Лабораторна робота 2.
ЗАПИТИ З ОБ'ЄДНАННЯМ ТАБЛИЦЬ. ЗАПИТИ JOIN-S. ВКЛАДЕНІ ЗАПИТИ.

Підзапити

До цього часу ми отримували дані з бази даних за допомогою простих запитів та одного оператора **SELECT**. Однак, все-таки, частіше нам потрібно буде вибирати дані, що відповідають багатьом умовам, і тут не обійтися без розширених запитів. Для цього в **SQL** існують підзапити або вкладені запити, коли один оператор **SELECT** вкладається у інший.

1. Фільтрування за допомогою підзапитів

Таблиці баз даних, які використовуються в СУБД являються реляційними таблицями, тобто всі таблиці можна пов'язати між собою по спільним полям. Припустимо в нас зберігаються дані в двох різних таблицях і нам потрібно вибрати дані в одній із них, в залежності від того, які дані в іншій. Для цього створимо ще одну таблицю в нашій базі даних. Це буде, наприклад, таблиця **Sellers** з інформацією про постачальників:

ID	Address	City	Seller_name	Country
1	500 Park Street	Montreal	Michelle Green	Canada
2	1000 5th Avenue	San Francisco	Kim Howard	USA
3	42 Galaxy Road	New York	John Smith	USA
4	123 Main Street	Toronto	Denise L. Stephens	Canada

Тепер ми маємо дві таблиці - **Sumproduct** та **Sellers**, які мають однакове поле **City**. Припустимо, нам потрібно порахувати скільки товарів було продано лише в Канаді. Зробити це нам допоможуть підзапити. Отже, спочатку напишемо запит для вибірки міст, які знаходяться в Канаді:

SELECT City FROM Sellers WHERE Country = *'Canada'*

City
Montreal
Toronto

Тепер передамо ці дані в наступний запит, який вибиратиме дані з таблиці **Sumproduct**:

SELECT SUM(Quantity) AS Qty_Canada FROM Sumproduct WHERE City IN (*'Montreal','Toronto'*)

Qty_Canada
10222

Також ми можемо об'єднати ці два запити в один. Таким чином, один запит, який виводить дані буде головним, а другий запит, який передає вхідні дані, буде допоміжним (підзапитом). Для вкладення підзапиту використаємо конструкцію **WHERE** ... **IN** (...), про яку говорилось в розділі **Розширене фільтрування**:

SELECT SUM(Quantity) AS Qty_Canada FROM Sumproduct WHERE City IN (SELECT City FROM Sellers WHERECountry = *'Canada'*)

Qty_Canada
10222

Бачимо, що ми отримали аналогічні дані, як і за допомогою двох окремих запитів. Таким же чином, ми можемо збільшувати глибину вкладеності запитів, вкладаючи підзапити скільки завгодно разів.

2. Використання підзапитів в якості розрахункових полів

Ми також можемо використовувати підзапити в ролі розрахункових полів. Відобразимо, наприклад, кількість реалізованої продукції по кожному продавцю за допомогою наступного запиту:

SELECT Seller_name, (SELECT SUM(Quantity) FROM Sumproduct WHERE Sellers.City = Sumproduct.City) AS Qty FROM Sellers

Seller_name	Qty
Michelle Green	5129
Kim Howard	5347
John Smith	3897
Denise L. Stephens	5093

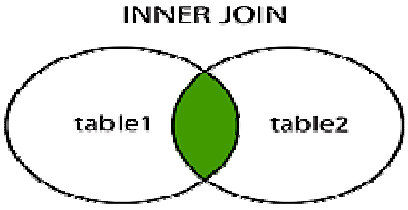
Перший оператор **SELECT** відображає два стовпця - **Seller_name** та **Qty**. Поле **Qty** являється розрахунковим, воно формується в результаті виконання підзапиту, який взятий в круглі дужки. Цей підзапит виконується по одному разу для кожного запису в полі **Seller_name** та взагальному буде виконаний чотири рази, оскільки вибрано імена чотирьох продавців.

Також, в підзапиті, речення **WHERE** виконує функцію поєднання, оскільки за допомогою **WHERE** ми з'єднали дві таблиці по полю **City**, використавши повні назви стовпців (*Таблиця.Поле*).

Поєднання таблиць (INNER JOIN)

Найбільш потужною особливістю мови SQL є можливіть поєднувати різні таблиці в оперативній пам'яті СУБД під час виконання запитів. Об'єднання дуже часто використовуються для аналізу даних. Як правило, дані знаходяться в різних таблицях, що дозволяє їх більш ефективно зберігати (оскільки інформація не дублюється), спрощує обробку даних та дозволяє масштабувати базу даних (можливо додавати нові таблиці з додатковою інформацією).

PS в **INNER JOIN** теж саме, що і **JOIN**.



1. Створення об'єднання таблиць

Об'єднання таблиць дуже проста процедура. Потрібно вказати всі таблиці, які будуть включені в об'єднання та "пояснити" СУБД, як вони будуть пов'язані між собою. Поєднання робиться за допомогою слова **WHERE**, наприклад:

SELECT DISTINCT Seller_name, Product FROM Sellers, Sumproduct WHERE Sellers.City = Sumproduct.City

Seller_name	Product
Denise L. Stephens	Bikes
Denise L. Stephens	Skates
Denise L. Stephens	Skis Long
Denise L. Stephens	Skis Short
Denise L. Stephens	Snow Board
John Smith	Bikes
John Smith	Skates
John Smith	Skis Long
John Smith	Skis Short
John Smith	Snow Board
Kim Howard	Bikes
Kim Howard	Skates
Kim Howard	Skis Long
Kim Howard	Skis Short
Kim Howard	Snow Board
Michelle Green	Bikes
Michelle Green	Skates
Michelle Green	Skis Long
Michelle Green	Skis Short
Michelle Green	Snow Board

Поєднавши дві таблиці, ми змогли побачити які товари реалізує кожен продавець. Розглянемо код запиту детальніше, оскільки він трохи відрізняється від звичайного запиту. Оператор **SELECT** починається з вказанням стовпців, які ми хочемо вивести, проте ці поля знаходяться в різних таблицях, речення **FROM** містить дві таблиці, які ми хочемо поєднати в операторі **SELECT**, таблиці поєднуються за допомогою слова **WHERE**, яке вказує стовпці для об'єднання. Обов'язково потрібно вказувати повну назву поля (*Таблиця.Поле*), оскільки поле **City** є в обидвох таблицях.

2. Внутрішнє об'єднання

В попередньому прикладі для об'єднання таблиць ми використали слово **WHERE**, яке здійснює перевірку на основі еквівалентності двох таблиць. Об'єднання такого типу називається також "*внутрішнім об'єднанням*". Існує також і інший спосіб об'єднання таблиць, який явно вказує на тип об'єднання. Розглянемо наступний приклад:

SELECT DISTINCT Seller_name, Product FROM Sellers INNER JOIN Sumproduct ON Sellers.City = Sumproduct.City

Seller_name	Product
Denise L. Stephens	Bikes
Denise L. Stephens	Skates
Denise L. Stephens	Skis Long
Denise L. Stephens	Skis Short
Denise L. Stephens	Snow Board
John Smith	Bikes
John Smith	Skates
John Smith	Skis Long
John Smith	Skis Short
John Smith	Snow Board
Kim Howard	Bikes
Kim Howard	Skates
Kim Howard	Skis Long
Kim Howard	Skis Short
Kim Howard	Snow Board
Michelle Green	Bikes
Michelle Green	Skates
Michelle Green	Skis Long
Michelle Green	Skis Short
Michelle Green	Snow Board

В цьому запиті замість **WHERE** ми використали конструкцію **INNER JOIN ... ON ...**, яка дала аналогічний результат. Незважаючи на те, що об'єднання з реченням **WHERE** є коротшим, все таки краще використовувати команду **INNER JOIN**, оскільки вона є більш гнучкою.

Символи підстановки та регулярні вирази (LIKE)

Часто, для фільтрації даних, нам потрібно буде здійснити вибірку не по точному співвпадінні умови, а по наближеному значенню. Тобто коли, наприклад, ми шукаємо товар, назва якого відповідає певному шаблону (**регулярному виразу**) або містить певні символи чи слова. Для таких цілей в**SQL** існує оператор **LIKE**, котрий шукає наближені значення. Для конструювання такого шаблону використовуються **метасимволи** (спеціальні символи, для пошуку частини значення), а саме: "*знак відсотка*" (**%**) або*зірочка* (*****), "*символ підкреслення*" (**_**) або "*знак питання*" (**?**), "*квадратні дужки*" (**[ ]**).

1. Метасимвол знак відсотка (%) або зірочка (*)

Давайте з нашої таблиці, наприклад, відберемо записи, які стосуються лише товарів, які містять у своїй назві слово ***Skis*** *(лижі)*. Для цього складемо відповідний шаблон:

SELECT * FROM Sumproduct WHERE Product LIKE *'%Skis%'*

ID	Month	Product	City	Quantity	Amount
10	April	Skis Long	Montreal	854	$209 230,00
11	April	Skis Long	Toronto	25	$6 125,00
12	April	Skis Long	San Francisco	663	$162 435,00
13	April	Skis Long	New York	21	$5 145,00
14	April	Skis Short	Montreal	21	$4 389,00
15	April	Skis Short	Toronto	4	$836,00
16	April	Skis Short	San Francisco	522	$109 098,00
17	April	Skis Short	New York	136	$28 424,00
30	February	Skis Long	Montreal	522	$127 890,00
31	February	Skis Long	Toronto	125	$30 625,00
32	February	Skis Long	San Francisco	663	$162 435,00
33	February	Skis Long	New York	21	$5 145,00

Як бачимо, **СУБД** відібрала лише ті записи, де в колонці **Product** були товари, які містять слово ***Skis***.

2. Метасимвол знак підкреслення (_) або знак питання (?)

Знак підкреслення або знак питання застосовується для того, щоб замінити один символ у слові. Давайте в слові***Bikes*** замінимо всі голосні літери на "знак питання" (**?**) і подивимось на результат:

SELECT * FROM Sumproduct WHERE Product LIKE *'B_k_s'*

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
22	February	Bikes	Montreal	663	$248 625,00
23	February	Bikes	San Francisco	21	$7 875,00
24	February	Bikes	San Francisco	54	$20 250,00
25	February	Bikes	New York	658	$246 750,00
42	January	Bikes	Montreal	75	$28 125,00

3. Метасимвол квадратні дужки ([ ])

Метасимвол "квадратні дужки" (**[ ]**) використовується для одночасного вказання набору символів, по яким потрібно здійснити пошук.

SELECT * FROM Sumproduct WHERE City LIKE '[TN]*'

ID	Month	Product	City	Quantity	Amount
5	April	Bikes	New York	25	$9 375,00
7	April	Skates	Toronto	854	$84 546,00
9	April	Skates	New York	663	$65 637,00
11	April	Skis Long	Toronto	25	$6 125,00
13	April	Skis Long	New York	21	$5 145,00
15	April	Skis Short	Toronto	4	$836,00
17	April	Skis Short	New York	136	$28 424,00
19	April	Snow Board	Toronto	522	$160 776,00
21	April	Snow Board	New York	663	$204 204,00

В прикладі вище, ми відібрали записи, де в полі **City** назви міст починаються з букви **T** або **N**. Також, в даному випадку, ми можемо використати ще один метасимвол, який виконує зворотню дію.

Добавимо у наш регулярний вираз знак оклику (**!**), що означатиме "не дорівнює" (для **СУБД Access**) або знак степені (^) (для інших **СУБД**).

SELECT * FROM Sumproduct WHERE City LIKE '[^TN]*'

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
6	April	Skates	Montreal	56	$5 544,00
8	April	Skates	San Francisco	25	$2 475,00
10	April	Skis Long	Montreal	854	$209 230,00
12	April	Skis Long	San Francisco	663	$162 435,00
14	April	Skis Short	Montreal	21	$4 389,00
16	April	Skis Short	San Francisco	522	$109 098,00

Тобто, останній створений нами запит читатиметься як: вибрати усі колонки з таблиці **Sumproduct**, та лише ті записи, де в полі **City** назви міст не починаються на букви **T** або **N**. Додатково зазначимо, що набір букв в метасимволі "квадратні дужки" відповідає лише за одну позицію в тексті.

Ми можемо отримати аналогічний результат, якщо скористатися вже відомим нам оператором **NOT**, проте зі знаком оклику (**!**) запис буде коротшим.

Розрахункові (обчислювальні) поля

Для чого потрібно використовувати розрахункові поля? Як правило, інформація в БД представлена в розрізі окремих фрагментів, оскільки так легше структуризувати дані та оперувати ними. Проте нам часто буде потрібно використовувати не окремі частини даних, а вже поєднану та оброблену інформацію. Наприклад, часто необхідно поєднувати ім'я та прізвище клієнтів, поєднувати елементи адрес, які знаходяться в різних стовпцях таблиці, обробляти текст та окремі слова, букви та символи, підсумовувати загальну вартість покупки, відображати статистику по інформації, яка знаходиться в БД. Дані, зазвичай, зберігаються окремими "кусками", що вимагає їх додаткового опрацювання на стороні клієнтської програми. Проте є можливість отримувати вже оброблену інформацію за допомогою СУБД. Саме в цьому випадку допомагають розрахункові поля. Вони автоматично створюються при виконанні запиту і мають вигляд та властивості звичайних стовпців, які є вже наявні в таблиці. Єдина відмінність полягає в тому, що фізично розрахункових полів немає, тому вони не займають додаткового місця в БД, а тимчасово існують в "оперативній пам'яті" СУБД. Перевагою виконання операцій на стороні СУБД являється швидкість опрацювання даних.

1. Виконання математичних операцій

Одним із способів використання розрахункових полів є виконання математичних операцій над вибраними даними. Давайте на прикладі розглянемо як це відбувається, використавши знову нашу таблицю **Sumproduct**. Припустимо, на потрібно вирахувати середню ціну придбання кожного товару. Для цього потрібно переділити колонку **Amount** (сума) на **Quantity** (кількість):

SELECT DISTINCT Product, Amount/Quantity FROM Sumproduct

Product	Expr1001
Bikes	375
Skates	99
Skis Long	245
Skis Short	209
Snow Board	308

Як бачимо, СУБД відібрала всі найменування товарів та відобразила їх середню вартість в окремому стовпці, який був створений під час виконання запиту. Також можна помітити, що ми використали додатковий оператор **DISTINCT**, який нам потрібен для відображення унікальних назв товарів (без нього ми б отримали дублювання записів).

2. Використання псевдонімів

В попередньому прикладі ми розраховували середню вартісь покупки кожного товару та відобразили значення в розрахунковому стовпці. Проте надалі, нам буде незручно звертатися до цього поля, оскільки його назва є неінформативною для нас (СУБД дала назву полю - **Expr1001**). Проте ми

можемо назвати поле самостійно, наперед вказавши його назву в запиті, тобто дати псевдонім. Давайте перепишемо попередній приклад та вкажемо псевдонім для розрахункового поля:

SELECT DISTINCT Product, Amount/Quantity AS AvgPrice FROM Sumproduct

Product	AvgPrice
Bikes	375
Skates	99
Skis Long	245
Skis Short	209
Snow Board	308

Бачимо, наше розрахункове поле отримало власну назву **AvgPrice**. Для цього ми використали оператор **AS**, після якого вказали необхідну нам назву. Варто зазначити, що в SQL підтримуються лише основні математичні операції: додавання(+), віднімання(-), множення(*), ділення(/). Також для зміни черговості виконання операції можна використовувати круглі дужки.

Часто псевдоніми використовують не тільки щоби називати розрахункові поля, але і для перейменування діючих. Це може бути необхідним, якщо діюче поле має довгу назву або назва не є достатньо інформативною.

3. З'єднання полів (конкатенація)

Крім математичних операцій ми також можемо поєднувати текст та виводити його в окремому полі. Давайте розглянемо, яким чином можна здійснити склеювання (конкатенацію) тексту. Маємо такий приклад:

SELECT Month + ' ' + Product AS NewField, Quantity FROM Sumproduct

NewField	Quantity
April Bikes	12
April Bikes	56
April Bikes	854
April Bikes	25
April Skates	56
April Skates	854
April Skates	25
April Skates	663
April Skis Long	854
April Skis Long	25
April Skis Long	663
April Skis Long	21

В цьому прикладі ми з'єднали значення в двох стовпцях та вивели результат в нове поле **NewField**.

Функції обробки даних

Як і в більшості мов програмування, в SQL існують функції для обробки даних. Варто зазначити, що на відміну від SQL-операторів, функції не є стандартизованими для усіх видів СУБД, тобто для виконання одних і тих самих операції над даними, різні СУБД мають свої власні назви функцій. Це означає, що код запиту написаний в одній СУБД може не працювати в іншій, і це потрібно враховувати в подальшому. Найбільше це стосується функцій для обробки текстових значень, перетворення типів даних та маніпуляцій над датами.

В більшості СУБД підтримується стандартний набір типів функцій, а саме:

- Текстові функції, що використовуються для обробки тексту (виокремлення частини символів з тексту, визначення довжини тексту, переведення символів в верхній або нижній регістр...);
- Числові функції. Використовуються для виконання математичних операцій над числовими значеннями;
- Функції дати та часу (здійснюються маніпулювання датою та часом, розраховуються період між датами, перевіряються дати на коректність тощо);
- Статистичні функції (для обчислення максимальних/мінімальних значень, середніх значень, підрахунок кількості та суми...);
- Системні функції (надають різного роду службову інформацію про СУБД, користувача та ін.).

1. Функції SQL для обробки тексту

Реалізація SQL в має наступні функції для обробки тексту:

Знак операції	Значення
LEFT()	Відбирає символи в тексті зліва
RIGHT()	Відбирає символи в тексті справа
MID()	Відбирає символи з середини тексту
UCase()	Переводить символи в верхній регістр
LCase()	Переводить символи в нижній регістр
LTrim()	Видаляє всі порожні символи зліва від тексту
RTrim()	Видаляє всі порожні символи справа від тексту
Trim()	Видаляє всі порожні символи з обох боків тексту

Переведемо назви товарів у верхній регістр за допомогою функції **UCase()**:

SELECT Product, UCase(Product) AS Product_UCase FROM Sumproduct

Product	Product_UCase
Bikes	BIKES
Bikes	BIKES
Bikes	BIKES
Bikes	BIKES
Skates	SKATES
Skates	SKATES
Skates	SKATES
Skates	SKATES
Skis Long	SKIS LONG
Skis Long	SKIS LONG

Виділимо перші три символи у тексті за допомогою функції **LEFT()**:

SELECT Product, LEFT(Product, 3) AS Product_LEFT FROM Sumproduct

Product	Product_LEFT
Bikes	Bik
Bikes	Bik
Bikes	Bik
Bikes	Bik
Skates	Ska
Skates	Ska
Skates	Ska
Skates	Ska
Skis Long	Ski
Skis Long	Ski

2. Функції SQL для обробки чисел

Функції обробки чисел призначені для виконання математичних операцій над числовими даними. Ці функції призначені для алгебраїчних та геометричних обчислень, тому вони використовуються значно рідше функцій обробки дати та часу. Проте числові функції є найбільш стандартизованими для усіх версій SQL. Давайте поглянемо на перелік числових функцій:

Знак операції	Значення
SQR()	Повертає корінь квадратний вказаного числа
ABS()	Повертає абсолютне значення числа
EXP()	Повертає експоненту вказаного числа
SIN()	Повертає синус вказаного кута
COS()	Повертає косинус вказаного кута
TAN()	Повертає тангенс вказаного кута

Ми навели лише кілька основних функцій, проте ви завжди можете звернутися до документації вашої СУБД, щоб переглянути повний перелік функцій, що підтримуються з їх детальним описом.

Наприклад, напишемо запит для отримання кореня квадратного для чисел в стовпці **Amount** за допомогою функції **SQR()**:

SELECT Amount, SQR(Amount) AS Amount_SQR FROM Sumproduct

Amount	Amount_SQR
$4 500,00	67,08203932499
$21 000,00	144,9137674619
$320 250,00	565,9063526768
$9 375,00	96,82458365519
$5 544,00	74,45804187595
$84 546,00	290,7679487151
$2 475,00	49,74937185533
$65 637,00	256,1971896801
$209 230,00	457,4166590757
$6 125,00	78,26237921249

3. Функції SQL для обробки дати та часу

Функції маніпулювання датою та часом являються одними з найважливіших і часто використовуваних функцій SQL. В базах даних значення дат та часу зберігаються в спеціальному форматі, тому їх неможливо використовувати напряму без додаткової обробки. Кожна СУБД має свій набір функцій для обробки дат, що, нажаль, не дозволяє переносити їх на інші платформи та реалізації SQL.

Знак операції	Значення
DatePart()	Повертає частину дати: рік, квартал, місяць, тиждень, день, годину, хвилини, секунди
Year(), Month()	Повертає рік та місяць відповідно
Hour(), Minute(), Second()	Повертає годину, хвилини та секунди вказаної дати
WeekdayName()	Повертає назву дня тижня

Подивимось на прикладі як працює функція **DatePart()**:

SELECT Date1, DatePart("m", Date1) AS Month1 FROM Sumproduct

Date1	Month1
11.02.2013	2

Функція **DatePart()** має додатковий параметр, який нам дозволяє відобразити необхідну частину дати. В прикладі ми використали значення параметра **"m"**, який відображає номер місяця (таким же чином ми можемо відобразити рік - **"yyyy"**, квартал - **"q"**, день - **"d"**, тиждень - **"w"**, годину - **"h"**, хвилини - **"n"**, секунди - **"s"** тощо).

4. Статистичні функції SQL

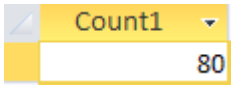
Статистичні функції допомагають нам отримати готові дані без їх вибірки. SQL-запити з цими функціями часто використовуються для аналізу та створення різноманітних звітів. Прикладом таких вибірок може бути: визначення кількості рядків в таблиці, отримання суми значень по певному полю, пошук найбільшого/найменшого або середнього значення в зазначеному стовпці таблиці. Також зазначимо, що статистичні функції підтримуються усіма СУБД без якихось особливих змін в написанні.

Знак операції	Значення
COUNT()	Повертає число рядків в таблиці або стовпці
SUM()	Повертає суму значень в стовпці
MIN()	Повертає найменше значення в стовпці

MAX()	Повертає найбільше значення в стовпці
AVG()	Повертає середнє значення в стовпці

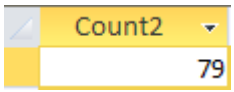
*Приклади використання функції **COUNT()**:*

SELECT COUNT(*) AS Count1 FROM Sumproduct - повертає кількість усіх рядків в таблиці



Count1
80

SELECT COUNT(Product) AS Count2 FROM Sumproduct - повертає кількість усіх непорожніх рядків в полі**Product**

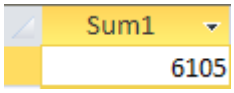


Count2
79

Ми навмисно видалили одне значення в стовпці **Product**, щоб показати різницю в роботі двох запитів.

*Приклади використання функції **SUM()**:*

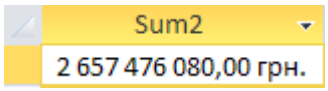
SELECT SUM(Quantity) AS Sum1 FROM Sumproduct WHERE Month = 'April'



Sum1
6105

Даним запитом ми відобразили загальну кількість проданого товару в квітні місяці.

SELECT SUM(Quantity*Amount) AS Sum2 FROM Sumproduct



Sum2
2 657 476 080,00 грн.

Як бачимо, в статистичних функціях ми також можемо здійснювати обчислення над кількома стовпцями з використанням стандартних математичних операторів.

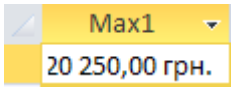
*Приклад використання функції **MIN()**:*

SELECT MIN(Amount) AS Min1 FROM Sumproduct

Min1
396,00 грн.

*Приклад використання функції **MAX()**:*

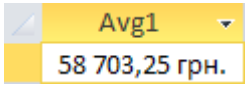
SELECT MAX(Amount) AS Max1 FROM Sumproduct



Max1
20 250,00 грн.

*Приклад використання функції **AVG()**:*

SELECT AVG(Amount) AS Avg1 FROM Sumproduct



Avg1
58 703,25 грн.

Об'єднання рядків оператором LEFT JOIN в SQL-запиті

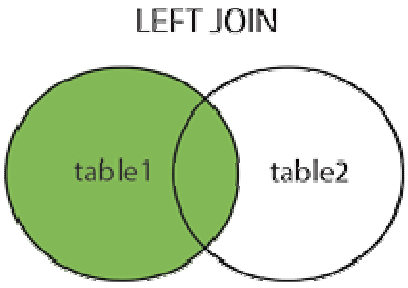
Ключове слово **LEFT JOIN** вибирає всі рядки в SQL-запиті з лівої таблиці (table1) з відповідними рядками в правій таблиці (table2).
Результат NULL коли не має відповідних рядків.
Синтаксис SQL-запиту:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;

чи
```

```
SELECT column_name(s)
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

PS в деяких Базах Даних **LEFT JOIN** теж саме, що і **LEFT OUTER JOIN**.



Приклад LEFT JOIN в SQL-запиті

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Зауважте. Оператор **LEFT JOIN** повертає всі значення даних з лівої таблиці, навіть якщо не було знайдено відповідних рядків в правій.

Об'єднання рядків з однієї чи декількох таблиць оператором RIGHT JOIN в SQL-запиті

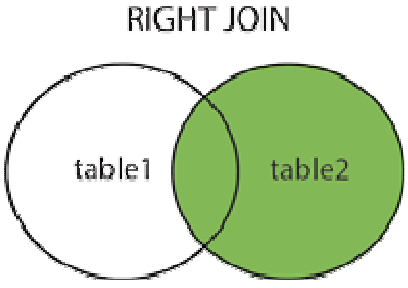
Ключове слово **RIGHT JOIN** вибирає всі рядки в SQL-запиті з правої таблиці (table2) з відповідними рядками в лівій таблиці (table1).
Результат NULL коли не має відповідних рядків.
Синтаксис SQL-запиту:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;

чи

SELECT column_name(s)
FROM table1
RIGHT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

PS в деяких Базах Даних **RIGHT JOIN** теж саме, що і **RIGHT OUTER JOIN**.



Приклад RIGHT JOIN в SQL-запиті

Слідуючий SQL-вираз повертає записи даних замовлень співробітників, що вони виконали.

```
SELECT Orders.OrderID, Employees.FirstName
FROM Orders
RIGHT JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID
ORDER BY Orders.OrderID;
```

Зауважте. Оператор **RIGHT JOIN** повертає всі значення даних з правої таблиці**Employees** (Співробітники), навіть якщо не було знайдено відповідних рядків в лівій таблиці **Orders**(Замовлення).

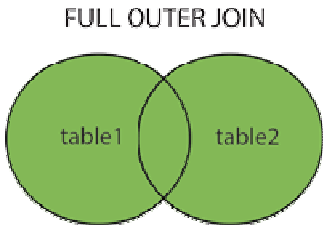
Об'єднання рядків оператором FULL OUTER JOIN в SQL-запиті

Ключове слово **FULL OUTER JOIN** повертає всі дані як з лівої таблиці так і з правої таблиці.

Синтаксис SQL-запиту:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

PS в деяких Базах Даних **FULL JOIN** теж саме, що і **FULL OUTER JOIN**.



Приклад Бази Даних

Таблиця **"Customers"**(Споживачі).

ID	CustomerName	ContactName	Address
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57
2	Ana Trujillo	Ana Trujillo	Constitucion 2222
3	Antonio Moreno	Antonio Moreno	Mataderos 2312

Таблиця **Orders**.

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Приклад FULL OUTER JOIN в SQL-запиті

Слідуючий SQL-вираз повертає записи всіх споживачів та всіх замовлень.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders
```

```
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Вибірка з таблиць буде мати вигляд:

CustomerName	OrderID
Alfreds Futterkiste	
Ana Trujillo	10308
Antonio Moreno Taquería	10365
	10382
	10351

Зауважте: Оператор **FULL OUTER JOIN** повертає всі рядки з лівої таблиці **"Customers"**(Споживачі), та всі записи з правої таблиці **Orders**(замовлення). Якщо рядок з таблиці**"Customers"** не має відповідного рядка в таблиці **"Orders"**, чи навпаки рядок з таблиці**"Orders"** не має відповідного з таблиці **"Customers"**, то він залишається пустим.

Повне зовнішнє об'єднання не підтримують такі СУБД: **Access, MySQL, SQL Server та Sybase**.

КОМБІНОВАНІ ЗАПИТИ (UNION)

В більшості **SQL-запитів** використовується один оператор, за допомогою якого повертаються дані із однієї або кількох таблиць. **SQL** також дозволяє виконувати одночасно кілька окремих запитів та відображати результат у вигляді єдиного набору даних. Такі комбіновані запити зазвичай називають*поєднаннями* або *складними запитами*.

1. Використання оператора UNION

Запити в мові **SQL** комбінуються за допомогою оператора **UNION**. Для цього необхідно вказати кожен запит **SELECT** та розмістити між ними ключове слово **UNION**. Обмежень щодо кількості використань оператора **UNION** в одному загальному запиті немає. В попередньому розділі ми зазначали, що **MySQL** немає можливості створювати *повне зовнішнє об'єднання*, тепер ми подивимось, як можна цього досягнути через оператор **UNION**.

```
SELECT *
FROM Sumproduct LEFT JOIN Sellers ON Sumproduct.City = Sellers.City
UNION
SELECT *
FROM Sumproduct RIGHT JOIN Sellers ON Sumproduct.City = Sellers.City
```

Sumproduct.ID	Month	Product	Sumproduct	Quantity	Amount	Sellers.ID	Address	Sellers.City	Seller_name	Country
						5	4th Avenue	Ottawa	Semuel Piter	Canada
2	April	Bikes	Montreal	12	4 500,00 грн.	1	500 Park Street	Montreal	Michelle Gree	Canada
3	April	Bikes	Montreal	56	21 000,00 грн.	1	500 Park Street	Montreal	Michelle Gree	Canada
4	April	Bikes	San Francisco	854	320 250,00 грн.	2	1000 5th Avenue	San Francisco	Kim Howard	USA
5	April	Bikes	New York	25	9 375,00 грн.	3	42 Galaxy Road	New York	John Smith	USA
6	April	Skates	Montreal	56	5 544,00 грн.	1	500 Park Street	Montreal	Michelle Gree	Canada
7	April	Skates	Toronto	854	84 546,00 грн.	4	123 Main Street	Toronto	Denise L. Step	Canada
8	April	Skates	San Francisco	25	2 475,00 грн.	2	1000 5th Avenue	San Francisco	Kim Howard	USA
9	April	Skates	New York	663	65 637,00 грн.	3	42 Galaxy Road	New York	John Smith	USA
10	April	Skis Long	Montreal	854	209 230,00 грн.	1	500 Park Street	Montreal	Michelle Gree	Canada

Бачимо, що запит відобразив як всі колонки з першої таблиці - так і з другої, незалежно від того, чи всі записи мають відповідники у іншій таблиці.

Також варто зазначити, що в багатьох випадках замість **UNION** ми можемо використовувати речення **WHERE** з багатьма умовами, та отримувати аналогічний результат. Проте через **UNION** записи

виглядають більш лаконічними та зрозумілими. Також необхідно дотримуватись певних правил при написанні комбінованих запитів:

- запит **UNION** повинен включати два і більше операторів **SELECT**, відділених між собою ключовим словом **UNION** (тобто якщо в запиті використовується чотири оператора SELECT, то повинно бути три ключових слова UNION);
- кожен запит в операторі **UNION** повинен мати одні й ті ж стовпці, вирази чи статистичні функції, які, до того ж, мають бути перераховані в однаковому порядку;
- типи даних стовпців мають бути сумісними. Вони не обов'язково мають бути одного типу, проте мають мати подібний тип, щоби **СУБД** могла їх однозначно перетворити (наприклад, це можуть бути різні числові типи даних або різні типи дати).

2. Включення або виключення повторюваних рядків

Запит з **UNION** автоматично видаляє усі повторювані рядки з набору результатів запиту (іншими словами, веде себе як речення **WHERE** з кількома умовами в одному операторі **SELECT**). Така поведінка оператора **UNION** по замовчуванню, але при бажанні ми можемо змінити це. Для цього нам варто використовувати оператор **UNION ALL** замість **UNION**.

3. Сортування результатів комбінованих запитів

Результати виконання оператора **SELECT** сортуються за допомогою речення **ORDER BY**. При комбінуванні запитів за допомогою **UNION** тільки одне речення **ORDER BY** може бути використане, і воно має бути проставлене в останньому операторі **SELECT**. Дійсно, на практиці немає особливого змісту частину результатів сортувати в одному порядку, а іншу частину - в іншому. Тому кілька речень **ORDER BY** застосовувати не дозволяється.

ЗАВДАННЯ 1. ПРОАНАЛІЗУВАТИ ЗАВДАННЯ (ТРАНЗАКЦІЇ) (!!!!)

(В кожній таблиці (відношенні) додати первинні та зовнішні ключі для з'єднання таблиць)
(Дані в таблицях продумати таким чином, щоб вони відповідали умовам майбутніх запитів)

Варіант № 1.
Предметна область - навчальна група факультету. Кожна навчальна група може бути описана структурою: назва факультету, код спеціальності, номер групи, склад групи. Склад групи може бути описаний списком структур, що описують окремого студента: прізвище, ім'я, по батькові, стать, рік народження, навчання на військовій кафедрі, зведена відомість. Зведена відомість може бути описана списком з наступних структур: предмет, вид атестації (іспит, залік), оцінка.
Реалізувати такі типи запитів:
1. Підрахувати число груп на зазначеному факультеті.
2. Знайти всіх студенток, що навчаються за вказаною спеціальністю.
3. Знайти всіх студентів, які мають заборгованості.
4. Підрахувати загальну кількість студентів на зазначеному факультеті.
5. Знайти групу, у якій найбільше відмінників.
6. Знайти всі предмети в групах зазначеної спеціальності, за якими здавався залік.
7. Знайти всіх студентів чоловічої статі, які навчаються на військовій кафедрі.
8. Знайти всіх студентів, народжених після заданого року народження;
9. Підрахувати середній бал сесії по факультету.

Варіант № 2.
Предметна область - відеотека. Кожна відеокасета може бути описана структурою: назва фільму, рік створення, кіностудія, атрибути фільму. Атрибути фільму можуть бути описані структурою: автор сценарію, режисер, виконавці головних ролей, премії. Виконавці головних ролей можуть бути описані списком з наступних структур: прізвище, роль. Премії можуть бути описані списком з наступних структур: назва фестивалю, рік проведення.
Реалізувати такі типи запитів:
1. Підрахувати число фільмів зазначеного режисера.
2. Знайти всі фільми, які отримали премії на певному фестивалі.
3. Знайти всіх режисерів, фільми яких створювалися на одній кіностудії.
4. Знайти всі ролі, заданого актора, які він зіграв у фільмах, які отримали будь-які премії.
5. Знайти режисерів і сценаристів, у яких всі фільми отримали премії.
6. Знайти всі кіностудії, які працювали з зазначеним режисером.
7. Знайти режисера, чиї фільми отримали максимальне число премій.
8. Знайти всі фільми, в яких зіграв зазначений актор.
9. Знайти всіх акторів, що знімалися у фільмах певного сценариста.

Варіант № 3.
Предметна область - бібліотека. Кожна книга може бути описана структурою: назва, автор, видання. Автор може бути описаний структурою: ім'я, прізвище, рік народження. Видання може бути описано структурою: видавництво, номер видання, рік видання, кількість сторінок, ціна, гонорар автора.
Реалізувати такі типи запитів:
1. Знайти всі книги, видані більш ніж один раз.
2. Знайти всі книги, видані в заданому році згруповані по видавництвах.
3. Знайти всі книги заданого автора.
4. Знайти всі книги, ціна яких перевищує задану суму.
5. Знайти книгу, у якої мінімальна ціна.
6. Знайти всіх авторів, книги яких видавалися тільки один раз.
7. Знайти всі книги зазначеного автора, у яких число сторінок більше заданої величини.
8. Знайти всі видавництва, що випускали книги до заданого року.
9. Знайти автора, у якого максимальний гонорар за видання книги.

Варіант № 4.
Предметна область - служба знайомств. Кожен клієнт може бути описаний структурою: прізвище, ім'я, по батькові, вік, національність, освіта, щомісячний дохід, додаткові умови, вимоги до партнера. Додаткові умови можуть бути описані структурою: володіння житловою площею, наявність дітей, шкідливі звички. Вимоги до партнера можуть бути описані структурою: освіта, вік, національність, щомісячний дохід, додаткові умови.
Реалізувати такі типи запитів:
1. Знайти всіх клієнтів, яким необхідний партнер без шкідливих звичок.
2. Знайти всіх клієнтів з вказаною національністю.
3. Знайти всіх клієнтів, яким необхідний партнер, не старше зазначеного віку, що не має дітей.
4. Знайти інформацію про найстаршого клієнта служби знайомств.
5. Знайти всіх клієнтів, для яких підходить партнер з вищою освітою і з зазначеним рівнем доходу.
6. Знайти всіх клієнтів, у яких не вказані додаткові умови.
7. Знайти всіх клієнтів молодше вказаного віку.
8. Знайти всіх клієнтів, у яких немає дітей.
9. Знайти всіх клієнтів зазначеної національності, не старше зазначеного віку.

Варіант № 5.
Предметна область - спортивні змагання. кожне змагання може бути описано структурою: вид змагання, команди-учасники. Вид змагання може бути описаний структурою: ранг змагань (чемпіонат Європи, чемпіонат світу, Олімпійські ігри), вид спорту, рік проведення, країна проведення. Команди-учасники можуть бути описані списком з наступних структур: назва команди, країна, результати змагань. Результати змагань можуть бути описані списком структур: назва команди-суперника, тип результату (виграш, програш, нічия).
Реалізувати такі типи запитів:
1. Знайти всі країни, де проводилися Олімпійські ігри до зазначеного року.
2. Знайти всіх суперників зазначеної команди в змаганнях заданого рангу.
3. Знайти інформацію про змагання, в яких брали участь команди зазначеної країни.
4. Знайти країну, де проводилося максимальне число змагань за вказаний період.
5. Знайти всі країни, де проводилися Чемпіонати світу із зазначеного виду спорту.
6. Знайти всіх суперників зазначеної команди в змаганнях в заданому році.
7. Знайти всі команди, що брали участь в зазначених змаганнях в заданій країні.
8. Знайти всі команди, які брали участь в змаганнях заданого рангу згруповані за видами спорту.
9. Знайти всі команди певної країни, у яких не було виграшів.

Варіант № 6.
Предметна область - біржа праці. Кожна вакансія може бути описана структурою: назва підприємства, посада, щомісячний дохід, вимоги до претендента. Вимоги до претендента можуть бути описані структурою: освіта, вік, стать, додаткові умови. Освіта може бути описана структурою:
рівень освіти (вища технічна, вища економічна, середня, середня спеціальна і т.д.), спеціальність. Додаткові умови можуть бути описані структурою: список іноземних мов, якими повинен володіти претендент, вміння працювати на ПК, стаж роботи за фахом.
Реалізувати такі типи запитів:
1. Знайти всі посади, для яких вік претендентів, нижчий за заданий.
2. Знайти всі вакансії для претендентів з вищою гуманітарною освітою.
3. Знайти всі вакансії для жінок, які вміють працювати на ПК та мають стаж роботи понад 5 років.
4. Знайти всі підприємства, які шукають на роботу жінок.
5. Знайти всі посади, для яких підходять претенденти з середньою спеціальною освітою.
6. Знайти підприємство, у якого найбільше вакансій.
7. Знайти всі вакансії для шукачів зазначеної спеціальності.
8. Знайти всі посади, для яких не потрібно знання іноземної мови.
9. Знайти посаду з мінімальним щомісячним доходом.

Варіант № 7.
Предметна область - сім'я. Кожна сім'я може бути описана структурою з трьох компонент: чоловік, дружина і кількість дітей. Кожен член сім'ї може бути описаний структурою: ім'я, по батькові, прізвище, рік народження, стать, робота. Структура робота описана: посада, щомісячний дохід.
Реалізувати такі типи запитів:
1.Перевірити, чи існує в БД задана людина (по ПІБ).
2.Знайти всіх працюючих чоловіків, чий дохід більше ніж у дружини.
3.Знайти всіх людей, які не працюють і народилися до зазначеного року.
4.Знайти всіх працюючих жінок, чий дохід більше заданої суми.
5.Знайти всіх людей, у яких є тільки одна дитина.
6.Знайти всіх людей, чий дохід менше заданого.
7.Знайти всіх непрацюючих дружин, які народилися пізніше заданого року.
8.Знайти сім'ї, у яких різниця у віці батьків перевищує задану величину.
9. Знайти кількість сімей, у яких є більше 3 дітей.

Варіант № 8.
Предметна область - країни світу. Кожна країна може бути описана структурою: назва, площа, географічне положення, населення. Географічне становище може бути описане структурою: частина світу, материк, океани, моря, гірські хребти. Населення може бути описане структурою: чисельність, державна мова, національний склад. Національний склад може бути описаний структурою: національність, чисельність, відсоток від усього населення.
Реалізувати такі типи запитів:
1. Знайти країну, у якій максимальна чисельність населення.
2. Знайти всі країни, що знаходяться на вказаному материку з населенням більше заданої величини.
3. Знайти всі країни, що мають вихід до зазначеного моря.
4. Знайти країни, які омиває найбільше морів.
5. Знайти всі країни, на території яких знаходиться зазначений гірський хребет.
6. Знайти всі країни, у яких назва частини світу збігається з назвою материка.
7. Знайти всі гірські хребти, що знаходяться на території зазначеної країни.
8. Знайти всі країни, у яких чисельність населення менше заданої величини.
9. Знайти всі моря, які омивають територію зазначеної країни.

Варіант № 9.
Предметна область - бібліотека. Кожна книга може бути описана структурою: назва, автор, видання. Автор може бути описаний структурою: ім'я, прізвище, рік народження. Видання може бути описано структурою: видавництво, номер видання, рік видання, кількість сторінок, ціна, гонорар автора.
Реалізувати такі типи запитів:
1. Знайти автора, у якого книга перевидавалася максимальне число раз.
2. Знайти всі книги, видані в одному видавництві.
3. Знайти всі книги, видані тільки один раз.
4. Знайти всіх авторів, які народилися пізніше зазначеного року.
5. Знайти всі видавництва, в яких була видана зазначена книга.
6. Знайти всі книги, ціна яких не змінювалася в різних виданнях.
7. Знайти всіх авторів, книги яких видавалися більше 2 разів.
8. Знайти всі книги одного автора, видані в певному році (групувати по авторах).
9. Знайти повну інформацію про книгу з максимальною ціною.

Варіант № 10.
Предметна область – прокат відеодисків. Кожен відеодиск може бути описаний структурою: назва фільму, рік створення, кіностудія, атрибути фільму. Атрибути фільму можуть бути описані структурою: автор сценарію, режисер, виконавці головних ролей, премії. Виконавці головних ролей можуть бути описані списком з наступних структур: прізвище, роль. Премії можуть бути описані списком з наступних структур: назва фестивалю, рік проведення.
Реалізувати такі типи запитів:
1. Знайти режисера, чиї фільми отримали максимальне число премій;
2. Знайти всі фільми, зняті на одній кіностудії, одним і тим же режисером;
3. Знайти акторів, що знімалися на одній кіностудії;
4. Знайти режисерів і сценаристів, у яких всі фільми отримали премії;
5. Знайти всі фільми зазначеного сценариста після зазначеного року;
6. Знайти всі фільми, які отримали премії в зазначеному році;
7. Знайти всі кіностудії, які працювали з зазначеним режисером.
8. Знайти сценаристів, які працювали з зазначеним режисером більше одного рази;
9. Знайти всі ролі, заданого актора, які він зіграв у фільмах, які ніколи не отримували премії.

Варіант № 11.
Предметна область - навчальна група факультету. Кожна навчальна група може бути описана структурою: назва факультету, код спеціальності, номер групи, склад групи. Склад групи може бути описаний списком структур, що описують окремого студента: прізвище, ім'я, по батькові, стать, рік народження, навчання на військовій кафедрі, зведена відомість. Зведена відомість може бути описана списком з наступних структур: предмет, вид атестації (іспит, залік), оцінка.
Реалізувати такі типи запитів:
1. Підрахувати число груп на зазначеному факультеті.
2. Знайти всіх студентів чоловічої статі, що навчаються за вказаною спеціальністю.
3. Знайти всіх студенток, які мають заборгованості.
4. Підрахувати загальну кількість студентів на зазначеному факультеті.
5. Знайти групу, у якій найменше відмінників.
6. Знайти всі предмети в групах зазначеної спеціальності, за якими здавався екзамен.
7. Знайти всіх студентів чоловічої статі, які навчаються на військовій кафедрі.
8. Знайти всіх студентів, молодших за заданий вік;
9. Підрахувати середній бал сесії по факультету.

Варіант № 12.
Предметна область - відеотека. Кожна відеокасета може бути описана структурою: назва фільму, рік створення, кіностудія, атрибути фільму. Атрибути фільму можуть бути описані структурою: автор сценарію, режисер, виконавці головних ролей, премії. Виконавці головних ролей можуть бути описані списком з наступних структур: прізвище, роль. Премії можуть бути описані списком з наступних структур: назва фестивалю, рік проведення.
Реалізувати такі типи запитів:
1. Підрахувати число фільмів зазначеного режисера.
2. Знайти всі фільми, які отримали премії на певному фестивалі.
3. Знайти всіх режисерів, фільми яких створювалися на одній кіностудії.
4. Знайти всі ролі, заданого актора, які він зіграв у фільмах, які отримали будь-які премії.
5. Знайти режисерів і сценаристів, у яких всі фільми отримали премії.
6. Знайти всі кіностудії, які працювали з зазначеним режисером.
7. Знайти режисера, чиї фільми отримали максимальне число премій.
8. Знайти всі фільми, в яких зіграв зазначений актор.
9. Знайти всіх акторів, що знімалися у фільмах певного сценариста.

Варіант № 13.

Предметна область - бібліотека. Кожна книга може бути описана структурою: назва, автор, видання. Автор може бути описаний структурою: ім'я, прізвище, рік народження. Видання може бути описано структурою: видавництво, номер видання, рік видання, кількість сторінок, ціна, гонорар автора.

Реалізувати такі типи запитів:

1. Знайти всі книги, видані більш ніж один раз.
2. Знайти всі книги, видані в заданому році згруповані по видавництвах.
3. Знайти всі книги заданого автора.
4. Знайти всі книги, ціна яких не перевищує задану суму.
5. Знайти книгу, у якої максимальна ціна.
6. Знайти всіх авторів, книги яких видавалися тільки один раз.
7. Знайти всі книги зазначеного автора, у яких число сторінок більше заданої величини.
8. Знайти всі видавництва, що випускали книги до заданого року.
9. Знайти автора, у якого максимальний гонорар за видання книги.

Варіант № 14.

Предметна область - служба знайомств. Кожен клієнт може бути описаний структурою: прізвище, ім'я, по батькові, вік, національність, освіта, щомісячний дохід, додаткові умови, вимоги до партнера. Додаткові умови можуть бути описані структурою: володіння житловою площею, наявність дітей, шкідливі звички. Вимоги до партнера можуть бути описані структурою: освіта, вік, національність, щомісячний дохід, додаткові умови.

Реалізувати такі типи запитів:

1. Знайти всіх клієнтів, яким необхідний партнер без шкідливих звичок.
2. Знайти всіх клієнтів з вказаною національністю.
3. Знайти всіх клієнтів, яким необхідний партнер, зазначеного в діапазоні віку, що не має дітей.
4. Знайти інформацію про наймолодшого клієнта служби знайомств.
5. Знайти всіх клієнтів, для яких підходить партнер з вищою освітою і з зазначеним рівнем доходу.
6. Знайти всіх клієнтів, у яких не вказані додаткові умови.
7. Знайти всіх клієнтів молодше вказаного віку.
8. Знайти всіх клієнтів, у яких немає дітей.
9. Знайти всіх клієнтів зазначеної національності, не старше зазначеного віку.

Варіант № 15.

Предметна область - спортивні змагання. кожне змагання може бути описано структурою: вид змагання, команди-учасники. Вид змагання може бути описаний структурою: ранг змагань (чемпіонат Європи, чемпіонат світу, Олімпійські ігри), вид спорту, рік проведення, країна проведення. Команди-учасники можуть бути описані списком з наступних структур: назва команди, країна, результати змагань. Результати змагань можуть бути описані списком структур: назва команди-суперника, тип результату (виграш, програш, нічия).

Реалізувати такі типи запитів:

1. Знайти всі країни, де проводилися Олімпійські ігри після вказаного року.
2. Знайти всіх суперників зазначеної команди в змаганнях заданого рангу.
3. Знайти інформацію про змагання, в яких брали участь команди зазначеної країни.
4. Знайти країну, де проводилося максимальне число змагань за вказаний період.
5. Знайти всі країни, де проводилися Чемпіонати світу із зазначеного виду спорту.
6. Знайти всіх суперників зазначеної команди в змаганнях в заданому році.
7. Знайти всі команди, що брали участь в зазначених змаганнях в заданій країні.
8. Знайти всі команди, які брали участь в змаганнях заданого рангу згруповані за видами спорту.
9. Знайти всі команди певної країни, у яких не було виграшів.

Варіант № 16.
Предметна область - біржа праці. Кожна вакансія може бути описана структурою: назва підприємства, посада, щомісячний дохід, вимоги до претендента. Вимоги до претендента можуть бути описані структурою: освіта, вік, стать, додаткові умови. Освіта може бути описана структурою:
рівень освіти (вища технічна, вища економічна, середня, середня спеціальна і т.д.), спеціальність. Додаткові умови можуть бути описані структурою: список іноземних мов, якими повинен володіти претендент, вміння працювати на ПК, стаж роботи за фахом.
Реалізувати такі типи запитів:
1. Знайти всі посади, для яких вік претендентів, нижчий за заданий.
2. Знайти всі вакансії для претендентів з вищою технічною освітою.
3. Знайти всі вакансії для жінок, які вміють працювати на ПК та мають стаж роботи понад 3 роки.
4. Знайти всі підприємства, які шукають на роботу жінок.
5. Знайти всі посади, для яких підходять претенденти з середньою спеціальною освітою.
6. Знайти підприємство, у якого найменше вакансій.
7. Знайти всі вакансії для шукачів зазначеної спеціальності.
8. Знайти всі посади, для яких не потрібно знання іноземної мови.
9. Знайти посаду з мінімальним щомісячним доходом.

Варіант № 17.
Предметна область - країни світу. Кожна країна може бути описана структурою: назва, площа, географічне положення, населення. Географічне становище може бути описане структурою: частина світу, материк, океани, моря, гірські хребти. Населення може бути описане структурою: чисельність, державна мова, національний склад. Національний склад може бути описаний структурою: національність, чисельність, відсоток від усього населення.
Реалізувати такі типи запитів:
1. Знайти країну, у якій мінімальна чисельність населення.
2. Знайти всі країни, що знаходяться на вказаному материку з населенням менше заданої величини.
3. Знайти всі країни, що не мають виходу до моря.
4. Знайти країни, які омиває найбільше морів.
5. Знайти всі країни, на території яких знаходиться зазначений гірський хребет.
6. Знайти всі країни, у яких назва частини світу збігається з назвою материка.
7. Знайти всі гірські хребти, що знаходяться на території зазначеної країни.
8. Знайти всі країни, у яких чисельність населення більша заданої величини, та материки, на яких вони знаходяться.
9. Знайти всі країни, які омиваються зазначеним морем.

РЕСУРСИ ДЛЯ ДОПОМОГИ:

http://beginner-sql-tutorial.com/sql.htm
http://bestwebit.biz.ua/w3c_1/mysql_w3c_syntax.php
http://moonexcel.com.ua/%D1%83%D1%80%D0%BE%D0%BA%D0%B8-sql_ua (уроки-sql_ua)
http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-data-definition.html
http://www.tutorialspoint.com/sql/sql-useful-functions.htm