МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
Факультет інформатики та обчислювальної техніки

Кафедра комп'ютерних
систем та мереж

Т.О. Паромова

ТЕКСТИ (конспект) лекцій з дисципліни

Організація баз даних
для студентів спеціальності 123 "Комп'ютерна інженерія"
усіх форм навчання
частина ІІІ

2020

Тексти (конспект) лекцій з дисципліни “Організація баз даних” для студентів спеціальності 123 “Комп’ютерна інженерія” усіх форм навчання частина ІІІ /Укл.: Т.О. Паромова. - Запоріжжя: ЗНТУ, 2020. – 38 с.

Укладачі: Т.О. Паромова, ст. викладач

Рецензент: С.Д. Точилін, доцент, к.ф.м.н.

Відповідальний
за випуск: Т.О. Паромова, ст. викладач

Затверджено
на засіданні кафедри
“Комп’ютерні системи
та мережі ”
Протокол № 1
від “27” серпня 2020.

Рекомендовано до видання
НМК факультету комп’ютерних
наук і технологій
Протокол № 1
від “28” серпня 2020.

ЗМІСТ

ТЕМА 1. РЕЛЯЦІЙНА АЛГЕБРА

1.1 Призначення, компоненти

Основним компонентом реляційної моделі (маніпулювання даними) є реляційна алгебра, яка складається в основному з набору операторів, які використовують відносини в якості операндів і повертають відносини в якості результату.

Реляційна алгебра - це теоретичний мову операцій, які на основі одного або декількох відносин, які дозволяють створити інше ставлення, не змінюючи вихідного. Реляційна алгебра використовує відносини в якості операндів, і результат є відношенням, яке може бути використане як операнд для наступної операції. Це дозволяє створювати вкладені вирази реляційної алгебри точно також як арифметичні вирази.

Реляційна алгебра є мовою послідовного використання відносин, в якому все кортежі обробляються (навіть взяті з різних відносин), однією командою без організації циклів.

Реляційна алгебра, певна Коддом, складається з 8 операторів:

1. операції над множинами: об'єднання (union), перетин (intersec-tion), віднімання (difference);

2. операції видалення частин відносини: вибірка (selection), проекція (projection);

3. операції поєднання кортежів двох відносин: декартово произведе-ня (Cartesian product), з'єднання (join);

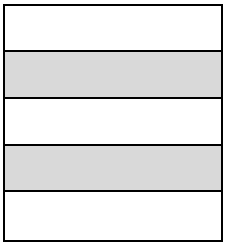
4. операція перейменування атрибутів або відносини цілком (renam-ing).

Вирази реляційної алгебри називаються запитами. Прикладом може служити операція об'єднання R□S, де R і S - атомарні операнди, що представляють відносини з будь-якою кількістю кортежей. Вказаний запит передбачає обчислення об'єднання відносин R і S.

1.2 Операція вибірки

Вибірка (select ... where) (обмеження) повертає відношення, яке містить все кортежі з певного ставлення, які задовольняють деяким умовам.

Схеми підсумкового і вихідного відносини не змінюються. Порядок проходження атрибутів зберігається



$\sigma_{\text{предікат}}$ (R) -операція вибірки працює з одним відношенням і віз-обертає відношення, все кортежі якого задовольняють деякому умові (предикат - умова вибірки).

Приклад

Скласти список всіх співробітників, зарплата яких перевищує 1000 грн.

$\sigma_{\text{salary}> 1000}$ (Співробітники) - вихідне відношення Співробітники, предикат умова salary> 1000, результат теж відношення Співробітники, але записи якого задовольняють предикату.

Більш складні предикати можуть містити логічні оператори **$\wedge$(AND), $\vee$(OR), ~(NOT)**

приклад

Скласти список всіх об'єктів нерухомості, для яких орендна плата перевищує 100 грн. у місяць.

$\sigma_{\text{платня}>100}$(Недвижимость)

Вихідне відношення

КодНедвиж.	Адрес	Плата	Код Влад	Владелец
К10	Победы 17-5	100	123	Сирко А.А.
К12	Ленина 14-7	150	124	Петрик В.В
К11	Победы 87-5	100	123	Сирко А.А.
К5	Гоголя 3-8	90	125	Кирпа А.А.

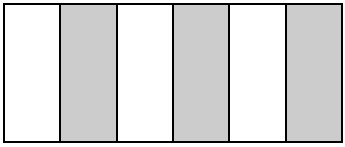
Результат выборки

КодНедвиж.	Адрес	Плата	Код Влад.	Владелец
К12	Ленина 14-7	150	124	Петрик В.В

1.3 Операція проекції

Проекція повертає відношення, що містить всі кортежі, вихідного відношення після виключення з нього деяких атрибутів, які задовольняють визначеним умовам.

Патр1, атр2 ... (R) - операція проекція використовує одне відношення R і визначає нове відношення, що містить тільки деякі атрибути від-носіння R, після вилучення зазначених атрибутів і виключення з ре-результату строк- дублікатів



Приклад: реалізувати проекцію відношення Студенти за схемою

Студенти (КодСтудента, Прізвище, Ім'я, По батькові, Адреса, Спеціальність, Група, ДомТелефон)

Проекція має містити атрибути Фамілія, Ім’я, По батькові, Група

Результат операції проекція:

$П_{фамілія,\ ім'я,\ по\ батькові,\ група}$ (Студенти) $\Rightarrow$ Проекція (Прізвище, Ім'я, По батькові, Група)

1.4 Операція декартового добутку

Операція декартового добутку (Cartesian product) повертає відношення, що містить всілякі кортежі, які є поєднанням двох кортежів, що належать відповідно двом певних відносин.

R×S- визначає нове відношення.

Оператор декартового добутку примножує два відносини, що призводить до створення іншого відношення, що складається з усіх можливих пар кортежів обох відносин. Отже, якщо відношення R містить I кортежів і N атрибутів, ставлення S містить J кортежів і M атрибутів, то буде створено відношення, що містить I * J кортежів і N + M атрибутів. Вихідні відносини можуть містити атрибути з однаковими іменами, тоді в атрибути будуть містити імена відносин у вигляді префіксів.

Приклад: знайти декартів добуток відносин R та S

R

a
b

X

S

1
2
3

=

R×S

a	**1**
a	**2**
a	**3**
b	**1**
b	**2**
b	**3**

Приклад 2

R

A	B
1	**2**
3	**4**

S

B	C	D
2	**5**	**6**
4	**7**	**8**
9	**10**	**11**

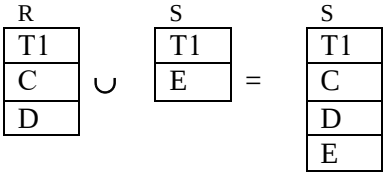
RxS

A	R.B	S.B	C	D
1	**2**	**2**	**5**	**6**
1	**2**	**4**	**7**	**8**
1	**2**	**9**	**10**	**11**
3	**4**	**2**	**5**	**6**
3	**4**	**4**	**7**	**8**
3	**4**	**9**	**10**	**11**

1.5 Операція об'єднання

Об'єднання (union) двох сумісних по типу відносин A і B повертає відношення з тим же заголовком, як у відносинах A і B, і тілом, що містить всі кортежі, які належать A або B або обом відношенням, при цьому всі дублікати виключаються, тобто кожен кортеж включається в підсумкове відношення тільки один раз. При цьому відносини повинні бути сумісні по об'єднанню: об'єднуються відносини повинні мати схеми з ідентичним набором атрибутів, що прямують в одному порядку і має збігаються домени $\Rightarrow R \cup S$.

Приклад виконання операції з’єднання



1.6 Операція перетину

Перетин (intersect) повертає відношення, що містить всі кортежі, які належать одночасно двом певним відношенням.

R∩S - позначення операції перетину

Перетин повертає відношення, яке містить кортежі, які належать обом відношенням.

R∩S= R-(R- S)

Приклад перетину відносин

R

A	B	C
a	1	x
a	1	y
b	2	z

S

A	B	C
a	1	x
b	2	y
a	2	z

R∩S

A	B	C
a	1	x

1.7 Операція віднімання

Віднімання (minus) повертає відношення, що містить всі кортежі, які належать до першого з двох певних відносин і не належать другому.

Різниця двох відносин R-S складається з кортежів, які є у відношенні R і відсутні в відношенні S. При цьому відносини повинні бути сумісні по об'єднанню – мати однакові заголовки

Приклад вирахування відносин

R

A	B	C
a	1	x
a	1	y
b	2	z

S

A	B	C
a	1	x
b	2	y
a	2	z

R-S

A	B	C
a	1	y
b	2	z

1.8 Операція з'єднання

З'єднання (join) є похідною від декартового добутку, тому що вона еквівалентна операції вибірки кортежів, які відповідають умові предиката, з декартового добутку двох відносин. У реляційній СУБД з'єднання є однією з основних причин зниження продуктивності.

Розрізняють:

1. тета соединеніе- з'єднання по еквівалентності
2. природне з'єднання
3. зовнішнє з'єднання
4. напівз'єднання

Природне з'єднання передбачає включення в підсумковий відносіння тих кортежів відносин R і S, які збігаються в атрибутах, загальних для схем R і S. Воно засноване на умови рівності вмісту компонентів, які відповідають атрибутам, загальним для обох відносин.

Приклад вирахування відносин

R

A	B
1	2
3	4

S

B	C
2	5
4	7
9	10

$R \triangleright\triangleleft S$

A	B	C
1	2	5
3	4	7

Для реалізації операції з'єднання відносини обов'язково повинні мати загальний атрибут B, тому для з'єднання необхідно, щоб відносини R і S збігалися по атрибуту B. З'єднаний кортеж буде складатись з атрибутів A, B, C.

Тета - з'єднання - визначає відношення, яке містить кортежі з декартового добутку відносин R і S і які задовольняють предикату F.

$R \triangleright\triangleleft_f S$

Предикат f має вигляд $\mathbf{R.a_i\ \Theta S.b_i}$, де **Θ** один з операторів порівняння**.**

Операція тета-з'єднання виконується у такий спосіб:

1. обчислюється декартовій добуток R і S

2. з результату добутку відбираються ті кортежі, які задовольняють визначеній умові f

$\mathbf{R \triangleright\triangleleft_f S = \sigma_f(R x S)}$

Схема підсумкового відношення є об'єднанням схем відношень R і S. При необхідності імена атрибутів забезпечуються префіксом-ім'ям відносини.

Приклад: з'єднати відношень R і S за умови A <D

$\mathbf{R} \triangleright\!\triangleleft_{a>d} \mathbf{S}$

R

A	B	C
1	2	3
6	7	8
9	7	8

S

B	C	D
2	3	4
2	3	5
7	8	10

$\mathbf{R} \triangleright\!\triangleleft_{a>d} \mathbf{S}$

A	R.B	R.C	S.B	S.C	D
6	7	8	2	3	5

$\mathbf{R} \triangleright\!\triangleleft_{a<d} \mathbf{S}$

A	B	C	D
1	2	3	4
9	7	8	10

Відношення R і S мають загальні атрибути B і C, значення яких збігаються тільки в 1 і 3 кортежі в обох відношеннях і 2 в першому відношенні. Але умова A <D, виконується тільки для 1 і 3 кортежів. Потому в підсумкове відношення входять тільки ці кортежі.

Зовнішнє з'єднання

Якщо при з'єднанні двох відносин кортежу одного відношення не відповідає жоден кортеж в іншому відношенні, тобто, в стовпах з'єднань опиняються неспівпадаючі значення, може знадобитися, щоб в результаті з'єднання була збережена неспівпадаючий рядок. Це досягається за допомогою зовнішнього з'єднання. Для відсутніх значень використовується визначник NULL.

$\mathbf{R} \supset\!\!< \mathbf{S}$ – ліве зовнішнім з'єднання - з'єднання, при якому кортежі відносини R, що не мають співпадаючих значень в загальних шпальтах відносини S, також включаються в результуючий відношення. Для відсутніх значень використовується визначник NULL. Перевагою зовнішнього з'єднання є збереження вихідної інформації, тобто,

зовнішнє з'єднання зберігає кортежі, які були втрачені при використанні інших типів з'єднань.

Пример.

R

A	B
a	1
в	2

S

B	C
1	x
1	y
3	y

R⊃<S

A	B	C
a	1	x
a	1	y
в	2	NULL

При створенні лівого зовнішнього з'єднання відносин R і S в вихідне відношення включаються всі кортежі відносини R. З відносини S додаються тільки ті атрибути, які збігаються в обох відносинах і по атрибуту і за значенням. Для атрибутів відношення R, значення яких не збігаються з атрибутами відносини S, приймається значення NULL (порожня клітина).

1.9 Набори операцій і формування запитів

Реляційна алгебра дозволяє створювати вирази різного ступеня складності і застосовувати оператори не тільки до вихідних відносин, але і до відносин, отриманим в результаті виконання інших операторів.

Вирази реляційної алгебри можуть містити оператори, призмінювані до більш приватним виразами, і дужки, що задають порядок групування операндів.

Приклад Дано відношення R

A	**B**	**C**	**D**	**E**	**f**
a	1	X	3	M	a
a	2	X	4	M	b
b	1	Y	1	N	C
b	2	Y	4	N	B
c	1	Y	5	N	У

1. вибрати з відносини R кортежі, для яких D> = 3

$\sigma_{D>=3}(R)$

A	**B**	**C**	**D**	**E**	**f**
a	1	X	3	M	a
a	2	X	4	M	b
b	2	Y	4	N	B
c	1	Y	5	N	У

2. вибрати з відносини R кортежі, для яких C = Y

$\sigma_{c=y}(R)$

A	B	C	D	E	f
b	1	Y	1	N	C
b	2	Y	4	N	B
c	1	Y	5	N	У

3. обчислити перетин відносин, отриманих при виконанні операцій п.п.1 і 2

$\sigma_{D>=3}(R) \cap \sigma_{c=y}(R)$

A	B	C	D	E	f
b	1	Y	1	N	C
b	2	Y	4	N	B
c	1	Y	5	N	У

4. здійснити проекцію підсумкового відносини п.3 на атрибути A, E, f

$\Pi_{A, E, f}(\sigma_{D>=3}(R) \cap \sigma_{c=y}(R))$

A	E	f
b	N	B
c	N	У

ТЕМА 2 МОВА SQL

2.1 Структура мови SQL

Стандарт мови SQL, хоч і заснований на реляційній теорії, але в багато в яких випадках відходить він її. Наприклад, відношення в реляційній моделі даних не припускає наявності однакових кортежів, а таблиці в термінології SQL можуть мати однакові рядки.

Мова SQL є реляційно повною. Це означає, що будь-який оператор реляційної алгебри може бути виражений відповідним оператором SQL.

На відміну від реляційної алгебри, де були представлені тільки операції запитів до БД, SQL є повною мовою, в нім присутні не тільки операції запитів, але і оператори, відповідні DDL - Data Definition Language - мові опису даних. Крім того, мова містить операторів, призначених для управління (адміністрування) БД.

Основу мови SQL складають оператори, умовно розбиті не декілька груп по виконуваних функціях (перераховані не всі оператори SQL):

Оператори DDL (Data Definition Language) - оператори визначення об'єктів бази даних

CREATE SCHEMA створити схему бази даних
DROP SHEMA видалити схему бази даних
CREATE TABLE створити таблицю
ALTER TABLE змінити таблицю
DROP TABLE видалити таблицю
CREATE DOMAIN створити домен
ALTER DOMAIN змінити домен
DROP DOMAIN видалити домен
CREATE COLLATION створити послідовність
DROP COLLATION видалити послідовність
CREATE VIEW створити уявлення
DROP VIEW видалити уявлення

Оператори DML (Data Manipulation Language) - оператори маніпулювання даними

SELECT вибрати рядки з таблиць
INSERT додати рядки в таблицю
UPDATE змінити рядки в таблиці
DELETE видалити рядки в таблиці
COMMIT зафіксувати внесені зміни
ROLLBACK відкотити внесені зміни

Оператори захисту і управління даними:

CREATE ASSERTION створити обмеження
DROP ASSERTION видалити обмеження
GRANT надати привілеї користувачеві або додатку на маніпулювання об'єктами
REVOKE відмінити привілеї користувача або додатки

Крім того, є групи операторів установки параметрів сеансу, отримання інформації про базу даних, оператори статичного SQL, оператори динамічного SQL.

Найбільш важливими для користувача є оператори маніпулювання даними (DML).

2.2 Оператор SELECT

Запис операторів SQL

- SQL -оператор складається із зарезервованих слів і слів, визначуваних користувачем.

- Зарезервовані слова не можна розбивати на частини для перенесення на інший рядок.

- Слова, визначуваних користувачем, є імена об'єктів ба-за даних.

- Слова в операторові розставляються відповідно до встановлених синтаксичних правил.

- В кінці оператора ставитися символ ; .

- Символьні літерали - дані повинні вводитися з урахуванням регістра.

Правила запису операторів SQL

- Кожна фраза починається з нового рядка.

- Початок фрази повинен бути вирівняний з початком решти фраз оператора.

- Якщо фраза має декілька частин, кожна повинна починатися з нового рядка з деяким відступом відносно почала фрази, що указує на її підлеглість.

BNF- нотація SQL -операторів

- Прописними буквами записуються зарезервовані слова

- Рядковими буквами записуються слова користувачів

- Вертикальна межа (|) означає обов'язковість вибору одного з зазначених значпараметрівень

- Фігурні дужки визначають обов'язковий елемент

- Квадратні дужки визначають необов'язковий елемент

- Багатокрапка (...) використовується для вказівки необов'язкового повторення конструкції, розділених комами.

Побудова простих запитів за допомогою оператора вибору SELECT

Призначення оператора SELECT полягає у вибірці і відображенні даних од-ной або більш за таблиці бази даних. Цього оператора забезпечує виконання дій, еквівалентних операторам реляційної алгебри: вибірка, об'єднання, проекция.

Цей єдиний оператор реалізує всі операції реляційної алгебри. Один і той же запит може бути реалізований декількома способами, і, будучи все правильними, вони, проте, можуть істотно відрізнятися за часом виконання, і це особливо важливо для великих баз даних.

Результатом виконання оператора SELECT завжди є таблиця. Будь-який оператор реляційної алгебри може бути виражений оператором SELECT.

2.3 Синтаксис оператора SELECT:

Порядок пропозицій в операторові SELECT не може бути змінений

SELECT [ALL | DISTINCT] {* [список полий [AS нове ім'я]][,..]|*)
FROM ім'я таблиці [alias][,.]
[WHERE Предикат-умова вибірки або з'єднання]
[GROUP BY Список полий]
[HAVING Предикат-умова для групи]
[ORDER BY <Список полий, по яких упорядкувати висновок>]

SELECT встановлюється, які стовпці повинні бути у вихідних даних

FROM визначаються імена використовуваних таблиць

WHERE виконується фільтрація рядків об'єкту відповідно до заданих умов

GROUP BY утворюються групи рядків, що мають одне і те ж значення в столб-це

HAVING фільтруються групи рядків відповідно до вказаної умови

ORDER BY визначається впорядкованість результатів виконання оператора

alias - скорочене ім'я таблиці або псевдонім, друге ім'я таблиці.

Ключове слово ALL - в результуючий набір рядків включаються всі рядки, що задовольняють умовам запиту, навіть однакові. І це порушення принципів теорії відносин (на відміну від реляційної алгебри, де за умовчанням передбачається відсутність дублікатів в кожному результуючому відношенні).

Ключове слово DISTINCT - в результуючий набір включаються тільки раз-особисті рядки, тобто дублікати рядків виключаються з результату.

2.4 Пропозиція SELECT

SELECT - ключове слово, яке повідомляє СУБД, що ця команда - запит. Всі запити починаються з цього слова з подальшим пропуском. За ним може слідувати спосіб вибірки - з видаленням дублікатів

(DISTINCT) або без видалення (ALL, за умовчанням), потім указуються імена полий, які повинні бути включені в результуюче відношення.

За ключовим словом AS указується підпис (назва) стовпця в вихідній таблиці.

Пропозиція SELECT може включати тільки наступні типи елементів:

•імена стовпців;

•агрегатні функції;

•константи;

•вирази, що включають комбінації перерахованих елементів;

•вкладені запити.

Символ * (зірочка) - в результуючий набір включаються всі стовпці з початкових таблиць запиту.

2.5 Пропозиція FROM

У пропозиції FROM задається перелік початкових відносин (таблиць) запиту.

FROM < Список таблиць >.

FROM - ключове слово, подібно SELECT, яке повинне бути в кожному запиті. Список таблиць може містити одну або декілька таблиць. При витяганні даних тільки з однієї таблиці можна не указувати її ім'я перед іменами стовпців в пропозиції SELECT. Якщо дані витягуються з д таблиць, декількох таблиць, обов'язково указується спосіб з'єднання таблиць.

Все інші розділи оператора SELECT є необов'язковими.

Приклад: Створити список студентів, з вказівкою їх ФІО і групи

SELECT ФІО, Група

FROM Студенти;

Приклад: створення обчислюваного поля: для кожного продукту вибрати найменування, ціну, кількість і загальну вартість.

SELECT Найменування, Ціна, кількість, (Ціна* Кількість) AS Вартість

FROM Продукти;

2.6 Пропозиція WHERE

Для обмеження набору рядків що поміщаються в результуючу таблицю запиту використовується пропозиція WHERE.

У пропозиції WHERE задаються умови відбору рядків в результуючу таблицю або умови з'єднання кортежів початкових

таблиць, тобто задаються обмеження набору рядків, що поміщаються в результуючу таблицю.

Вибір рядків WHERE <Предикат-умова вибірки або з'єднання >.

Існує п'ять основних типів умов пошуку (предикатів - термінологія ISO).

• **Порівняння** - порівнюються результати обчислень двох виразів.

Предикати порівняння {=, <> >, <, >=, <=}, які мають традиційний сенс

Приклад: вибрати всі продукти, ціна яких більше 25 грн.

SELECT Найменування

FROM Продукти

WHERE Ціна > 25;

• **Діапазон**-перевіряється, чи потрапляє результат обчислення в заданий діапазон значень.

Предикат Between A and B - приймає значення між A і B. Предикат виконується, коли порівнюване значення потрапляє в заданий діапазон, включаючи границі діапазону. Одночасно в стандарті заданий і протилежний предикат Not Between A and B, який істинний тоді, коли порівнюване значення не потрапляє в заданий інтервал, включаючи його межі.

Приклад: використання діапазонів в умовах пошуку.

Скласти список викладачів з 1949 по 1953 рік народження

SELECT Викладачі.*, Викладачі.[Дата народження]

FROM Викладачі

WHERE ((year (Викладачі.[Дата народження]) Between 1949 And 1953));

• **Приналежність множині** - перевіряється, yeагчи належить результат до задан-ному безлічі значень.

Предикат входження в безліч IN (множина) істинний тоді, коли значення, яке перевіряється, входить в безліч заданих значень. При цьому безліч значень може бути задана простим перерахуванням або вбудованим запитом. Протилежний предикат NOT IN (множина) істинний тоді, коли значення, яке перевіряється, не входить в задану множину.

Приклад: умови пошуку з перевіркою входження в множину (IN / NOT IN)

Список викладачів - доцентів

SELECT Викладачі.*
FROM Викладачі
WHERE Посада IN ("Доцент");

• **Відповідність шаблону** - перевіряється, чи відповідає деяке рядкове вираз заданому шаблону.

Предикати порівняння із зразком LIKE і NOT LIKE. Предикат LIKE вимагає завдання шаблону, з яким порівнюється задане значення. Предикат істинний, ес-лі порівнюване значення відповідає шаблону, і помилковий інакше. Предикат NOT LIKE має протилежний сенс.

За стандартом в шаблон можуть бути включені спеціальні символи:

- символ підкреслення (_) - для позначення будь-якого одиночного символу;
- символ відсотка (%) - для позначення будь-якої довільної послідовності символів;
- решта символів, заданих в шаблоні, позначає самих себе.

Приклад: умови пошуку з вказівкою шаблонів.

Список всіх співробітників, що проживають на вулиці Горького.

SELECT Викладачі.*
FROM Викладачі
WHERE Адреса LIKE "*Горького*"; // адреса, що містить набір символів Горького (при цьому допустиме використання символу * для позначення будь-якої кількості символів до і після вказаної комбінації символів).

• **Визначник NULL**- перевіряється, чи містить заданий стовпець визначник NULL

Предикат порівняння з невизначеним значенням IS NULL. Невизначене значення інтерпретується як значення, невідоме на даний момент часу. Це значення при появі додаткової інформації у будь-який момент часу може бути замінено деяким конкретним значенням. При порівнянні невизначених значень не діють стандартні правила порівняння: одне невизначене значення ніколи не вважається рівним іншому невизначеному значенню. Для виявлення рівності значення деякого атрибуту невизначеному використовують спеціальні стандартні предикати:

<ім'я атрибуту>**IS NULL** і <ім'я атрибуту> **IS NOT NULL**.

Якщо в даному кортежі (у даному рядку) вказаний атрибут має невизначене значення, то предикат **IS NULL** приймає значення «Істина» (TRUE), а предикат IS NOT NULL - «ЛОЖЬ» (FALSE), інакше предикат IS NULL приймає значення «ЛОЖЬ», а предикат IS NOT NULL приймає значення «Істина».

Приклад: список всіх співробітників, що мають телефони.

SELECT Викладачі.*

FROM Викладачі

WHERE Телефон IS NOT NULL;

Приклад: вибрати всіх студентів, які не складали іспитів.

SELECT Прізвище, Іспит

FROM Сесія

WHERE Оцінка is NULL;

• **Предикати існуван**ня EXIST і не існування NOT EXIST. Ці предикати використовуються з вбудованими під запитами.

Для побудови складніших запитів предикати можуть бути побудовані з використанням логічних операторів And, OR, Not.

Обчислення у виразах виконуються за наступними правилами:

1. Вирази обчислюються зліва направо.

Першими обчислюються вирази в дужках.

3. Оператори NOT обчислюються до виконання операторів AND або OR.

4. Оператори AND обчислюються до виконання операторів OR.

5. Для усунення яких-небудь неоднозначностей рекомендується використовувати дужки.

Приклад: складні умови пошуку.

Перерахувати всіх викладачів - доцентів кафедри Фізики.

SELECT Викладачі.* //вибірка всіх полів з таблиці Викладачі

FROM Викладачі

WHERE Кафедра= «Фізики» and Посада = «Доцент»;

Все подальші розділи оператора SELECT є необов'язковими

2.7 Пропозиція ORDER BY

Сортування здійснюється за допомогою пропозиції ORDER BY.

Сортування результатів ORDER BY <Список полий>

У загальному випадку результати запитів не впорядковані певним чином.

За умовчанням сортування встановлене за збільшенням ASC для сортування по убуванню DESC. Можливе сортування по декількох полях. При цьому сортування по полях виконується в тій послідовності, в якій ці поля вказані в пропозиції ORDER BY

Приклад: відсортувати список студентів по убуванню.

SELECT Студенті.*,

FROM Студенті

ORDER BY Студенті.Прізвище DESC;

Приклад: впорядкування результатів запиту по декількох полях із зростанням або убуванням (ключові слова ASC, DESC):

В результаті отримаємо таблицю, в якого рядка йдуть в порядку зростання значення поля DNUM, а рядки, з однаковим значенням DNUM йдуть в порядку убування значення поля VOLUME

SELECT PD.PNUM, PD.DNUM, PD.VOLUME

FROM PD

ORDER BY DNUM ASC, VOLUME DESC;

При використанні впорядковування по декількох стовпцях спочатку упорядковуються значення в стовпці, вказаному першим в пропозиції SELECT, і так далі по порядку. При цьому можна отримати різне відображення результуючої таблиці.

2.8 Пропозиція GROUP BY

При необхідності послідовного відображення кортежів з однаковим значенням деякого атрибуту використовується групування кортежів поодинці або декільком атрибутам.

Групування результатів [GROUP BY <Список полів результату >].

При угрупуванні вся безліч кортежів відношення розбивається на групи, в які збираються кортежі, що мають однакові значення відповідних атрибутів, заданих в списку угрупування.

Якщо задано декілька атрибутів групування, то спочатку кортежі гриппуют по першому стовпцю, потім по другому і т.д. Найчастіше групування використовуються спільно з агрегатними функціями.

Відповідно до стандарту ISO всі імена стовпців перерахованих в списку SELECT повинні бути присутніми і в пропозиції GROUP BY, окрім імен стовбців, використовуваних в агрегатній функції. У пропозиції GROUP BY можуть бути присутніми імена стовпців, отсутствующих в списку SELECT. Якщо спільно з пропозицією GROUP BY використовується пропозиція WHERE, спочатку

здійснюється вибірка значень що задовольняють умовам в пропозиції WHERE, а потім вони групуються.

Приклад: застосування групування результатів запиту.

Створити список студентів з вказівкою оцінки по дисциплінах.

SELECT Іспит.Іспит, Студенті. Прізвище, Сесія.Оцінка

FROM Студенті INNER JOIN (Іспит INNER JOIN Сесія ON Іспит.Код = Сесія.[Код іспиту]) ON Студенті.[Код студента] = Сесія.[Код студента]

GROUP BY Іспит.Іспит, Студенті.Призвище, Сесія.Оцінка;

2.9 Застосування агрегатних функцій

У SQL додані додаткові функції, які дозволяють обчислювати узагальнені групові значення. Для застосування агрегатних функцій передбачається попередня операція угрупування. Агрегатні функції повертають одне значення для всієї групи.

При використанні агрегатних функцій в запитах з угрупуванням все п дані групованого стовпця стискаються в один рядок, для якого обчислюється значення агрегатної функції. У такому запиті після виконання оператору SELECT для кожної окремої групи створюється підсумковий рядок, для якій обчислюється задана агрегатна функція. При цьому кожен елемент списку SELECT повинен мати єдине значення для всієї групи.

Можна застосовувати агрегатні функції також і без операції попереднього угрупування, в цьому випадку все відношення розглядається як одна група і для цієї групи можна обчислити одне значення на групу.

Агрегатні функції можуть застосовуватися як у виразі виведення результатів рядка SELECT, так і у виразі умови обробки сформованих груп HAVING. В цьому випадку кожна агрегатна функція обчислюється для кожної окремої групи. Значення, отримані при обчисленні агрегатних функцій, можуть бути використані для виведення відповідних результатів або для умови от-бора груп.

Стандарт ISO передбачає 5 агрегатних функцій:

COUNT - підрахунок кількості рядків або непорожніх значень полий, які вибрав запит. Застосовується для числових і нечислових полів.

Приклад: визначити кількість студентів, що складали іспит по кожній дисциплине

SELECT P1.Дисциплина, Count(*) AS Кількість

FROM P1
GROUP BY P1.Дисциплина
ORDER by P1.Дисциплина;

Приклад: визначити число успішно складених іспитів

SELECT COUNT(*) as Кількість
FROM Сесія
WHERE Оцінка > 2;

Приклад: визначити студентів тих, що отримали двійки на сесії.

SELECT DISTINCT Сесія.[Код студента], Count(Сесія.Оцінка) AS [Кількість]
FROM Сесія
WHERE (((Сесія.Оцінка)=2))
GROUP BY Сесія.[Код студента];

SUM - визначається сума всіх вибраних значень даного поля, використовується тільки для числових полів

Приклад: визначити кількість кожного товару на складі

SELECT Найменування, sum(Кількість) as разом
FROM склад
GROUP BY Найменування;

AVG - середньоарифметичне значення всіх вибраних значень даного поля, застосовується тільки для числових полів.

Приклад: визначити середню ціну кожного товару на складі

SELECT Найменування, avg(ціна) as ср_ціна
FROM склад
GROUP BY Найменування;

MIN - визначається найменше зі всіх вибраних значень даного поля, застосовується для числових і нечислових полів

Приклад: визначити мінімальну ціну кожного товару на складі

SELECT Товар, min(ціна) as min_ціна
FROM склад
GROUP BY Товар;

MAX - найбільше зі всіх вибраних значень даного поля, застосовується для числових і нечислових полів

Всі ці функції працюють із значеннями в єдиному стовпці таблиці і повертають єдине значення.

Агрегатні функції використовуються подібно до імен полий в операторові SELECT, але з одним виключенням: вони беруть ім'я поля як аргумент. З функціями SUM і AVG можуть використовуватися

тільки числові поля. З функціями COUNT, MAX і MIN можуть використовуватися як числові, так і символьні поля. При користуванні з символьними полями MAX і MIN транслюватимуть їх в еквівалент стрічок ASCII коду і обробляти в алфавітному порядку.

Окрім функції COUNT (*) при обчисленні результату спочатку виключаються порожні рядки, після чого вказана операція застосовується тільки до конкретних значень стовпця, що залишилися. Варіант COUNT (*) є особливим випадком функції COUNT (*) - в цьому випадку підраховуються всі рядки в результуючій таблиці, незалежно від того чи є порожні, дублюючі або інші значення.

Якщо до застосування агрегатної функції необхідно виключити дублюючі значення, слід перед ім'ям стовпця у визначенні функції використовувати ключове слово DISTING. Це слово не може бути використане з функціями MAX і MIN. Проте його використання перед функціями SUM і AVG може привести до зміни результату. Крім того ключове поле DISTING може використовуватись в запиті тільки один раз.

Агрегатні функції можуть бути використані тільки в списку пропозиції SELECT або HAVING. Якщо список в SELECT містить агрегатні функції, і в тексті запиту немає пропозиції GROUP BY, то жоден з елементів списку пропозиції SELECT не може містити посилань на стовпці, крім випадку, коли це посилання є аргументів агрегатної функції.

Приклад: Визначити кількість всіх студентів, що здавали сесію і середній бал на сесії

```
SELECT COUNT(DISTINCT, Код_Студента) as Кількість, AVG(Оцінка) as CP_бал
FROM Сесія
WHERE Оцінка IS NOT NULL;
```

Приклад: Отримати загальну, максимальну, мінімальну і середню кількості деталей, що поставляються (ключові слова SUM, MAX, MIN, AVG):

```
SELECT
 SUM(PD.VOLUME) AS SM
 MAX(PD.VOLUME) AS MX
 MIN(PD.VOLUME) AS MN
 AVG(PD.VOLUME) AS AV
 FROM PD;
```

Агрегатні функції не можна використовувати в пропозиції WHERE, тому що предикати оцінюються в термінах одиночного рядка, а агрегатні функції - в термінах груп рядків.

2.10 Пропозиція HAVING

У пропозиції HAVING задаються обмеження на виконання групування.

[HAVING <Предикат-умова для групи>]

Пропозиція HAVING використовується спільно з пропозицією GROUP BY для завдання обмежень на відбір груп, які будуть поміщені в вихідну таблицю. Т.ч. пропозиція HAVING використовується для фільтрації груп, на відміну від пропозиції WHERE, яка використовується для фільтрації рядків.

Відповідно до стандарту ISO всі імена стовпців пропозиції HAVING повинні бути присутніми в пропозиції GROUP BY або застосовуватися в агрегатній функції.

Приклад: визначити групи, в яких по одній дисципліні на іспитах отримано більше однієї двійки

```
SELECT R2.Группа
FROM R1, R2
WHERE Rl.ПІБ = R2.ФИО AND Rl.Оцінка = 2
GROUP BY R2.Группа, R1.Дисциплина
HAVING count(*)> 1
```

Приклад: Отримати номери деталей, сумарна кількість яких, що поставляється, перевершує 400 (ключове слово HAVING.):

```
SELECT
 PD.DNUM,
SUM(PD.VOLUME) AS SM
GROUP BY PD.DNUM
HAVING SUM(PD.VOLUME)> 400;
```

Зауваження. Умова, що сумарна кількість, що поставляється, повинна бути більше 400 не може бути сформульовано в пропозиції WHERE, оскільки в цьому розділі не можна використовувати агрегатні функції. Умови, що використовують агрегатні функції повинні бути розміщені в спеціальному розділі HAVING:

2.11 Вкладені запити

Зовнішнього оператора SELECT використовує результат внутрішнього запиту для визначення змісту остаточного результату всієї операції. Внутрішні за-проси можуть бути поміщені пропозиція

WHERE і HAVING, FROM зовнішнього оператора SELECT, крім того, внутрішні запити можуть використовуватися в опера-торах INSERT, UPDATE, DELETE.

Існує три типи внутрішніх запитів:

- скалярний вкладений запит повертає значення, вибирані з перетину одного стовпа з одним рядком, - єдине значення. В принципі, скалярний вкладений запит може використовуватися скрізь, де необхідно набути єдиного значення.
- рядковий вкладений запит повертає значення декількох стовпців таблиці, але у вигляді єдиного рядка. Рядковий вкладений запит може використовуватися скрізь, де застосовується конструктор рядкових значень.
- табличний вкладений запит повертає значення одного або більш за стовпці таблиці, розміщені більш ніж в одному рядку. Табличний вкладений запит може використовуватись скрізь, де допускається указувати таблицю.

вкладений запит є інструмент створення тимчасової таблиці, зміст якою витягується і обробляється зовнішнім оператором. Вкладений запит може указуватися безпосередньо після оператора порівняння (=, <і т.д.) в пропозиціях WHERE і HAVING. Текст вкладений запит повинен бути поміщений в дужки.

Приклад: отримати список постачальників, статус яких менше максимального статусу в таблиці постачальників (порівняння з вкладеним запитом):

```
SELECT *
 FROM P
 WHERE P.STATYS <
 (SELECT MAX(P.STATUS)
 FROM P);
```

Оскільки поле P.STATUS порівнюється з результатом вкладеного запиту, то вкладений запит має бути сформульований так, щоб повертати єдине значення.

Результат виконання запиту буде еквівалентний результату наступної послідовності дій:

1. Виконати один раз вкладений запит і отримати максимальне значення статусу.

2. Переглянути таблицю постачальників P, кожного разу порівнюючи значення статуса постачальника з результатом вкладеного запиту, і відібрати тільки ті рядки, в яких статус менше максимального.

Використання вкладеного запиту з агрегатними функціями

Приклад: Скласти список всіх співробітників, які одержують зарплату вище середньою по підприємству з вказівкою, на скільки ця зарплата більше середньою.

```
SELECT Код_працівника, ФІО, Зарплата- (SELECT avg(зарплата) FROM Штат)
FROM Штат
WHERE Зарплата >
  (SELECT avg(зарплата)
  FROM Штат);
```

В цьому випадку необхідно використовувати вкладений запит, тому що не можна використовувати агрегатні функції в пропозиції WHERE.

Приклад: знайти студентів, що мають максимальний середній бал на сесії

```
SELECT Сесія.[Код студента], Avg(Сесія.Оцінка) AS [Avg-оцінка]
FROM Сесія
GROUP BY Сесія.[Код студента]
HAVING Avg(Сесія.Оцінка>=All (SELECT Avg(Сесія.Оцінка)
FROM Сесія
GROUP BY Сесія.[Код студента]);
```

В цьому випадку необхідно використовувати вкладений запит, тому що не можна використовувати як аргумент агрегатної функції нічого окрім поля таблиці.

Використання вкладеного запиту з предикатом IN.

Використовується у разі, коли вкладений запит, тому що не можна використовувати як аргумент агрегатної функції нічого окрім поля таблиці.

повертає таблицю

Приклад: отримати список постачальників, що поставляють деталь номер 2:

```
SELECT *
 FROM P
 WHERE P.PNUM IN
```

```
(SELECT DISTINCT PD.PNUM
FROM PD
WHERE PD.DNUM = 2);
```

В даному випадку вкладений запит може повертати таблицю, яка містить декілька рядків.

Результат виконання запиту буде еквівалентний результату наступної послідовності дій:

1. Виконати один раз вкладений запит, тому що не можна використовувати як аргумент агрегатної функції нічого окрім поля таблиці.

і отримати список номерів Поставників, що поставляють деталь номер 2.

2. переглянути таблицю постачальників P, при цьому кожного разу перевіряти, чи міститься номер постачальника в результаті вкладений запит, тому що не можна використовувати як аргумент агрегатної функції нічого окрім поля таблиці.

Використання предиката EXIST.

Приклад: отримати список постачальників, що поставляють деталь номер 2.

```
SELECT *
FROM P
WHERE PD EXIST
(SELECT *
FROM PD
WHERE
PD.PNUM = P.PNUM AND PD.DNUM = 2);
```

Результат виконання запиту буде еквівалентний результату наступної послідовності дій:

1. переглянути таблицю постачальників P, кожного разу виконуючи вкладений запит з новим значенням номера постачальника з таблиці P.

2. У результат запиту включити тільки ті рядки з таблиці постачальників, для яких вкладений запит повернув непорожню безліч рядків.

На відміну від двох попередніх прикладів, вкладений запит містить параметр (зовнішнє посилання), що передається з основного запиту, - номер постачальника P.PNUM. Такі вкладені запити називаються корельованими (correlated). Зовнішнє посилання може

приймати різні значення для кожного рядка-кандидата, якій оцінюється за допомогою вкладеного запиту, тому вкладений запит повинен виконуватися наново для кожного рядка, що відбирається в основному запиті. Такі вкладений запити характерні для предиката EXIST, але можуть бути використані і в інших вкладених запитах.

Використання предиката NOT EXIST.

Приклад: отримати список постачальників, що не поставляють деталь номер 2.

```
SELECT *
 FROM P
 WHERE NOT EXIST
 (SELECT *
 FROM PD
 WHERE
PD.PNUM = P.PNUM AND
 PD.DNUM = 2);
```

Також як і в попередньому прикладі, тут використовується корельований вкладений запит. Відмінність в тому, що в основному запиті будуть відібрані ті рядки з таблиці постачальників, для яких вкладений запит не містить жодного рядка.

Приклад: отримати імена постачальників, що поставляють всі деталі.

```
SELECT DISTINCT PNAME
 FROM P
 WHERE NOT EXIST
 (SELECT *
 FROM D
 WHERE NOT EXIST
 (SELECT *
 FROM PD
 WHERE
 PD.DNUM = D.DNUM AND
 PD.PNUM = P.PNUM));
```

Даний запит містить два вкладені вкладених запита і реалізує реляційну операцію ділення відносин.

Самий внутрішній вкладений запит використовує два параметри (D.DNUM, P.PNUM) і має наступний сенс: відібрати всі рядки, що містять дані про постачання постачальника з номером PNUM деталі з

номером DNUM. Вираз NOT EXIST говорить про те, що даний постачальник не поставляє зазначену деталь. Зовнішній до нього вкладений запит, що визначається параметром P.PNUM, має сенс: відібрати список деталей, які не поставляються постачальником PNUM. Вираз NOT EXIST говорить про те, що для постачальника з номером PNUM не повинно бути деталей, які не поставлялися б цим поставником. Це в точності означає, що в зовнішньому запиті відбираються тільки поставники, що поставляють всі деталі.

Використання вкладеного запиту з перевіркою рівності

Приклад: скласти список персоналу, що працює у філіалі, розташованому за адресою ул.Горького, 5

```
SELECT *
FROM Штат
WHERE Код_Філіалу=(SELECT Код_Філіалу
                   FROM Філіал
                   WHERE адреса =’ул.Горького,5’);
```

У внутрішньому запиті визначається номер філіалу, розташованого по адре-су. У зовнішньому підзапиті визначається список працівників даного філіалу.

Спільно з вкладеним запитом можна використовувати предикат EXISTS, який повертає істину, якщо результат вкладеного запиту не порожній.

Ключові слова ANY і ALL

Ключові слова ANY і ALL можуть використовуватися з вкладеними запитами, які повертають один стовпець чисел. Якщо у вкладений запит передуватиме ключове слово ALL, умова вважається виконаною, якщо воно виконується для всіх значень в результуючому стовпці запиту. Якщо вкладеному запиту передує ключове слово ANY, умова вважається виконаною, якщо воно виконується хоч би для одного значення в результуючому стовпці запиту. Якщо результатом виконання вкладеного запиту буде порожнє значення, то для ALL умова вважається виконаною, а для ANY - ні.

Використання ключового слова ALL

Приклад: знайти всіх працівників, чия зарплата більше зарплати будь-якого працівника філіалу 13.

```
SELECT *
FROM Штат
```

```
WHERE Зарплата> All
  (SELECT Латка
  FROM Штат
  WHERE Филиал=’13’);
```

2. 13 Багатотабличні запити

Для об'єднання в результуючій таблиці стовпців з декількох таблиць виконується з'єднання таблиць. У мові SQL операція з'єднання використовується для об'єднання інформації з двох таблиць шляхом утворення двох зв'язаних рядків, вибраних з кожної таблиці.

Для виконання з'єднання досить в пропозиції FROM вказати імена декількох таблиць через кому, а в пропозиції WHERE вказати стовпці, які використовуються для з'єднання таблиць.

Іноді доводиться виконувати запити, в яких таблиця з'єднується сама з собою, або одна таблиця з'єднується двічі з іншою таблицею. При цьому використовуються імена кореляції (аліаси, псевдоніми), які дозволяють розрізняти копії таблиць, що зєднуюься. Імена кореляції вводяться в пропозиції FROM через пропуск після імені таблиці. Імена кореляції повинні використовуватися як префікс перед ім'ям стовпця і відділяються від імені стовпця крапкою. Якщо в запиті вказуються одні і ті ж поля з різних екземплярів однієї таблиці, вони повинні бути перейменовані для усунення неоднозначності в іменуваннях стовпців результуючої таблиці. Визначення імені кореляції діє тільки під час виконання запиту.

Приклад використання псевдонімів: відібрати всі пари постачальників так, щоб перший постачальник в парі мав статус, більший статусу другого поставника:

```
SELECT
 P1.PNAME AS PNAME1
 P1.PSTATUS AS PSTATUS1
 P2.PNAME AS PNAME2
 P2.PSTATUS AS PSTATUS2
FROM
 P P1, P P2
WHERE P1.PSTATUS1 > P2.PSTATUS2
```

Виконання з'єднань

Процедура генерації результуючої таблиці оператора SELECT, який містить з'єднання:

1. Формується декартовий добуток таблиць, вказаних в пропозиції FROM.

2. Якщо в запиті присутня пропозиція WHERE, то після застосування умов пошуку в таблиці залишаються тільки ті рядки, які задовольняють задан-ним умовам, - обмеження декартового добутку.

3. Для кожного рядка, що залишився, визначається значення кожного елементу, зазначеного в списку SELECT, в результаті формується окремий рядок вихідної таблиці

4. Якщо в запиті є DISTINCT, то віддаляються рядки-дублі.

5. Якщо запит містить пропозиція ORDER BY, проводиться впорядкування рядків результуючої таблиці.

Просте з'єднання

Скласти список імен всіх клієнтів, які вже оглянули хоч би один об'єкт нерухомості і висловили свою думку

SELECT r.rno, r.fname, r.lname, v.pno, v.comment

FROM renter r, viewing v

WHERE r.rno= v.r.rno

Оскільки r.rno може бути вибраний з будь-якої таблиці, то перед ним указується ім'я таблиць

Скласти список імен всіх клієнтів, які вже оглянули хоч би один об'єкт нерухомості і висловили свою думку

SELECT r.rno, r.fname, r.lname, v.pno, v.comment

FROM renter r, viewing v

WHERE r.rno= v.r.rno

Оскільки r.rno може бути вибраний з будь-якої таблиці, то перед ним указується ім'я таблиці з якої потрібно вибрати цей атрибут. Для побудови результуючої таблиці вибираються тільки рядки, для яких r.rno= v.r.rno - поєднувані стовпці

Найчастіше багатотабличні запити застосовуються для отримання інформації з таблиць, що мають зв'язок 1:М.

Синтаксис з’єднуваних таблиць

При виконанні операції з'єднання дані з двох таблиць комбінуються з утворенням зв'язаних рядків, в яких значення загального атрибуту рівнозначні.

У пропозиції FROM можна використовувати сполучені таблиці. Хай в результаті деяких операцій виходять таблиці A і B. Такими

операціями можуть бути, наприклад, оператор SELECT або інша сполучена таблиця. Тоді синтаксис сполученої таблиці має наступний вигляд:

Таблиця А [тип з'єднання] JOIN Таблиця В

При цьому можливі наступні варіанти з'єднань:

Перехресне з'єднання ::= таблиця А CROSS JOIN таблиця В.

Природне з'єднання ::= таблиця А [NATURAL] [тип з'єднання] JOIN таблиця В.

З'єднання за допомогою предиката ::= таблиця А [тип з'єднання] JOIN таблиця В ON предикат.

З'єднання за допомогою імен стовпців ::= таблиця А [тип з'єднання] JOIN таблиця В USING (ім'я стовпця.,..).

З'єднання об'єднання ::= таблиця А UNION JOIN таблиця В.

CROSS JOIN - перехресне з'єднання повертає просто декартовий добуток таблиць. Таке з'єднання в пропозиції FROM може бути замінене списком таблиць через кому.

NATURAL JOIN - природне з'єднання проводиться по всіх стовпцях таблиць А і В, що має однакові імена. У результуючу таблицю однакові стовпці вставляються тільки один раз.

JOIN . ON - З'єднання за допомогою предиката сполучає рядки таблиць А і В за допомогою вказаного предиката.

Приклад: знайти середній бал по кожному екзаменаційному предмету.

SELECT Іспит.Іспит AS Іспит, Avg(Сесія.Оцінка) AS Ср_бал

FROM Іспит INNER JOIN Сесія ON Іспит.Код= Сесія.[Код іспиту]

GROUP BY Іспит. Іспит;

JOIN . USING - з'єднання за допомогою імен стовпців сполучає відносини подібно до природного з'єднання по тих загальних стовпцях таблиць А і Б, які вказані в списку USING.

OUTER - ключове слово OUTER (зовнішній) не є обов'язковим, воно не використовується ні в яких операціях з даними.

INNER - тип з'єднання "внутрішнє". Внутрішній тип з'єднання використовується за умовчанням, коли тип з'єднання явно не заданий. У таблицях А і В з'єднуються тільки ті рядки, для яких знайдений збіг.

Приклад: Створити список студентів, що проживають в гуртожитку, з вказівкою всіх даних студентів.

SELECT Студентsи.*, Гуртожиток.Гуртожиток, Гуртожиток.Кімната

FROM Студенти INNER JOIN Гуртожиток ON Студенті.Код_студента = Гуртожиток.[Код студента];

LEFT (OUTER) - Тип з'єднання "ліве (зовнішнє)". Ліве з'єднання таблиць A і У включає всі рядки з лівої таблиці A і ті рядки з правої таблиці B, для яких виявлений однакові значення. Для рядків з таблиці A, для яких не знайдено відповідності в таблиці B, в стовпці з таблиці B заносяться значення NULL.

Приклад: перерахувати філіали компанії і об'єкти, що здаються в оренду, які розташовані в одному і тому ж місті, а також інші філіали компанії.

SELECT b.*, p.*

FROM філіал b LEFT JOIN Оренда a ON b.bcity=a.acity

У результуючу таблицю потрапляють не тільки ті рядки, які знайшли собі пару в таблицях b і p, але і ті рядки з лівої таблиці, що сполучається, які не знайшли відповідності в другій таблиці. У цьому рядку всі поля правої таблиці заповнюються значеннями NULL.

RIGHT (OUTER) - Тип з'єднання "праве (зовнішнє)". Праве з'єднання таблиць A і У включає всі рядки з правої таблиці B і ті рядки з лівої таблиці A, для яких виявлений збіг. Для рядків з таблиці B, для яких не знайдено відповідності в таблиці A заносяться значення NULL.

FULL (OUTER) - Тип з'єднання "повне (зовнішнє)". Це комбінація лівого і правого з'єднань. У повне з'єднання включаються всі рядки з обох таблиць. Для співпадаючих рядків поля заповнюються реальними значеннями, для неспівпадаючих рядків поля заповнюються відповідно до правил лівого і правого з'єднань.

UNION JOIN - З'єднання об'єднання є зворотним по відношенню до внутрішнього з'єднання. Воно включає тільки ті рядки з таблиць A і B, для яких не знайдено збігів. У них використовуються значення NULL для стовпців, отриманих з іншої таблиці. Якщо узяти повне зовнішнє з'єднання і видалити з нього рядки, отримані в результаті внутрішнього з'єднання, то вийде з'єднання об'єднання.

2.14 Комбінування результуючих таблиць

У мові SQL можна використовувати звичайні операції над множинами - об'єднання, перетин, різниця, що дозволяють комбінувати результати виконання два і більш за запити в єдину результуючу таблицю. Таблиці, які можуть комбінуватися за допомогою операцій

над множинами, повинні бути сумісні по з'єднанню, тобто вони повинні мати однакову структуру: однакова кількість стовпців, у відповідних стовпцях повинні розміщуватися дані одного типу.

Стандартом ISO передбачено три операції над безліччю UNION, INTERSECT, EXCEPT

Використання операції UNION- об'єднання

СИНТАКСИС: (Select ..) union (Select..)

Створюється результуюча таблиця, в яку включаються результати об'єднуваних запитів, при цьому об'єднувані запити повинні мати однакові схеми.

Приклад: створити список всіх регіонів, в яких або знаходиться філіал компанії, або є об'єкти, що здаються.

```
(SELECT регіон
FROM філіал
WHERE регіон Is Not Null )
UNION
(SELECT регіон
FROM оренда
WHERE регіон Is Not Null);
```

Використання операції INTERSECT- перетин

Синтаксис: (Select ..) Intersect (Select..)

Створюється результуюча таблиця, в яку включаються тільки ті записи, які є загальними для результатів обох запитів

Приклад: створити список віх міст, в яких є філіали компанії і є об'єкти, що здаються.

```
(SELECT Місто
FROM Філіал )
INTERSECT
(SELECT Місто
FROM Оренда)
```

Приклад: створити список всіх студентів, що проживають в гуртожитку і які отримали на іспитах хоч би одну двійку.

```
SELECT Прізвище
FROM Студенти
WHERE код_студента EXIST
( (SELECT DISTINCT код_студента
FROM Сесія
WHERE Оценка=2)
```

INTERSECT
(SELECT код_студента
FROM Гуртожиток));

Використання операції EXCEPT - РІЗНИЦЯ

Синтаксис: (Select ..) EXCEPT (Select..)

Створюється результуюча таблиця, в яку включаються тільки ті з результатів першого запиту, яких немає в результатах другого запиту.

Приклад: створити список всіх міст, в яких є філіали компанії, і немає об'єктів, що здаються.

(SELECT Місто
FROM Філіал)
EXCEPT
(SELECT Місто
FROM Оренда

Приклад: створити список студентів, які отримали двійки на іспиті і не живуть в гуртожитку.

SELECT Прізвище
FROM Студенти
WHERE код_студента EXIST
((SELECT DISTINCT код_студента
FROM Сесія
WHERE Оцінка=2)
EXCEPT
(SELECT код_студента
FROM Гуртожиток));

Приведений вище запит але без використання оператора EXCEPT.

Приклад:

SELECT DISTINCT Місто
FROM Філіал
WHERE Місто NOT IN
(SELECT Місто
FROM Оренда)

Зміна вмісту бази даних

INSERT - додати рядки в таблицю
UPDATE - змінити рядки в таблиці
DELETE - видалити рядки в таблиці

INSERT - вставка рядків в таблицю

Синтаксис INSERT INTO [схема таблиці]

VALUES (значення атрибутів, що вставляються);

Якщо вставляється текстові значення, то вони указуються в апострофах.

Оператор вставки складається з наступних частин:

1. Пара службових слів INSERT INTO;
2. Схеми відношення - імені таблиці R, списку атрибутів таблиці R в круглих дужках;
3. Службового слова values

Вирази кортежу, поміщеного в круглі дужки списку значень, поодинці на кожен атрибут списку, заданого в схемі відношення RA1,A2.An v1,v2..vn. Якщо заданий список атрибутів не охоплює всі атрибути відношення R, всім компонентам кортежу для атрибутів, відсутнім в списку, привласнюється значення за умовчанням. Звичайно це значення Null, але можливі і інші значення. Якщо список атрибутів у виразі містить всі атрибути відношення, то його можна опускати. Якщо інструкція не містить списку атрибутів, то компонентам кортежу значення, що вставляється, привласнюються в тому порядку, який визначений схемою відношення.

Приклад: вставка одного рядка в таблицю з використанням значень за умовчанням.

```
INSERT INTO
 P (PNUM, PNAME)
 VALUES (4, "Іванов");
```

Приклад: вставка в таблицю декількох рядків, вибраних з іншої таблиці (у таблицю TMP_TABLE вставляються дані про постачальників з таблиці P, що мають номери, великі 2):

```
INSERT INTO
 TMP_TABLE (PNUM, PNAME)
 (SELECT PNUM, PNAME
 FROM P
 WHERE P.PNUM>2);
```

UPDATE - оновлення рядків в таблиці

Синтаксис UPDATE [ім'я таблиці]

SET [атрибут]= [привласнюване значення]

WHERE [умова відбору];

Приклад: оновлення декількох рядків в таблиці.

```
UPDATE P
 SET PNAME = "Хутровиків"
 WHERE P.PNUM = 1;
```

Приклад: оновлення всіх рядків в таблиці.

```
UPDATE P
 SET Salary=salary*1.5
 WHERE position = ‘Менеджер’;
```

Приклад: оновлення декількох рядків в таблиці.

```
UPDATE Гуртожиток
SET Гуртожиток.Гуртожиток = 4, Гуртожиток.Кімната = 46
WHERE Гуртожиток.Общежитие=3 AND Гуртожиток.Комната=22;
```

Приклад: оновлення декількох стовпців в таблиці: перевести Петрова (123) на посаду Менеджера і підвищити йому зарплату в 1,5раза.

```
UPDATE P
 SET Salary=salary*1.5, position = ‘Менеджер’
 WHERE Код_Працівника= 123;
```

DELETE - видалення рядків в таблиці

```
DELETE
FROM [ім'я таблиці]
WHERE [ім'я атрибуту] [предікат- умова] [значення]
```

Приклад: Видалення декількох рядків в таблиці:

```
DELETE
FROM P
WHERE P.PNUM = 1;
```

Приклад: Видалення всіх рядків в таблиці:

```
DELETE
FROM P;
```

Приклад: видалення рядків таблиці з використанням умови

```
DELETE Студенті.*, Гуртожіток.Кімната
FROM Студенті, Гуртожіток
WHERE (((Гуртожіток. Кімната)=[Введіть номер кімнати:])
```

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Коннолли Т., Бегг К., Страчан А., Базы данных. Проектирование, реализация и сопровождение. Теория и практика.- Москва-Санкт-Петербург- Киев: Вильямс, 2006.-1111с.
2. Дейт К. Дж. Введение в системы баз данных.- Киев-Москва: Диалектика, 2008. - 782с.
3. Г. Гарсия-Молина, Дж. Ульман, Дж. Уидом. Системы баз данных. Полный курс.- Москва-Санкт-Петербург -Киев: Вильямс, 2003.- 1082с
4. Д.Кренке. Теория и практика построения баз данных. Москва: Питер, 2005: - 800с– 354с.
5. http://kafinfor.petrsu.ru/mysql/index.htm
6. http://www.site-do.ru/db/sql3.php
7. http://phpclub.ru/mysql/doc
8. http://www.sql.ru/
9. http://www.cyberforum.ru