>
6	1	Maldives	1000-01-01	9999-12-31	<null>
7	1	Nepal	1000-01-01	9999-12-31	<null>
8	1	Pakistan	1000-01-01	9999-12-31	<null>
9	1	Sri Lanka	1000-01-01	9999-12-31	<null>
10	2	Armenia	1000-01-01	9999-12-31	<null>
11	2	Azerbaijan	1000-01-01	9999-12-31	<null>
12	2	Bahrain	1000-01-01	9999-12-31	<null>
13	2	Cyprus	1000-01-01	9999-12-31	<null>

Рисунок 5.3 - Дані таблиці dim_country

Locati...	CountryID	LocationName	Latitude	Longitude
3	183	Vanuatu-SW Pacific	-16.250	168.120
4	248	Alaska-E	61.380	-141.750
5	75	Italy	40.821	14.426
6	27	Arabia-S	15.630	44.080
7	177	New Zealand	-38.820	176.000
8	248	Kamchatka	51.800	157.530
9	248	Alaska-SW	59.370	-153.420
10	30	Indonesia	-6.102	105.423
11	212	El Salvador	13.672	-89.053
12	212	El Salvador	13.736	-89.286
13	23	Arabia-W	27.080	37.250
14	181	New Britain-SW Pac	-5.056	150.108
15	181	New Britain-SW Pac	-4.271	152.203
16	181	New Britain-SW Pac	-5.580	150.520
17	41	Kyushu-Japan	31.580	130.670
18	58	Iceland-S	63.630	-19.050

Рисунок5.4 - Дані таблиці dim_Location

DamageDes...	DescriptionStatus	Lower...	UpperBound	DamageType
1	None	0	0	damageMillionDollars
2	Limited	0	1	damageMillionDollars
3	Moderate	1	5	damageMillionDollars
4	Severe	5	24	damageMillionDollars
5	Extreme	25	<null>	damageMillionDollars
6	None	0	0	people
7	Few	1	50	people
8	Some	51	100	people
9	Many	101	1000	people
10	Very	1001	<null>	people
11	None	0	0	houses
12	Few	1	50	houses
13	Some	51	100	houses
14	Many	101	1000	houses
15	Very	1001	<null>	houses

Рисунок 5.5 - Дані таблиці dim_DamageDescription

DateID	Year	Month	Day
1764	1804	7	10
1765	1804	7	31
1766	1804	8	25
1767	1805	<null>	<null>
1768	1805	1	8
1769	1805	1	27
1770	1805	5	8
1771	1805	6	16
1772	1805	7	26
1773	1805	8	11

Рисунок 5.6 - Дані таблиці dim_Date

EventNameID	EventName
1	Adatara
2	Agung
3	Akagi
4	Akita-Yake-yama
5	Alaid
6	Alayta
7	Ambalatungan Group
8	Ambrym
9	Aoba
10	Aoga-shima
11	Ararat
12	Arenal

Рисунок 5.7 - Дані таблиці dim_EventName

EventTypeID	EventTypeName
1	volcano
2	earthquake
3	tsunami

Рисунок 5.8 - Дані таблиці dim_EventType

EventID	Loca...	StartDateID	EndDateID	EventTy...	Deaths	Deat...	Missing	Event...	Missi...
1	2	62	<null>	1	<null>	<null>	<null>	200	<null>
2	3	64	<null>	1	<null>	<null>	<null>	8	<null>
3	4	68	<null>	1	<null>	<null>	<null>	33	<null>
4	5	75	<null>	1	2100	10	<null>	248	<null>
5	6	95	<null>	1	<null>	<null>	<null>	13	<null>
6	7	96	<null>	1	<null>	<null>	<null>	227	<null>
7	8	97	<null>	1	<null>	<null>	<null>	120	<null>
8	9	118	<null>	1	<null>	<null>	<null>	18	<null>
9	10	135	<null>	1	<null>	7	<null>	118	<null>
10	11	144	<null>	1	30000	10	<null>	92	<null>
11	6	154	<null>	1	<null>	<null>	<null>	13	<null>
12	12	185	<null>	1	<null>	7	<null>	196	<null>
13	13	189	<null>	1	<null>	7	<null>	246	<null>
14	14	191	<null>	1	<null>	<null>	<null>	53	<null>
15	15	195	<null>	1	<null>	<null>	<null>	179	<null>
16	16	200	<null>	1	<null>	<null>	<null>	166	<null>
17	17	217	<null>	1	80	8	<null>	192	<null>
18	17	219	<null>	1	<null>	9	<null>	192	<null>
19	5	223	<null>	1	<null>	7	<null>	248	<null>

Рисунок 5.9 - Дані перших 10 колонок таблиці fact_Event

Для демонстрації роботи функції scd_country(old_name, new_name) зафіксуємо рядок зі старим значенням назви країни:

78	7	Macedonia	1000-01-01	9999-12-31	<null>

Рисунок 5.10 - Запис до виклику процедури

LocationID	CountryID	LocationName	Latitude	Longitude
2516	78	MACEDONIA: STOBI	41.600	21.500
2537	78	MACEDONIA: SKOPJE, STAMER	42.000	21.400
2625	78	MACEDONIA	41.700	22.900
2963	78	MACEDONIA: RAZLOVCI	42.800	22.800

Рисунок5.11 - Ключ старого запису у таблиці dim_Location

Викликаємо процедуру для зміни старого значення назви на нове:

CountryID	Sub...	CountryName	Load_Date	End_Load_Date	Old_refe...
78	7	Macedonia	1000-01-01	2023-03-07	<null>

Рисунок 5.12 - старий запис містить поточну дату у якості End_Load_Date

CountryID	SubCo...	CountryName	Load_Date	End_Load...	Old_reference_ID
256	7	North Macedonia	2023-03-07	9999-12-31	78

Рисунок5.13 - Нове значення назви, що містить посилання на старий запис

LocationID	CountryID	LocationName	Latitude	Longitude
2516	256	MACEDONIA: STOBI	41.600	21.500
2537	256	MACEDONIA: SKOPJE, STAMER	42.000	21.400
2625	256	MACEDONIA	41.700	22.900
2963	256	MACEDONIA: RAZLOVCI	42.800	22.800

Рисунок 5.14 - Оновлення FK у таблиці Location

Додаток А Створення таблиць Stage зони

```
use disasters_stage;

drop table if exists Country;
drop table if exists Earthquakes;
drop table if exists TsunamiEvent;
drop table if exists VolcanoEvent;
drop table if exists PeopleEffectDescription;
drop table if exists HouseEffectDescription;
drop table if exists DamageEffectDescription;

create table Country
(
  CountryID          int                    not null AUTO_INCREMENT,
  CountryName        varchar(70)            null,
  alpha2             varchar(3)             null,
  alpha3             varchar(4)             null,
  CountryCode        int                    null,
  iso_3166_2         varchar(17)            null,
  Region             varchar(17)            null,
  SubRegion          varchar(40)            null,
  IntermediateRegion varchar(40)            null,
  RegionCode         int                    null,
  SubRegionCode      int                    null,
  IntermidiateRegionCode int                null,
  constraint PK_COUNTRY primary key (CountryID)
);

create table Earthquakes
(
  EarthquakeEventID  int                    not null AUTO_INCREMENT,
  FlagTsunami        varchar(4)             null,
  Year               int                    null,
  Mo                 int                    null,
  Dy                 int                    null,
  Hr                 int                    null,
  Mn                 int                    null,
  Sec                varchar(6)             null,
  FocalDepth         varchar(6)             null,
  EQ_Primary         varchar(6)             null,
  EQ_MAG_MV          varchar(6)             null,
  EQ_MAG_MS          varchar(6)             null,
  EQ_MAG_MB          varchar(6)             null,
  EQ_MAG_ML          varchar(6)             null,
  EQ_MAG_MFA         varchar(6)             null,
  EQ_MAG_UNK         varchar(6)             null,
  Intensity          varchar(6)             null,
```

```
  Country             varchar(70)                 null,
  State               varchar(15)                 null,
  Location            varchar(150)                null,
  Latitude            varchar(10)                 null,
  Longitude           varchar(10)                 null,
  RegionCode          varchar(6)                  null,
  Deaths              int                         null,
  DeathsDescription   tinyint                     null,
  Missing             int                         null,
  MissingDescription  tinyint                     null,
  Injuries            int                         null,
  InjuriesDescription tinyint                     null,
  DamageMillionsDollars float                     null,
  DamageDescription   tinyint                     null,
  HousesDestroyed     float                       null,
  HousesDestroyedDescription tinyint              null,
  HousesDamaged       float                       null,
  HousesDamagedDescription tinyint                null,
  TotalDeaths         int                         null,
  TotalDeathsDescription tinyint                  null,
  TotalMissing        int                         null,
  TotalMissingDescription tinyint                 null,
  TotalInjuries       int                         null,
  TotalInjuriesDescription tinyint                null,
  TotalDamage         float                       null,
  TotalDamageDescription tinyint                  null,
  TotalHousesDestroyed int                        null,
  TotalHousesDestroyedDescription tinyint         null,
  TotalHousesDamaged  int                         null,
  TotalHousesDamageDescription tinyint            null,
  constraint PK_EARTHQUAKES primary key (EarthquakeEventID)
);

create table TsunamiEvent
(
  TsunamiEventID      int                         not null AUTO_INCREMENT,
  Year                int                         null,
  Mo                  int                         null,
  Dy                  int                         null,
  Hr                  int                         null,
  Mn                  int                         null,
  Sec                 int                         null,
  TsunamiEventValidity tinyint                    null,
  TsunamiCauseCode    tinyint                     null,
  EarthquakeMagnitude float                       null,
  Deposits            int                         null,
  Country             varchar(70)                 null,
  LocationName        varchar(150)                null,
```

```
  Latitude            decimal(10,3)                 null,
  Longitude           decimal(10,3)                 null,
  MaximumWaterHeightInMeters float                  null,
  NumberOfRunups      int                           null,
  TsunamiMagnitudeAbe  float                        null,
  TsunamiMagnitudeIida float                        null,
  TsunamiIntensity    float                         null,
  Deaths              int                           null,
  DeathsDescription   int                           null,
  Missing             int                           null,
  MissingDescription  int                           null,
  Injuries            int                           null,
  InjuriesDescription  int                          null,
  DamageMillionsDollars int                         null,
  DamageMillionsDollarsDescription int              null,
  HousesDestroyed     int                           null,
  HousesDestroyedDescription int                    null,
  HousesDamaged       int                           null,
  HousesDamagedDescription int                      null,
  TotalDeaths         int                           null,
  TotalDeathsDescription int                        null,
  TotalMissing        int                           null,
  TotalMissingDescription int                       null,
  TotalInjuries       int                           null,
  TotalInjuriesDescription int                      null,
  TotalDamage         int                           null,
  TotalDamageDescription int                        null,
  TotalHousesDestroyed int                          null,
  TotalHousesDestroyedDescription int               null,
  TotalHousesDamaged  int                           null,
  TotalHousesDamagedDescription int                 null,
  constraint PK_TSUNAMIEVENT primary key (TsunamiEventID)
);

create table VolcanoEvent
(
  VolcanoEventID      integer                       not null AUTO_INCREMENT,
  Year                int                           null,
  Mo                  int                           null,
  Dy                  int                           null,
  TsunamiEvent        int                           null,
  EarthquakeEvent     int                           null,
  VolcanoName         varchar(60)                   null,
  Location            varchar(70)                   null,
  Country             varchar(70)                   null,
  Latitude            decimal(10,3)                 null,
  Longitude           decimal(10,3)                 null,
  Elevation           int                           null,
```

```
  Type              varchar(40)              null,
  VEI               tinyint                  null,
  Agent             varchar(15)              null,
  Deaths            int                      null,
  DeathsDescription tinyint                  null,
  Missing           int                      null,
  MissingDescription int                     null,
  Injuries          int                      null,
  InjuriesDescription int                    null,
  Damage            float                    null,
  DamageDescription tinyint                  null,
  HousesDestroyed   int                      null,
  HousesDestroyedDescription tinyint         null,
  TotalDeaths       int                      null,
  TotalDeathsDescription tinyint             null,
  TotalMissing      int                      null,
  TotalMissingDescription tinyint            null,
  TotalInjuries     int                      null,
  TotalInjuriesDescription tinyint           null,
  TotalDamage       int                      null,
  TotalDamageDescription tinyint             null,
  TotalHousesDestroyed int                   null,
  TotalHousesDestroyedDescription tinyint    null,
  constraint PK_VOLCANOEVENT primary key (VolcanoEventID)
);

create table PeopleEffectDescription
(
  id                int                      not null AUTO_INCREMENT,
  description       varchar(35)              null,
  constraint PK_PEOPLEEFFECTDESCRIPTION primary key (id)
);

create table HouseEffectDescription
(
  id                int                      not null AUTO_INCREMENT,
  description       varchar(35)              null,
  constraint PK_HOUSEEFFECTDESCRIPTION primary key (id)
);

create table DamageEffectDescription
(
  id                int                      not null AUTO_INCREMENT,
  description       varchar(35)              null,
  constraint PK_DAMAGEEFFECTDESCRIPTION primary key (id)
);
```

Додаток Б Створення таблиць сховища даних

```
use disasters_dwh;

drop table if exists Fact_Event;
drop table if exists Dim_Location;
drop table if exists Dim_Country;
drop table if exists Dim_SubContinent;
drop table if exists Dim_Continent;
drop table if exists Dim_DamageDescription;
drop table if exists Dim_Date;
drop table if exists Dim_EventType;
drop table if exists Dim_EventName;

create table Dim_Continent
(
  ContinentID         int                     AUTO_INCREMENT,
  ContinentName       varchar(17)                 not null,
  constraint PK_DIM_CONTINENT primary key (ContinentID)
);

create table Dim_SubContinent
(
  SubContinentID      int                       AUTO_INCREMENT,
  ContinentID         int                     not null,
  SubContinentName    varchar(40)                 not null,
  constraint PK_DIM_SUBCONTINENT primary key (SubContinentID)
);

create table Dim_Country
(
  CountryID           int                     AUTO_INCREMENT,
  SubContinentID      int                       not null,
  CountryName         varchar(70)                 not null,
  Load_Date           date                      not null DEFAULT '1000-01-01',
  End_Load_Date       date                        not null DEFAULT '9999-12-31',
  Old_reference_ID    int                       ,
  constraint PK_DIM_COUNTRY primary key (CountryID)
);

create table Dim_Location
(
  LocationID          int                     AUTO_INCREMENT,
  CountryID           int                     ,
  LocationName        varchar(100)                ,
  Latitude            decimal(10,3)               ,
  Longitude           decimal(10,3)               ,
  constraint PK_DIM_LOCATION primary key (LocationID)
```

```
);

create table Dim_DamageDescription
(
  DamageDescriptionID  tinyint                  AUTO_INCREMENT,
  DescriptionStatus    varchar(15)              not null,
  LowerBound           int                      ,
  UpperBound           int                      ,
  DamageType           enum('people', 'houses', 'damageMillionDollars') not null,
  constraint PK_DIM_DAMAGEDESCRIPTION primary key (DamageDescriptionID)
);

create table Dim_Date
(
  DateID               int                      AUTO_INCREMENT,
  Year                 int                      not null,
  Month                tinyint                  ,
  Day                  tinyint                  ,
  constraint PK_DIM_DATE primary key (DateID)
);

create table Dim_EventType
(
  EventTypeID          int                      AUTO_INCREMENT,
  EventTypeName        varchar(18)              not null,
  constraint PK_DIM_EVENTTYPE primary key (EventTypeID)
);

create table Dim_EventName
(
  EventNameID          int                      AUTO_INCREMENT,
  EventName            varchar(30)              not null,
  constraint PK_EVENTNAME primary key (EventNameID)
);

create table Fact_Event
(
  EventID              int                      AUTO_INCREMENT,
  LocationID           int                      not null,
  StartDateID          int                      not null,
  EndDateID            int                      ,
  EventTypeID          int                      not null,
  Deaths               int                      ,
  DeathsDescription    tinyint                  ,
  Missing              int                      ,
  EventNameID          int                      ,
  MissingDescription   tinyint                  ,
  Injuries             int                      ,
```

```
  InjuriesDescription  tinyint                ,
  DamageMillionsDollars int                   ,
  DamageMillionsDollarsDescription tinyint    ,
  HousesDamaged        int                    ,
  HousesDamagedDescription tinyint            ,
  HousesDestroyed      int                    ,
  HousesDestroyedDescription tinyint          ,
  TotalDeaths          int                    ,
  TotalDeathsDescription tinyint              ,
  TotalMissing         int                    ,
  TotalMissingDescription tinyint             ,
  TotalInjuries        int                    ,
  TotalInjuriesDescription tinyint            ,
  TotalDamageMillionsDollars int              ,
  TotalDamageMillionsDollarsDescription tinyint ,
  TotalHousesDamaged   int                    ,
  TotalHousesDamagedDescription tinyint       ,
  TotalHousesDestroyed int                    ,
  TotalHousesDestroyedDescription tinyint     ,
  constraint PK_FACT_EVENT primary key (EventID),
  constraint FK_TotalHousesDestroyedDescription foreign key (TotalHousesDestroyedDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_TotalHousesDamagedDescription foreign key (TotalHousesDamagedDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_TotalDamageMillionsDollarsDescription foreign key (TotalDamageMillionsDollarsDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_TotalInjuriesDescription foreign key (TotalInjuriesDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_TotalMissingDescription foreign key (TotalMissingDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_TotalDeathsDescription foreign key (TotalDeathsDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_HousesDestroyedDescription foreign key (HousesDestroyedDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_HousesDamagedDescription foreign key (HousesDamagedDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_DamageMillionsDollarsDescription foreign key (DamageMillionsDollarsDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_InjuriesDescription foreign key (InjuriesDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_MissingDescription foreign key (MissingDescription) references Dim_DamageDescription (DamageDescriptionID),
  constraint FK_DeathsDescription foreign key (DeathsDescription) references Dim_DamageDescription (DamageDescriptionID)
);

alter table Dim_Country
  add constraint FK_DIM_COUN_REFERENCE_DIM_COUN foreign key (Old_reference_ID)
```

```
    references Dim_Country (CountryID);

alter table Dim_Country
  add constraint FK_DIM_COUN_REFERENCE_DIM_SUBC foreign key (SubContinentID)
    references Dim_SubContinent (SubContinentID);

alter table Dim_Location
  add constraint FK_DIM_LOCA_REFERENCE_DIM_COUN foreign key (CountryID)
    references Dim_Country (CountryID);

alter table Dim_SubContinent
  add constraint FK_DIM_SUBC_REFERENCE_DIM_CONT foreign key (ContinentID)
    references Dim_Continent (ContinentID);

alter table Fact_Event
  add constraint FK_FACT_EVE_REFERENCE_DIM_LOCA foreign key (LocationID)
    references Dim_Location (LocationID);

alter table Fact_Event
  add constraint FK_FACT_EVE_REFERENCE_DIM_DATE foreign key (StartDateID)
    references Dim_Date (DateID);

alter table Fact_Event
  add constraint FK_FACT_EVE_REFERENCE_DIM_EVEN foreign key (EventTypeID)
    references Dim_EventType (EventTypeID);

alter table Fact_Event
  add constraint FK_FACT_EVE_REFERENCE_EVENTNAM foreign key (EventNameID)
    references Dim_EventName (EventNameID)
```

Додаток В Програмний код заповнення таблиць Stage зони за допомогою Python

main.py

```
import pymysql
from config import host, password, stage_area_user, dwh_area_user,
disasters_stage_db, disasters_dwh_db
from extraction import Extraction

try:
    connection = pymysql.connect(
        host=host,
        port=3306,
        user=stage_area_user,
        password=password,
        database=disasters_stage_db,
        cursorclass=pymysql.cursors.DictCursor
    )
    print("Connection is successfull")
    extr = Extraction(connection)
    extr.initial_extraction()

except Exception as e:
    print("Connection FAILED!")
    print(e)
```

extraction.py

```
# initial load
import pandas as pd
import numpy as np
import pymysql
from pathes import *
from stage_area_model import *
from config import *

class Extraction:
    def __init__(self, connection: pymysql.Connection):
        self.connection = connection

    def initial_extraction(self):
        self.continents_extraction()
        self.earthquakes_extraction()
        self.volcanos_extraction()
        self.tsunami_extraction_initial()
        self.effect_extraction(damage_description_file_name,
damage_description_table)
        self.effect_extraction(people_description_file_name,
people_description_table)
        self.effect_extraction(houses_description_file_name,
house_description_table)

    def continents_extraction(self, file_path=continents_file_name):
        data = pd.read_csv(file_path, delimiter=',')
        data = data.replace({np.nan: None})
        data = data.to_numpy()
        columns = get_columns_without_id(self.connection, disasters_stage_db,
country_table)
        self.write_data_to_database(country_table, ', '.join(columns),
len(columns), data)


    def earthquakes_extraction(self, file_path=earthquakes_file_name):
```

```
        data = pd.read_csv(file_path, delimiter=',')
        data = data.drop('I_D', axis=1)
        data = data.replace({np.nan: None})
        data = data.to_numpy()
        columns = get_columns_without_id(self.connection, disasters_stage_db,
earthquake_table)
        self.write_data_to_database(earthquake_table, ', '.join(columns),
len(columns), data)

    def volcanos_extraction(self, file_path=volcanos_file_name):
        data = pd.read_csv(file_path, delimiter=',')
        data = data.replace({np.nan: None})
        data = data.to_numpy()
        columns = get_columns_without_id(self.connection, disasters_stage_db,
volcano_table)
        self.write_data_to_database(volcano_table, ', '.join(columns),
len(columns), data)

    def tsunami_extraction_initial(self, file_path=tsunamis_file_name_initial):
        data = pd.read_csv(file_path, delimiter='\t')
        data = data.replace({np.nan: None})
        data = data.drop('Search Parameters', axis=1)
        data = data.drop('Vol', axis=1)
        data = data.drop('More Info', axis=1)
        data = data.dropna(how='all')
        data = data.to_numpy()

        columns = get_columns_without_id(self.connection, disasters_stage_db,
tsunami_table)
        self.write_data_to_database(tsunami_table, ', '.join(columns),
len(columns), data)

    def effect_extraction(self, filename, table_name):
        data = pd.read_json(filename)
        columns = get_columns_without_id(self.connection, disasters_stage_db,
table_name)
        with self.connection.cursor() as cursor:
            for record in data['items']:
                cursor.execute(f"insert into {table_name}({', '.join(columns)})
values ({self.get_format_string(len(columns))})",
                               record['description'])
            self.connection.commit()

    def write_data_to_database(self, table_name, columns_string, column_number,
data):
        with self.connection.cursor() as cursor:
            cursor.executemany(
                f"insert into {table_name}({columns_string}) values
({self.get_format_string(column_number)})",
                tuple(map(tuple, data)))
            self.connection.commit()

    def get_format_string(self, table_num):
        return ','.join(['%s']*table_num)
```

stage_area_model.py

```
import pymysql
country_table = 'Country'
earthquake_table = 'Earthquakes'
tsunami_table = 'TsunamiEvent'
volcano_table = 'VolcanoEvent'
damage_description_table = 'DamageEffectDescription'
house_description_table = 'HouseEffectDescription'
people_description_table = 'PeopleEffectDescription'
```

```
def get_all_column_names(connection: pymysql.Connection, database_name: str,
table_name:str):
    cols = []
    with connection.cursor() as cursor:
        cursor.execute(
            f"""SELECT `COLUMN_NAME`
               FROM `INFORMATION_SCHEMA`.`COLUMNS`
               WHERE `TABLE_SCHEMA`='{database_name}'
               AND `TABLE_NAME`='{table_name}'
               ORDER BY ordinal_position;"""
        )
        cols = cursor.fetchall()
    return cols

def get_columns_without_id(connection: pymysql.Connection, database_name: str,
table_name:str):
    cols_objects = get_all_column_names(connection, database_name,
table_name)[1:]
    cols = []
    for col_obj in cols_objects:
        cols.append(col_obj['COLUMN_NAME'])
    return cols
```

Додаток Г SQL Скрипти обробки та завантаження даних до сховища даних

-- country-transforming.sql

```
UPDATE disasters_stage.country
SET CountryName = 'Palestine'
WHERE CountryName LIKE 'Palestine,%';

UPDATE disasters_stage.country
SET CountryName = 'Democratic Republic of the Congo'
WHERE CountryName LIKE 'Congo (D%';

UPDATE disasters_stage.tsunamievent
SET Country = disasters_dwh.transform_country_name(Country);

UPDATE disasters_stage.volcanoevent
SET Country = disasters_dwh.transform_country_name(Country);

UPDATE disasters_stage.earthquakes
SET Country = disasters_dwh.transform_country_name(Country);
```

-- dim tables insertion

```
CALL disasters_dwh.insert_damage_effect_table();
```

```
INSERT INTO disasters_dwh.dim_eventname (EventName)
SELECT DISTINCT VolcanoName
FROM disasters_stage.volcanoevent
WHERE VolcanoName IS NOT NULL
ORDER BY VolcanoName;
```

```
INSERT INTO disasters_dwh.dim_eventtype(EventTypeName) VALUES ('volcano'),
                                                              ('earthquake'),
                                                              ('tsunami');
```

```
INSERT INTO disasters_dwh.dim_continent (ContinentName)
SELECT DISTINCT Region
FROM disasters_stage.country
WHERE Region IS NOT NULL;

INSERT INTO disasters_dwh.dim_subcontinent (ContinentID, SubContinentName)
SELECT ContinentID, c.SubRegion
FROM disasters_dwh.dim_continent dc, (
    SELECT DISTINCT Region, SubRegion
    FROM disasters_stage.country) AS c
WHERE dc.ContinentName = c.Region;

INSERT INTO disasters_dwh.dim_country(SubContinentID, CountryName)
SELECT SubContinentID, s.CountryName
FROM disasters_dwh.dim_subcontinent ds, (
    SELECT DISTINCT SubRegion, CountryName
    FROM disasters_stage.country) AS s
WHERE ds.SubContinentName = s.SubRegion;

INSERT INTO disasters_dwh.dim_location(CountryID, LocationName, Latitude,
Longitude)
    SELECT CountryID, t1.Location, t1.Latitude, t1.Longitude
        FROM(
    SELECT CountryID, l.Location, l.Latitude, l.Longitude
```

```
    FROM disasters_dwh.dim_country ds right join (
        SELECT DISTINCT LOWER(Country) c, Location, Latitude, Longitude
        FROM disasters_stage.volcanoevent
        UNION
        SELECT DISTINCT LOWER(Country) c, LocationName, Latitude, Longitude
        FROM disasters_stage.tsunamievent
        UNION
        SELECT DISTINCT LOWER(Country) c,
disasters_dwh.delete_country_from_location(Location) l,

cast(disasters_dwh.if_is_not_alphanumeric_set_null(Latitude) as decimal (10,3))
la,

cast(disasters_dwh.if_is_not_alphanumeric_set_null(Longitude) as decimal (10,3))
lo
        FROM disasters_stage.earthquakes
        GROUP BY c, l, la, lo
    ) AS l on CONCAT('%',l.c,'%') like
              CONCAT('%', ds.CountryName, '%')
    ) as t1
group by t1.Location, t1.Latitude, t1.Longitude;
```

```
INSERT INTO disasters_dwh.dim_date(Year, Month, Day)
SELECT t.Year, MO, Dy
FROM (SELECT DISTINCT Year, MO, DY
FROM disasters_stage.volcanoevent
UNION
SELECT DISTINCT Year, MO, DY
FROM disasters_stage.earthquakes
UNION
SELECT DISTINCT Year, MO, DY
FROM disasters_stage.tsunamievent) as t
WHERE Year IS NOT NULL
ORDER BY t.Year, MO, DY;
```

-- fact table insertion

```
use disasters_dwh;
INSERT INTO disasters_dwh.fact_event (LocationID, StartDateID, EndDateID,
EventTypeID, EventNameID, Deaths, DeathsDescription, Missing,
MissingDescription, Injuries, InjuriesDescription, DamageMillionsDollars,
DamageMillionsDollarsDescription, HousesDamaged, HousesDamagedDescription,
HousesDestroyed, HousesDestroyedDescription, TotalDeaths,
TotalDeathsDescription, TotalMissing, TotalMissingDescription, TotalInjuries,
TotalInjuriesDescription, TotalDamageMillionsDollars,
TotalDamageMillionsDollarsDescription, TotalHousesDamaged,
TotalHousesDamagedDescription, TotalHousesDestroyed,
TotalHousesDestroyedDescription)

SELECT dl.LocationID, dd.DateID, null , de.EventTypeID, den.EventNameID,
       Deaths, find_damage_description_id('people', DeathsDescription) ,
       Missing, find_damage_description_id('people', MissingDescription),
       Injuries, find_damage_description_id('people', InjuriesDescription),
       Damage, find_damage_description_id('damageMillionDollars',
DamageDescription) ,
       null , null ,
       HousesDestroyed, find_damage_description_id('houses',
HousesDestroyedDescription),
       TotalDeaths, find_damage_description_id('people',
TotalDeathsDescription),
       TotalMissing, find_damage_description_id('people',
TotalMissingDescription),
```

```
       TotalInjuries, find_damage_description_id('people',
TotalInjuriesDescription),
       TotalDamage, find_damage_description_id('damageMillionDollars',
TotalDamageDescription),
       null , null ,
       TotalHousesDestroyed, find_damage_description_id('houses',
TotalHousesDestroyedDescription)
FROM
    disasters_stage.volcanoevent sv left join disasters_dwh.dim_location dl on
(sv.Latitude <=> dl.Latitude AND sv.Longitude <=> dl.Longitude AND sv.Location
<=> dl.LocationName) left join disasters_dwh.dim_date dd on (dd.Year = sv.Year
AND dd.Month <=> sv.Mo AND dd.Day <=> sv.Dy) join  disasters_dwh.dim_eventname
den on sv.VolcanoName = den.EventName, disasters_dwh.dim_eventtype de
WHERE
     de.EventTypeName = 'volcano'
UNION
SELECT dl.LocationID, dd.DateID, null , de.EventTypeID, null ,
       Deaths, find_damage_description_id('people', DeathsDescription),
       Missing, find_damage_description_id('people', MissingDescription),
       Injuries, find_damage_description_id('people', InjuriesDescription),
       DamageMillionsDollars, find_damage_description_id('damageMillionDollars',
DamageMillionsDollarsDescription),
       HousesDamaged, find_damage_description_id('houses',
HousesDamagedDescription),
       HousesDestroyed, find_damage_description_id('houses',
HousesDestroyedDescription),
       TotalDeaths, find_damage_description_id('people',
TotalDeathsDescription),
       TotalMissing, find_damage_description_id('people',
TotalMissingDescription),
       TotalInjuries, find_damage_description_id('people',
TotalInjuriesDescription),
       TotalDamage, find_damage_description_id('damageMillionDollars',
TotalDamageDescription),
       TotalHousesDamaged, find_damage_description_id('houses',
TotalHousesDamagedDescription), TotalHousesDestroyed,
find_damage_description_id('houses', TotalHousesDestroyedDescription)
FROM
    disasters_stage.tsunamievent st left join disasters_dwh.dim_location dl on
dl.Latitude <=> st.Latitude AND dl.Longitude <=> st.Longitude AND
dl.LocationName <=> st.LocationName join disasters_dwh.dim_date dd on (st.Year =
dd.Year AND st.Mo <=> dd.Month AND st.Dy <=> dd.Day),
disasters_dwh.dim_eventtype de
WHERE
    de.EventTypeName = 'tsunami'
UNION
SELECT dl.LocationID, dd.DateID, null , de.EventTypeID, null ,
       Deaths, find_damage_description_id('people', DeathsDescription),
       Missing, find_damage_description_id('people', MissingDescription),
       Injuries, find_damage_description_id('people', InjuriesDescription),
       DamageMillionsDollars, find_damage_description_id('damageMillionDollars',
DamageDescription),
       HousesDamaged, find_damage_description_id('houses',
HousesDamagedDescription),
       HousesDestroyed, find_damage_description_id('houses',
HousesDestroyedDescription),
       TotalDeaths, find_damage_description_id('people',
TotalDeathsDescription),
       TotalMissing, find_damage_description_id('people',
TotalMissingDescription),
       TotalInjuries, find_damage_description_id('people',
TotalInjuriesDescription),
       TotalDamage, find_damage_description_id('damageMillionDollars',
TotalDamageDescription),
       TotalHousesDamaged, find_damage_description_id('houses',
TotalHousesDamageDescription),
```

```
       TotalHousesDestroyed, find_damage_description_id('houses',
TotalHousesDestroyedDescription)
FROM
        disasters_stage.earthquakes se left join disasters_dwh.dim_location dl
on
        ((dl.Latitude <=> se.Latitude OR dl.Latitude IS NULL AND NOT
EXISTS(select se.Latitude where se.Latitude REGEXP '^[A-Za-z0-9]+$'))
             AND
          (dl.Longitude <=> se.Longitude OR dl.Longitude IS NULL AND NOT
EXISTS(select se.Longitude where se.Longitude REGEXP '^[A-Za-z0-9]+$')) AND
dl.LocationName <=> disasters_dwh.delete_country_from_location(se.Location))
     join disasters_dwh.dim_date dd on (dd.Year = se.Year AND dd.Month <=> se.Mo
AND dd.Day <=> se.Dy), disasters_dwh.dim_eventtype de
WHERE
     de.EventTypeName = 'earthquake';
```

Додаток Д Засоби інкрементного завантаження та реалізації Slowly Changing Dimensions type 2

extraction.py

```
def incremental_extraction(self):
    self.tsunami_extraction_initial(tsunamis_file_name_incremental)
```

sql-скрипт інкрементного завантаження

```
-- date
INSERT INTO disasters_dwh.dim_date(Year, Month, Day)
SELECT DISTINCT Year, Mo, Dy
FROM disasters_stage.tsunamievent stage
WHERE Year IS NOT NULL AND
    NOT EXISTS(
    SELECT DateID
    FROM disasters_dwh.dim_date dd
    WHERE dd.Year = stage.Year AND dd.Month <=> stage.Mo AND dd.Day <=> stage.Dy
    );

-- location
INSERT INTO disasters_dwh.dim_location(CountryID, LocationName, Latitude,
Longitude)
SELECT DISTINCT CountryID, LocationName, Latitude, Longitude
        FROM disasters_stage.tsunamievent l left join disasters_dwh.dim_country
ds on CONCAT('%',LOWER(l.Country),'%') like
              CONCAT('%', ds.CountryName, '%')
WHERE NOT EXISTS(
    SELECT LocationID
    FROM disasters_dwh.dim_location dl
    WHERE dl.CountryID <=> ds.CountryID
      AND CONCAT('%', dl.LocationName, '%') like CONCAT('%', l.LocationName,
'%')
      AND dl.Latitude <=> l.Latitude
      AND dl.Longitude <=> l.Longitude
    );

-- tsunami-to-fact-table
INSERT INTO disasters_dwh.fact_event (LocationID, StartDateID, EndDateID,
EventTypeID, EventNameID, Deaths, DeathsDescription, Missing,
MissingDescription, Injuries, InjuriesDescription, DamageMillionsDollars,
DamageMillionsDollarsDescription, HousesDamaged, HousesDamagedDescription,
HousesDestroyed, HousesDestroyedDescription, TotalDeaths,
TotalDeathsDescription, TotalMissing, TotalMissingDescription, TotalInjuries,
TotalInjuriesDescription, TotalDamageMillionsDollars,
TotalDamageMillionsDollarsDescription, TotalHousesDamaged,
TotalHousesDamagedDescription, TotalHousesDestroyed,
TotalHousesDestroyedDescription)
SELECT dl.LocationID, dd.DateID, null , de.EventTypeID, null ,
       Deaths, find_damage_description_id('people', DeathsDescription),
       Missing, find_damage_description_id('people', MissingDescription),
       Injuries, find_damage_description_id('people', InjuriesDescription),
       DamageMillionsDollars, find_damage_description_id('damageMillionDollars',
DamageMillionsDollarsDescription),
       HousesDamaged, find_damage_description_id('houses',
HousesDamagedDescription),
       HousesDestroyed, find_damage_description_id('houses',
HousesDestroyedDescription),
       TotalDeaths, find_damage_description_id('people',
TotalDeathsDescription),
       TotalMissing, find_damage_description_id('people',
TotalMissingDescription),
       TotalInjuries, find_damage_description_id('people',
```

```
TotalInjuriesDescription),
       TotalDamage, find_damage_description_id('damageMillionDollars',
TotalDamageDescription),
       TotalHousesDamaged, find_damage_description_id('houses',
TotalHousesDamagedDescription), TotalHousesDestroyed,
find_damage_description_id('houses', TotalHousesDestroyedDescription)
FROM
    disasters_stage.tsunamievent st left join disasters_dwh.dim_location dl
        on dl.Latitude <=> st.Latitude AND dl.Longitude <=> st.Longitude AND
CONCAT('%', dl.LocationName, '%') like CONCAT('%', st.LocationName, '%')
        join disasters_dwh.dim_date dd on (st.Year = dd.Year AND st.Mo <=>
dd.Month AND st.Dy <=> dd.Day), disasters_dwh.dim_eventtype de
WHERE
    de.EventTypeName = 'tsunami'
```

sql-скрипт процедури, що забезпечує SCD type 2 для виміру dim_Country:

```
DROP PROCEDURE IF EXISTS disasters_dwh.scd_country;

DELIMITER $$
CREATE PROCEDURE disasters_dwh.scd_country(old_name varchar(70), new_name
varchar(70))
BEGIN
    DECLARE old_id int DEFAULT null;
    DECLARE old_subCont_id int;
    DECLARE new_id int;

    SELECT CountryID, SubContinentID into old_id, old_subCont_id
    FROM dim_country
    WHERE CountryName = old_name;
  IF old_id IS NULL THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Old country name has not been
found!';
    ELSE
      INSERT INTO dim_country (SubContinentID, CountryName, Load_Date,
Old_reference_ID)
          VALUE(old_subCont_id, new_name, CURRENT_DATE, old_id);
      UPDATE dim_country
      SET End_Load_Date = CURRENT_DATE
      WHERE CountryID = old_id;

      SELECT CountryID into new_id
      FROM dim_country
      WHERE CountryName LIKE new_name;
      UPDATE dim_location
      SET CountryID = new_id
      WHERE CountryID = old_id;
      -- NEEED TO FINISH!!!!!!!!!!!!!!!!!!!!!
  end if ;
end $$

DELIMITER ;
```