SQL ЗАПИТИ ДЛЯ ГІС

Кривий Ріг
Видавець Роман Козлов
2022

УДК 681.518.3:528
ББК 32.973.202(26.12)
Н73

Рецензенти: завідувач кафедри геоінформаційних систем, оцінки землі та нерухомого майна Харківського національного університету міського господарства імені О. М. Бекетова, доктор економічних наук, професор К. А. Мамонов;

професор кафедри геоекології та землеустрою Таврійського державного агротехнологічного університету, доктор технічних наук, Л. М. Даценко;

декан бакалаврату Інституту геодезії Національного університету «Львівська політехніка», кандидат технічних наук, доцент М. С. Маланчук.

Новікова О. М., Паламар А. Ю., Мазикіна О. Б., Сидоренко В. Д. SQL запити для ГІС : навчальний посібник / Олена Миколаївна Новікова, Альона Юріївна Паламар, Ольга Борисівна Мазикіна, Віктор Дмитрович Сидоренко. – Кривий Ріг : Видавець Р. А. Козлов, 2022. – 130 с.

Н 73

ISBN 978-617-8096-06-9

Навчальний посібник присвячений окремій, досить складній темі: «SQL запити» дисципліни «ГІС і бази даних». Зважаючи на те, що SQL мова є частиною реляційної бази даних, в рамках посібника коротко розглянуті елементи реляційної моделі та складові SQL запиту. Більша частина посібника присвячена SQL запитам для ГІС на прикладі найбільш розповсюдженої геоінформаційної системи MapInfo. Теоретичний матеріал супроводжується великою кількістю практичних прикладів та задач.

Посібник складений відповідно до програми нормативної дисципліни «ГІС і бази даних», яку вивчають студенти спеціальності «Геодезія та землеустрій» першого (бакалаврського) рівня.

Посібник призначений для студентів спеціальності «Геодезія та землеустрій». Він може бути корисним викладачам з дисципліни «ГІС і бази даних», а також аспірантам, магістрантам, студентам та науковим співробітникам в галузі геодезії, картографії та землеустрою.

УДК 681.518.3:528
ББК 32.973.202(26.12)

ISBN 978-617-8096-06-9

Зміст

Вступ

Як тільки були створені перші комп'ютери, користувачі почали вводити інформацію в їх пам'ять. Спочатку цієї інформації було досить мало, в ній легко було орієнтуватися і знаходити потрібну. Але згодом інформації стало вже настільки багато, що з'явилася необхідність її впорядкувати, тобто створити певну структуру. Так були розроблені комп'ютерні програми, які мають назву «Бази даних».

Отже уявимо, що існує потужна база даних, наприклад по населеним пунктам Криворізького району. В цій базі по кожному населеному пункту є безліч самої різноманітної інформації, про кількість мешканців, про їх вік, стать, місце роботи, навіть про їх вакцинування. Необхідно з цієї бази даних обрати інформацію, наприклад, про дітей, які вчаться у першому класі. Можна відкрити базу даних і розпочати пошук потрібної інформації. Але, якщо в базі є декілька терабайт інформації, то виникає питання, коли скінчиться пошук: через день, або через місяць. Саме тому для обслуговування реляційної бази даних була створена спеціальна комп'ютерна мова, SQL. В рамках цього навчального посібника розглядаються такі питання, як історія появи та склад реляційної моделі бази даних, мова SQL, та один з головних операторів мови - оператор Select.

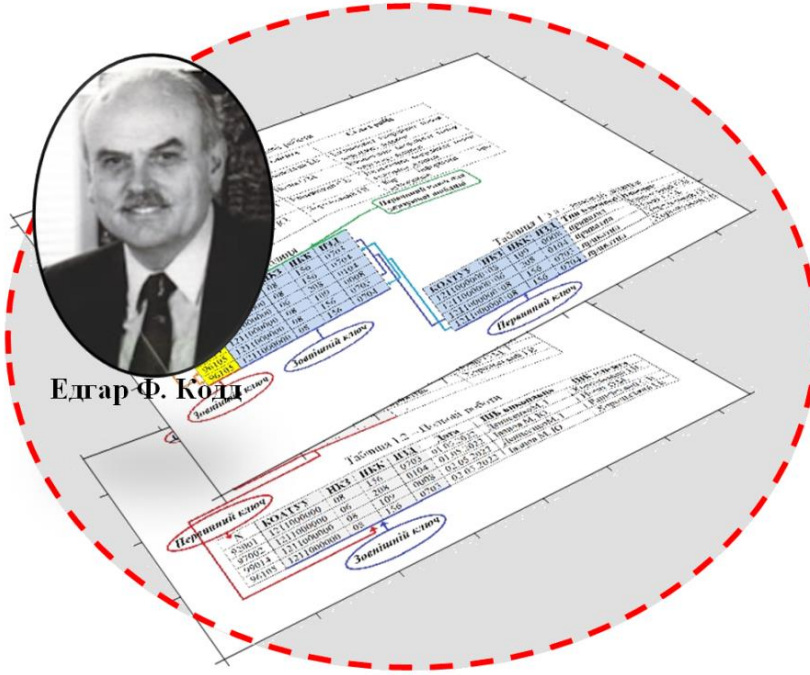
Вивчення SQL дає навички, необхідні для отримання інформації з будь-якої реляційної бази даних. Знання та розуміння SQL дозволяє проектувати як прості, так і складні запити до бази даних. Оскільки SQL застосовується в самих різних програмних продуктах, достатньо вивчити основи SQL, щоб далі використовувати його для роботи в різноманітних програмах, таких як класичні бази даних: Microsoft Access, Sybase SQL Server, та геоінформаційні системи: MapInfo, ArcGIS. AutoCAD тощо.

Головна мета цього посібника – навчити студентів виконувати запити саме до даних в геоінформаційних

системах. Тому більша частина посібника присвячена створенню SQL запитів в найбільш відомій та розповсюдженій ГІС - MapInfo. В рамках цього посібника розглянуті особливості процедури вибору об'єктів в MapInfo, варіанти формування SQL запитів. Показано, що є дві команди створення SQL запитів, кожна з яких має свої особливості та можливості. Зазначена аналогія між реченням звичайної мови та виразом для запиту. Проаналізовані можливості використання основних елементів запиту, таких як константи, колонки, оператори, функції та ключові слова. Особливу увагу приділено таким елементам, які є унікальними для ГІС, а саме: географічним операторам та функціям. Теоретичний матеріал супроводжується великою кількістю практичних прикладів та задач, найбільш складні з яких проілюстровані детальними рішеннями.

Навчальний посібник складений відповідно до робочої програми нормативної дисципліни «ГІС і бази даних», розробленої на основі галузевого стандарту вищої освіти підготовки бакалаврів спеціальності 193 «Геодезія та землеустрій». Він може бути корисним не тільки студентам вищезгаданої спеціальності, а також викладачам з дисципліни «ГІС і бази даних», аспірантам, магістрантам, та науковим співробітникам в галузі геодезії, картографії та землеустрою.

1. Реляційна модель



«Геніальна·…,·революційна·робота·видатного·математика-програміста·Едгара·Кодда·–·…розробка·ідеї·реляційної·моделі…»·¶

«Інтернет-видання·«ToWave»¶

1.1. Моделі представлення даних

На даний час існує досить багато визначень поняття бази даних [1,5,6,9,13]. Найбільш поширене визначення формулюється так:

База даних є впорядкованим набором даних, який використовується для моделювання деякого типу організації або організаційного процесу.

Перші бази даних створювалися на папері. В даний час для збору та зберігання даних використовуються, в основному комп'ютерні програми, які мають назву баз даних.

Якби дані не мали структури, то дуже скоро накопичена інформація перетворилася б у «річ в собі» (*Ding an sich)*, тобто у таку річ, яку не можливо використовувати. Тому, щоб не втратити дані, їх потрібно структурувати.

Дані, що зберігаються в базі, мають певну логічну структуру, яка називається моделлю представлення даних. До класичних моделей відносяться [1, 5, 6]:

• Ієрархічна;
• Мережева;
• Реляційна.

Крім того, в останні роки з'явилися і стали більш ефективно використовувати такі моделі даних [1, 5, 6, 13]:

• Постреляційна;
• Багатомірна;
• Об'єктно-орієнтована.

У деяких базах даних одночасно підтримуються кілька моделей даних, однак найбільш поширеними є саме класичні моделі. Саме тому, коротко розглянемо кожну з класичних моделей.

1.1.1. Ієрархічна модель

Інформація у комп'ютері має певну структуру. Вона розділена на диски та зберігається у файлах. Усередині диска

файли об’єднуються у папки. Така структура називається ієрархічною.

В ієрархічній моделі зв’язки між даними можна описати за допомогою впорядкованого графа, який називається деревом [1, 6, 9, 13] (див. рис. 1.1).

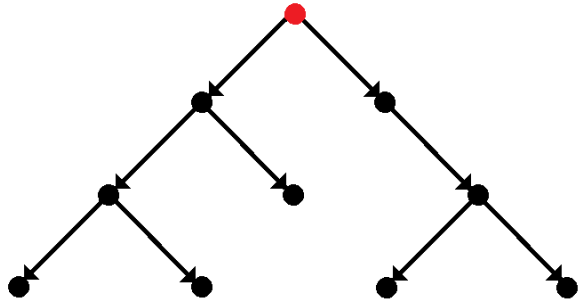


Рис. 1.1 – Схема ієрархічної моделі даних

В ієрархічній моделі всі елементи, які називаються записами, зв’язані між собою зв’язком, в якому один елемент підпорядкований іншому. Саме тому один запис називається предком, інший - нащадком. Є запис в ієрархічній моделі, який має особливе значення, це кореневий запис. Він не має предка. На схемі рис. 1.1 він показаний червоним кольором. Обхід всіх записів ієрархічної бази даних зазвичай виконується зверху донизу від кореневого типу до підлеглих. Між двома записами існує лише один зв’язок, що дозволяє визначити, який запис є предком, а який – нащадком.

Для того, щоб база даних існувала як єдине ціле, вона повинна задовольняти певним правилам. Ці правила називаються правилами контролю цілісності. Основне правило контролю цілісності ієрархічної бази даних формулюється так [5, 13]:

Нащадок не може існувати без предка, у предків може бути довільна кількість нащадків.

Прикладом ієрархічної бази даних є класифікатор програми «Credo», де зберігається інформація про умовні знаки, що використовуються в проекті. Дані, про земельну

ділянку, які зберігаються в обмінному файлі кадастрових даних з розширенням *.xml, також мають ієрархічну структуру. На рис. 1.2 показана структура цього файлу, побудована в Менеджері Обмінних файлів.

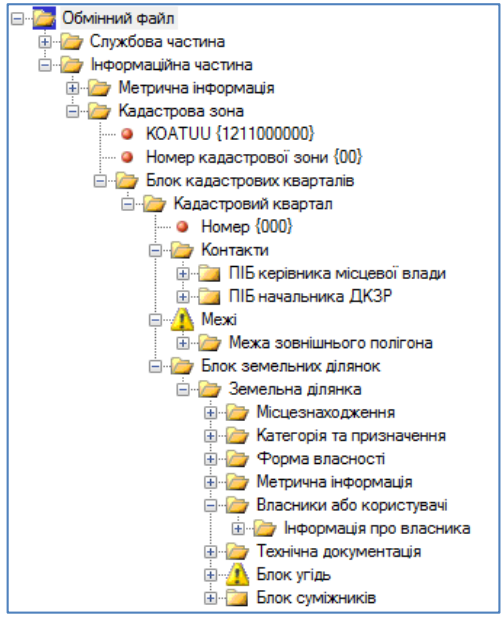


Рис. 1.2 – Ієрархічна структура обмінного файлу земельної ділянки

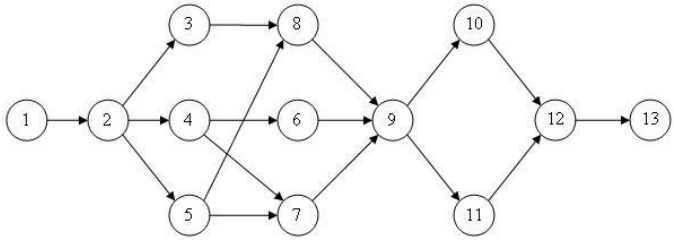
В 60-х роках минулого століття саме ця модель використовувалася в перших базах даних, таких як DBOMP (https://ru.wikipedia.org/wiki/DBOMP).

Однією з переваг ієрархічної моделі є можливість швидкого пошуку за простими запитами. Однак, якщо запит є складним, то ієрархічна модель виявляється досить громіздкою.

Ієрархічна модель зручна для роботи з ієрархічно упорядкованою інформацією. Якщо інформація має іншу структуру, ієрархічна модель стає не ефективною. Саме тому була створена модель даних з більшими можливостями, ніж ієрархічна. Це мережева модель, яка узагальнює ієрархічну модель.

1.1.2. Мережева модель

Класичним прикладом мережевої моделі є мережевий графік, який використовується для організації, планування та керування роботою в економіці (див. рис. 1.3).



1.3 - Мережевий графік планування робіт, згідно [12]

Мережева модель даних дозволяє відображати різноманітні взаємозв'язки елементів даних за допомогою довільного орієнтованого графа [1, 5, 9], так як це показано на рис. 1.4.

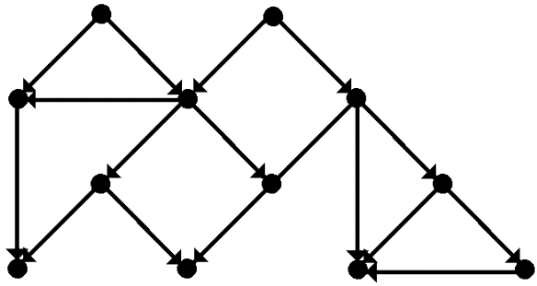


Рис. 1.4 – Схема мережевої моделі даних

Як і ієрархічна, мережева модель складається з набору записів та набору відповідних зв'язків. Записи можуть бути предками або нащадками. Однак, на відміну від ієрархічної моделі один нащадок може мати довільну кількість предків.

Перевагами цієї моделі є [6,9]:

- економія витрат пам'яті ЕОМ;
- великі можливості при утворенні довільних зв'язків;
- більша гнучкість пошуку даних у порівнянні з ієрархічною моделлю.

Недоліком мережевої моделі є [5-6]:

- висока складність та жорсткість схеми бази даних, побудованої на її основі;
- складність для розуміння та обробки інформації;
- ослаблений контроль цілісності зв'язків між записами.

Мережева модель даних реалізована в схемі документообігу програми 1С:Підприємство, як це показано на рис. 1.5.

Рис. 1.5 – Використання мережевої моделі даних для реалізації документообігу в 1С:Підприємстві, згідно (https://www.ekam.ru/blogs/pos/1s-upravlenie-torgovley)

1.2. Історія виникнення реляційної моделі

Як ієрархічна так і мережева моделі мали суттєві проблеми, такі як погана цілісність даних та їх дублювання. Досить проблематично було управляти великими базами даних, сформованих по цим моделям. Тому в 60-х роках минулого століття стало ясно, що необхідна інша модель для баз даних, яка б не мала недоліків, притаманних вищезгаданим моделям. В 1970 році доктор Едгар Ф. Кодд опублікував роботу [16], в якій сформулював основні принципи формування реляційної бази даних. Оскільки бази даних, засновані на інших моделях, не задовольняли потребам користувачів баз даних, а реляційна база даних

поки теоретично мала суттєві переваги, то різноманітні організації, такі як університети, дослідницькі лабораторії та комерційні фірми, почали розробляти мову, яка могла би використовуватися як фундамент для баз даних, що підтримують реляційну модель.

1.3. Елементи реляційної БД

Відповідно до реляційної моделі дані в реляційній базі даних зберігаються у відносинах (relations), які сприймаються користувачем як таблиці. Однак, слід зазначити, що таблиця реляційної моделі не є в повному розумінні таблицею, до якої всі звикли, і яку можна побудувати на папері, або в програмі Excel. Таблиця реляційної моделі повинна підкорятися жорстким правилам, які називаються нормальними формами (normal forms) [16]. Нехтування цим правилам призводить до неприємних наслідків, коли керування базою стає дуже складною процедурою. Сам автор реляційної бази даних, сформулював 12 правил [17]. Згодом ці правила були переглянуті, переосмислені та скорочені. Зараз використовують 8 правил, які, як було сказано вище, мають назву нормальних форм [18].

1.3.1. Таблиці

Для того, щоб зрозуміти різницю між звичайною таблицею та таблицею реляційної бази даних, розглянемо структуру таблиці на прикладі таблиць бази даних геодезичної фірми. Нижче представлена табл. 1.1[1] з цієї бази даних, яка містить інформацію про співробітників фірми.

[1] Таблиці, представлені в розд. 1-2, належать до навчальної бази, дані якої були складені авторами цього підручника та студентами спеціальності «Геодезія та землеустрій» методом випадкової вибірки. Зміст таблиць не має нічого спільного з реальними даними.

Таблиця 1.1 – Співробітники

Код	Ім’я	Прізвище	По_батькові	Вулиця_буд _кв	Місто	Область	Поштовий_ індекс	
1010	Михайло	Денисенко	Іванович	Миру, 20, 3	Київ	-	01005	**Записи (рядки)**
1011	Маргарита	Панасюк	Михайлівна	Пушкіна, 1,1	Херсон	Херсонська	73000	
1012	Олена	Лужанська	Ігорівна	Гагаріна, 2	Кривий Ріг	Дніпропетровська	50027	
1013	Михайло	Іванов	Вадимович	Зелена, 19, 4	Кривий Ріг	Дніпропетровська	53800	
1014	Руслан	Сергєєв	Денисович	Зарічна, 49	Слов’янськ	Донецька	84100	
Поля (колонки)								

Таблиця 1.2 – Польові роботи

N	Дата	Виконавець_ Прізвище	Клієнт_ Прізвище	Склад_роботи
92001	01.05.2022	Денисенко.	Хорольський	Визначення координат точок земельної ділянки
97002	01.05.2022	Іванов	Носик.	Визначення координат точок земельної ділянки
99014	02.05.2022	Денисенко.	Рашевський.	Визначення координат точок земельної ділянки
96105	02.05.2022	Іванов	Хорольський	Збір інформації про суміжників

Таблиця є основною структурою реляційної бази даних. Як і звичайна таблиця, вона має рядки та колонки. Кожен рядок таблиці належить певному предмету, який може бути або об'єктом, або подією. Коли предмет є об'єктом, таблиця описує щось реальне. Це можуть бути клієнти, населені пункти, річки. Якщо предметом є подія, то таблиця представляє щось, що відбувається в заданий момент часу. Незалежно від свого типу, кожен предмет має характеристики, які зберігаються в колонках, які в реляційній базі даних називаються полями. В табл. 1.2 наведений приклад таблиці, яка описує подію, виїзд бригади геодезистів до клієнта для виконання комплексу геодезичних та землевпорядних робіт.

Поле (колонка) є характеристикою предмета таблиці, до якої належить. Поля реально використовуються для зберігання даних.

Якщо структура бази даних сформована не належним чином, обов'язково з'являться проблеми. Наприклад цілісність даних, зв'язки таблиць і можливість отримати точну інформацію порушуються, якщо структура бази спроектована погано. Для того, щоб уникнути деяких з цих проблем достатньо під час створення бази даних виконувати досить прості правила. Частина з цих правил пов'язана з назвою таблиць та колонок в таблицях.

Правило 1 напряму зв'язано зі стандартом SQL [19], згідно якому назва таблиць та колонок в таблицях повинна бути регулярним ідентифікатором, тобто починатися з букви і може містити тільки букви, цифри і символ підкреслення; пропуски не допускаються. Оскільки багато версій SQL підтримують тільки регулярні ідентифікатори, рекомендовано використовувати для назв таблиць та полів саме це правило.

Правило 2. Як вказано в [13, 14], таблиця повинна представляти тільки один предмет. Ознакою того, що таблиця представляє більш ніж один предмет, можуть служити такі

слова в назві, як «і», «та», «або», чи такі символи, як похила риса (\), (/), дефіс (-), амперсанд (&). Аналогічно, назва поля повинна бути однозначною. Використання в назві вищезгаданих літер та символів може свідчити про неоднозначність самого поля, що є протиріччям першої нормальної форми. Більш докладно поняття неоднозначності полів розглянуто в розд. 1.3.4.

Якщо таблиця, або поле таблиці представляє більш ніж один предмет, таблицю і поле необхідно розділити.

Правило 3. Згідно [8, 14], назва таблиці та кожного її поля повинна бути ясною для розуміння кожного користувача бази даних.

Правило 4. Назва таблиці повинна чітко відображати предмет, та бути унікальною для бази даних. Унікальні назви таблиць свідчать про той факт, що кожна таблиця у базі даних представляє інший предмет [8, 14].

Аналогічно, необхідно уникати однакових назв полів в різних таблицях. Вони також повинні бути унікальними. Якщо в різних таблицях використовуються одні й ті ж самі об’єкти, наприклад Прізвище, або Місто (як частина адреси), є досить простий прийом, який дозволяє зробити поля в базі даних унікальними. Згідно цьому прийому, перед основною назвою поля ставиться повна, або скорочена назва таблиці. Наприклад, в двох таблицях Клієнти та Співробітники є поле Прізвище. Достатньо в першій таблиці використати назву Клієнт_Прізвище, в другій – Співробітник_Прізвище. Таким чином, кожне поле у базі даних набуває унікальну назву у всій структурі бази даних. Єдиним виключенням з цього правила є випадок, коли поле використовується для установки зв'язку між двома таблицями. Нижче, на прикладі програми MapInfo, буде показано, як автоматично використовується вищеописаний прийом під час встановлення зв’язку між таблицями.

Правило 5. Для вибору назви таблиці або поля не бажано використовувати мало відомі акроніми[2], та скорочення.

Правило 6. Бажано, щоб назва таблиці використовувалася у множині, а назви полів – в однині. Множина для таблиці необхідна тому, що таблиця зберігає набір екземплярів предмета таблиці. З другого боку назва поля використовується в однині, тому що поле є окремою характеристикою предмету, описуваного таблицею, до якої воно належить. Використання цього правила допоможе відрізнити назву таблиці від назви поля [5, 8, 14].

1.3.2. Ключі

Зверніть увагу на перші колонки таблиць 1.1-1.2. Вони мають різні назви (Код - в табл. 1.1, N – в табл. 1.2), але не дають ніякої кількісної або якісної інформації про об'єкти, які описуються в таблицях. Це особливі колонки, які називаються первинними ключами таблиці.

Первинний ключ — це поле (або група полів), яке унікальним чином ідентифікує кожний запис в таблиці.

Кожна таблиця повинна мати первинний ключ. В таблицях 1.1-1.2 первинний ключ займає тільки одну колонку, такий ключ називається простим. Однак, в деяких випадках ідентифікація об'єкта в таблиці потребує не одного а декількох полів. Наприклад, під номером 1.3 дається фрагмент таблиці з характеристикою земельних ділянок, на які було складено проекти відведення. Кадастровий номер земельної ділянки є унікальним і може бути використаний в якості первинного ключа. Однак відомо, що кадастровий номер складається з декількох частин, кожна частина

[2] Акроніми – скорочення, які вимовляються разом, наприклад виш, НАТО, НАСА. Деякі з них стали самостійними словами та вживаються як іменники з відповідними відмінками, наприклад виш, вишу, вишах тощо. На відміну від акронімів, класичні скорочення вимовляються по буквам, наприклад РНБО – «ер-ен-бе-о». (http://ru.wikipedia.org/wiki/).

визначає положення ділянки в окремій структурі. Наприклад перші 10 цифр номера, які мають назву КОАТУУ визначають положення ділянки всередині одиниці адміністративно-територіального устрою України, наступні дві цифри, визначають номер кадастрової зони, всередині якої знаходиться ділянка. Більш детально структуру кадастрового номера земельної ділянки див. [10].

Таблиця 1.3 – Земельні ділянки[3]

КОАТУУ	НКЗ	НКК	НЗД	Тип_власності	Власник_Прізвище
1211000000	08	109	0008	приватна	Рашевський
1211000000	06	208	0104	приватна	Носик
1211000000	08	156	0703	приватна	Хорольський
…	…	…	…	…	…

Якщо первинний ключ складається з двох або більш полів, він називається **складовим первинним ключем**. Табл. 1.3 представляє складовий первинний ключ, створений на основі кадастрового номера земельної ділянки. Поля складового первинного ключа зафарбовані сірим кольором.

Вважається [13], що більш кращим у застосуванні є простий, а не складовий первинний ключ, оскільки його легше застосовувати при створенні зв'язків між таблицями.

Зважаючи на те, що первинний ключ є найбільш важливим полем таблиці, до нього пред'являють особливі вимоги, а саме [6, 13]:

1. Поле первинного ключа повинно однозначно ідентифікувати кожний предмет в таблиці. Не повинно бути однакових значень в рядках цього поля.

[3] Табл. 1.2 та 1.3 не задовольняють вимогам, які предявляються до таблиць. Зокрема, вони містять інформацію не про один, а про декілька об'єктів. Наприклад, в табл. 1.3 є інформація про ділянки та про власників цих ділянок. Однак, не будучи досконалими, вони дозволяють наочно показати процедуру встановлення зв'язку між таблицями.

2. Поле первинного ключа не повинно бути складовим. В даному випадку не слід плутати такі поняття як складове поле та складовий первинний ключ. Приклад складового первинного ключа представлений в табл. 1.3. Такий ключ складається з декількох полів. Складове поле розглянуте нижче в розд. 1.3.4.
3. Поле первинного ключа не повинно бути багатозначним. Приклад багатозначного поля також розглянутий в розд. 1.3.4.
4. Поле первинного ключа не повинно містити невідомі або необов'язкові значення.

Поле повинне задовольняти всім вимогам даного списку, для того, щоб його можна було використовувати в якості первинного ключа. Якщо в таблиці немає поля, яке б задовольняло вимогам, які висуваються до первинного ключа, то досить просто додати поле з номерами, так як це показано в табл. 1.1-1.2. Таке поле повністю відповідає всім вимогам, які висуваються до первинного ключа.

Це довільне поле, яке визначається і додається до таблиці з єдиною метою: використовувати його в якості первинного ключа. Перевага додавання довільного поля полягає в тому, що воно гарантовано задовольняє всім вимогам, які висуваються до полів первинного ключа.

Крім первинних ключів в базі даних використовуються зовнішні ключі. Вони застосовуються під час створення зв'язків між таблицями, які розглядаються нижче.

1.3.3. Зв'язки між таблицями

Геодезична фірма виконує роботи по оформленню документації на земельні ділянки. Частина цих робіт відображена в табл. 1.2. В табл. 1.3 надана інформація по цим ділянкам. Очевидно, обидві таблиці зв'язані між собою. Роботи, які відображені в табл. 1.2, виконувалися для

ділянок, представлених в табл. 1.3. Можна було б відобразити всю інформацію табл. 1.2-1.3 в одній таблиці, однак це суперечить умовам формування реляційної бази даних, оскільки обидві таблиці описують різні об’єкти. Тому, ще одним елементом реляційної бази даних є зв’язки між таблицями.

Якщо записи однієї таблиці можуть бути пов'язані деяким чином із записами в іншій таблиці, то в цьому випадку між таблицями можна встановити зв'язок. Спосіб встановлення цього зв'язку залежить від типу зв'язку. Розрізняють наступні типи зв’язків між таблицями[14]:

- Один до одного;
- Один до декількох;
- Декілька до декількох.

На рис. 1.6 наочно показаний зв’язок між таблицями 1.2 та 1.3. Незалежно від типу зв’язку, одна з таблиць обирається в якості первинної, друга – в якості вторинної. На рис. 1.6 первинною є табл. 1.3, вторинною – табл. 1.2.

Польові роботи по визначенню різноманітної інформації по земельним ділянкам здійснювалися різними виконавцями в різні дати, так як це показано в табл. 1.2. Вони виконувалися для конкретних ділянок, інформація по яким показана в табл. 1.3. Саме тому між таблицями 1.2 та 1.3 є зв’язок. При встановленні зв’язку між двома таблицями беруть копію первинного ключа з першої таблиці і вставляють її в другу таблицю, де вона стає зовнішнім ключем. На рис. 1.6 показана процедура переносу первинного ключа першої таблиці (табл. 1.3) в другу таблицю (табл. 1.2) та перетворення його в зовнішній ключ. Таким чином в другій таблиці представлені два ключа, перший – власний первинний ключ, другий – первинний ключ з першої таблиці, який для другої таблиці стає зовнішнім ключем.

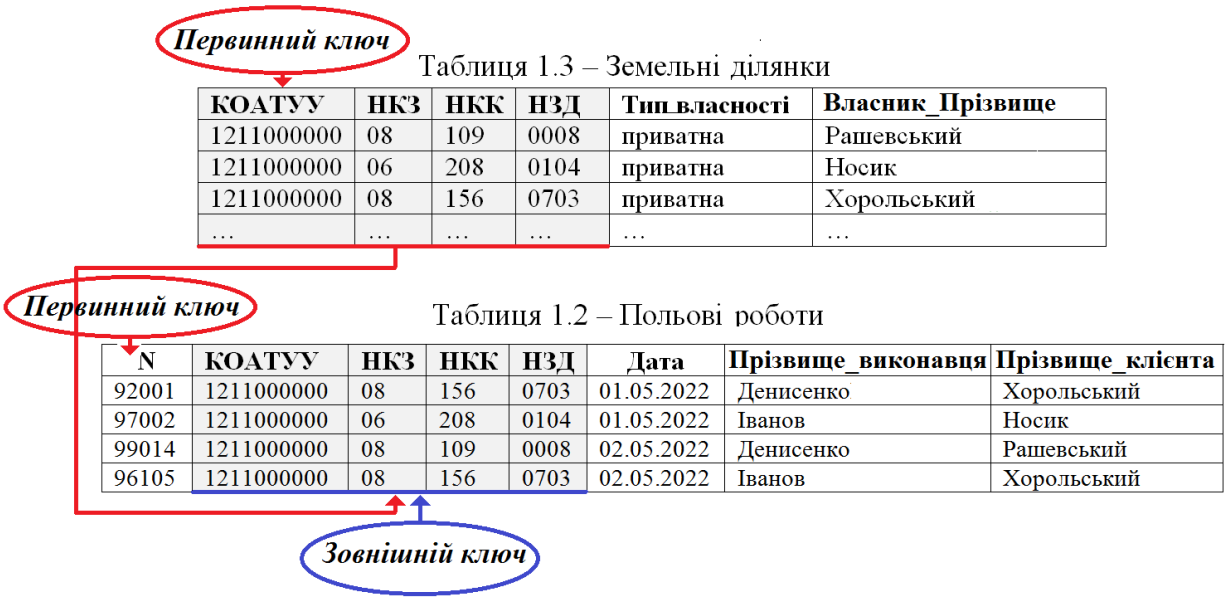


Рис. 1.6 – Процедура зв’язування таблиць на прикладі зв’язку між табл. 1.2 та 1.3

Як вказано в [6, 9, 11, 14], зовнішній ключ грає важливу роль в реляційній базі даних, не тільки тому що він допомагає встановити зв’язок між двома таблицями. Він також забезпечує цілісність на рівні зв’язків бази даних. Розглянемо варіанти зв’язків між таблицями більш детально.

Зв'язок «один-до-одного»

Згідно [6, 11, 14], дві таблиці зв’язані відношенням один-до-одного, якщо один запис в першій таблиці пов’язаний тільки з одним записом в другій таблиці, а один запис в другій таблиці пов’язаний тільки з одним записом в першій таблиці.

Це особливий тип зв’язку, який не є поширеним. Його встановлюють в тому випадку, коли частину таблиці хочуть приховати від всіх користувачів, залишивши доступ тільки для обраних. Під час встановлення зв’язку, одна з таблиць повинна бути обрана як первинна, інша – як вторинна. Для цього типу зв’язку в більшості випадків не має значення, яка таблиця буде обрана як первинна, а яка вторинна.

Зв'язок «один-до декількох»

У випадку, коли пара таблиць зв’язана між собою по типу один-до-декількох, один запис в першій таблиці може бути зв’язаний з декількома записами в другій таблиці, але один запис в другій таблиці може бути зв’язаний тільки з одним записом в першій таблиці. При встановленні такого зв'язку береться первинний ключ в таблиці зі сторони «один» і вставляється в таблицю зі сторони «декількох», в якій він стає зовнішнім ключем. На рис. 1.6 показаний зв’язок саме «один до декількох». На одній і тій же земельній ділянці можуть виконувати геодезичні та землевпорядні роботи різні виконавці в різні інтервали часу. Зверніть увагу на земельну ділянку з номером 1211000000:08:156:0703. Польові роботи на цій ділянці виконувалися 01.05.2022 та 02.05.2022.

Зв'язок «декілька до декількох»

Дві таблиці зв’язані відношенням «Декілька до декількох», коли один запис в першій таблиці може бути пов’язаний з декількома записами в другій таблиці, а один запис в другій таблиці може бути пов’язаний з декількома записами в першій таблиці. Це найбільш складний тип зв’язку. Для його коректного встановлення створюється додаткова зв’язуюча таблиця, яка ділить зв’язок «декілька до декількох» на два зв’язки один до декількох. Розглянемо цей зв’язок на наступному прикладі. Як відомо один власник може мати не одну, а декілька ділянок. В табл. 1.3а представлений саме такий варіант. Власник Хорольський має дві ділянки, які вимірювалися в одні й ті ж самі дні 01.05.2022 та 02.05.2022. З появою двох ділянок, які належать одному власнику, зв’язок «один до декількох» між табл. 1.3 та 1.2 руйнується, натомість з’являється зв’язок «декілька до декількох», який наочно показаний на рис. 1.7.

Таблиця 1.3 а – Земельні_ділянки

КОАТУУ	НКЗ	НКК	НЗД	Тип власності	Власник_ Прізвище
1211000000	08	109	0008	приватна	Рашевський .
1211000000	06	208	0104	приватна	Носик .
1211000000	08	156	0703	приватна	Хорольський.
1211000000	08	156	0704	приватна	Хорольський

Таким чином між табл. 1.3а та 1.2 існує зв’язок типу «Декілька до декількох». Дві ділянки з таблиці 1.3.а зв’язані з двома записами про виконання польових робіт в табл. 1.2.

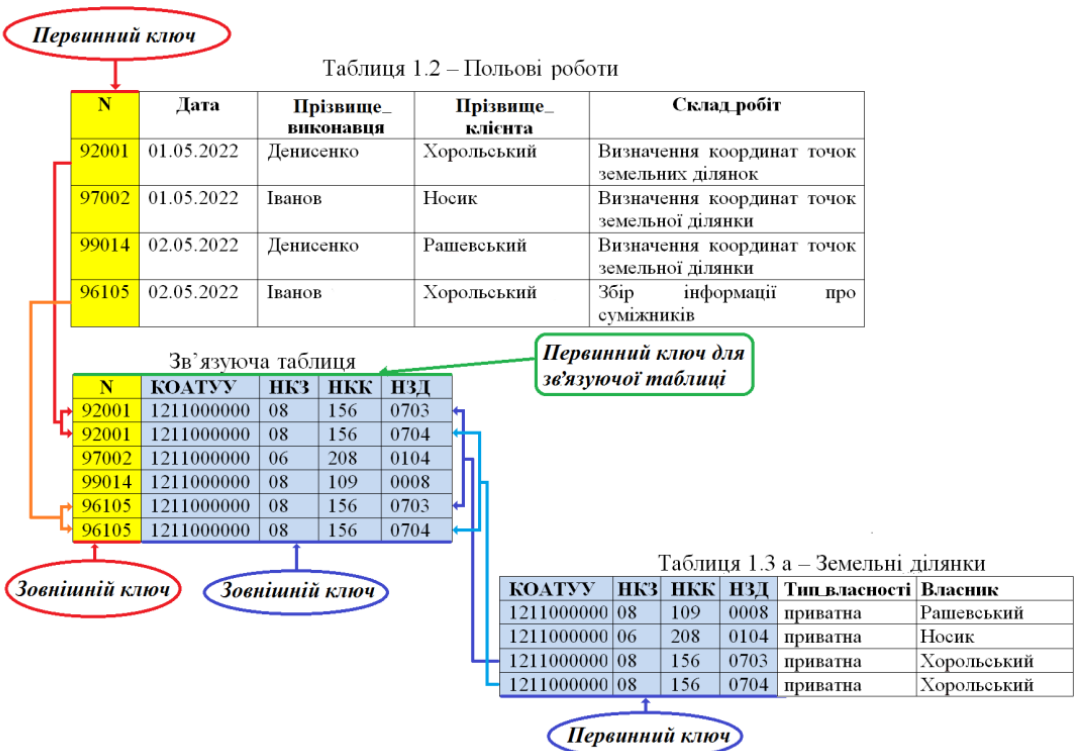


Рис. 1.7 - Схема зв'язку типу «декілька до декількох»

На рис. 1.7 показана процедура формування зв'язуючої таблиці. Згідно цьому рисунку, копії первинних ключів вихідних таблиць копіюються в зв'язуючу таблицю. В цій таблиці два поля одночасно є зовнішніми полями та складовим первинним ключем зв'язуючої таблиці.

Зв'язки грають важливу роль при створенні запитів в тому випадку, коли запити формуються до декількох таблиць. Більш детально запити розглянуто в розд. 2, 3.

1.3.4. Нормальні форми

Згідно [20], перші 5 правил нормальних форм були сформульовані доктором Уїльямом Кентом. В подальшому вони були доповнені ще трьома нормальними формами [6, 14, 18]. Процес перетворення структури бази даних таким чином, щоб вони задовольняли нормальним формам, називається нормалізацією. Для нормалізації бази даних іноді достатньо дотримання перших трьох нормальних форм. Розглянемо їх більш детально.

Перша нормальна форма (1NF): Таблиця повинна складатися з рядків і колонок і, оскільки колонки використовуватимуться як ключі пошуку, в кожній з них на кожному рядку повинне знаходитися тільки одне значення. Це значення повинне бути скаляром [11, 14].

Головний висновок з першої нормальної форми – поле в таблиці повинно бути однозначним.

Поле, потенційно зберігаюче декілька екземплярів одного і того ж значення, називається **багатозначним** полем.

Поле, яке може потенційно зберігати два або більш різних значень, називається **складовим** полем.

Багатозначні і складові поля не задовільняють правилу першої нормальної форми і можуть зруйнувати базу даних, особливо при спробах редагування, видалення або сортування даних. Тому, при проектуванні бази даних необхідно уникати багатозначних та складових полів. Якщо

ж база даних спроектована з порушенням першої нормальної форми, то такі поля необхідно перетворити.

Табл. 1.4 має ту ж саму назву та містить ту ж саму інформацію, що й табл. 1.1. Однак, на відміну від табл. 1.1, інформація в табл. 1.4 подана в більш компактному вигляді, та додані ще дві колонки: Телефон/Факс та Сертифікати.

Табл. 1.4 містить як складові, так і багатозначні поля. Складові поля позначені блакитним кольором, багатозначні – жовтим. Складовими полями є такі поля як «ПІБ», «Адреса». Як зазначено в [11, 14], складові поля досить легко перетворити на однозначні. Для цього достатньо розділити інформацію складових полів на декілька частин, та помістити кожну частину в окреме поле. Результат розділення показаний вище в табл. 1.1. Очевидно, замість двох полів «ПІБ», «Адреса» в таблиці з'явилося 7 полів: «Ім'я», «Прізвище», «По_батькові», «Вулиця, буд, кв.», «Місто», «Область», «Поштовий індекс».

Табл. 1.1 має більш просту структуру, ніж табл. 1.4, що дозволяє отримати інформацію по ній за допомогою досить простих запитів.

Більш складним є перетворення багатозначних полів. В табл. 1.4 багатозначним є поле «Сертифікати». Зверніть увагу на назву поля, воно є у множині. Один співробітник може мати декілька сертифікатів. Тому не можливо назвати це поле у однині. Назва у множині може вказувати на те, що поле є багатозначним.

Таблиця 1.4 – Співробітники

Код	ПІБ	Адреса	Телефон/Факс	Сертифікати
1010	ДенисенкоМихайло Іванович	Миру, 20, 3, Київ, 01005	(056)-111-11-11, (086)-111-11-11	Сертифікат Геодезиста, Сертифікат Землевпорядника
1011	Панасюк Маргарита Михайлівна	Пушкина, 1,1, Херсон, 73000	(098)-222-22-22 (068)-222-22-22, (050)-222-22-22	Сертифікат Геодезиста
1012	Лужанська Олена Ігорівна	Гагарина, 2, Кривий Ріг, 50027	(056)-333-33-33 (068)-333-33-33	Сертифікат Землевпорядника, По оцінці землі
1013	Іванов Михайло Вадимович	Зелена, 19, 4, Кривий Ріг, Дніпропетровська область, 53800	(098)-444-44-44	-
1014	Сергеєв Руслан Денисович	Зарічна, 49, Славянськ, 84100	(050)-555-55-55	-

Складові поля

Багатозначні поля

Поле Телефон/факс одночасно є складовим та багатозначним. Поле є складовим, оскільки містить такі дані як телефон, та факс. Факс можна використовувати в якості телефону, однак незрозуміло, який саме номер: факсу, або звичайного телефону представлений в колонці. Розділення поля на два окремих «Телефон», та «Факс» дозволяє уникнути складових полів. Проте поле «Телефон» є, в свою чергу, багатозначним. У користувачів може бути декілька телефонів, причому окремо можуть бути телефони мобільні, домашні та службові.

Можна розділити багатозначні поля таким же способом як і складові, виділивши для кожного значення окреме поле. Однак в [11, 14], рекомендується для багатозначних полів використовувати той же метод, що і для зв'язку «Декілька до декількох», а саме: за допомогою спеціальної зв'язуючої таблиці. Це пов'язано з тим фактом, що значення в багатозначному полі подібні зв'язку «Декілька до декількох». Одне конкретне значення багатозначного поля може бути пов'язано з будь-якою кількістю записів таблиці, а окремий запис в таблиці можна співвіднести з будь-якою кількістю значень в багатозначному полі.

Окрім зв'язуючої таблиці необхідно створити ще одну таблицю, яка буде відображати інформацію багатозначного поля. В нашому випадку, це табл. 1.5. Кожний з співробітників геодезичної фірми може мати сертифікат геодезиста, що дозволяє не тільки виконувати, а ще й підписувати всю геодезичну документацію, та сертифікат землевпорядника, що дозволяє розробляти проекти землеустрою та іншу землевпорядну документацію. Крім вищезгаданих сертифікатів співробітник геодезичної фірми може мати сертифікат оцінщика землі. Всі ці сертифікати відображені в табл. 1.5.

Зв'язок табл. 1.5 з вихідною табл. 1.4 (1.1) представлений на рис. 1.8.

Таблиця 1.5 – Сертифікати

Код_сертифікату	Кваліфікаційний_сертифікат
1	Інженера геодезиста
2	Інженера землевпорядника
3	Спеціаліста з оцінки земельних ділянок
4	-

Для створення зв'язуючої таблиці, так як і в попередньому випадку під час створення зв'язку «Декілька до декількох», первинні ключі з вихідної таблиці Співробітники та з таблиці Сертифікати вставляються в зв'язуючу таблицю. Нарізно ці ключі є зовнішніми ключами, а сумісно є первинним складовим ключем зв'язуючої таблиці.

Таким чином, процес видалення з таблиці багатозначного поля не є простим. В результаті цього процесу замість однієї таблиці, з'являються три. Однак, нормалізована таким чином база даних є більш прозорою для створення різноманітних запитів.

Друга і третя нормальні форми встановлюють місце первинного ключа по відношенню до інших полів таблиці.

Друга нормальна форма (2NF): Кожна колонка, що не є первинним ключем, повинна повністю залежати від первинного ключа [11].

Третя нормальна форма (3NF): Колонки, які не є первинним ключем, повинні залежати від первинного ключа, в той час, як первинний ключ не залежить від будь-якого не первинного ключа [11].

Рис 1.8 - Схема перетворення багатозначного поля

Друга і третя нормальні форми ще більш зміцнюють і підсилюють значення первинного ключа в таблиці. Як вказано в [5, 6, 9, 11, 14], із застосуванням другої та третьої нормальних форм, первинний ключ використовують для пошуку інформації по всій таблиці, однак, для визначення самого первинного ключа не потрібна інформація з інших полів таблиці. Є думка [11, 13], що 2NF та 3NF спрощують таблиці і зменшують надмірність інформації в базі даних.

1.3.5. Поля

Вище були розглянуті правила, які використовують для назв таблиць і полів. Нижче сформульовані правила для самих полів [9, 11]:

1. Поле повинно бути характеристикою предмета таблиці, а не окремим предметом;
2. Поле таблиці не може бути обчислювальним, тобто таким, значення якого може бути визначено за допомогою формули. Аргументи формули можуть з часом змінитися, в той же самий час, в таблиці зберігається не формула, а результат її обчислення. Тому, після зміни аргументів, поле буде показувати неправдивий результат;
3. Якщо зв'язки в базі даних встановлюються заздалегідь, то поле повинно з'являлися в базі даних тільки один раз. Наприклад, поля Прізвище (табл. 1.1) та Виконавець_Прізвище (табл. 1.2) дають одну й ту ж саму інформацію – прізвище співробітника. Якщо одна й та ж інформація представлена в різних таблицях, зміна цієї інформації може привести до руйнування цілісності бази даних. Поля різних таблиць, які містять однакову інформацію називаються **дублюючими**. Припустимо, один з співробітників, жінка, вийшла заміж та змінила прізвище. У всіх таблицях бази даних, які містять

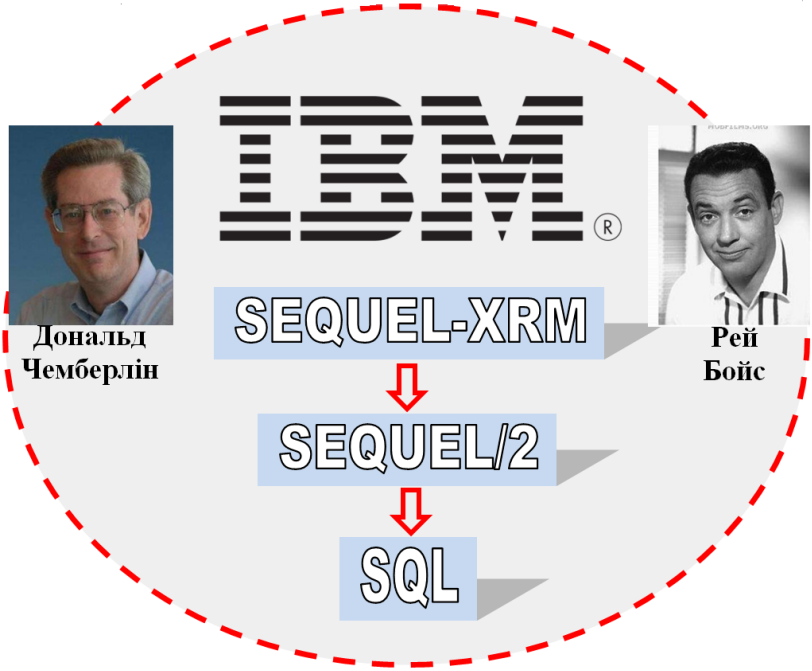
Прізвище співробітників, необхідно змінити застарілу інформацію. Забудькуватість користувачів бази даних може призвести до того, що в одній таблиці буде нова інформація, в іншій – стара. Саме тому бажано, щоб ПІБ співробітників з'являлося тільки в одній таблиці бази даних.

Тільки в тому випадку, якщо в базі даних зв'язки не встановлені, однакові значення в таблицях можуть бути використані для встановлення зв'язків. Більш детально використання дублюючих полів для встановлення зв'язків між таблицями див. розд. 3.4.2.

1.3.6. Запити

Запит є таблицею, яка сформована на підставі інформації, обраної з полів однієї або декількох таблиць бази даних. Таблиці, з яких обирається інформація для запиту, називаються **базовими**. Таблиця, сформована в результаті запиту, вважається **віртуальною**, оскільки після закінчення роботи в базі даних вона, як правило, знищується. На відміну від таблиць самої бази даних, в запитній таблиці можуть використовувати обчислювальні поля. Більш детально процедура формування запиту див. наступні розд.

2. SQL-запити



«SQL - мова світу баз даних. Більшість користувачів застосовують SQL лише формально і навіть не підозрюють про ту потужність, яка є в їхньому розпорядженні.»

Ентоні Молінаро «SQL-збірник рецептів»

2.1. Історія створення SQL

Як, було зазначено вище, в 1970-х роках доктор Едгар Кодд теоретично розробив реляційну модель бази даних. Оскільки бази даних, засновані на інших моделях, не задовольняли потребам користувачів, а реляційна база даних поки теоретично мала суттєві переваги, то різноманітні організації, такі як університети, дослідницькі лабораторії та комерційні фірми, почали розробляти мову, яка могла би використовуватися для баз даних, що підтримують реляційну модель.

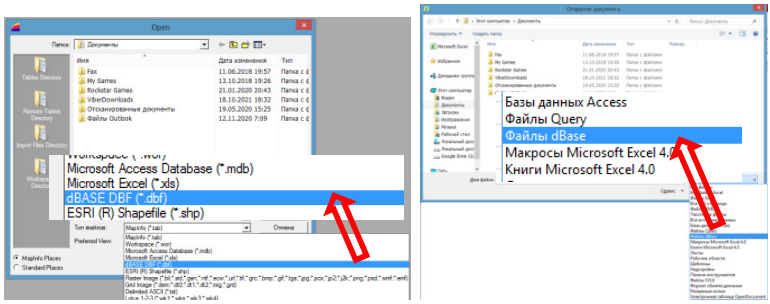
Згідно [6, 11, 14], на початку 1970-х років фірма IBM розпочала дослідницький проект «System/R», який повинен був довести життєздатність реляційної моделі і дати деякий досвід проектування і реалізації реляційної бази даних. Проект досяг успіху: вдалося створити базовий прототип реляційної бази даних.

Одночасно з розробкою реляційної бази даних, були виконані роботи по визначенню її мови. Робота, завершена в рамках цього проекту, виявилася найбільш значущою в цьому напрямку. Співробітники IBM доктор Дональд Чемберлін і Реймонд Бойс у 1974 році розробили структуровану англійську мову запитів, яку спочатку назвали SEQUEL **(**Structured English Query Language). Ця мова дозволяла здійснювати запити до реляційної бази даних, використовуючи чітко визначені пропозиції, близькі до англійської мови.

Розроблена в середині 1970-х років Д. Чемберліном і Реймондом Бойсом, мова SEQUEL стала першою комерційно успішною мовою для реляційних баз даних. Початкові відгуки і успіх SEQUEL підтримали наміри Д. Чемберліна і його співробітників продовжувати дослідження. Згодом, з юридичних причин, найменування SEQUEL було змінено на SQL (Structured Query Language) - виявилося, скорочення SEQUEL вже використовувалося в якості торгової марки фірмою «Hawker Siddeley aircraft» [2, 14].

У 1983 році на основі «System/R» IBM створила базу даних з ім'ям «Database 2», скорочено DB2, яка стала головною базою даних від IBM, а її технологія була впроваджена на всю серію продуктів IBM. У 1988 році проект «System/R» отримав престижну премію «ACM Software System Award», як успішний проект по створенню бази даних.

Далеко в минулому 80-ті роки. Вже немає у вжитку жодної з програм, які використовувалися в комп'ютерах 80- х років, і немає самих комп'ютерів, але і досі існує формат файлу, який застосовувався для збереження даних в програмі Database 2. На рис. 2.1 показані два діалогові вікна. Ліве вікно використовується для відкривання файлу в системі MapInfo, праве вікно - для відкривання файлу в системі Excel. Зверху представлені збільшені фрагменти цих вікон, на яких видно, що кожна з цих програм дозволяє відкрити файл формату DBASE, так званий файл *.dbf. Саме так називалася більш досконала версія Database 2 – DBASE.



a) b)

Рис. 2.1 – Використання файлів бази даних dBASE в програмах a) MapInfo, b) Excel

IBM була не єдиною організацією, яка у 80-х роках розробляла реляційну базу даних [11, 14]. У 1977 році молодий програміст Ларрі Еллісон, українець за походженням, створив компанію «Software Development Laboratories», яка в 1979 році випустила першу реляційну

базу даних «Oracle». В основі цієї бази даних була мова SQL. Слід зазначити, що «Oracle» випередив першу реляційну базу даних IBM на ринку на два роки, представивши першу комерційно доступну систему керування базами даних. На сьогоднішній день компанія «Oracle» є одним з провідних постачальників баз даних.

В той же самий час викладачі лабораторії обчислювальних систем Каліфорнійського університету під керівництвом професора Майкла Стоунбрекера також займалися дослідженнями технології реляційних баз даних. Подібно IBM, вони розробили прототип реляційної бази даних з назвою «INGRES». До складу «INGRES» була включена мова бази даних, названа мовою запитів QUEL (Query Language), яка в порівнянні з SQL була більш структурованою, але одночасно і більш складною для простих користувачів [6, 11].

Зрештою, коли стало ясно, що мова SQL перетворилася в стандартну мову баз даних, «INGRES» була перетворена в базу даних на основі SQL. Сьогодні «INGRES» є одним з лідируючих продуктів баз даних в цій області.

Пройшло багато років з 1970 року. Дослідники, які почали створювати перші реляційні бази даних та мову для цієї бази даних, заснували успішні корпорації. Їх програмні продукти і досі використовуються, а мова SQL перетворилася в стандартну мову баз даних [9, 11, 13, 14].

2.2. Склад SQL

Під час створення теоретичної основи реляційної бази даних доктор Едгар Кодд сформулював 12 правил, яким на його думку повинна задовольняти ця база даних [17]. Одним з цих правил є правило за номером 5, яке стверджує, що мова доступу до даних повинна бути єдиним способом доступу до даних у реляційній базі даних. Як зазначено вище, було

декілька мов, для роботи з базою даних. Однак, мова SQL мала перевагу в тому, що її могли використовувати не тільки програмісти, але й прості користувачі, які не мають навиків у програмуванні. Саме тому вона стала стандартом і є єдиною мовою для реляційних баз даних [19].

В склад SQL мови входять 4 групи операторів, а саме [15]:

- Оператори визначення даних (Data Definition Language, DDL). Це та частина SQL, яка використовується для створення бази даних, зміни її структури та знищення бази, після того, як вона стає не потрібною (Create, Alter, Drop).
- Оператори маніпулювання даними (Data Manipulation Language, DML) призначені для підтримки бази даних. За допомогою цього потужного інструмента можна точно вказати, що саме потрібно виконати з даними, які знаходяться в базі: ввести, змінити, або обрати необхідні (Select, Delete, Update, Insert).
- Оператори контролювання доступу до даних (Data Control Language, DCL). Призначення цих операторів - захист бази даних від різних варіантів пошкоджень (Grant, Revoke, Deny).
- Оператори управління транзакціями (Transaction Control Language, TCL). Як правило, слово транзакція використовується в контексті банківських операцій. Однак, транзакція має місце і в інформатиці. Згідно визначенню [15], транзакція в інформатиці - це група логічно об'єднаних послідовних операцій по роботі з базою даних, які виконуються або відміняються сумісно. Для транзакцій в інформатиці виконується правило «все, або нічого». Якщо неможливо виконати хоча б одну операцію, то вся група операцій, які в даному випадку називаються транзакцією, відміняється. (Commit, Rollback, SavePoint).

Коротка характеристика зазначених операторів представлена в Додатку А. Для створення SQL – запитів в

MapInfo використовується тільки один з вищезазначених операторів. Це оператор Select. Саме йому присвячені наступні рядки цього підручника.

2.3. Оператор Select

На відміну від всіх інших операторів, оператор Select дійсно є серцем SQL. Він є найпотужнішим оператором мови і засобом отримання інформації з таблиць бази даних. Оператор Select використовується спільно з ключовими словами і умовами пошуку для отримання найрізноманітної інформації.

Залежно від програми оператор Select може виконуватися по-різному: безпосередньо з вікна командного рядка, або з блоку програмного коду. Реляційні бази даних ГІС, такі як MapInfo, використовують спеціальні діалогові вікна. Незалежно від того, яким чином визначається і виконується оператор Select, його синтаксис завжди один і той же.

Схема оператора з ключовими словами і атрибутами показана на рис. 2.2. Це класична схема спрощеного оператора Select [14]. Розглянемо її більш детально.

Спочатку розберемо значення ключових слів оператора, їх шість. На схемі вони позначені великими літерами.

- **SELECT** — це основне ключове слово оператора Select, і його наявність абсолютно обов'язкова. Воно використовується для визначення колонок, які необхідно отримати в наборі результату для запиту. Самі колонки обирають з таблиць, які визначені в іншому ключовому слові FROM. Крім вже готових колонок, які є в наявності в таблицях, можуть бути використані обчислювальні колонки. Їх формують за допомогою функцій.

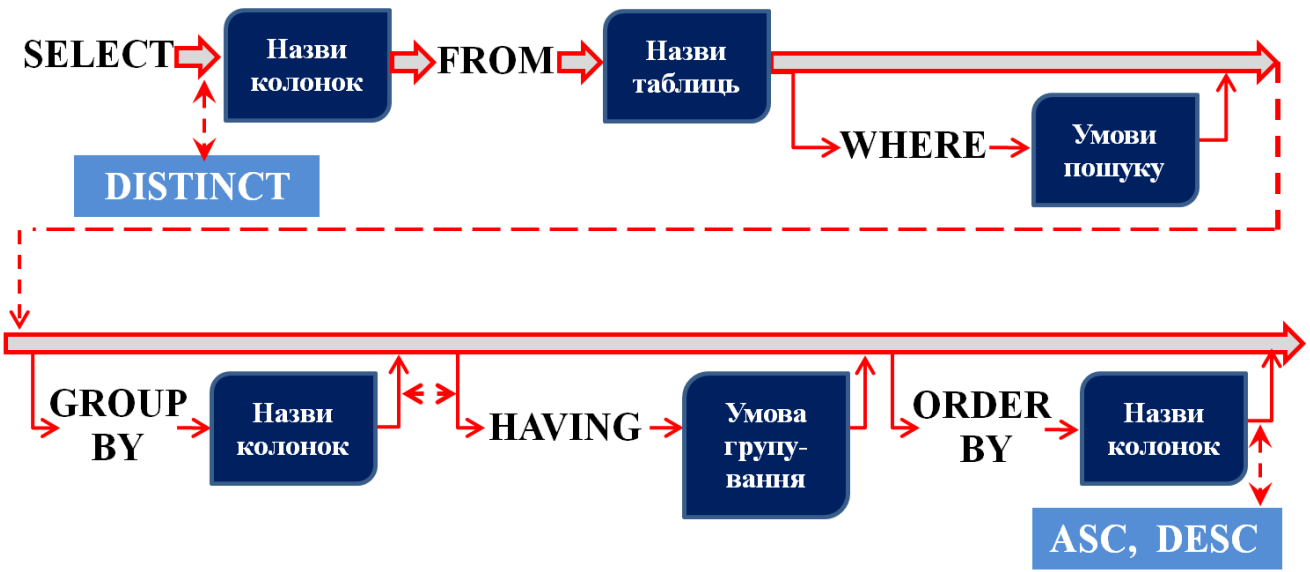


Рис 2.2 - Схема оператора Select

- **FROM** — це друге найбільш важливе ключове слово в операторі SELECT, і воно також є обов'язковим. Воно використовується для визначення таблиць, з яких повинні вилучатись стовпці, перераховані після ключового слова SELECT.
- **WHERE** — це необов'язкове ключове слово, яке використовується для фільтрації рядків, повернутих умовою FROM. Ключове слово WHERE супроводжується логічною умовою, яка називається предикатом, значення якого оцінюється як True (Істина), False (Брехня) або Unknown (Невідомо). Цей вислів можна перевірити, використовуючи стандартні оператори порівняння, булеві оператори, або спеціальні оператори, наприклад географічні (див. розд. 3.3.5).
- **GROUP BY.** Коли в ключовому слові SELECT використовуються агрегатні функції для отримання зведеної інформації, слід використовувати умову GROUP BY для об'єднання інформації в окремі групи. Програма використовує будь-яку колонку або список колонок, розташованих після ключових слів GROUP BY, для групування в них інформації. Ключове слово GROUP BY є необов'язковим. Агрегатні функції будуть детально розглянуті далі в розд. 3.4.3.
- **HAVING —** це слово, яке спеціально пов'язано з умовою GROUP BY, використовується для фільтрації згрупованої інформації. Воно подібно до ключового слова WHERE, в якому за ключовим словом HAVING слідує вираз, що оцінюється як True, False або Unknown. Ключове слово HAVING також необов'язкове.
- **ORDER BY.** Це останнє ключове слово, яке використовується для сортування інформації по рядках.

Давайте проаналізуємо схему оператора Select загалом. Є два ключових слова, це SELECT та FROM, які розташовані вздовж основної осі схеми. Таким розташуванням підкреслюється той факт, що ці ключові слова є обов'язковими. Без них виконання оператора Select неможливе. Інші ключові слова розташовані під основною віссю. Таким чином показується їх необов'язковість.

Розглянемо на простих прикладах дію кожного ключового слова в операторі Select.

2.3.1. Запити, сформовані за допомогою тільки обов'язкових ключових слів

Найпростішими є запити, які складаються тільки з обов'язкових ключових слів SELECT та FROM. На рис. 2.2 показаний фрагмент схеми оператора Select, який складається тільки з обов'язкових ключових слів.

Рис. 2.3 – Фрагмент схеми оператора Select, складений з обов'язкових ключових слів

Табл. 2.1 представляє більш розширений фрагмент таблиці, яка раніше в розд. 1.3.2 була дана під номером 1.3 – Земельні ділянки. Саме до цієї таблиці будуть сформовані наступні запити.

Найпростіший запит до табл. 2.1 представлений нижче. Завдяки йому обирається одна колонка.

SELECT Власник FROM Ділянки . (2.1)

Таблиця 2.1 – Ділянки

КОАТУУ	НКЗ	НКК	НЗД	Тип_власності	Власник	Площа_(га)
1211000000	08	109	0008	приватна	Рашевський С.В.	2,003
1211000000	06	208	0104	приватна	Носик О.М.	1,904
1211000000	08	156	0703	приватна	Хорольський І.В.	6,392
1211000000	08	156	0704	приватна	Хорольський І.В.	4,221
1221884001	01	001	1111	приватна	Хорольський І.В.	2,481
1211000000	06	001	1111	комунальна	Криворізька територіальна громада	19,536
1221884001	01	001	1111	комунальна	Лозуватська територіальна громада	25,923

Результуюча таблиця показана під номером 2.2.

Таблиця 2.2 – Результат запиту (2.1)

Власник
Рашевський С.В.
Носик О.М.
Хорольський І.В.
Хорольський І.В.
Хорольський І.В.
Криворізька територіальна громада
Лозуватська територіальна громада

За допомогою оператора Select декілька колонок можна отримати з основної бази даних так само просто, як і одну колонку. Імена колонок відділяються одна від одної комою. Нижче показаний запит, за допомогою якого обираються дві колонки Власник та Площа. Послідовність стовпців в ключовому слові SELECT не важлива вони можуть бути перераховані в будь-якому порядку. Це забезпечує гнучкість при перегляді однієї і тієї ж інформації безліччю способів.

SELECT Власник, Площа_(га) FROM Ділянки . (2.2)

Таблиця 2.3 – Результат запиту (2.2)

Власник	Площа_(га)
Рашевський С.В.	2,003
Носик О.М.	1,904
Хорольський І.В.	6,392
Хорольський І.В.	4,221
Хорольський І.В.	2,481
Криворізька територіальна громада	19,536
Лозуватська територіальна громада	25,923

Кількість стовпців, які можна визначити в умові SELECT, не обмежена. Фактично можна перерахувати всі колонки з вихідної таблиці. Однак, деякі таблиці бази даних можуть

містити досить велику кількість колонок, декілька десятків. Якщо необхідно обрати всі колонки, то замість назв стовпців ставиться зірка, так як це показано в прикладі наступного запиту.

SELECT * FROM Ділянки . (2.3)

Результатом цього запиту буде таблиця, повністю співпадаюча з вихідною (табл. 2.1).

В списку стовпців можуть бути обчислювальні колонки, які формуються за допомогою формул. Нижче представлений запит з формулою, яка об'єднує складові номера земельної ділянки, КОАТУУ, НКЗ, НКК та НЗД в кадастровий номер, так як це показано за допомогою табл. 2.4.

SELECT "КОАТУУ"+":"+"НКЗ"+":"+"НКК"+":"+" НЗД"

FROM Ділянки . (2.4)

В формулах обчислювальних колонок можна використовувати різноманітні оператори та функції. Аргументами функцій повинні бути дані з існуючих колонок таблиці та константи. Більш детально застосування формул див. розд. 3.3.

Таблиця 2.4 – Результат виконання запиту (2.4)

"КОАТУУ"+":"+"НКЗ"+":"+"НКК"+":"+"НЗД"
1211000000:08:109:0008
1211000000:06:208:0104
1211000000:08:156:0703
1211000000:08:156:0704
1211000000:06:001:1111
1221884001:01:001:1111

2.3.2. Запити з виключенням дублікатів рядків

Уважно подивіться на табл. 2.3. Вона містить список власників земельних ділянок. Оскільки одному власнику належать три ділянки, то таблиця містить три однакові рядки в колонці Власник. Аналогічна ситуація буде в тому випадку,

якщо поставити запит до колонки Тип_власності. Якщо сформувати запит, в якому необхідно показати одну колонку, наприклад Тип_власності, то буде отримана таблиця, в якій буде міститися 4 рядка з типом власності приватна і два – з типом власності комунальна. Для видалення однакових рядків в результуючій таблиці, використовується додаткове ключове слово DISTINCT, яке зв'язане з основним ключовим словом SELECT (рис. 2.3). Запит без однакових рядків буде виглядати так як показано нижче.

SELECT DISTINCT Власник FROM Ділянки . (2.5)

Як показано на схемі рис. 2.3, DISTINCT є необов'язковим ключовим словом, яке передує списку колон, зазначених в умові SELECT. Ключове слово DISTINCT вимагає від бази даних оцінити значення всіх колонок кожного рядка. Якщо буде виявлений повтор, то в результуючій таблиці всі повтори будуть виключені. Результатом дії запиту (2.5) є табл. 2.5.

Таблиця 2.5 – Результат запиту (2.5)

Власник
Рашевський С.В.
Носик О.М.
Хорольський І.В.
Криворізька територіальна громада
Лозуватська територіальна громада

Ключове слово DISTINCT можна використовувати для довільної кількості колонок. В цьому випадку з загального списку рядків будуть виключені тільки ті, в яких значення колонок, визначених в ключовому слові SELECT повністю співпадають. Змінимо попередній приклад, запросивши з таблиці Ділянки вже не одну, а дві колонки Власник та КОАТУУ, так як це показано в запиті (2.6).

SELECT DISTINCT Власник, КОАТУУ FROM Ділянки . (2.6)

Таблиця 2.6 – Результат запиту (2.6)

Власник	КОАТУУ
Рашевський С.В.	1211000000
Носик О.М.	1211000000
Хорольський І.В.	1211000000
Хорольський І.В.	1221884001
Криворізька територіальна громада	1211000000
Лозуватська територіальна громада	1221884001

Порівняйте два запити (2.5), (2.6), та результуючі таблиці. Якщо для однієї колонки результуюча таблиця має 5 рядків, то для двох колонок рядків більше. Як було сказано раніше, КОАТУУ відображає адміністративне територіальне положення ділянки. Як видно з табл. 2.1, власник Хорольський І.В. володіє трьома ділянками, дві ділянки розташовані на території Кривого Рогу (КОАТУУ=1211000000), одна – біля Лозуватки (КОАТУУ=1221884001). В табл. 2.5 необхідно було показати тільки власників ділянок. Тому в таблиці представлені саме власники. В табл. 2.6 необхідно було показати не тільки власників а ще й адміністративне розташування ділянок власників. Так як ділянки одного з власників розташовані в двох різних адміністративних одиницях, результуюча таблиця має два рядки для цього власника.

Ключове слово DISTINCT - це дуже корисний інструмент в певних обставинах. Його потрібно використовувати, коли дійсно необхідно отримати унікальні рядки в наборі результатів.

2.3.3. Запити з ключовим словом WHERE

Умова вибору, яка записується після атрибута WHERE може складатися з констант, функцій, операторів, та іншої умови. Який би не був вираз, його значення повинно бути

логічним. Тобто воно може приймати одно з трьох значень True (Істина), False (Брехня) або Unknown (Невідомо).

SELECT * FROM Ділянки WHERE Тип_власності = “комунальна” . (2.7)

Вище представлений приклад запиту, в якому обираються всі колонки, в той же самий час умова після слова WHERE дозволяє обрати тільки ті рядки, які задовольняють цій умові. Тобто в результуючій таблиці представлені рядки, в яких осередки в колонці Тип_власності дорівнюють значенню «комунальна».

Таблиця 2.7 – Результат запиту (2.7)

КОАТУУ	НКЗ	НКК	НЗД	Тип_власності	Власник	Площа_(га)
1211000000	06	001	1111	комунальна	Криворізька територіальна громада	19.536
1221884001	01	001	1111	комунальна	Лозуватська територіальна громада	25.923

Більш детально створення умови вибору розглянуто нижче в розд. 3.3.

2.3.4. Запити з ключовим словосполученням GROUP BY

Ключове слово GROUP BY використовується для об’єднання результатів вибірки по одному або кількох колонках. З використанням ключового слова GROUP BY тісно пов’язане використання агрегатних функцій і ключового слова HAVING.

Агрегатною функцією в мові SQL називається функція, яка повертає якесь одне значення з набору значень рядків колонки. Таких функцій всього 6. Їх коротка характеристика представлена в табл. 3.2.

Припустимо, що необхідно обчислити загальну площу всіх ділянок кожного власника. В цьому випадку групування

повинно бути виконано по колонці Власник. Для кожного власника обчислюється сума значень, представлених в колонці Площа_(га). Запит виглядає так:

SELECT Власник, Sum(Площа_(га)) FROM Ділянки GROUP BY Власник . (2.8)

Уважно погляньте на текст запиту, в ньому після слова SELECT представлені назви двох колонок, «Власник» та агрегатної функції Sum(), яка сумує рядки, які мають однакові значення в колонці «Власник». Умова за якої виконується групування задано після ключових слів Group By.

В табл. 2.8 представлений результат дії запиту з тим же номером.

Таблиця 2.8 – Результат запиту (2.8)

Власник	Площа_(га)
Рашевський С.В.	2,003
Носик О.М.	1,904
Хорольський І.В.	13,094
Криворізька територіальна громада	19,536
Лозуватська територіальна громада	25,923

Якщо необхідно показати не всю згруповану інформацію, а тільки її частину, то використовується ключове слово HAVING.

Ключове слово HAVING аналогічне ключовому слову WHERE за тим винятком, що застосовується не для всього набору колонок таблиці, а для набору створеного ключовим словом GROUP BY і застосовується завжди виключно після нього. Результатом ключового слова HAVING є логічна величина, також як і ключового слова WHERE, тобто, True (Істина), False (Брехня) або Unknown (Невідомо). Обов'язково в умові повинна бути агрегатна функція. Тобто умова ставиться до агрегатної функції. Наприклад, обираються тільки ті власники, площа ділянок яких не перевищує визначеної величини, наприклад 2 га. В цьому

випадку до результуючої таблиці попадуть не всі власники, а тільки ті, загальна площа ділянок яких не перевищує задане число. В нашому випадку це тільки один власник (див. табл. 2.9).

SELECT Власник, Sum(Площа_(га)) FROM Ділянки GROUP BY Власник HAVING Sum(Площа_(га))<=2 . (2.9)

Таблиця 2.9 – Результат запиту (2.9)

Власник	Площа_(га)
Носик О.М.	1,904

2.3.5. Запити з ключовим словосполученням SORT BY

За визначенням, рядки набору результатів, повернуті оператором Select, не впорядковані. Послідовність, в якій вони з'являються, ґрунтується на їх фізичному розташуванні в таблиці. Єдиний спосіб сортування набору результатів полягає у встановленні оператора Select з ключовим словом ORDER BY. Саме ключове слово ORDER BY дозволяє визначити послідовність рядків в остаточному наборі результатів.

Ключове словосполучення ORDER BY дозволяє впорядкувати набір результатів зазначеного оператора Select по одному або кількох колонках, а також містить опцію упорядкування за зростанням або зменшенням для кожної колонки. В ключовому слові ORDER BY можна використовувати тільки ті колонки, які перераховані в ключовому слові SELECT. Хоча ця вимога зазначена в стандарті SQL, деякі реалізації його повністю ігнорують. Проте в програмі MapInfo, яка буде вивчатися далі, ця умова виконується точно.

Коли в ключовому слові ORDER BY використовуються дві або більше колонок, кожна колонка відокремлюється комою.

За замовчуванням, сортування виконується за зростанням. Саме цей варіант використаний в запиті, представленому під номером (2.10).

SELECT Власник, Площа_(га) FROM Ділянки SORT BY Власник . (2.10)

Таблиця 2.10 – Результат запиту (2.10)

Власник	Площа_(га)
Криворізька територіальна громада	19,536
Лозуватська територіальна громада	25,923
Носик О.М.	1,904
Рашевський С.В.	2,003
Хорольський І.В.	6,392
Хорольський І.В.	4,221
Хорольський І.В.	2,481

Згідно запиту (2.10), необхідно показати дві колонки Власник та Площа_(га). Сортування необхідно виконати по колонці Власник. Оскільки не вказаний варіант сортування, то виконується сортування за зростанням. Результат виконання запиту показаний в табл. 2.10. Табл. 2.3 створена з тими ж самими колонками, але без сортування. Якщо порівняти ці таблиці, то досить легко зрозуміти, що порядок рядків змінився, спочатку йде інформація по Криворізькій та Лозуватській громадам, а потім по іншим власникам згідно порядку літер в українському алфавіті.

Сортування можна виконати по декільком колонкам. Спочатку виконується сортування по колонці, яка представлена відразу після ключового слова ORDER BY. Якщо в цій колонці зустрічаються однакові дані, так як в нашому випадку, прізвища власників повторюються, то сортування виконується вже по другій колонці. В нашому

випадку це колонка Площа_(га). Результат сортування по двом колонкам, Власник та Площа_(га), сформований запитом (2.11), показаний в табл. 2.11. Уважно подивіться на результуючі таблиці 2.10, 2.11. Якщо перша колонка Власник осталася незмінною, то друга колонка змінилася. Порядок виводу ділянок відсортований за зростанням площі.

SELECT Власник, Площа_(га) FROM Ділянки SORT BY Власник, Площа_(га) . (2.11)

Таблиця 2.11 – Результат запиту (2.11)

Власник	Площа_(га)
Криворізька територіальна громада	19,536
Лозуватська територіальна громада	25,923
Носик О.М.	1,904
Рашевський С.В.	2,003
Хорольський І.В.	2,481
Хорольський І.В.	4,221
Хорольський І.В.	6,392

SELECT Власник, Площа_(га) FROM Ділянки SORT BY Власник DESC . (2.12)

Таблиця 2.12 – Результат запиту (2.12)

Власник	Площа_(га)
Хорольський І.В.	6,392
Хорольський І.В.	4,221
Хорольський І.В.	2,481
Рашевський С.В.	2,003
Носик О.М.	1,904
Лозуватська територіальна громада	25,923
Криворізька територіальна громада	19,536

Табл. 2.12 є результатом виконання запиту з тим же номером, згідно якому рядки в колонці Власник повинні бути відсортовані за зменшенням. Для порівняння табл. 2.10 також відсортована, але за зростанням даних в колонці Власник. Порівняйте ці таблиці. Положення всіх рядків змінилося. Ті що були зверху, зараз находяться знизу і навпаки.

Як вже було сказано, за замовчуванням сортування виконується за зростанням. Однак, можна встановити сортування за зростанням явно, за допомогою ключового слова ASC (від Ascending - за зростанням), а саме:

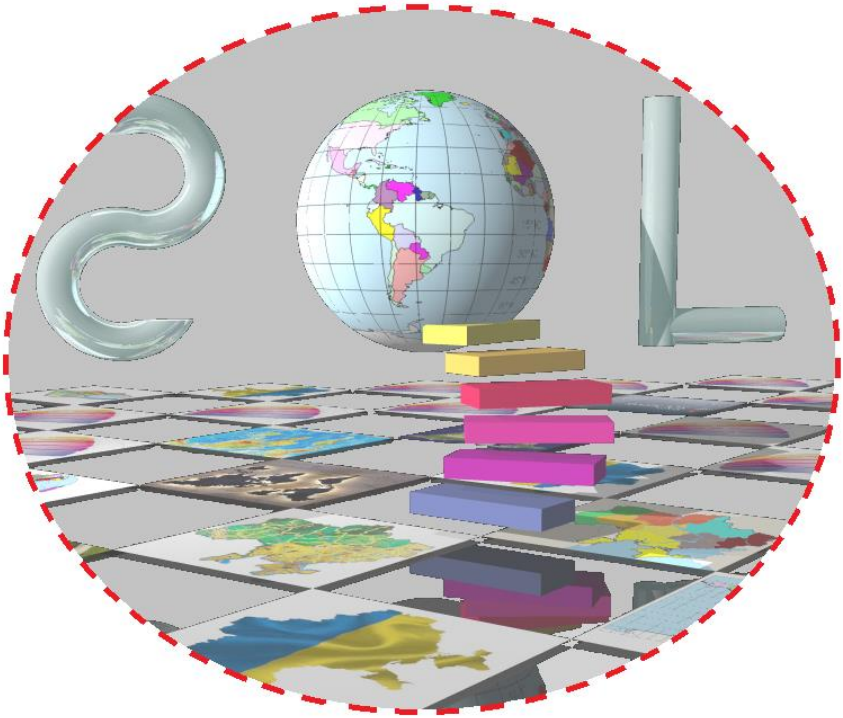
SELECT Власник, Площа_(га) FROM Ділянки SORT BY Власник DESC, Площа_(га) ASC . (2.13)

Таблиця 2.13 – Результат запиту (2.13)

Власник	Площа_(га)
Хорольський І.В.	2,481
Хорольський І.В.	4,221
Хорольський І.В.	6,392
Рашевський С.В.	2,003
Носик О.М.	1,904
Лозуватська територіальна громада	25,923
Криворізька територіальна громада	19,536

Більш детально запити розглянуті в наступному розділі на прикладі реляційної бази даних MapInfo.

3. Оператор Select в MapInfo



«MapInfo… – розвинена система електронної картографії, яка дозволяє вирішувати складні завдання географічного аналізу…»

Посібник користувача

3.1. Процедура вибору в MapInfo

На перший погляд процедура вибору об'єктів в MapInfo не відрізняється від процедури виділення об'єктів в інших Windows - додатках. Також як і в інших сучасних графічних програмах, в MapInfo, перш ніж виконувати будь-які дії з об'єктом, його слід вибрати. Однак, насправді, є суттєва відмінність процедури вибору, закладена в структурі програми. Це зв'язано з тим фактом, що MapInfo є реляційною базою даних. Тому кожний вибір якогось елемента – це для MapInfo SQL-запит. Для кожного комплекту вибраних об'єктів MapInfo створює тимчасову таблицю, яка називається Selection, структура якої повністю збігається зі структурою таблиці, з якої були обрані об'єкти. З цією таблицею можна працювати також як і зі звичайною таблицею, зокрема, її можна відкрити в вікнах Карта (Map) і Список (Browser). Якщо не звертатися якимось чином до таблиці Selection, вона знищується при скасуванні вибору об'єктів, або замінюється іншою таблицею з тією ж назвою при виборі інших об'єктів.

Якщо, навпаки, звернутися до таблиці Selection, наприклад відкрити її в вікні Карта (Map) або Список (Browser), вона отримує назву Query n, де n – номер запиту, яка відкривається, і зберігається незмінною протягом усього сеансу роботи в MapInfo. Можна записати таблицю з іншою назвою, використовуючи команду Save Copy As (Зберегти копію) з меню File (Файл).

На рис. 3.1 показані етапи процедури вибору об'єктів в MapInfo, зокрема зверху представлений фрагмент таблиці World[4], відкритою у вікні Список. В цій таблиці обрано 4 об'єкти. Вони позначені в таблиці World чорними квадратами

[4] Як зазначено в [21], таблиця World є частиною бази даних географічних об'єктів, які поставляються сумісно з програмними продуктами MapInfo.

та обведені червоною стрічкою. Програма для обраних об'єктів створила тимчасову таблицю з назвою Selection. Для того, щоб її побачити, слід в меню Window (Вікно) обрати варіант New Browser Window (Новий список), або New Map Window (Нова карта). Відкриється вікно запитів з назвами доступних таблиць. Це вікно показано всередині рис. 3.1. У вікні дається список всіх відкритих таблиць. Першим у списку представлена назва таблиці, яка тільки була створена за допомогою процедури вибору. Вона має назву Selection.

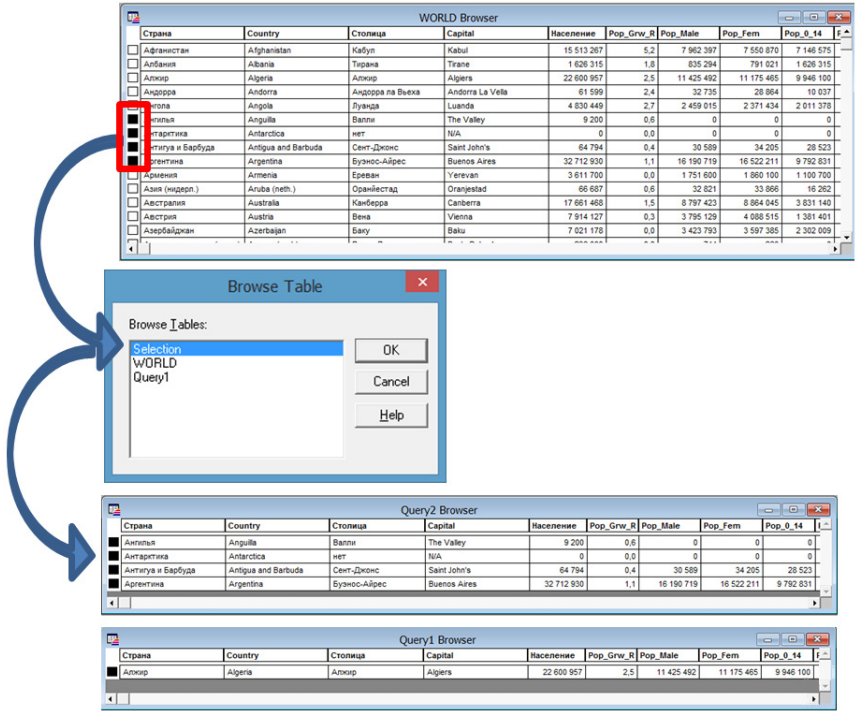


Рис. 3.1 – Етапи процедури вибору в MapInfo

Уважно подивіться на список доступних таблиць. Крім таблиці World, та зовсім нової таблиці Selection, є таблиця з назвою Query 1. Це означає, що перед вибором 4 об'єктів, показаних чорними квадратиками в першому списку, була виконана ще одна процедура вибору, і результат цієї процедури був раніше відкритий або у вікні Browser, або у

вікні Map, та отримав назву Query 1. Наступна таблиця Query 2 була створена після того як була обрана таблиця Selection в діалоговому вікні вибору таблиць. Подивіться на таблиці в нижній частині рис. 3.1, вони мають назву Query 1 та Query 2. Вони були створені за допомогою вибору об'єктів у вікні Browser. Структура таблиць Query 1 та Query 2 повністю співпадає зі структурою вихідної таблиці.

Виконати процедуру вибору в MapInfo можна одним з трьох способів, представлених нижче. Не зважаючи на вибір способу, в програмі виконується SQL запит.

Варіанти SQL запиту в MapInfo

1. **Вибір об'єктів за допомогою кнопок вибору;**
2. **Команда Select (Вибір) з меню Query (Запит);**
3. **Команда SQL Select (SQL Запит) з меню Query (Запит)**

Перший спосіб SQL запиту - за допомогою миші, попередньо вибравши одну з кнопок вибору на панелі Main. Кнопки цієї панелі показані в лівій частині абзацу та обведені червоною стрічкою. Верхня ліва кнопка є універсальною. Її можна використовувати в довільному робочому вікні програми. Інші 6 кнопок можна використовувати тільки в графічному вікні Map. Остання сьома кнопка, яка є нижньою правою кнопкою в групі кнопок вибору, є кнопкою вибору у вікні діаграми.

Другий та третій способи вибору, це створення SQL запиту, який можна сформувати за допомогою двох різних команд з менюQuery.

Перша команда має назву Select (Вибір), друга команда називається SQL Select (SQL Вибір). Обидві команди формують SQL запит. Можливості команд різні. Команда

Select (Вибір) має менше можливостей ніж команда SQL Select. Після вибору команди Select, відкривається вікно опцій вибору, показане на рис. 3.2. Для команди SQL Select відкривається вікно запиту, показане на рис. 3.3. Вже с першого погляду на ці вікна зрозуміло, що можливості команд різні. Розберемо які задачі можна вирішити за допомогою цих команд більш детально.

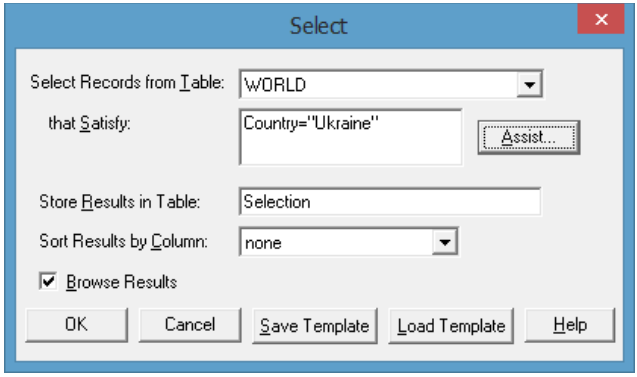


Рис. 3.2 – Діалогове вікно Select

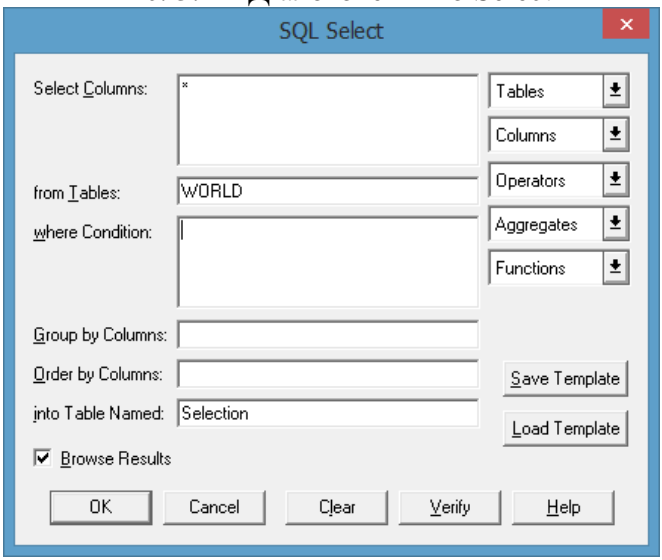


Рис. 3.3 – Діалогове вікно SQL Select

Почнемо з більш простої команди Select (Вибір).

3.2. SQL запит за допомогою команди Select

Схема SQL запита за допомогою команди Select (Вибір) показана на рис. 3.4. Якщо порівняти її зі схемою стандартного запиту на рис. 2.2, то можна помітити такі відмінності.

1. Запит здійснюється тільки до однієї таблиці. Не має можливості зробити запит до декількох таблиць одночасно. Уважно подивіться на атрибут після ключового слова FROM. Згідно атрибуту можна обрати тільки одну таблицю.
2. Немає можливості обрати тільки ті колонки, які необхідні. Структура запитної результуючої таблиці повністю співпадає зі структурою вихідної таблиці. Це показано за допомогою зірки, яка показана після ключового слова SELECT. Більш того, не можливо створювати обчислювальні колонки.
3. Відсутні ключові слова GROUP BY та HAVING. Це означає, що немає можливості групування інформації в колонках та не має доступу до агрегатних функцій.
4. Сортування виконується тільки по одній колонці і тільки за зростанням. Тобто за допомогою команди Select (Вибір) можна виконати дуже спрощений варіант SQL запиту.

Подивіться на схему запиту ще раз. Зеленими стрілками показано, яким чином за допомогою діалогового вікна виконується вибір атрибутів запиту. Зокрема вибір атрибута до ключового слова FROM, Назва таблиці обирається за допомогою кнопки зі стрілкою в полі зі словами Select Records from Table (Обрати записи з таблиці).

Для формування логічної умови пошуку, яка є атрибутом ключового слова WHERE, використовується центральне текстове поле, яке знаходиться біля слів that Satisfy (Згідно умові). Найпростіший спосіб сформувати логічну умову без помилок - натиснути кнопку Assist (Скласти).

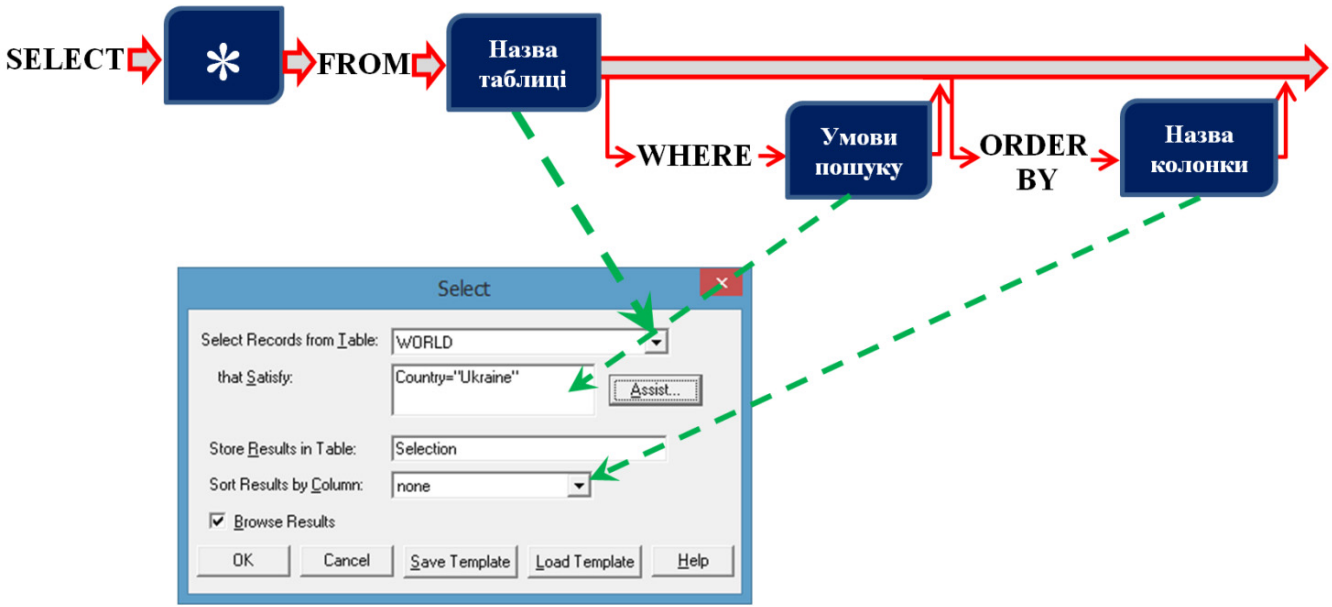


Рис 3.4 - Схема оператора Select для команди Select (Вибір)

Для вибору колонки для сортування необхідно натиснути кнопку зі стрілкою в полі біля слів Sort Results by Column (Впорядкувати за колонкою).

Крім полів, позначених зеленими стрілками та трьох обов'язкових кнопок OK, Cancel, Help, в діалоговому вікні рис. 3.4 є ще два поля та дві кнопки. Поле Store Results in Table за замовчуванням пропонує використати для запитної таблиці назву Selection. Як показано на рис. 3.1, ця назва під час відкриття таблиці замінюється на назву Query n. Однак, якщо є необхідність назвати запитну таблицю по-іншому, то цю назву можна вказати в полі Store Results in Table заздалегідь під час створення запиту. Прапорець Browse Results у включеному стані дає змогу відкрити запитну таблицю у вигляді списку (Browser). Якщо прапорець відключений, то запитна таблиця відкривається у вікні карти (Map).

Якщо необхідно зберегти всі атрибути вікна Select, то для цього можна використати кнопку Save Template. Під час натискання цієї кнопки створюється текстовий файл з розширенням *.qry, в якому зберігається запит, створений у вікні. На рис. 3.5 показаний файл, створений за даними, представленими в діалоговому вікні Select рис. 3.4.

```
Tables {WORLD}
Where {Country="Ukraine"}
Order {}
Into {Selection}
Browse
```

Рис. 3.5 – Файл з атрибутами запиту

Для повторного використання запитного файлу можна вже створений файл загрузити в діалогове вікно. Для цього потрібно натиснути кнопку Load Templatета, та обрати потрібний файл.

Як було зазначено вище, за допомогою команди Select (Вибрати) виконується досить спрощений варіант SQL

запиту. Однак, це не означає, що завдання, які вирішує ця команда, примітивні. Не зважаючи на просту схему запиту, за допомогою цієї команди можна отримувати цікаву і різноманітну інформацію з бази даних. Все різноманіття додається за допомогою атрибуту ключового слова WHERE, тобто за допомогою логічного виразу. Можливості побудови цього виразу ми розглянемо далі.

3.3. Складання виразів для запитів

Вирази, які створюються в програмі MapInfo, можна розділити на дві групи:

1. Вирази, в результаті обчислення яких виходить логічна величина (TRUE або FALSE).

2. Вирази, в результаті обчислення яких виходить чисельна, строкова величина, або дата.

Вирази першої групи можуть складатися з декількох підвиразів і операторів порівняння та географічних операторів (див. розд. 3.3.5). Саме логічні вирази беруть участь в SQL - запитах об'єктів.

Вирази другої групи не використовують оператори порівняння.

Вирази першої та другої груп можуть бути використані для створення обчислювальних колонок в ключовому слові SELECT. Однак, для команди Select (рис. 3.4), обчислювальні колонки не можливі, тому вирази другої групи не використовуються в цій команді. Можливі тільки вирази першої групи і тільки в атрибуті Умови пошуку ключового слова SELECT.

Є два способи створення виразів. По-перше, можна ввести вираз безпосередньо у поле that Satisfy (Згідно умові). Цей спосіб працює зазвичай швидше при формуванні простих виразів. Другий спосіб полягає в тому, що можна натиснути кнопку Assist (Скласти) в діалозі Select (Вибрати) (рис. 3.2) й побудувати вираз в віконцях діалогу Espression (Вираз).

Такий спосіб можна рекомендувати під час вивчення процедури створення виразу та при побудові складних виразів.

Кнопка Assist (Скласти) відкриває діалогове вікно створення виразу, показане на рис. 3.6. Вікно має три списки Columns (Колонки), Operators (Оператори) і Functions (Функції).

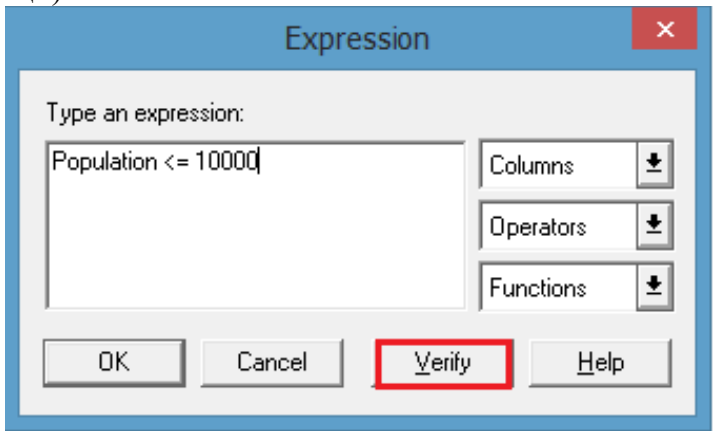


Рис. 3.6 – Діалогове вікно створення виразу (Expression)

Список Columns (Колонки) містить список всіх колонок таблиці, в якій здійснюється вибір. Останнім у списку колонок є слово obj, що є скороченням слова object. Вибір цього слова дозволяє обрати графічні об'єкти, замість колон. Детально про варіант колонки obj буде розглянуто далі в розд. 3.3.5. Якщо таблиця містить колонки, обчислені за попередніми запитами, то показуються і ці колонки.

В списку Operators (Оператори) міститься перелік всіх доступних операторів. Оператори діляться на певні групи. Їх список показаний на рис. 3.8. Це по перше круглі дужки, далі математичні, оператори порівняння, географічні та логічні оператори. За допомогою цих операторів можна створювати формули.

Список функцій вікна 3.6 містить різноманітні функції, які мають один і більше параметрів і повертають певні

значення. Як і оператори, функції діляться на групи, список яких показаний на рис. 3.8. Це, по перше, математичні функції, такі як синус, косинус тощо, функції роботи з текстовими строками, функції дати та часу, та зовсім особливі функції, які мають назву географічних. Вони у списку на рис. 3.8 показані червоним кольором.

У вікні Expression (Вираз) є кнопка Verify (Перевірити). Вона обведена червоною стрічкою. Ця кнопка дає команду перевірити лексику і логіку складеного виразу. Це корисно для перевірки створених нових виразів.

Створення виразів, подібно створенню речень. Процес складання виразів MapInfo схожий на те, як сформовано речення на звичайній мові. У нашому розпорядженні є набір слів, які можна використовувати, і синтаксичні правила для з'єднання цих слів. Синтаксис виразів MapInfo набагато простіший синтаксису звичайної мови. Правила звичайної мови здаються простими тільки тому, що всі користуються ними постійно, а правила складання виразів можуть тільки на перший погляд здатися складними. Втім, як і речення звичайної мови, вирази в MapInfo можна зробити досить складними.

Можна розглядати назви колонок і константи як підмет у реченні, а функції та оператори - як присудок (див. рис. 3.7-3.8). В кожному реченні є як мінімум один підмет і один присудок. Аналогічно, у будь-якому виразі необхідні, як мінімум: ім'я однієї колонки або константи, та один з операторів, або функцій. Але для логічного виразу обов'язкова присутність операторів порівняння, або географічних операторів, тобто операторів шостого рівня пріоритету. Тільки в цьому випадку результат буде логічною величиною. Як багато операторів і функцій буде використовуватися в виразі залежить від того, що саме необхідно отримати.

Рис. 3.7 – Елементи виразу, які можуть розглядатися як підмет

Рис. 3.8 - Елементи виразу, які можуть розглядатися як присудок

Розберемо спочатку варіанти підмету. Це колонки та константи.

3.3.1. Константи в MapInfo

При вживанні в виразах фіксованих значень текстових рядків, числових констант і дат необхідно дотримуватися наступних правил.

Правила для числових констант. Як вказується в [21], при завданні числових констант не допустимі кома або знак долара. Допустимі тільки цифри, десяткова точка, яка використовується як роздільник цілої та дробової частини, знак мінус для від'ємних чисел та англійська літера «e» для

завдання числа з плаваючою точкою. В останньому випадку для відображення ступеня числа можна використовувати як велику так і маленьку літеру «е». Далі наведений приклад з двома варіантами запису одного й того же числа, перший варіант – з фіксованою точкою, другий – з плаваючою.

Приклад:
ЧИСЛО: з фіксованою точкою **-0.123456789**
з плаваючою точкою **-1.23456789e-1**

Нижче представлений запит до таблиці World, завдяки якому визначаються країни, кількість мешканців яких не перевищує 10 тисяч чоловік. Можна набрати цифру 1 з чотирма нулями, а можна показати число з плаваючою точкою. Другий запит праворуч використовує саме число з плаваючою точкою.

Population<=**10000** , Population<=**1e4** . (3.1)

Правила відображення дат. Дати складаються з номерів дня, місяця і року. Рік позначається двома, або чотирма цифрами. Рік можна не вказувати. В цьому випадку вважається поточний рік. Вся дата відокремлюється подвійними лапками, причому місяці, дні і роки відокремлюються одна від одної точкою, так як це показано в нижченаведеному прикладі.

Приклад:
ДАТА: з чотирма цифрами року **“31.12.2022”**
з двома цифрами року **“01.01.22”**
без цифр року (поточний рік) **“01.01”**

Нижче показаний запит до бази даних за допомогою якого вирішується задача вибору співробітників фірми, які народилися не раніше 2000 року.

Дата_народження<=**“01.01.2000”** . (3.2)

Правило для текстових рядків. Якщо в виразі є рядок символів, то цей рядок має бути укладений в подвійні лапки. Таким чином MapInfo відрізняє рядки символів від назв колонок. Подивіться на приклад запиту (3.3). Зліва стоїть назва колонки Country, праворуч, після знаку дорівнює, в подвійних лапках стоїть назва країни, Ukraine.

Country=**“Ukraine”** . (3.3)

Правило для логічних величин. Логічна величина може набувати значення True (Істина), False (Брехня)[5]. Можна замість цілих слів, використовувати скорочення, складене з першої букви слова. Наприклад True, можна замінити англійською літерою T, False можна замінити англійською літерою F. Насправді, комп’ютер розглядає логічні величини, як особливі числа, які можуть приймати одно з двох значень, або 0, - це буде False, або 1, - це буде True. Якщо логічна величина задається своєю назвою, або літерою, то і назва і літера повинні бути укладені в подвійні лапки. Якщо логічна величина задається цифрами, або 0, або 1, то подвійні лапки не потрібні.

Приклад:

Логічна величина:	**“True”, “False”**
скорочений формат	**“T”, “F”**
цифровий формат	**1, 0**

Нижче представлений вираз запиту до бази даних геодезичної фірми. Серед іншої інформації в цій базі даних є колонка з логічною величиною, в якій дається інформація про наявність у геодезиста сертифіката. Якщо сертифікат є, то в відповідній комірці представлена логічна величина True. Три версії виразу показують можливості представлення логічної величини.

[5] Варіант Unknown (Невідомо) в MapInfo не передбачений [21].

Сертифікат=**“True”**, Сертифікат=**“T”**, Сертифікат=**1**. (3.4)

3.3.2. Колонки в MapInfo

Другий варіант підмету у виразах (рис. 3.7) – колонки. Як було зазначено вище, вони задають структуру таблиці. Таблиця в MapInfo - це об’єднання об’єктів, що мають однакову структуру. Структура об’єктів формується при створенні таблиці і визначається обраною проекцією, кількістю, типом та порядком завдання колонок, які, як правило називаються полями (Fields) і їх індексацією. Кожна колонка характеризує яку-небудь властивість об’єкта, що входить в таблицю. Наприклад, таблиця населених пунктів може містити колонки - назва міста, кількість мешканців, адміністративне значення міста і т.п. Колонки, (поля) можуть бути таких же типів, як і типи констант:

- числового;
- символьного (текстового);
- типу дати;
- логічного типу.

Варіанти типів колонок показані на рис. 3.9.

Згідно рис. 3.9, перший тип в списку доступних типів даних в колонках – це символьний тип (Character). Кількість символів в символьному типі задається при його встановленні і не може бути більше 254. Якщо в поле цього типу записані цифри, над ними не можна виконувати арифметичні операції. Для програми це текст.

Наступний тип – чисельний, який в колонках може бути таких варіантів:

1. Ціле (Integer) - варіант поля цілих чисел. Цілі числа, введені в це поле, по абсолютній величині повинні бути не більше 2^{12};

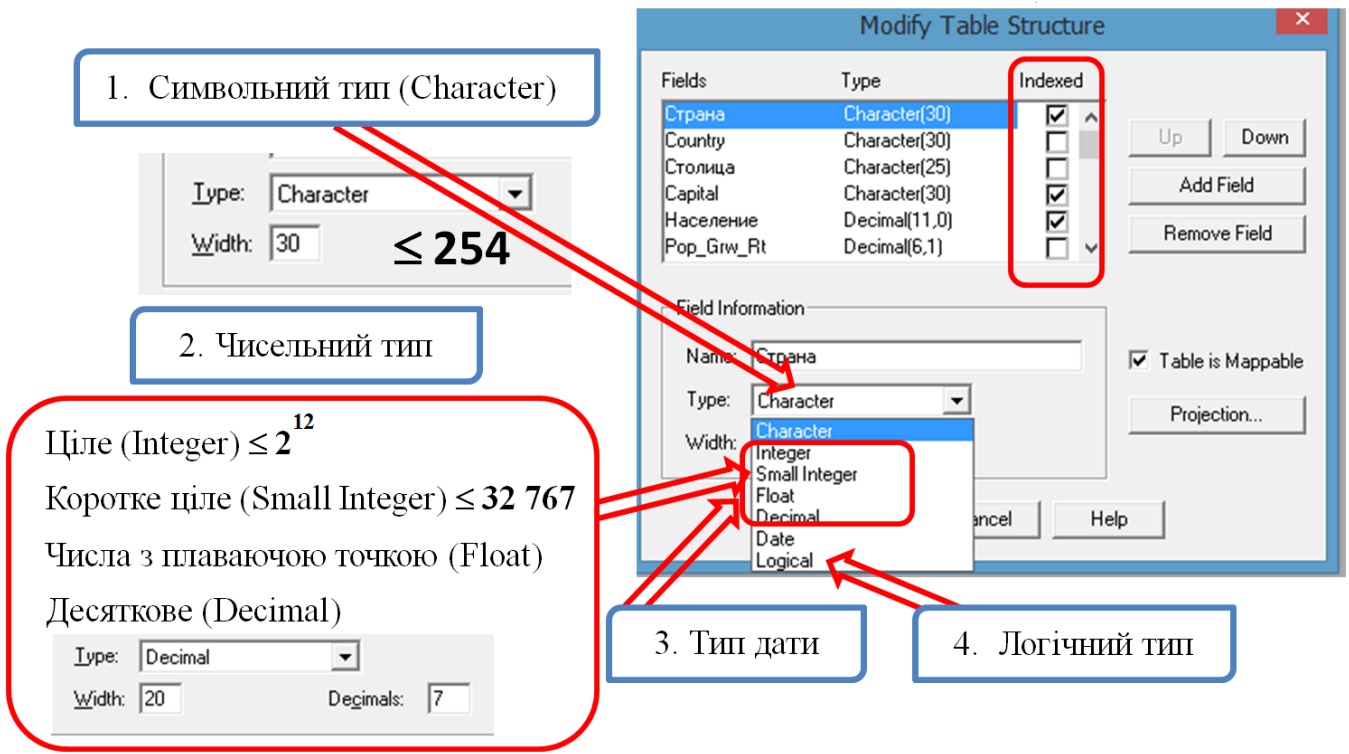


Рис 3.9 – Колонки в таблиці MapInfo

2. Коротке ціле (Small Integer) - варіант поля малих цілих чисел. Цілі числа, збережені в цьому полі, повинні бути не менше -32768 і не більше +32767. Цілі числа даного варіанту займають менший об'єм комп'ютерної пам'яті і швидше обробляються, в порівнянні з числами попереднього варіанту;
3. Числа з плаваючою точкою (Float);
4. Десятковий (Decimal) - варіант поля дійсних чисел з фіксованою точкою. При встановленні варіанту задається загальна кількість знаків (Width) і кількість цифр після крапки (Decimals). У загальне число позицій входять загальна кількість цифр, знак мінус, якщо він є, і десяткова крапка. Ця величина повинна бути не більше 20.

Тип дати і логічний тип, аналогічні типам, які задаються для констант.

Колонки можуть бути індексованими і неіндексованими. Індексація полів необхідна для таких процедур, як геокодування і пошук даних.

При побудові структури таблиці значення має не тільки кількість і типи полів, але також порядок записів полів в структурі. Тобто, дві таблиці з однаковою кількістю колонок одних типів і навіть з одним і тим же змістом в колонках, для MapInfo матимуть різну структуру, якщо ці колонки записані в таблиці в різному порядку.

Наприклад, таблиця 1 має Колонку1 (Дата), Колонку2 (Логічна величина), а Таблиця 2 має Колонку 1 (Логічна величина), Колонку2 (Дата). Оскільки порядок цих колонок різний, таблиці мають різну структуру. MapInfo коректно об'єднує кілька таблиць в одну, але тільки в тому випадку, якщо вони мають однакову структуру.

Під час створення виразу, в якому використовується колонка, обов'язково необхідно врахувати тип колонки. Зокрема, неможливо використовувати математичні функції

для строкових колон, або функції дати та часу для чисельних колон.

3.3.3. Оператори в MapInfo

Першим варіантом присудка є оператори. В MapInfo є 5 різних груп операторів. В табл. 3.1 ці оператори показані по пріоритетам. Найбільший пріоритет мають круглі дужки. Саме вони обробляються в першу чергу. Наступні пріоритети мають математичні оператори. На шостому місці стоять оператори порівняння та географічні оператори, ще нижче, по пріоритету, стоять логічні оператори. Всі ці оператори за винятком географічних, студенти спеціальності 193 «Геодезія та землеустрій» вивчають в рамках дисципліни «Інформатика». Географічні оператори є унікальними операторами програми MapInfo. Наряду з операторами порівняння вони є складовими запитів. Результатом виконання географічних операторів, як і операторів порівняння є логічні величини. Тому розберемо їх використання більш детально. Однак, спочатку розглянемо визначення такого поняття, як центроїд.

3.3.4. Центроїд графічного обєкта

Згідно табл. 3.1, в MapInfo є 5 географічних операторів. Дія деяких з них заснована на понятті центроїда.

Центроїд – центральна точка графічного об'єкта

На жаль в ГІС немає точного визначення для місцеположення центроїда. В одних ГІС центроїд встановлюється в центрі тяжіння об'єкта, в інших – в центрі виділяючого прямокутника для об'єкта. В MapInfo всі графічні об'єкти мають центроїди. Для об'єктів різних типів використовуються різні варіанти встановлення центроїда.

Таблиця 3.1 – Пріоритет операторів в MapInfo [21]

Пріоритет	Група	Оператор	Опис оператора
1	Круглі дужки	()	Круглі дужки
2	Математичні оператори	^	Зведення в ступень
3		-	Від'ємний знак
4		*, /	Множення, ділення
5		+, -	Додавання, віднімання
6	Оператори порівняння	=, <>, >, <, >=, <=	Рівно, не рівно, більше, менше, більше рівно, менше рівно
	Географічні оператори	Contains, Contains Entire, Within, Entirely Within, Intersects	Містить, Повністю містить, Всередині, Повністю всередині, Перетинає
7	Логічні оператори	Not	Оператор заперечення
8		And	Оператор логічного сумування
9		Or	Оператор логічного множення
10		Like	Оператор логічного порівняння (як)

Наприклад, точкові об'єкти, мають всього один вузол на карті, до якого прив'язується умовний знак. Ця точка одночасно є центроїдом точкового об'єкта.

Центроїд лінійного об’єкту в MapInfo встановлюється в залежності від типу лінійного об’єкта. Для відрізка прямої лінії центроїд встановлюється в середині відрізка, так як показано на рис. 3.10 a). Таким чином центроїд ділить відрізок на дві рівні частини. Положення центроїду полілінії з точки зору здорового глузду також повинно встановлюватися в середині та ділити відстань між початковим та кінцевим вузлами навпіл. Однак в MapInfo з полілінією ситуація не така однозначна. Положення центроїда полілінії залежить від кількості вузлів. Якщо кількість вузлів полілінії - не парна, то центроїд встановлюється в центральному вузлі, так як це показано на рис. 3.10 c). Якщо кількість вузлів полілінії парна, то центроїд встановлюється в центрі центрального сегмента, так як це показано на рис. 3.10 b). Для дуги еліпса центроїд взагалі знаходиться за межами об’єкта (див. рис. 3.10 d). Дуга еліпса - це єдиний лінійний об’єкт, центроїд якого знаходиться так як і центроїд площинного об’єкта, в центрі виділяючого прямокутника об’єкта. Саме тому центроїд не знаходиться в середній точці лінії, а за її межами.

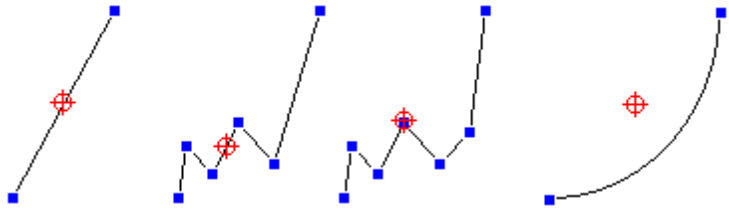


Рис. 3.10 – Положення центроїда a) відрізка прямої лінії, b)полілінії з парною кількістю вузлів, c) з непарною кількістю вузлів, d) дуги еліпса

Виділяючий прямокутник –найменший прямокутник, повністю оконтурюючиий графічний об’єкт

Якщо центр виділяючого прямокутника площинного об’єкта знаходиться всередині цього об’єкта, то в MapInfo

центроїд розташовується в центрі виділяючого прямокутника об’єкта, так як це показано на рис. 3.11.

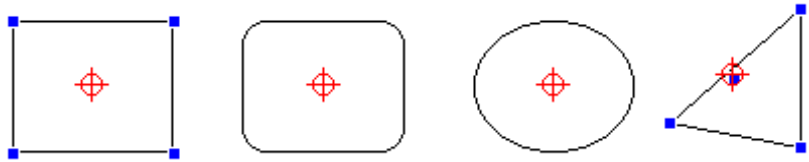


Рис. 3.11 – Положення центроїда a) прямокутника, b) прямокутника з округленими вершинами, c) еліпса, d) багатокутника

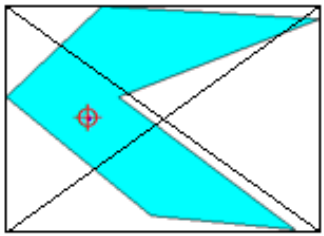


Рис. 3.12 – Положення центроїда багатокутника в тому випадку, коли центр виділяючого прямокутника знаходиться за межами багатокутника

В деяких випадках центр виділяючого прямокутника площинного об’єкта знаходиться за межами цього об’єкта (рис. 3.12). Однак, головне правило встановлення центроїда – він обов’язково повинен знаходиться всередині площинного об’єкта. Тому якщо центр виділяючого прямокутника знаходиться за межами площинного об’єкта, то центроїд все одно встановлюється всередині об’єкта по можливості на однаковій відстані від границь об’єкта. На рис. 3.12, як і на всіх інших рисунках, центроїд показаний червоним кольором.

Для багатокутників положення центроїда можна змінити вручну. Це зв’язано з тим фактом, що в географії, центроїд деякої території на земній поверхні, розглядається як географічний центр. Географічним центром країни є столиця

цієї країни. Вона розташовується в межах країни, але її положення зовсім не співпадає з центром багатокутника, межі якого співпадають з межами країни. Аналогічно, кожний населений пункт має свій центр, Найчастіше це центральна площа, в межах якої розташовані центральні магазини, пункти обслуговування, будівлі з органами управління, найвидатніші історичні пам'ятники. На рис. 3.13 a) показана схема ідеального міста, Палманова, спроектованого та побудованого ще у 16 столітті італійським архітектором Вінченцо Скамоцці [4]. Центр цього міста розташований в центрі схеми. Однак праворуч, на рис. 3.13 b) показана схема реального населеного пункту, розташованого на березі річки. Уважно подивіться на рис. 3.13 b). Центр міста знаходиться зовсім не в центрі схеми, він розташований біля границі міста, яка співпадає з береговою лінією. Саме тому в MapInfo центроїд багатокутника можна змінити в ручному режимі, за допомогою миші та клавіш.

a)

b)

Рис. 3.13 – Центр населеного пункту a) ідеального (http://www.alltravels.com.ua/2017/03/14/ideal-city-palmanova/), b) реального (https://kostromka.ru/bochkov/maps.php)

3.3.5. Географічні оператори

Географічний оператор Contains (Містить). Його дія заснована на розташуванні центроїда об'єкта. В операції беруть участь два графічних об'єкти: об'єкт A, та об'єкт B.

Дія оператора Contains (Містить) може бути сформульована так:

Об'єкт A Contains (Містить) об'єкт B, якщо центроїд об’єкта B лежить в межах об’єкта A

Під час виконання оператора Contains перевіряється розташування центроїда об’єкта B по відношенню до границь об’єкта A (рис. 3.14). Якщо центроїд об’єкта B знаходиться всередині границь об’єкта A, то результатом дії цього оператора буде логічна величина True, якщо навпаки центроїд об’єкта B знаходиться за межами об’єкта A, то результатом дії цього оператора буде логічна величина False.

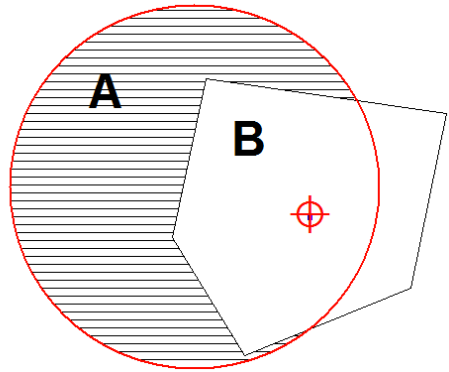


Рис. 3.14 – Виконання оператора Contains (Містить) між об’єктом A (коло) та об’єктом B (багатокутник)

Географічні оператори діють по відношенню до всього графічного об’єкта, а не до деякої колонки в таблиці. Тому серед списку колонок таблиці в самому його кінці є особлива колонка, яка в самій таблиці відсутня. Це колонка з назвою obj, скорочений варіант слова object. Цим словом позначається графічний об’єкт. На рис. 3.15 показано діалогове вікно для створення наступного виразу:

objA Contains objB. (3.5)

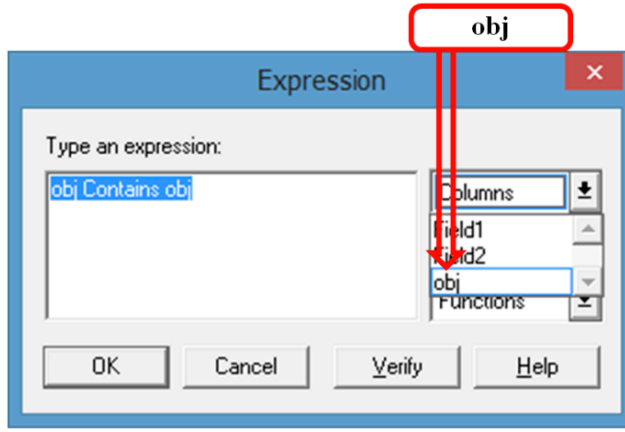


Рис. 3.15 – Вибір в списку Columns варіанту obj

Як показано на рис. 3.15, в самому кінці списку Columns діалогового вінка Expression представлена віртуальна колонка з назвою obj. Вона позначена червоною стрілкою.

Для формування запиту з географічним оператором спочатку необхідно вибрати графічний об'єкт зі списку Columns, потім сам географічний оператор зі списку Operators, а потім знов графічний об'єкт зі списку Columns.

Географічний оператор Contains Entire (Повністю містить).

Об'єкт A Contains Entire (Повністю містить) об'єкт B, якщо межі об'єкта B повністю лежать всередині кордонів об'єкта A

В даному випадку перевіряються кордони обох об'єктів, а не розташування центроїда. Формування запиту аналогічно попередньому. Він виглядає так як показано в нижче.

objA Contains Entire objB. (3.6)

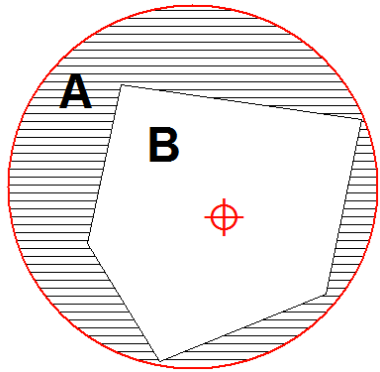


Рис. 3.16 - Виконання оператора Contains Entire (Повністю містить) між об’єктом А (коло) та об’єктом В (багатокутник)

Географічний оператор Within (Всередині).

Об’єкт А Within (Всередині) об’єкта В, якщо центроїд об’єкта А знаходиться в межах об’єкта В

Як і в першому випадку перевіряється розташування центроїда. Однак на відміну від першого випадку перевіряється розташування центроїда не об’єкта В, а об’єкта А, тобто кола, по відношенню до границь об’єкта В. Приклад запиту з оператором Within виглядає так:

objA Within objB. (3.7)

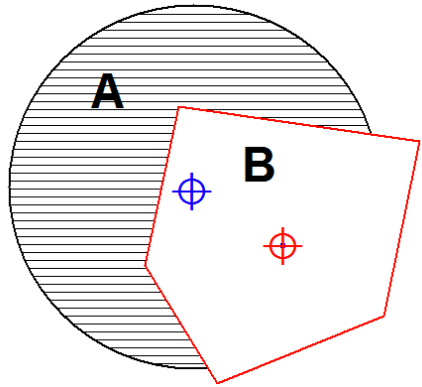


Рис. 3.17 - Виконання оператора Within (Всередині) між об’єктом А (коло) та об’єктом В (багатокутник)

На рис. 3.17 показана дія оператора Within, згідно якому центроїд кола знаходиться в межах границі багатокутника.

Географічний оператор Entirely Within (Повністю Всередині).

Об’єкт А Entirely Within (Повністю Всередині) об’єкта В, якщо його межа повністю знаходиться всередині кордонів об’єкта В

На рис. 3.18 показаний об’єкт А, коло, яке знаходиться повністю всередині об’єкта В, багатокутника. Приклад запиту виглядає так:

objA Entirely Within objB. (3.8)

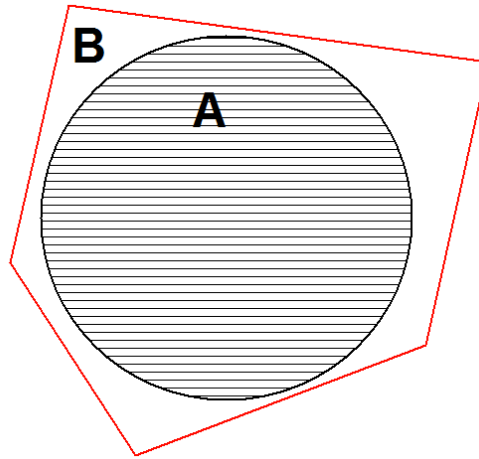


Рис. 3.18 - Виконання оператора Entirely Within (Повністю Всередині) між об’єктом А (коло) та об’єктом В (багатокутник)

Географічний оператор Intersects (Перетинає).

Об’єкт A Intersects (Перетинає) об’єкт B, якщо об’єкти мають хоча б одну спільну точку

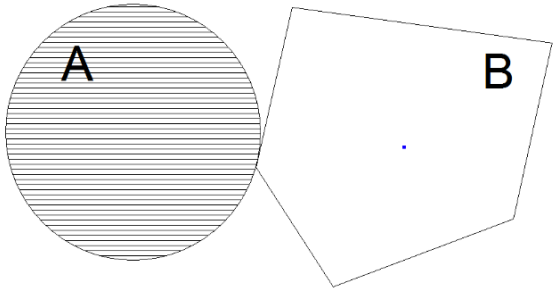


Рис. 3.19 – Виконання оператора Intersects між об’єктом A (коло) та об’єктом B (багатокутник)

На рис. 3.19 представлений варіант перетинання об’єктів A (кола) та B (багатокутника). Вираз з цим оператором виглядає так:

objA Intersects objB. (3.9)

Між першими чотирма географічними операторами: Contains; Contains Entire; Within; Entirely Within та оператором Intersects є суттєва різниця. Це зв’язано з тим фактом, що один з об’єктів в перших чотирьох географічних операторах обов’язково повинен бути площинним, так як межі мають тільки площинні об’єкти. На рис. 3.14-3.19 об’єкти, які обов’язково повинні бути площинними, показані за допомогою червоних кордонів. Для оператора Intersects обидві об’єкти можуть бути довільними, головна умова, вони повинні мати спільні точки.

Всі географічні оператори доступні в вікні Select (рис. 3.2), тобто їх формально можна використовувати для формування запиту за допомогою команди Select (Вибрати). Однак, слід зазначити, що дія цих операторів призводить, в цьому випадку, до парадоксальних результатів. Це зв’язано з тим фактом, що об’єкти, які беруть участь в виразах з географічними операторами, знаходяться в одній таблиці. Під

час виконання одного з цих операторів формуються два списки графічних об'єктів і кожен об'єкт з першого списку (це об'єкт А) перевіряється по відношенню до всіх об'єктів другого списку (це об'єкти В). Оскільки запит за допомогою команди Select (Вибір) формується по відношенню тільки до однієї таблиці, то обидва списки повністю ідентичні. Якщо об'єкт знаходиться в першому списку, то він знаходиться і в другому. Тобто в даному випадку кожен об'єкт А має свого повного двійника в якості об'єкта В. Очевидно, об'єкт А повністю співпадає по всім точкам зі своїм двійником, об'єктом В. Це призводить до того, що в результаті дії графічних операторів всі без винятку об'єкти будуть обрані і попадуть в таблицю запиту. Такий запит не має ніякої позитивної інформації. На рис. 3.20 показаний результат запиту з виразом, створеним за допомогою оператора Intersects. Не зважаючи на те, що об'єкти не мають ні однієї спільної точки, вони все одно є обраними, так як мають безліч спільних точок зі своїми копіями, в обох списках об'єктів, як у списку А, так і у списку В.

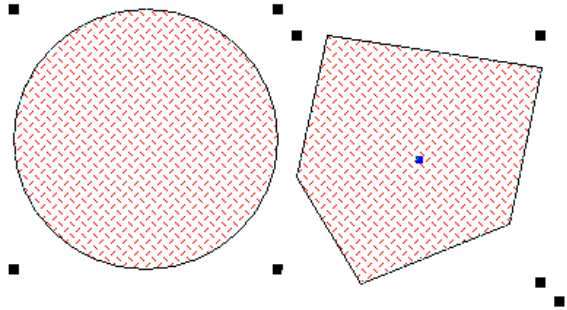


Рис. 3.20 – Результат виконання оператора Intersects для запиту, сформованого до однієї таблиці

Тому [21], використовувати географічні оператори для запитів, сформованих до однієї таблиці, не рекомендується. Географічні оператори дають позитивну інформацію тільки в тому випадку, якщо об'єкт А знаходиться в одній таблиці, а

об’єкт В – в інший. Докладніше про формування запитів до декількох таблиць дивиться далі, розд. 3.4.2.

3.3.6. Географічні функції

Другим варіантом присудка є функції. В MapInfo, згідно рис. 3.8, використовується 4 групи різноманітних функцій. Математичні, текстові дати та часу – це стандартні функції, з якими користувачі знайомляться, використовуючи різні програми, наприклад, Excel, Access тощо. Однак, як і серед операторів, серед функцій MapInfo є функції, які є унікальними. Це географічні функції.

Є 6 основних географічних функцій програми MapInfo:

- **Area(obj, одиниці_вимірювання);**
- **CentroidX(obj);**
- **CentroidY(obj);**
- **ObjectLen(obj, одиниці_вимірювання);**
- **Perimeter(obj, одиниці_вимірювання);**
- **Distance(x1, y1, x2, y2, одиниці_вимірювання)**

Як і географічні оператори, географічні функції діють по відношенню не однієї, або декількох колонок, або констант, а по відношенню до графічних об’єктів. Тому першим аргументом більшості географічних функцій є ключове слово obj (object). Таких функцій в програмі 5. Це функції Area (), CentroidX(), CentroidY(), ObjectLen(), та Perimeter(). Остання, шоста функція Distance(), також вважається географічною, але не є класичною географічною функцією, так як не має в якості аргументу ключового слова obj, тобто не відноситься до всього графічного об’єкту. Але дія цієї функції практична така сама, що й інших географічних функцій.

Слід зазначити, що в сучасних версіях програми з’явилися нові модифікації цих географічних функцій. Більш детально див. [21].

Всі географічні функції, за винятком двох функцій CentroidX() та CentroidY(), в якості аргументу мають таку величину, як одиниці вимірювання. Це можуть бути одиниці вимірювання довжин ліній, або одиниці вимірювання площ. Варіанти цих одиниць представлені в додатках Б та В.

Географічна функція Area(obj, одиниці_вимірювання площі). Призначення функції – обчислення площі графічного об'єкта. Функція має два аргументи. Перший аргумент – obj, свідчить про те, що функція відноситься до всього графічного об'єкта. Другий аргумент – одиниці вимірювання. За замовчуванням функція Area() обчислюється в одиницях вимірювання sq mi (квадратні мілі). Їх досить легко поміняти на необхідні одиниці вимірювання площі (див. Додаток В).

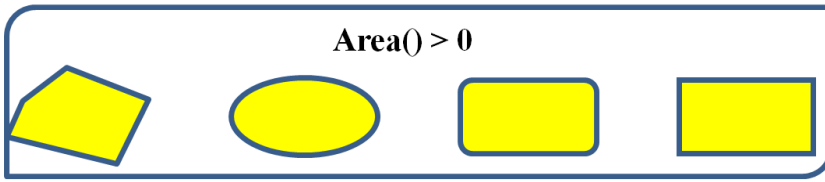


Рис. 3.21 – Об'єкти, для яких функція Area() > 0

Тільки полігони, еліпси, прямокутники та прямокутники із округленими кутами мають позитивну величину площі, обчисленої за допомогою функції Area() (рис. 3.21). Результат застосування функції до точкових, текстових об'єктів, а також до прямої лінії, дуги еліпса та полілінії буде дорівнювати нулю.

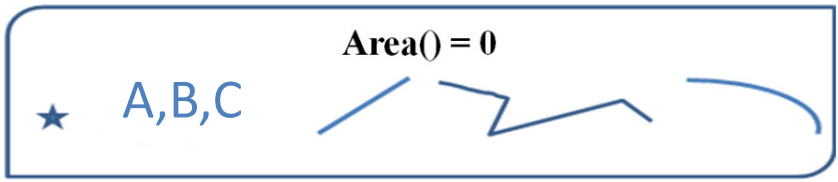


Рис. 3.22 – Об'єкти, для яких функція Area() = 0

Для округленого прямокутника функція Area() повертає приблизне значення, так як обчислює площу прямокутника із округленими кутами, якби він був звичайним прямокутником

без округлених кутів. Результат обчислення функції Area() не є однозначним, оскільки MapInfo проводить або Декартові обчислення площі на площині, або обчислення площі на поверхні земного шару. За замовчуванням обчислення площі виконується на поверхні земного шару. Тільки в тому випадку, якщо в якості проекції для таблиці обраний варіант Non-Earth (План-схема), виконуються обчислення площі на площині.

В деяких випадках необхідно обчислити площу об'єкта на площині в проекції. Однак, якщо проекція таблиці – варіант, відмінний від Non-Earth, функція Area() такого значення не дасть. Саме тому в подальших версіях програми MapInfo були додатково додані функції CartesianArea() та SphericalArea(). Якщо необхідно обчислити площу на площині, то обирається функція CartesianArea(), якщо навпаки потрібно обчислити площу на поверхні Земного шару, то обирається функція SphericalArea(). Обидві функції аналогічні функції Area(), зокрема мають ті ж самі аргументи.

Проте, якщо функція CartesianArea() використовується для обчислення площі об'єктів таблиці, для якої обрана проекція Longitude/Latitude (Широта/Довгота), тобто не має ніякої проекції, то функція повертає замість площі, величину, рівну мінус одиниці. Аналогічно, якщо функція SphericalArea() використовується для обчислення площі об'єктів таблиці, для якої обрана проекція Non-Earth, тобто не має ніякої прив'язки до поверхні Земного шару, то функція повертає замість площі мінус одиницю. Така парадоксальна величина повинна насторожити користувача програми, та змусити його переобчислити площу за допомогою іншої функції. Уважно подивіться на табл. 3.2. Вона дозволяє зорієнтуватися в тому, як правильно обирати функцію обчислення площі.

Географічні функції CentroidX(obj), CentroidY(obj) повертають координату центральної точки (центроїда)

довільного графічного об'єкта по осі X та по осі Y, відповідно. Якщо координати точок проекції передбачають формат широти - довготи, то функція CentroidX(obj) повертає довготу, а функція CentroidY(obj) - широту.

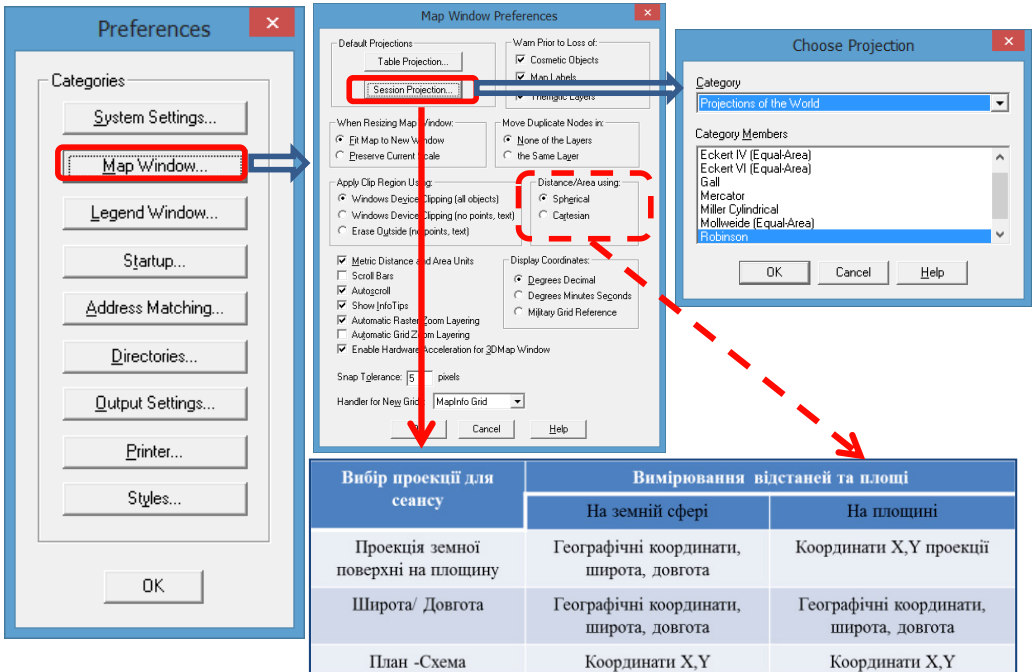
Таблиця 3.2 – Результат використання географічних функцій MapInfo в залежності від обраної проекції [21]

Проекція	Функція	
	CartesianArea() CartesianObjectLen() CartesianPerimeter() CartesianDistance()	SphericalArea() SphericalObjectLen() SphericalPerimeter() SphericalDistance()
Проекція земної поверхні на площину	>0	>0
Longitude / Latitude	-1	>0
Non-Earth	>0	-1

Самі функції не дозволяють обрати потрібні координати. Програму попередньо слід налаштувати на необхідний варіант координат. Є два варіанти: географічні координати: широта та довгота, та прямокутні координати X,Y. В першому випадку координати за допомогою функцій CentroidX() та CentroidY() будуть отримані в градусах та долях градусів. В другому випадку координати за допомогою функцій CentroidX() та CentroidY() будуть отримані в метрах та долях метрів.

Для виконання налаштування програми на необхідні системи координат, потрібно в меню Options (Настройки) обрати команду Preferences (Режими). Відкриється діалогове вікно з тією ж назвою, показане на рис. 3.23 ліворуч. В цьому вікні слід обрати кнопку Map Window (Вікно карти). На рис. 3.23 вона обведена червоною стрічкою. Відкриється вікно налаштувань режимів карти, показане в центрі рис. 3.23.

Координати центроїдів об’єктів напряму залежать від вибору варіанта проекції для сеансу роботи в програмі. Для вибору проекції сеансу необхідно в вікні Map Window Preferences натиснути кнопку з назвою Session Projection (Проекція для Сеансу). Вона як і попередня кнопка, обведена червоною стрічкою в лівій частині вікна. Відкриється вікно вибору Проекції, показане в правій частині рис. 3.23. Перехід між діалоговими вікнами позначений синіми стрілками. Якщо в вікні вибору проекції обрати варіант Longitude/Latitude (Широта-Довгота), то незалежно від того, яка насправді проекція таблиці і який варіант обчислення довжин та площ обраний в вікні режимів карти, для центроїдів будуть обчислені географічні координати, широта та довгота в градусах. Функція CentroidX(obj) обчислить довготу, CentroidY(obj) – широту. Якщо в вікні вибору проекцій буде обраний варіант Non-Earth (План-схема), то навпаки, не залежно від того, яка проекція таблиці та який обраний варіант обчислення довжин та площ, координати центроїда будуть обчислені в метрах. Тільки в тому випадку, якщо у вікні вибору проекції обраний варіант конкретної картографічної проекції, яка має формули зв’язку географічних координат з прямокутними, вибір варіанту вимірювання відстаней та площ має сенс. Цей вибір встановлюється за допомогою перемикача Distance/Area using (Вибір одиниць вимірювання довжин/площ), який показаний червоною переривчастою стрічкою в правій частині вікна Map Window Preferences. Якщо перемикач встановлений на варіант Spherical (На сфері), то результатом виконання функцій CentroidX(obj) та CentroidY(obj) є широта та довгота в градусах, Якщо перемикач встановлений на варіант Cartesian (На площині), то результат виконання функцій CentroidX(obj) та CentroidY(obj) є координати X, Y в метрах. Все вищесказане наочно показано в таблиці, яка представлена в нижній частині рис. 3.23.



Вибір проекції для сеансу	Вимірювання відстаней та площі	
	На земній сфері	На площині
Проекція земної поверхні на площину	Географічні координати, широта, довгота	Координати X,Y проекції
Широта/ Довгота	Географічні координати, широта, довгота	Географічні координати, широта, довгота
План -Схема	Координати X,Y	Координати X,Y

Рис 3.23 – Процедура налаштування MapInfo на необхідний варіант координат

Географічна функція ObjectLen(obj, одиниці_вимірювання_довжини) призначена для обчислення довжини лінії таких лінійних об'єктів, як відрізок прямої лінії та полілінії. Для всіх інших об'єктів функція дає значення, рівне нулю. Навіть для лінійного об'єкта дуги еліпса, ця функція дає нульове значення. Подивіться на об'єкти на рис. 3.24, 3.25. Серед площинних об'єктів, точкових та тексту, праворуч зеленим кольором показана дуга еліпса. Це лінійний об'єкт, але довжина цього об'єкта за допомогою функції ObjectLen() не обчислюється.

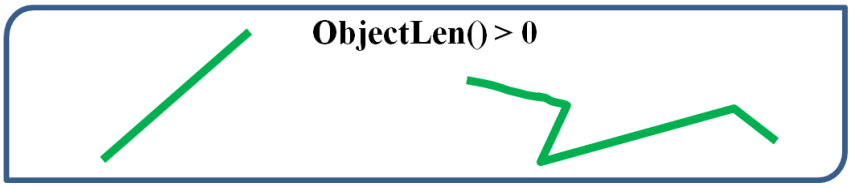


Рис. 3.24 - Об'єкти, для яких функція ObjectLen() > 0

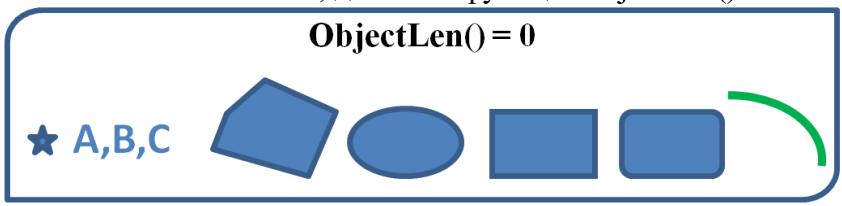


Рис. 3.25 - Об'єкти, для яких функція ObjectLen() = 0

Як і функція Area(), функція ObjectLen() має два аргументи. Перший аргумент, obj, свідчить про те, що функція виконується для всього графічного об'єкта, а не для окремої колонки в таблиці об'єкта. Другий аргумент – одиниці вимірювання довжини. Доступні одиниці вимірювання довжин представлені в додатку Б. Довжина обчислюється в тих одиницях, які задані в другому аргументі функції.

За замовчуванням, обчислюється довжина лінії на земному шарі. Тільки в тому випадку, коли в якості проекції обраний варіант Non-Earth (План-схема), обчислюється довжина лінії на площині.

Як і функція Area(), функція ObjectLen() має дві модифікації: CartesianObjectLen() та SphericalObjectLen(). Перша функція обчислює довжину лінії на площині, в проекції. Друга функція обчислює довжину лінії на сфері. Вільний вибір функцій, як і для функцій обчислення площі є тільки в тому випадку, коли в якості проекції обрана реальна картографічна проекція, для якої є формули зв'язку між географічними координатами на сфері, або еліпсоїді та прямокутними координатами на площині в проекції. Якщо в якості проекції обраний варіант Longitude/Latitude (Широта-Довгота), то функція CartesianObjectLen() повертає значення, рівне мінус одиниці. Аналогічно, якщо в якості проекції обраний варіант Non-Earth (План-схема), то функція SphericalObjectLen() повертає значення, рівне мінус одиниця. Варіанти некоректних та коректних значень функцій CartesianObjectLen() та SphericalObjectLen() показані в табл. 3.2.

Географічна функція Perimeter(obj, одиниці_вимірювання_довжини) обчислює периметр площинного об'єкта. Це може бути полігон, прямокутник, прямокутник з округленими вершинами, або еліпс (рис. 3.26). Функція за своїми властивостями повністю співпадає з функцією обчислення площі для площинних об'єктів. Зокрема, вона обчислює периметр площинного об'єкта на поверхні земної сфери. Тільки в тому випадку, коли проекція – Non-Earth (План – схема), обчислення виконуються на площині.

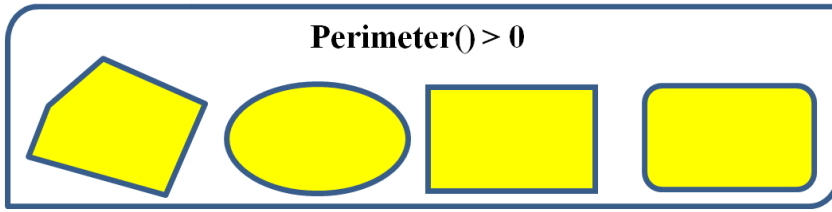


Рис. 3.26 - Об'єкти, для яких функція Perimeter() > 0

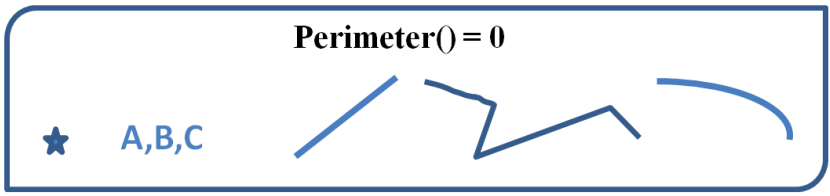


Рис. 3.27 - Об’єкти, для яких функція Perimeter() = 0

Як і у випадку обчислення площі, для обчислення периметру в більш нових версіях програми MapInfo використовуються дві додаткові функції CartesianPerimeter() і SphericalPerimeter(), які дозволяють конкретизувати поверхню, на якій виконуються обчислення (див. табл. 3.2).

Географічна функція Distance(X1,Y1,X2,Y2,одиниці_вимірювання_довжини) обчислює відстань між двома точковими об’єктами. Саме тому першими аргументами цієї функції є координати першої точки, X1 Y1, та координати другої точки X2 та Y2. П’ятий, останній аргумент цієї функції повністю співпадає з останнім аргументом попередніх географічних функцій. Це одиниця вимірювання відстані. Перші чотири аргументи можуть бути або константами, або функціями, які визначають координати точок. Нижче показаний приклад, який найбільш часто використовується в виразах, тому його можна вважати стандартним варіантом використання функції Distance(). Він дозволяє обчислити відстань між фіксованою точкою, яка визначена двома константами, та центроїдом довільного графічного об’єкта.

Distance (33.38044, 47.90966, CentroidX(obj), CentroidY(obj), “km”). (3.11)

Як і попередні функції, такі як Area(), ObjectLen(), та Perimeter(), функція Distance() обчислює відстань на поверхні земного шару. Тобто обчислюється відстань між двома точками на земному шарі вздовж великого круга, який, як відомо, має назву ортодромії (геодезичної лінії) для поверхні шару, або дуги великого кола. Тільки в тому випадку, якщо в

групі проекцій обраний варіант Non-Earth (План-схема), обчислюється відстань між двома точками на площині.

Для того, щоб можна було вільно обчислювати відстань в проекції на площині, або на поверхні сфери, в подальших варіантах програми введені дві функції, повністю подібні попереднім: CartesianDistance() і SphericalDistance(). Варіанти їх використання показані в табл. 3.2.

3.3.7. Ключові словосполучення в виразах

Згідно [MapInfo], крім операторів та функцій в виразах MapInfo можна використовувати ключові словосполучення. Їх не багато, всього 5.

Ключові словосполучення:

- **any(…)**
- **all(…)**
- **in(…)**
- **not in (…)**
- **between … and…**

Без них, в принципі, можна обійтися і використовувати тільки оператори та функції. Але їх використання досить сильно наближає запит до звичайного питання звичайною мовою, зрозуміло англійською мовою. За допомогою ключових слів запит стає більш простим для сприйняття та розуміння.

Аналіз списку ключових слів показує, що перші два ключових слова any() та all(), як вказано в [19] визначені як особливі агрегатні функції, які мають назву **кванторних функцій**. В MapInfo ці елементи також використовуються як кванторні функції, однак їх нема в списку функцій тому їх, як

і інші ключові слова, неможливо обрати зі списку, їх можна тільки набрати вручну на клавіатурі.

Стандартна схема використання цих функцій в виразах виглядає так:

Назва Колонки + **Оператор 6 рівня пріоритету** + **Кванторна функція (аргументи)**. (3.12)

Більш детально схема використання кванторних функцій показана на рис. 3.28.

Згідно цій схемі першим у виразі, який містить кванторну функцію розташовується назва колонки. Це може бути назва реальної колонки в таблиці, або віртуальної, тобто obj. Згідно схемі (3.12), функції any() та all() використовуються сумісно з операторами шостого рівня пріоритету. Це оператори порівняння та географічні оператори (табл. 3.1). Всі ці оператори показані на рис. 3.28 зеленим кольором.

Якщо у виразі використовується реальна колонка, то далі обирається один з операторів порівняння, якщо застосовується obj, то обирається один з географічних операторів. Наступний крок – вибір однієї з двох кванторних функцій. Для реальної колонки аргументами функції можуть бути значення, які є в цій реальній колонці.

Якщо використовується віртуальна колонка obj, то аргументом функції може бути тільки графічний об’єкт, тобто obj. Аргументи можуть бути представлені у вигляді констант, виразів або підзапитів. Незалежно від того, який вигляд мають аргументи, вони повинні бути того ж типу, що і початкова колонка. Тобто, якщо колонка реальна, то аргументи порівнюються зі значенням цієї колонки, якщо колонка віртуальна (obj), то аргументом функції може бути тільки графічний об’єкт.

Рис. 3.28 – Схема використання кванторних функцій

Зі знаком «=» кванторна функція any() означає, що необхідно вибрати довільний елемент з обраних. Наприклад, необхідно обрати країни з таблиці World, які знаходяться на континентах Азії, або Європи. Для цього можна використати вираз (3.13). Згідно цьому виразу обираються всі країни, для яких значення колонки Continent дорівнює текстовій величині «Asia», або «Europe». Тобто будуть обрані всі країни Азії, та Європи.

Continent =any("Asia", "Europe") . (3.13)

Функція any() сумісно з оператором «<>» (не рівно) дає досить парадоксальний результат, так як обираються записи, для яких континент має значення, ні Азія ні Європа одночасно.

Continent <> any("Asia", "Europe"). (3.14)

Країни Азії не знаходяться в Європі. Тому попадуть в список. Аналогічно країни Європи не знаходяться в Азії, тому теж попадуть в список. Так як в колонці Continent записано тільки одне слово, або Asia, або Europe, то результат буде повністю співпадати з вихідною таблицею. Нічого, по великому рахунку, обиратися не буде. Досить коректно функція буде працювати сумісно з оператором «<>» тоді, коли в якості аргументу буде всього одне значення, наприклад Asia. Тоді результатом буде таблиця, в яку увійдуть країни, не розташовані на континенті Азія.

Continent <> any("Asia"). (3.15)

Використання функції all() з оператором «=» в тому випадку, коли аргументами є декілька значень так як це показано в прикладі (3.16), дає також парадоксальний результат.

Continent =all("Asia", "Europe"). (3.16)

При виконанні умови (3.16) повинні обратися всі записи, для яких Continent має значення, одночасно і «Asia» і

«Europe». Так як таких записів немає, результат – ні одного запису.

Коректний результат дає функція all() з оператором «<>». В прикладі за номером (3.17) обираються записи, для яких рядки в колонці Continent мають значення, ні «Asia» ні «Europe». Результат – записи, відмінні від Азії, або Європи, тобто всі країни інших континентів.

Continent <> all("Asia", "Europe"). (3.17)

Як було зазначено в розд. 3.3.5, географічні оператори дають позитивний результат в тому випадку, коли об'єкти, які є їх аргументами, знаходяться в різних таблицях. Тому приклади з варіантом віртуальної таблиці представлені далі в розд. 3.4.2.

Наступні ключові словосполучення in() та not in(), є частковими випадками кванторних функцій any(), all() (табл. 3.2). Зокрема [21], результат використання слова in() еквівалентний використанню варіанту =any(). Наприклад, результат виконання запиту з виразом (3.13) повністю співпадає з результатом виконання виразу (3.18), в якому обираються країни з Азії або Європи.

Continent in("Asia", "Europe"). (3.18)

Аналогічно, використання словосполучення not in() еквівалентно використанню варіанту <>all(). Це можна простежити по результатам виконання виразів (3.17) та (3.19). В обох випадках результатом буде таблиця, в якій записи, відмінні від Азії, або Європи, тобто всі країни з інших континентів. Використання ключових словосполучень in() та not in() дозволяє уникнути парадоксальних результатів на відміну від функцій any(), або all().

Continent not in("Asia", "Europe"). (3.19)

Таблиця 3.2 – Часткові варіанти кванторних функцій [21]

Оператор	any()	all()
=	in()	-
<>	-	not in()

Останнє ключове словосполучення складається з двох слів between та and та має два аргументи, на відміну від попередніх, кількість аргументів яких не обмежена.

Нижче представлені два приклади використання ключових слів between … and. Приклад (3.20) використовує чисельні значення в колонці Population таблиці World та дозволяє обрати записи, значення яких в колонці більше 50 тисяч та менше 100 тисяч.

Population between 50000 and 100000. (3.20)

Цим виразом обираються країни, кількість мешканців яких більша ніж 50 тисяч чоловік та не перевищує 100 чоловік.

Приклад (3.21) використовує літери для порівняння. Як відомо, літери також можна порівнювати. Літера А менше літери Б, яка в свою чергу менше літери В. Тому літери, та не тільки літери, а цілі слова можна порівнювати один з одним. В прикладі (3.21) представлений вираз за допомогою якого обираються країни, назви яких починаються на літеру В.

Country between "B" and "C". (3.21)

3.4. Побудова запиту командою SQL Select

Всі запити, сформовані і виконані за допомогою команди Select (Вибрати), можна також виконати за допомогою команди SQL Select. Зворотне не вірно, тому що команда SQL Select має більш широкі можливості ніж команда Select, та більш близька до стандарту SQL.

Давайте розберемо властивості команди SQL Select за допомогою її схеми, показаній на рис. 3.29.

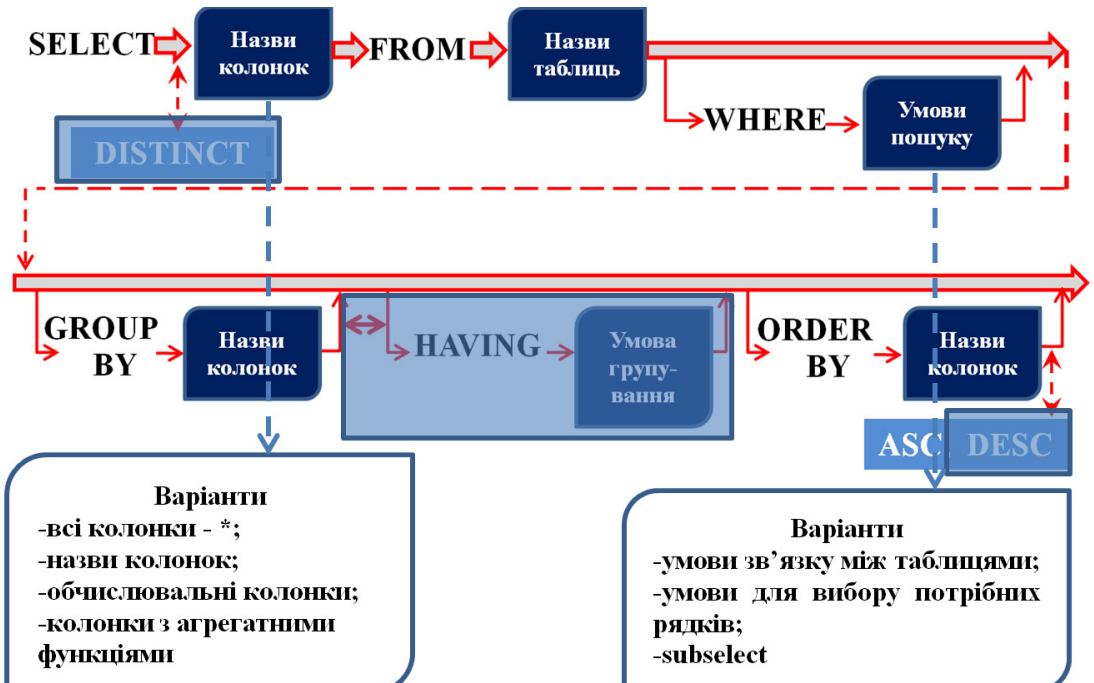


Рис 3.29 - Схема оператора Select для команди SQL Select

На рис.3.29 показана стандартна схема SQL – запиту. Однак, три елементи цієї схеми закрити напівпрозорими прямокутниками. Вони показують ті елементи схеми, які відрізняють стандарт SQL – запиту від SQL – запиту в програмі MapInfo.

Перша відмінність: не має можливості використати ключове слово Distinct. Як вже було зазначено в розд. 2.3.1, це слово дозволяє не виводити однакові рядки в результуючий таблиці. Таким чином, в SQL – запиту програми MapInfo такої можливості не має.

Якщо уважно подивитися на схему запиту, то можна виявити ключове слово Groupe By, що свідчить про можливість групування інформації по рядках в SQL – запиту програми MapInfo. Однак, зв'язане зі словом Groupe By, інше ключове слово HAVING разом зі своїм атрибутом, умовою пошуку, закрито напівпрозорим прямокутником. Це свідчить про той факт, що групування в SQL – запиту програми MapInfo є спрощеним, без можливості фільтрування не потрібних даних.

Останній напівпрозорий прямокутник встановлений на ключовому слові Desc, яке є скороченням від descending -. сортування за зменшуванням даних. Отже в SQL – запиту програми MapInfo є можливість сортувати дані по декільком колонкам, але тільки за збільшуванням, а не зменшуванням даних.

Ліворуч показані варіанти вибору колонок. Вони повністю співпадають зі стандартом SQL – запиту. Зокрема, можна вибрати всі колонки, можна вибрати тільки ті колонки, які потрібні для запиту за їх назвами, можна створити нові колонки, використовуючи функції та оператори програми.

Аналогічно праворуч показані варіанти умови пошуку в ключовому слові WHERE. Наряду з класичними умовами, які використовуються для фільтрації рядків, в поле для умов

вставляються ще два варіанти умов: умови зв'язку між таблицями та підзапит (Subselect).

На відміну від класичних реляційних баз даних, в яких зв'язки між таблицями встановлюються заздалегідь під час створення бази даних та використовуються в запитах вже готовими, в MapInfo не має зв'язків між таблицями, вони створюються під час запиту і показуються в вікні SQL Select. Ці зв'язки розглядаються як додаткові умови пошуку.

Другий варіант умов, відмінних від фільтрації рядків, це S**ubselect.** Тобто в один вибір, можна вложити інший вибір. В класичному SQL – запиту допускається 4 рівня вкладення. В MapInfo – тільки один рівень. Тобто можна застосувати тільки одне слово Select в умові пошуку.

Отже давайте сформулюємо особливості SQL запиту з командою SQL-Select.

1. Запит може здійснюватися не по одній, а по декількох таблицях;
2. Умови зв'язку між таблицями формуються під час запиту і є першими з умов в ключовому слові WHERE;
3. Кількість колонок – довільна, може бути обчислювальні колонки, в тому числі з агрегатними функціями;
4. Не має можливості використовувати ключове слово DISTINCT, що унеможливлює виключення дублікатів рядків;
5. Групування виконується за декількома колонками, однак не має можливості відсортувати згруповані дані, так як відсутнє ключове слово HAVING;
6. Відсутня можливість сортування за зменшенням даних. Є тільки одна можливість сортування за збільшенням даних. Однак сортувати можна за декількома колонками.

На рис. 3.30 показаний зв'язок схеми SQL – запиту з діалоговим вікном SQL Select програми MapInfo.

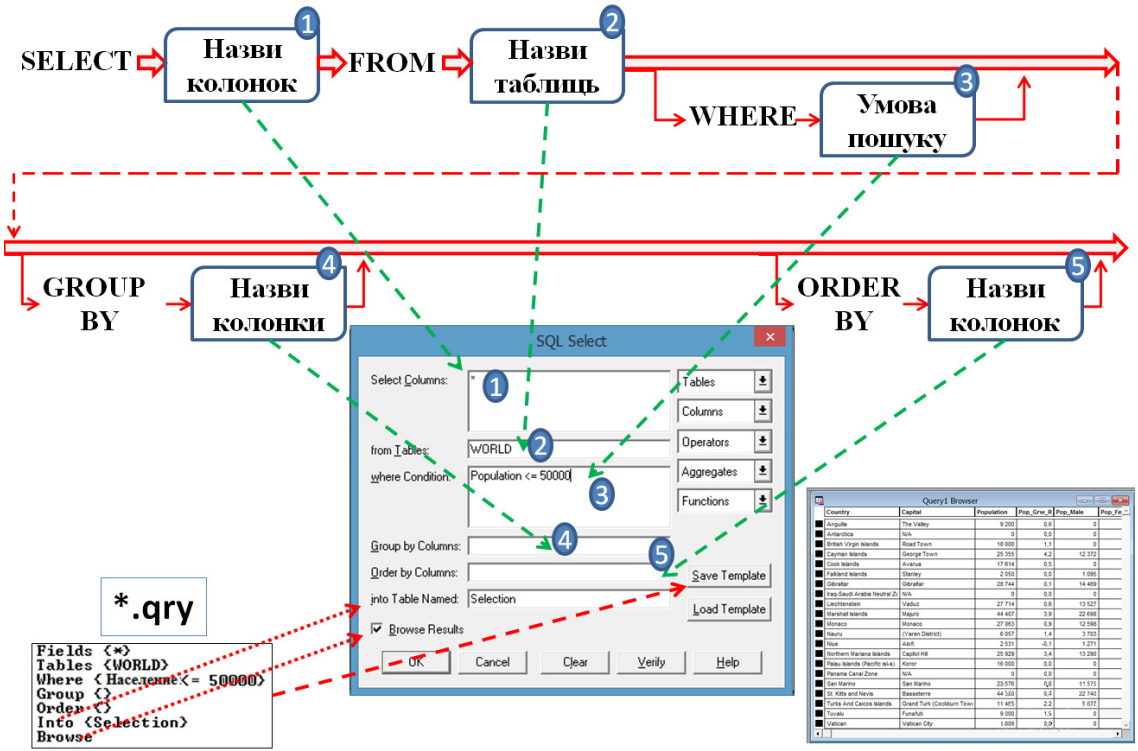


Рис 3.30 – Зв'язок схеми SQL – запиту з діалоговим вікном SQL Select

Це вікно, як вже було зазначено вище, на порядок більш складне, ніж діалогове вікно команди Select (Вибір). Для того, щоб зрозуміти структуру цього вікна, його поля на рис. 3.30 напряму зв'язані з атрибутами схеми запиту. Зв'язок показаний зеленими стрілками. Кожний атрибут позначений на схемі номером від одного до п'яти. В діалоговому вікні відповідні їм поля також позначені ти ж самими номерами. Зокрема, атрибут вибору колонок, тобто атрибут ключового слова SELECT зв'язаний з полем Select Columns (Вибрати колонки). За замовчуванням в це поле загружається зірка, так як показано у вікні. Для того, щоб вибрати конкретні поля з таблиці, необхідно видалити зірку та обрати назви полів зі списку Columns (Колонки).

Атрибут ключового слова FROM є зв'язаним з полем, розташованим біля слів from Tables (з таблиць). Назви таблиць обираються зі списку Tables (Таблиці).

Третій атрибут, умова пошуку ключового слова WHERE, зв'язаний з полем where Condition (з умовою). Саме в це поле вставляється вираз, який дозволяє отфільтрувати необхідні рядки, зв'язати таблиці та виконати Subselect.

Колонки для групування даних вставляються в поле (Group By Columns) (Згрупувати по колонках), так як це показано зеленою стрілкою, проведеною між атрибутом ключового слова GROUP BY та цим полем.

Для вибору колонок для сортування використовується поле Order by Columns (Порядок задати по колонкам), як показано в нижній частині діалогового вікна.

Таким чином, для формування запиту за допомогою команди **SQL Select, необхідно заповнити 5 різних полів діалогового вікна.**

Як і для діалогового вікна Select, для вікна SQL Select передбачено збереження умов запиту в текстовому файлі з розширенням *.qry. Файл, створений за атрибутами запиту, представленого в діалоговому вікні, показаний на рис. 3.30 ліворуч знизу. За допомогою цього запиту з таблиці World

обираються країни, кількість мешканців в яких не перевищує 50 тисяч чоловік.

Праворуч від діалогового вікна показаний результат цього запиту.

3.4.1. Вибір колонок для результуючої таблиці

Для того щоб коректно обрати назви колонок, спочатку необхідно обрати назву таблиці, або декількох таблиць для запиту. Тільки в цьому випадку в список колонок вставляються назви колонок з обраних таблиць. Таблиці попередньо повинні бути відкритими для роботи з ними в програмі.

Перша дія у вікні SQLSelect:
Вибір однієї таблиці, або декількох таблиць в списку Таблиці

Це перша дія, яку необхідно виконати. Без неї неможливо правильно сформувати запит. Вибір таблиць формується за допомогою списку Tables (Таблиці). Цей список представлений в діалоговому вікні SQL Select.

За замовчуванням в полі Select Columns встановлюється зірка (рис. 3.31 a). Це означає, що будуть обрані всі колонки зі списку доступних колонок.

Для того щоб обрати конкретні колонки, необхідно зірку видалити з поля Select Columns та обрати необхідні колонки зі списку Columns. На рис. 3.31 b) представлений варіант вибору двох колонок Country та Population. В результаті виконання запиту з таким вибором, буде сформована таблиця з двома а не зі всіма колонками.

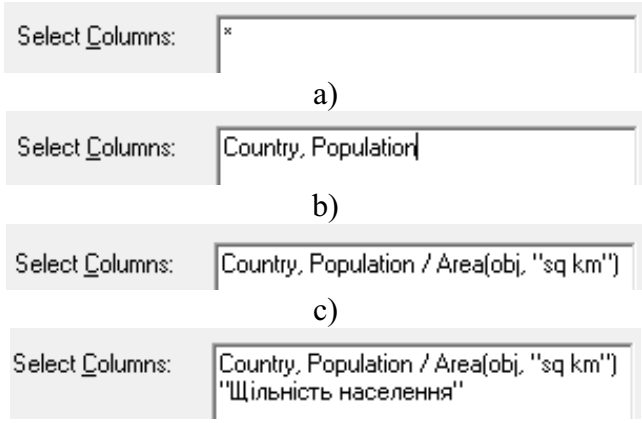


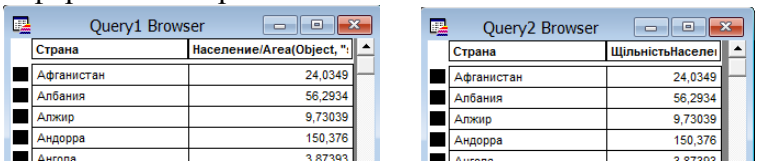
Рис. 3.31 – Варіанти заповнення поля Select Columns a) вибір всіх колонок b) вибір конкретних колонок за їх назвами; c) створення обчислювальної колонки; d) створення обчислювальної колонки з назвою

На рис. 3.31 c) представлений варіант формування обчислювальних колонок. Друга колонка в списку колонок під номером три – це обчислювальна колонка. В полі Select Columns сформований вираз, який дозволяє за кількістю мешканців та за площею обчислити щільність мешканців для кожної країни.

Назва для обчислювальної колонки за замовчуванням така ж сама як і сама формула для обчислювання. Однак, можна сформувати назву з текстового рядка. Для цього потрібно після формули в полі вибору колонок набрати пробіл, а потім в подвійних лапках набрати з клавіатури назву обчислювальної колонки. Для колонки, яка обчислює щільність населення саме таким чином сформована назва, яка показана на рис. 3.31 d).

На рис. 3.32 представлені таблиці з результуючими колонками, в яких друга колонка - щільність населення. Перша таблиця має колонку, назва якої співпадає з формулою

для обчислення щільності, друга таблиця має колонку, для якої сформована окрема назва.



a) b)

Рис. 3.32 – Варіанти назв обчислювальних колонок

3.4.2. Вибір декількох таблиць

Як вже було сказано раніше, команда SQL Select дозволяє сформувати запит до декількох таблиць. Для цього таблиці повинні бути зв'язані між собою, як це зазначено в розд. 1.3.3, одним з трьох типів зв'язку: «один до одного», «один до декількох». «декілька до декількох».

Таблиці в системі MapInfo не містять зв'язків. Тому під час створення запиту до декількох таблиць одночасно слід встановити зв'язок між ними. На підставі встановленого зв'язку формується загальна таблиця, для якої потім формується запит і створюється результуюча таблиця. Тип зв'язку програма визначає автоматично. Не зважаючи на тип зв'язку, під час процедури запиту створюється зв'язуюча таблиця (див. розд. 1.3.3), яка містить всі колонки таблиць, що зв'язуються. Саме до неї формується запит.

Є два способи завдання зв'язку в MapInfo [21].

Перший спосіб: формування умови з оператором дорівнює «=», в якому зліва від оператора задається колона однієї таблиці, справа – колона іншої. Наприклад:

Таблиця1.Поле1=Таблиця2.Поле2. (3.22)

Обидві колони, що беруть участь в умові, повинні мати один тип. Для формування зв'язку передбачається, що в колонах міститися одні і ті ж величини, наприклад скорочена назва штату. Або назва населеного пункту.

Другий спосіб: географічне об’єднання таблиць за допомогою географічних операторів.

Якщо дві таблиці мають графічні об’єкти, то MapInfo може об’єднати ці таблиці на основі просторових відносин між об’єктами цих таблиць. Тому якщо таблиці не містять загальної колонки, то можна об’єднати їх географічно за допомогою географічних операторів. (див. розд. 3.3.5). Вони використовуються для вибору об’єктів на підставі їх взаємного розташування в просторі. З географічними операторами в MapInfo Professional використовується віртуальна колонка obj, яка обирається в нижній частині списку колонок відповідних таблиць.

Ім’я географічного оператора вказується між географічними об’єктами; вибрати його можна в списку Operators в діалозі SQL Select. Схема географічного зв’язку має наступний вигляд:

Таблиця1.obj **Within** Таблиця2.obj. (3.23)

Умова поміщається в полі where Condition діалогового вікна SQL Select. Якщо в цьому полі передбачається помістити декілька умов, то вони повинні бути з’єднані оператором and. Причому умови зв’язку між таблицями повинні бути першими.

3.4.3. Узагальнення даних

На відміну від команди Select (Вибрати), яка дозволяє виконувати математичні операції з окремими записами, команда SQL Select дає можливість узагальнювати дані зі всіх записів в заданій колонці. Процедуру узагальнення умовно можна розділити на дві складові.

Перша частина процедури узагальнення зв’язана з використанням агрегатних функцій, які заносяться в поле Select Columns. Таким чином вони в результуючій таблиці будуть представлені у вигляді колонки. Ця частина

процедури узагальнення показана на рис. 3.33 верхньою зеленою стрілкою.

Друга частина процедури узагальнення - це угрупування даних, яке пов'язано з полем Group By Columns. Для того, щоб сумісні процедури узагальнення були виконані коректно, колонка, по якій повинно бути виконано угрупування, має бути вказана в полі Select Columns. Процедура угрупування показана на рис. 3.33 нижньою зеленою стрілкою.

Сумісне виконання процедур агрегування та угрупування наочно показане ліворуч знизу рис. 3.33. Для спрощення з вихідної таблиці World показані тільки ті колонки, які беруть участь в запиті. Крім того показана колонка з назвами країн. Процедура агрегування сформована за допомогою агрегатної функції Sum(), яка сумує населення країн. Процедура угрупування групує сумування по континентах. Тобто, підсумовується чисельність населення кожної країни по континентах: окремо по Азії, окремо по Європі і т.д. Результуюча таблиця містить суми населення кожного континенту, так як це показано в центральній нижній частині рис. 3.33.

В MapInfo є шість агрегатних функцій (табл. 3.3), які беруть участь в процедурі узагальнення.

Таблиця 3.3 - Агрегатні функції SQL в MapInfo [21]

Функція	Характеристика
COUNT(*)	Обчислює кількість записів в групі
MIN(вираз)	Знаходить мінімальне значення в групі
MAX(вираз)	Знаходить максимальне значення в групі
SUM(вираз)	Обчислює суму записів в групі
AVG(вираз)	Обчислює середнє арифметичне всіх записів в групі
WtAvg(вираз, вираз)	Обчислює середнє вагове всіх записів в групі

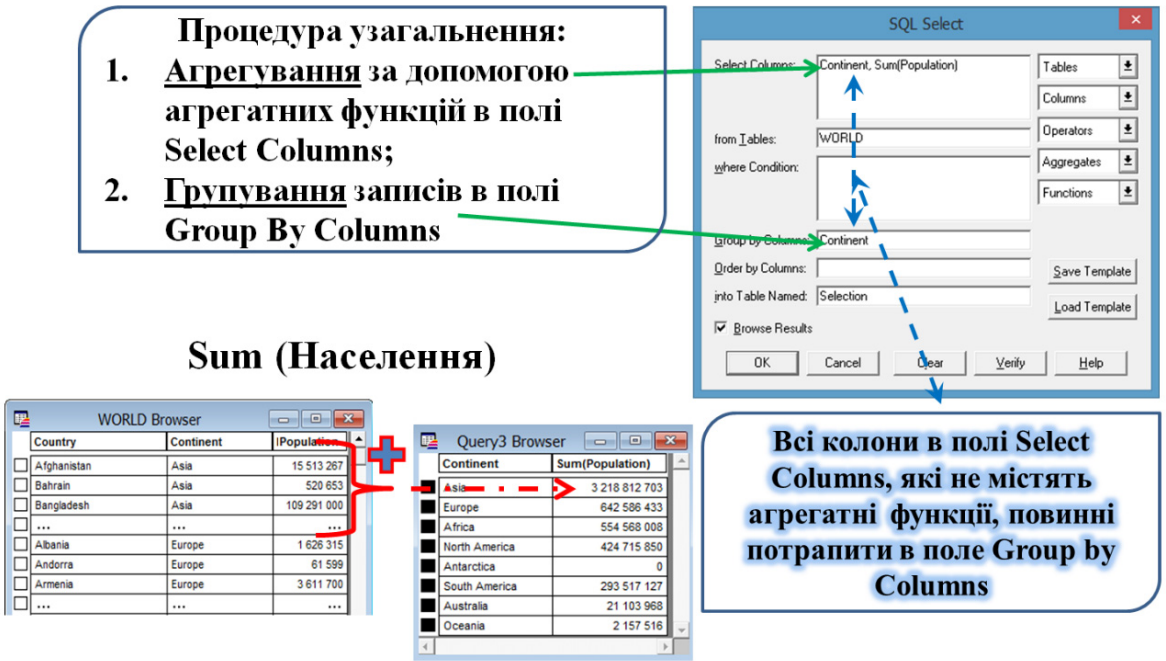
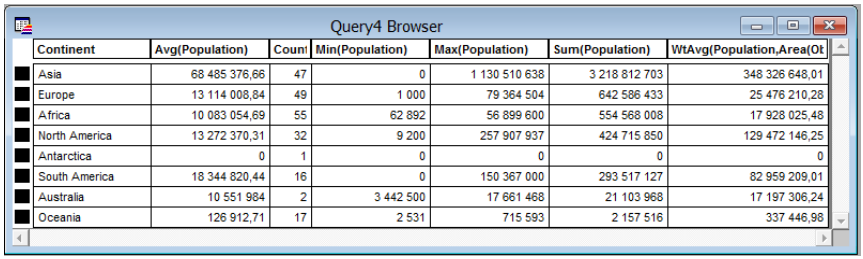


Рис 3.33 – Процедура узагальнення в MapInfo

Агрегатна функція Count(*) підраховує число записів в групі. В якості аргументу указується зірка, так як функція застосовується до всіх записів, а не до якогось окремого поля запису. Відокремлення одного запису від іншого виконується за допомогою колонки, яка показана в полі групування даних, Groupe By.

На рис. 3.34 показаний результат запиту до таблиці World з агрегатними функціями, згрупованими за колонкою Continent. В колонці Count кількість записів сумується для однакових значень в рядку колонки Continent. Це дозволило отримати кількість країн кожного континенту, для яких є інформація в таблиці World. Згідно результуючій таблиці, в Азії налічується 47 країн, в Європі – 49.


Query4 Browser

Continent	Avg(Population)	Count	Min(Population)	Max(Population)	Sum(Population)	WtAvg(Population,Area(Ot
Asia	68 485 376,66	47	0	1 130 510 638	3 218 812 703	348 326 648,01
Europe	13 114 008,84	49	1 000	79 364 504	642 586 433	25 476 210,28
Africa	10 083 054,69	55	62 892	56 899 600	554 568 008	17 928 025,48
North America	13 272 370,31	32	9 200	257 907 937	424 715 850	129 472 146,25
Antarctica	0	1	0	0	0	0
South America	18 344 820,44	16	0	150 367 000	293 517 127	82 959 209,01
Australia	10 551 984	2	3 442 500	17 661 468	21 103 968	17 197 306,24
Oceania	126 912,71	17	2 531	715 593	2 157 516	337 446,98

Рис. 3.34 – Результат використання агрегатних функцій в запиті до таблиці World з групуючою колонкою Continent

Агрегатна функція Sum (вираз) обчислює суму значень в виразі для всіх записів групи. В якості аргументу використовується вираз, який має чисельне значення. Якщо у виразі є колонки, вони також повинні мати чисельний тип. В табл. рис. 3.34 в колонці Sum(Population) представлений результат сумування кількості мешканців по кожному континенту. Таким чином згідно отриманому запиту, в Європі мешкає 642,5 млн., в Азії - 3 мільярди.

Агрегатні функції Min(вираз), Max(вираз) знаходять якнайменше та якнайбільше значення в виразі серед всіх записів групи. Вираз в обох функціях повинен бути

чисельного типу. В середній частині табл. рис. 3.34 показані результати запиту, в якому було визначені мінімальна та максимальна кількість мешканців для країн кожного континенту. Наприклад в Європі найменша кількість мешканців в країні дорівнює 1 тис., найбільша - 79 тис.

Агрегатна функція Avg(вираз) обчислює середнє значення в виразі для всіх записів групи. Наприклад необхідно обчислити середнє значення чисельності населення по країнах світу для кожного континенту Результат показаний в табл. на рис.3.34 дозволяє зробити висновок, що середня кількість мешканців в країнах Європи складає 13 млн., в Азії - 68 млн.

Агрегатна функція WtAvg(вираз, вираз) обчислює зважене середнє вагове значення виразу для всіх записів групи. На відміну від інших агрегатних функцій вона має не один а два аргументи. Перший аргумент - величини, для яких обчислюється середнє значення, другий аргумент – вага. На рис. 3.34, остання колонка в таблиці містить значення, отримані агрегатною функцією, яка обчислила середнє вагове значення кількості мешканців кожного регіону. В якості ваги використовувалася площа кожної країни.

Для того, щоб процедура запиту з узагальненням даних була виконана коректно, необхідно дотримуватися певного правила. Воно показано в нижній частині рис. 3.33 праворуч. Згідно цьому правилу, всі колони в полі Select Columns, які не засновані на агрегатних функціях, необхідно показати в полі групування колон, тобто в полі Group By. Ці колони MapInfo використовує для угрупування. В прикладі на рис. 3.33 це одна колонна Continent. Дія правила в діалоговому вікні показана за допомогою переривчастої стрілки блакитного кольору.

Якщо в полі вибору колонок немає ніяких інших колонок, крім колонок з агрегатними функціями, то групування буде виконано за всіма рядками. Тобто, по кожній агрегатній функції буде отримано одне число, так як це показано на рис. 3.35.

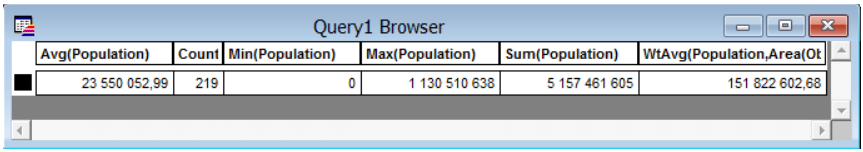


Рис. 3.35 – Результат використання агрегатних функцій в запиті до таблиці World без групуючих колонок

3.4.4. Виконання підзапитів (Subselect)

Subselect - це оператор Select, який знаходиться в середині іншого оператора Select. В MapInfo Subselect вставляється в атрибут Умови пошуку ключового слова WHERE. Наприклад, необхідно обрати країни, кількість мешканців яких не перевищує середню кількість за всіма країнами світу. Можна вирішити таку задачу простим запитом, навіть без використання складної команди SQL Select. Однак, для цього потрібно заздалегідь знати таку величину як середня кількість мешканців в країнах по всьому світу. Якщо така величина невідома, то для вирішення задачі без залучення Subselect потрібно зробити не один, а два запити. В першому запиті вирішується задача знаходження середньої кількості населення по всьому світу. І тільки в другій задачі, визначаються країни, кількість мешканців яких не перевищує середньої кількості населення по всьому світу.

На рис. 3.36 представлена процедура створення запиту з підзапитом на основі двох запитів, які будуються послідовно. Другий запит формується на основі даних, отриманих в першому запиті.

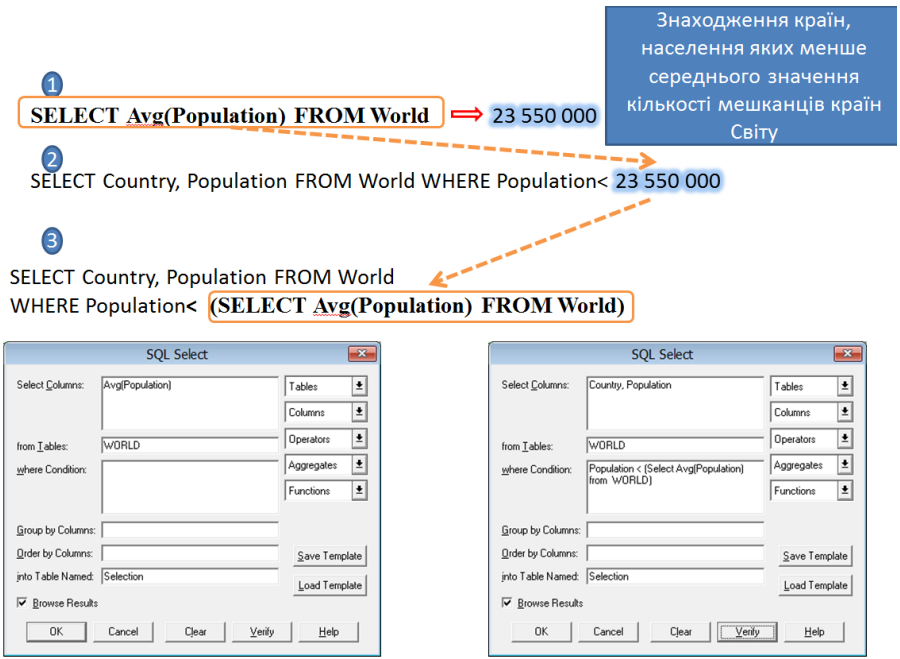


Рис 3.36 – Створення запиту з підзапитом на основі двох послідовних запитів для вирішення задачі знаходження країн, кількість мешканців яких менше середньої кількості мешканців країн в Світі

Перша задача, визначення середньої кількості мешканців в країні, вирішується за допомогою запиту під номером 1 на рис. 3.36, а саме:

SELECT Avg(Population) FROM World . (3.24)

Оскільки для колонок використовується одна агрегатна функція,Avg(), немає інших колонок, які б могли використовуватися для групування даних, то обчислення середнього арифметичного виконується по всіх рядках таблиці. Заповнення полів діалогового вікна SQL Select для цього запиту показано в нижній частині рис. 3.36 ліворуч. Результатом виконання запиту з цим виразом, буде одне число. Припустимо ми отримали таке число. Воно виявилося рівним 23 550 000. Далі слід сформувати інший запит, який повинен виглядати так, як показано на рис. 3.36 під номером 2, а саме:

SELECT Country, Population FROM World WHERE Population< 23 550 000 . (3.25)

В цьому випадку по запиту обираються дві колонки Country та Population з таблиці World, де Population менше константи 23 550 000.

Subselect дозволяє вирішити таку задачу одним запитом. Для цього треба об'єднати обидва запити таким чином. Замість числа 23 550 000 в другому запиті в круглих скобках треба поставити перший запит, який визначає це число. В результаті отримаємо оператор Select, в якому міститься ще один оператор Select. Результуючий запит показаний на рис. 3.36 під номером 3. Він отриманий в результаті об'єднання двох попередніх запитів.

В діалоговому вікні праворуч на рис. 3.36 показано заповнення полів запиту з підзапитом, який вирішує задачу знаходження країн, кількість мешканців яких менше середньої кількості мешканців по країнам всього світу.

Наступну задачу, яка розглядається на рис. 3.37, також неможливо вирішити одним запитом без використання

Subselect. Згідно умові цієї задачі, необхідно знайти країни, які межують з іншою країною, наприклад з Україною.

Якщо розділити цю задачу на дві окремі запити, то перший запит повинен сформувати таблицю, в якій буде міститься інформація саме про країну, для якої шукають сусідні країни, тобто таблицю з однією країною – Україною. Ця задача вирішується за допомогою запиту з номером 1 на рис. 3.37. Далі таблицю з цим запитом необхідно відкрити, так щоб вона отримала назву Query з якимось номером, наприклад з номером 1.

Наступний крок – формування запиту до обох таблиць, який показаний номером 2. В цьому запиті обирається колонка з назвами країн в таблиці World, та обираються дві таблиці перша World, в якій міститься інформація по всім країнам, друга таблиця Query1, в якій представлений тільки один графічний об’єкт, Україна. В умові ключового слова WHERE перевіряється чи має кожний об’єкт в таблиці World хоча б одну спільну точку з об’єктом таблиці Query1. Якщо має, то умова виконується і такий об’єкт, тобто країна попадає в список межуючих країн.

Як і в попередньому випадку, два запити можна об’єднати за допомогою Subselect в один. Об’єднаний запит показаний на рис. 3.37 номером 3. Цей запит сформований таким же чином, як і попередній. В другий запит замість об’єкта Query1, тобто Query1.obj вставляється перший запит, який визначає саме цей об’єкт. Оскільки в третьому запиті вже не беруть участі дві таблиці, тобто відсутня таблиця з назвою Query1, то назви колонок спрощені. Перший запит показаний в діалоговому вікні рис. 3.37 ліворуч, третій запит показаний в діалоговому вікні праворуч.

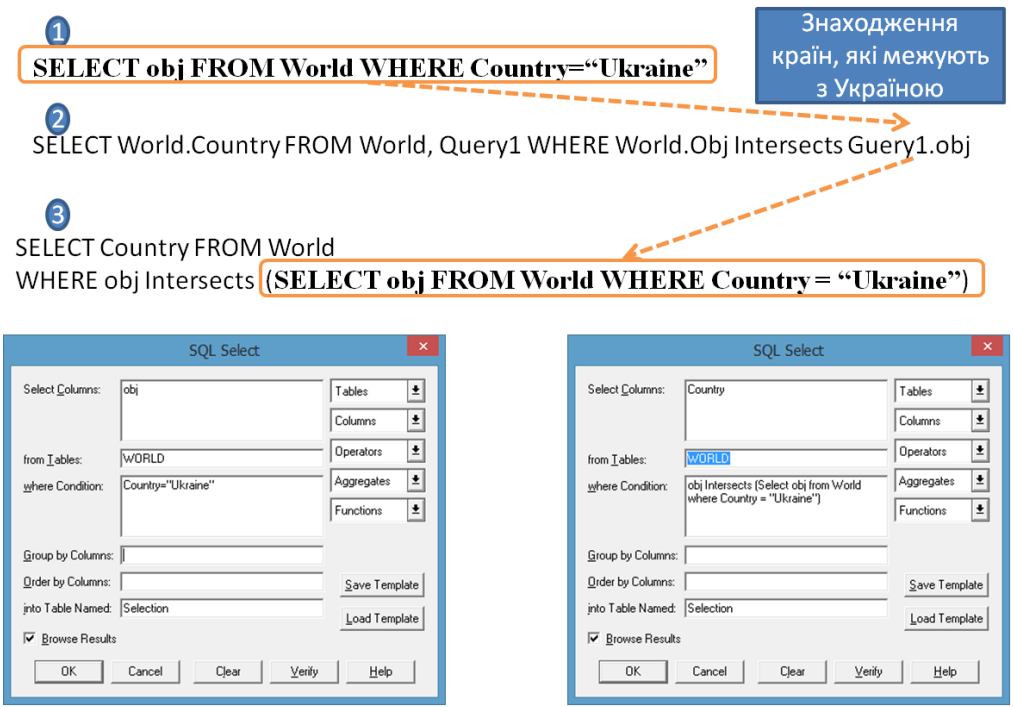


Рис 3.37 – Створення запиту з підзапитом на основі двох послідовних запитів для вирішення задачі знаходження країн, які межують з іншою країною

Не всі запити, можна використати як підзапити. Є досить жорсткі правила використання підзапитів. Subselect дозволяє виконати обчислення тільки з однією колонкою таблиці, або з декількома рядками. В деяких випадках можливо використання тільки одного рядка. Нижче сформульовані основні правила використання Subselect, а саме [21]:

1. **Subselect вставляється в круглі скобки;**
2. **Всі слова Subselect набираються з клавіатури;**
3. **В Subselect можна використовувати таблиці, які не вказані в списку таблиць основного запиту, однак вони попередньо повинні бути відкриті командою Open з меню File;**
4. **Якщо в Subselect використовується одно з ключових слів any, all, in, то аргументом ключового слова може бути тільки одна колонка;**
5. **Якщо в Subselect не використовується ключове слово any, all, in, то результат повинен бути одним рядком. Це може бути одна константа, або один графічний об'єкт;**
6. **Якщо в Subselect не використовується ключове слово any, all, in, то не можна використовувати групування в полі Group By.**

Не зважаючи на досить жорсткі обмеження для використання Subselect, це дуже потужний інструмент, який дозволяє сформувати один запит, замість декількох.

Питання для самоконтролю

1. Дайте визначення моделі представлення даних.
2. Яка модель представлення даних використовуються в обмінному файлі кадастрових даних XML?
3. Що таке реляційна модель даних?
4. Дайте визначення первинному ключу реляційної бази даних.
5. Сформулюйте основні вимоги до первинного ключа.
6. Яка різниця між складовим первинним ключем та складовим полем?
7. Які зв'язки виникають між таблицями в реляційній базі даних?
8. Для яких цілей створюється зв'язуюча таблиця?
9. Сформулюйте I, II, III нормальні форми.
10. Дайте визначення для складового та багатозначного полів.
11. Які групи операторів входять до складу SQL мови?
12. Який оператор вважається серцем SQL мови?
13. Назвіть варіанти створення SQL запиту в MapInfo.
14. Які елементи виразу в SQL запиті можуть розглядатися як підмет?
15. Сформулюйте правило використання числових констант в виразах SQL запитів.
16. В якому випадку вираз Table1.obj Intersects Table2.obj має значення True?
17. Де знаходиться центроїд об'єкта B, якщо вираз Table1.objA Contains Table2.objB має значення True?
18. Для яких геометричних фігур географічна функція ObjectLen() позитивна?
19. Яким чином можна з'єднати таблиці в MapInfo?
20. Що таке Subselect?

Задачі для самостійного виконання

Задачі на SQL запити для команди Select (Вибір) з використанням операторів

Задача 1. Вибрати з таблиці WORLD країни, населення яких не перевищує 10000.

Задача 2. Вибрати з таблиці WORLD країни, розташовані на континенті Europe.

Задача 3. В таблиці WORLD є дані по дитячому (Pop_0_14) та працездатному (Pop_15_64) населенню. Знайдіть частку дитячого населення по відношенню до працездатного населення.
Зауваження. Безпосередньо вирішити задачу за допомогою таблиці WORLD неможливо, оскільки в ній є рядок з відсутніми даними в колонах Pop_0_14, Pop_15_64, (в цих колонах записані числа, що дорівнюють 0). Ділення на 0 дасть помилку в обчисленнях. Для рішення задачі створіть таблицю, в якій будуть відсутні ці рядки. Далі запит сформуйте саме до створеної запитної таблиці.

Задача 4. Знайдіть в таблиці WORLD країни, назва яких починається на букву D
Примітка: для рішення задачі слід використати властивість букв аналогічну властивості чисел. Букви та слова можна порівнювати один з одним за допомогою операторів порівняння. Наприклад, “A”< “B”, “B” < “C”. Для складання умови, за якої всі назви країн будуть починатися на букву D, слід врахувати, що всі слова, які починаються на букву D не менше цієї букви, але менше наступної букви в алфавіті, тобто букви E. Для об’єднання умов слід використати оператор And.

Задача 5. Знайдіть в таблиці WORLD країни, назва яких починається на букву, відмінну від D
Примітка: Задача вирішується аналогічно задачі 4, з тією різницею, що всі оператори міняються на протилежні:
«>=» → «<», «<» → «>=», And → Or.

Задача 6. Вибрати з таблиці WORLD інформацію про три країни, столиці, яких мають назву: Brazzaville, Baku, Bogota.
Примітка: для рішення задачі слід використати ключове слово any.

Задача 7. Вибрати з таблиці WORLD інформацію про країни, де населення знаходиться в межах від 2000000 до 3000000.
Примітка: Для рішення задачі слід використати ключове словосполучення between… and.

Задача 8. Знайдіть в таблиці WORLD країни, для яких одночасно задовольняються такі критерії:

- населення менше 1 млн.;
- міське населення перевищує сільське;
- чоловіків більше ніж жінок.

Примітка: Для об'єднання умов слід використати оператор And.

Задача 9. Знайдіть в таблиці WORLD країни, для яких задовольняється один критерій:

- населення менше 1 млн.;

або обидва критерії:

- міське населення перевищує сільське;
- чоловіків більше ніж жінок.

Примітка: Для формування складної умови слід використати оператори And, Or та врахувати пріоритети операторів (табл. 3.1).

Задачі на SQL запити для команди Select (Вибір) з використанням операторів і функцій

Задача 10. Знайдіть в таблиці WORLD країни, назва яких складається не більш як з 4 букв.
Примітка: для рішення задачі слід використати функцію Len, яка має один аргумент, який може бути назвою колонки. Функція повертає кількість символів в виразі.

Задача 11. Оберіть з таблиці WORLD всі країни, назва яких починається з букви E.
Примітка: задача аналогічна задачі 4, однак для її рішення слід використати функцію Left$(рядок, номер символу в рядку). Замість аргументу рядок можна обрати назву колонки. Функція повертає значення у вигляді букви, яка стоїть під номером, визначеним в другому аргументі. Наприклад, якщо необхідно обрати першу букву в назві Ukraine, то буде повернута літера “U”.

Задача 12. Оберіть з таблиці WORLD всі країни, назва яких починається не з букви E.
Примітка: Як і для попередньої задачі слід використати функцію Left$(рядок, номер символу в рядку). Однак замість оператора «=» для порівняння значення функції з літерою E, слід навпаки обрати оператор «<>».

Задача 13. Оберіть з таблиці WORLD країни, щільність населення яких більше 1000 на 1 кв км.
Примітка: для рішення задачі слід використати функцію Area(obj, "sq km").

Задача 14. Знайдіть в таблиці WORLDCAP столиці країн, які знаходяться північніше від столиці України, Києва.
Примітка. Попередньо треба знайти широту Києва.

Для отримання запиту слід використати вираз з функцією CentroidY (Obj).

Задача 15. Знайдіть в таблиці WORLDCAP населені пункти, які знаходяться північно-західніше від міста Warsaw (Poland).
Примітка. Попередньо треба знайти географічні координати міста. Для отримання запиту слід використати вираз з функціями CentroidX(Obj) та CentroidY (Obj).

Задача 16. Знайдіть в таблиці WORLD країни, які знаходяться на північний захід від України.
Примітка. Попередньо треба знайти географічні координати центроїду країни. Для отримання запиту слід використати вираз з функціями CentroidX(Obj) та CentroidY (Obj).

Задача 17. Знайдіть в таблиці WORLDCAP столиці, які знаходяться в радіусі 1000 км від Києва
Примітка. Попередньо треба знайти географічні координати Києва.
Для отримання запиту слід використати вираз з функціями Distance(), CentroidX(Obj) та CentroidY (Obj). Наприклад,
Distance(CentroidX(obj),CentroidY(obj),30.5,50.5,"km")<= 1000

Задача 18. Знайдіть в таблиці WORLD країни, в назві яких третя буква є буквою A.
Примітка. Для отримання запиту слід використати вираз з функцією Mid$ (Вираз,N1,N2). Функція має три аргументи. Перший аргумент, текстовий вираз може бути назвою колонки, другий аргумент N1 – число, яке вказує номер позиції з якого обирається текстовий рядок, N2 - число, яке вказує кількість символів в текстовому рядку, які обираються функцією Mid$(). Наприклад, Mid$(Country,3,1)="A"

Задачі на SQL запити для команди SQL Select (SQL Вибір)

Задача 19. Виконайте задачу 1 за умовою, що результуюча таблиця повинна мати всього дві колонки - Country, Population.

Задача 20. Створіть таблицю з щільністю населення для кожної країни із таблиці WORLD. В результуючій таблиці повинні бути дві колони Country, Щільність, визначена в "sq km".

Задача 21. Виконайте задачу 3 так, щоб не потрібно було створювати додаткову таблицю.
Рішення:
Select Columns: Pop_0_14/Pop_15_64
from Tables: WORLD
where Condition: Pop_15_64 <> 0 and Pop_0_14/Pop_15_64 >= 1

Задача 22. В таблиці WORLDCAP знаходиться список столиць країн світу з кількістю мешканців. В таблиці WORLD є інформація про кількість мешканців всієї країни. Знайдіть відсоток мешканців, столиці по відношенню до всіх мешканців країни.
Підказка: Для вирішення задачі необхідно виключити з таблиці WORLD рядки з даними, рівними 0. Необхідно створити таблицю QueryN, в якій ці рядки відсутні. Для створення запиту необхідно обрати дві таблиці (QueryN та WORLDCAP) та зв’язати їх одним з двох можливих методів (розд. 3.4.2).

Задача 23. Сформуйте результуючу таблицю на основі такої схеми запиту:
Select * from Us_custg,States,City_125

where States.state = City_125.state and States.state = Us_custg.state and Us_custg.order_amt > 10000

Задача 24. За допомогою таблиці World покажіть країни, що межують з країною Poland.
Примітка. Для вирішення задачі без використання Subselect, слід виконати такі дії:

1. Необхідно створити запитну таблицю з назвою обраної країни. Таблицю можна створити за допомогою команди Select з використання умови:
   Country = "Poland"
2. Збережіть створену таблицю за допомогою команди Save Copy As з меню File. Назвіть таблицю іменем країни (Poland). Відкрийте цю таблицю вибравши команду Open з меню File.
3. Виберіть команду SQL Select. Заповніть поля так як показано нижче.

Select Columns: WORLD.Country
from Tables: WORLD, Poland
where Condition: WORLD.Obj Intersects Poland.Obj And WORLD.Country <> "Poland"
Вирішення задачі за допомогою Subselect представлено на рис. 3.37.

Задача 25. Знайдіть номери автострад (таблиця US_HIWAY), які проходять через штат Alabama (таблиця STATES). Обидві таблиці знаходяться в папці USA.
Рішення:
Select Columns: US_HIWAY.Highway
from Tables: STATES, US_HIWAY
where Condition: STATES.Obj Intersects US_HIWAY.Obj And STATES.State_Name = “Alabama”

Задача 26. Знайдіть населені пункти (таблиця CITY_125), які знаходяться в штаті Alabama (таблиця STATES)

Рішення:
Select Columns: STATES.State_Name, CITY_125.City
from Tables: STATES, CITY_125
where Condition: STATES.State = CITY_125.State And
STATES.State_Name="Alabama"

Задача 27. Знайдіть штати в таблиці STATES, де в 1990 році не менше як третина населення мешкала в столиці цього штату (таблиця STATECAP).
Рішення:
Select Columns: STATES.State_Name, STATES.Pop_1990,
STATECAP.Pop_1990
from Tables: STATES, STATECAP
where Condition: STATES.State = STATECAP.State And
STATECAP.Pop_1990 / STATES.Pop_1990 >= 0.3

Задача 28. Використовуючи таблицю US_CUSTG (група USA), показати загальну і середню кількість продаж (Order_amt) в штатах і кількість компаній в кожному штаті. Примітка групування необхідно виконати по колонці State.
Рішення:
Select Columns: State, Sum(Order_amt), Avg(Order_amt), Count(*)
from Tables: US_CUSTG
Group By Columns: State

Задача 29. Використовуючи таблицю CITY_125, показати загальну кількість населених пунктів і загальну кількість мешканців в цих пунктах по штатах.
Примітка: групування необхідно виконати по колонці State
Рішення:
Select Columns: State, Count(*), Sum(Tot_pop)
from Tables: CITY_125
Group By Columns: State

Задача 30. Використовуючи таблицю WORLD, показати кількість країн, кількість мешканців та середнє значення кількості мешканців по країнах кожного континенту. Примітка: групування необхідно виконати по колонці Continent.

Задача 31. Знайти штати в таблиці STATES, кількість мешканців в яких більше середньої кількості мешканців по кожному штату в країні.
Підказка: задача вирішується аналогічно задачі, представленій на рис. 3.36.

Задача 32. Використовуючи таблицю CITY_125, оберіть всі міста в штатах, загальна кількість мешканців в яких менше 2 000 000 чоловік.
Рішення:
Select Columns: *
from Tables: CITY_125
where Condition: obj within any(select obj from States where POP_1990 < 2000000)

Літературні джерела

1. 11 типов современных баз данных: краткие описания, схемы и примеры БД // Proglib: Библиотека программиста. URL: https://proglib.io/p/11-tipov-sovremennyh-baz-dannyh-kratkie-opisaniya-shemy-i-primery-bd-2020-01-07 (дата звернення 03.01.2022).
2. В чем разница между "sequel" и "SQL"? CodeRoad. URL: https://coderoad.ru/about.html. (дата звернення 05.01.2022).
3. Грабер М. Введение в SQL. Пер. с англ. М.: ЛОРИ, 1996, 382 с.
4. Гутнов А., Глызычев В. Мир архитектуры. М.: Молодая гвардия, 1990, с. 352. URL: http://architecture.artyx.ru/books/item/f00/s00/z0000003/index.shtml.
5. Дейт К. Дж. SQL и реляционная теория. Пер. с англ. СПб.: Символ-Плюс, 2010, 480 с., ил.
6. Зацерковний В.І., Бурачек В.Г., Железняк О.О., Терещенко А.О. Геоінформаційні системи і бази даних: монографія. Кн. 1. Ніжин: НДУ ім. Гоголя, 2014, 492 с.
7. Маркин А.В. Построение запросов и программирование на SQL: учеб. пособие. Рязань: РГРТУ, 2008, 312 с.
8. Молинаро Э. SQL. Сборник рецептов: Пер. с англ. СПб: Символ-Плюс, 2009, 672 с., ил. ISBN: 978-5-93286-125-7.
9. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. Ужгород: Електронне видання, 2018, 118 с. URL: https://dspace.uzhnu.edu.ua/jspui/.
10. Про Державний земельний кадастр: Закон України від 07.07.2011 № 3613-VI. Відомості Верховної Ради

України (ВВР), 2012, № 8, ст. 61. URL: https://zakon.rada.gov.ua/laws/show/3613-17#Text.
11. Райордан Р. Основы реляционных баз данных/Пер, с англ. М.: Издательско-торговый дом «Русская Редакция», 2001, 384 с.: ил. ISBN 5-7502-0150-3.
12. Сетевое планирование и управление: Лекция 2. ИНТУИТ: Национальный открытый университет. URL: https://intuit.ru/studies/professional_retraining/964/courses/352/lecture/8389?page=3#image.2.9 (дата звернення 03.01.2022).
13. Ульман Дж., Ундом Дж. Введение в системы баз данных. Пер. с англ. Лори, 2006, 224 с., ил. ISBN: 5-85582-069-6. URL: http://www.db.stanford.edu/~ullman/fcdb.html.
14. Хернандес М. Дж., Вьескас Дж. Л. SQL – запросы для простых смертных: практ. руководство по манипулированию данными в SQL. Пер. с англ. Лори, 2006, 224 с., ил. ISBN: 5-85582-069-6. URL: http:www.natahaus.ru.
15. Что такое DDL, DML, DCL и TCL в языке SQL: Заметки IT специалиста. URL: https://info-comp.ru/what-is-ddl-dml-dcl-tcl (дата звернення 05.01.2022).
16. Codd E.F. A Relational Model of Data for Large Shared Banks //*Communications of the ACM*. 1970. Vol. 13, № 6. C. 377-387.
17. Codd's 12 rules: From Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Codd%27s_12_rules.
18. Database normalization: From Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Database_normalization. (дата звернення 03.01.2022).

19. ISO/IEC 9075 standard: "Information technology - Database languages - SQL". URL: https://docs.oracle.com/cd/B28359_01/server.111/b28286/intro002.htm#SQLRF50928 . (дата звернення 03.01.2022).
20. Kent W., A Simple Guide to Five Normal Forms in Relational Database Theory // *Communications of the ACM.* 1983. Vol. 26, № 2. С. 120-125.
21. MapInfo Professional: Справочник. Пер. с англ. В. Журавлев, А. Колотов, В. Николаев. MapInfo Corporation, Troy, New York, 2002, 658 с. URL: http://www.mapinfo.com.

ДОДАТКИ

ДОДАТОК А

Коротка характеристика операторів мови SQL [15]

Група	Оператор	Використовується:
Оператори визначення даних (D D L)	Create	для створення об’єктів бази даних (самої бази даних та її частин)
	Alter	для додавання, видалення та модифікації колонки в таблиці, яка вже є в наявності в базі даних
	Drop	для знищення таблиці, індексів та навіть самої бази даних
Оператори маніпулювання даними (DML)	Select	для вибору інформації з бази даних
	Delete	для видалення даних з таблиці
	Update	для оновлення даних в таблиці
	Insert	для додавання даних у таблицю
Оператори контролювання доступу до даних (DCL)	Grant	для встановлення дозволу користуватися базою даних
	Revoke	для видалення вже існуючого дозволу доступу до бази даних
	Deny	для відміни дозволу користуватися базою даних
Оператори управління транзакціями (TCL)	Commit	для виконання процедури транзакції
	Rollback	для відміни всіх змін, які були створені під час транзакції
	SavePoint	для встановлення точки збереження всередині транзакції

ДОДАТОК Б

Одиниці вимірювання довжини в MapInfo [21]

Одиниця вимірювання	Назва
mi	мілі
km	кілометри
in	дюйми
ft	фути
survey ft	топографічні фути
yd	ярди
mm	міліметри
cm	сантиметри
m	метри
nmi	морські мілі (1 морська міля дорівнює 1852 метрам)

ДОДАТОК В

Одиниці вимірювання площі в MapInfo [21]

Одиниця вимірювання	Назва
sq mi	квадратні мілі
sq km	квадратні кілометрі
sq in	квадратні дюйми
sq ft	квадратні фути
sq survey ft	квадратні топографічні фути
sq yd	квадратні ярди
sq mm	квадратні міліметри
sq cm	квадратні сантиметри
sq m	квадратні метри
acre	акри
hectare	гектари

Навчальне видання

НОВІКОВА Олена Миколаївна
ПАЛАМАР Альона Юріївна,
МАЗИКІНА Ольга Борисівна
СИДОРЕНКО Віктор Дмитрович

SQL запити для ГІС

Навчальний посібник

Підписано до друку 04.02.2022.
Формат 60x84/16. Ум. др. арк. – 7,58. Обл.-вид. арк. – 4,95.
Тираж – 50 пр.

Видавець Р. А. Козлов
вул. Рокоссовського, 5/3, м. Кривий Ріг, 50027
097-192-20-77
Свідоцтво суб'єкта видавничої справи ДК № 4514 від 01.04.2013 р.

Друкарня С. Г. Щербенка «Літерія»
вул. Рокоссовського, 5/3, м. Кривий Ріг, 50027
097-192-20-77
Свідоцтво суб'єкта видавничої справи ДК № 4561 від 13.06.2013 р.