МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ

Кафедра інформаційних технологій

СКБД MySQL і робота з інтернет-базами даних засобами phpMyAdmin

Методичні вказівки для виконання лабораторних робіт з дисципліни "Створення та опрацювання баз даних"

Одеса 2021

УДК 004.43 План НМВ 2021 р.

Рецензент: к.т.н., доцент Флейта Ю. В.

СКБД MySQL і робота з інтернет базами даних: метод. вказівки для виконання лабораторних робіт роботи з дисципліни "Створення та опрацювання баз даних" / Трофименко О. Г., Прокоп Ю. В., Буката Л.М. Одеса, 2021. 56 с.

Методичні вказівки призначені для студентів при підготовці до лабораторних робіт з дисципліни “Створення та опрацювання баз даних”. Містить теоретичні відомості щодо проєктування та створення баз даних засобами phpMyAdmin. Надано відомості про засоби MySQL при роботі з інтернет базами даних. Зазначено, що популярна реляційна СКБД MySQL працює як сервер для забезпечення багатокористувацького доступу до значної кількості баз даних і завдяки своїй доступності, швидкості та безпеці добре підходить для роботи з БД через інтернет. Розглянуто засоби вебінтерфейсу phpMyAdmin для адміністрування сервера MySQL. Детально описано етапи створення БД, на численних прикладах розглянуто специфіку пошуку та відбору певних даних, різноманітного маніпулюваннями даними у таблицях БД.

СХВАЛЕНО
на засіданні кафедри
інформаційних технологій
і рекомендовано до друку.

Протокол № 6 від 22.01.2021 р.

ЗАТВЕРДЖЕНО
методичною радою університету

Протокол № 1 від 11.03.2021 р.

ЗМІСТ

Лабораторна робота 1

Вставновлення Denver для роботи з phpMyAdmin

Лабораторне завдання

1. Скачати безкоштовну інсталяцію **Denver** за посиланням www.denwer.ru або базову редакцію **Open Server**[1] за доступом https://ospanel.io/download.

2. Встановити програму **Denver** або **Open Server** на своєму ПК, тобто найти скачаний файл з інсталяцією Open Server і запустити його.

При встановленні **Denver** на робочому столі комп'ютера будуть створені ярлики для запуску Денвера – *Start Denver*, перезапуску Денвера *Restart Denver* і зупинки Денвера – *Stop Denver*.

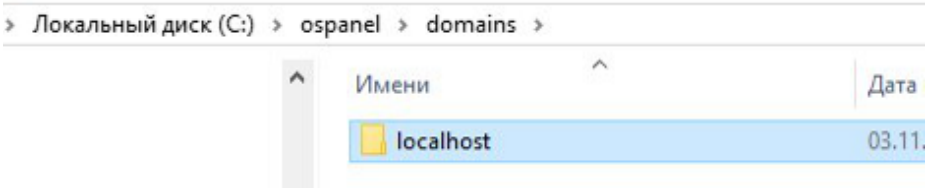
А при встановленні **Open Server** у системному треї (system tray, область повідомлень) потрібно знайти червоний прапорець, клацнути на ньому, вибрати команду *Запустити* і чекати доки прапорець стане зеленим.

3. Запустити **Denver** або **Open Server**.

Для запуску **Denver** треба клацнути ярлик *Start Denver* і відкрити phpMyAdmin у браузері, для цього потрібно:

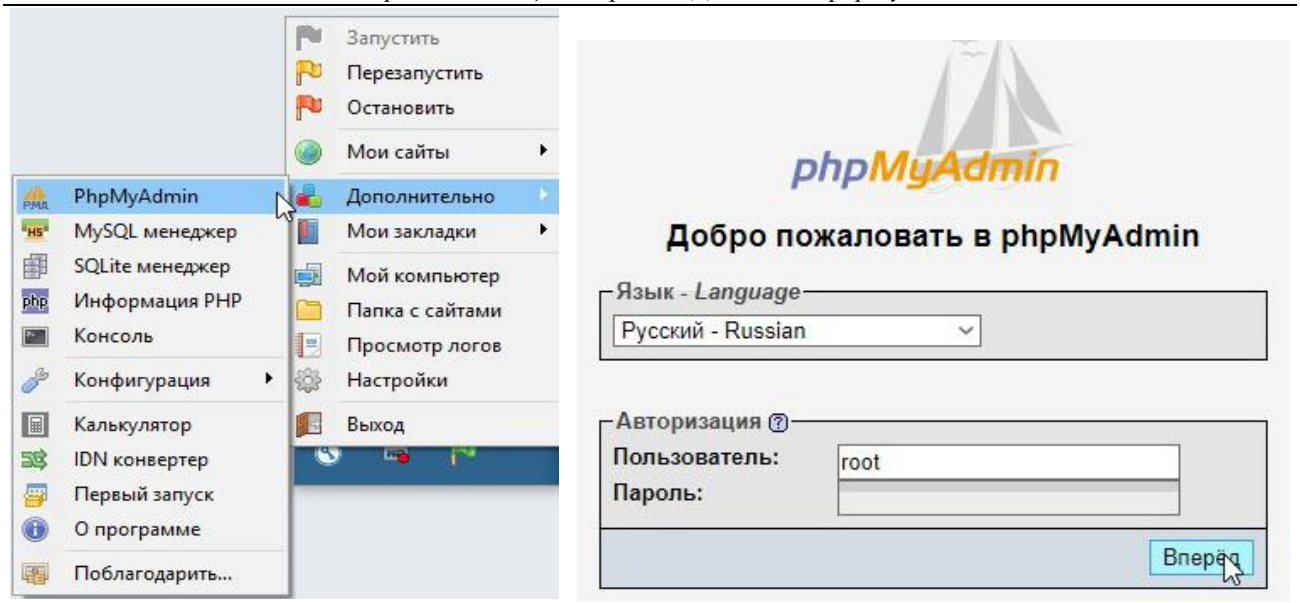
– набрати в адресному рядку **localhost** і натиснути [Enter];
– на вебсторінці у розділі *Тестування Денвера в таблиці* знайти таблицю з посиланням **http://localhost/Tools/phpMyAdmin** та перейти за цим посиланням.

Для запуску **Open Server** вписати в адресному рядку браузера **http://localhost**, щоб з'явилася сторінка Open Server з привітанням. Далі треба натиснути у треї на прапорець, а потім на *Папка з сайтами*, що призведе до відкривання програми *Провідник*. У папці *localhost* треба створити папку (наприклад, *mysite*). Після цього у браузері на цьому комп'ютері він буде доступний за адресою *localhost/mysite*.

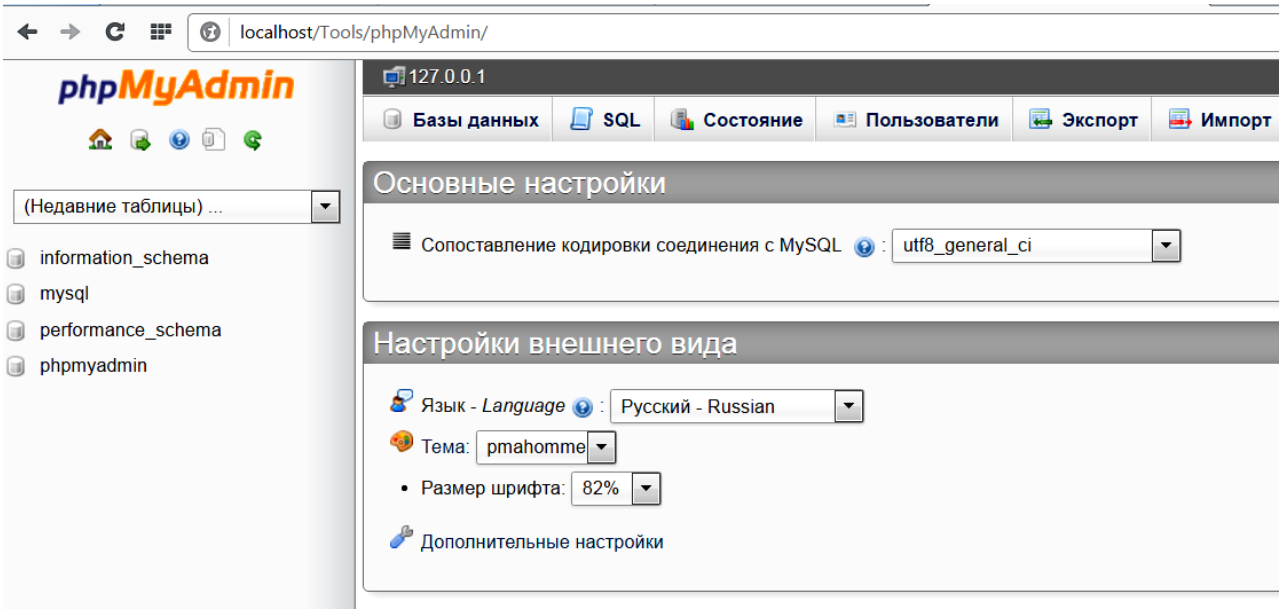


Далі для запуску **Open Server** у треї треба вибрати *Дополнительно* та *phpMyAdmin*. У браузері відкриється стартове вікно phpMyAdmin, в якому потрібно ввести логін *root*, а пароль залишити порожнім.

[1] Безкоштовне скачування триватиме приблизно 60 хвилин, після чого треба знайти скачаний файл з інсталяцією Open Server і запустити його



Якщо все зроблено правильно, незалежно від різновиду вибраної програми Denver або Open Server, з'явиться вікно phpMyAdmin. Це вікно складається з двох частин – фреймів. Лівий фрейм використовують для навігації та вибору поточної бази даних. Правий фрейм – робочий, у ньому редагують структуру таблиць, здійснюють редагування записів та інші операції з вибраною базою. Якщо БД не вибрано, то в правому фреймі відображатимуться загальні функції роботи з цілим сервером MySQL та налаштування phpMyAdmin. Після вибору потрібної бази даних у лівому фреймі відображатиметься список її таблиць, а в правому – ці самі таблиці, але в розширеному вигляді.



4. Перейти на вкладку *Базы данных* і створити БД *Деканат*. Для цього у лівому фреймі треба натиснути *Создать БД*, у правому фреймі ввести назву *Decanat* і кодування *utf-8_general_ci*, після чого натиснути кнопку *Создать*.

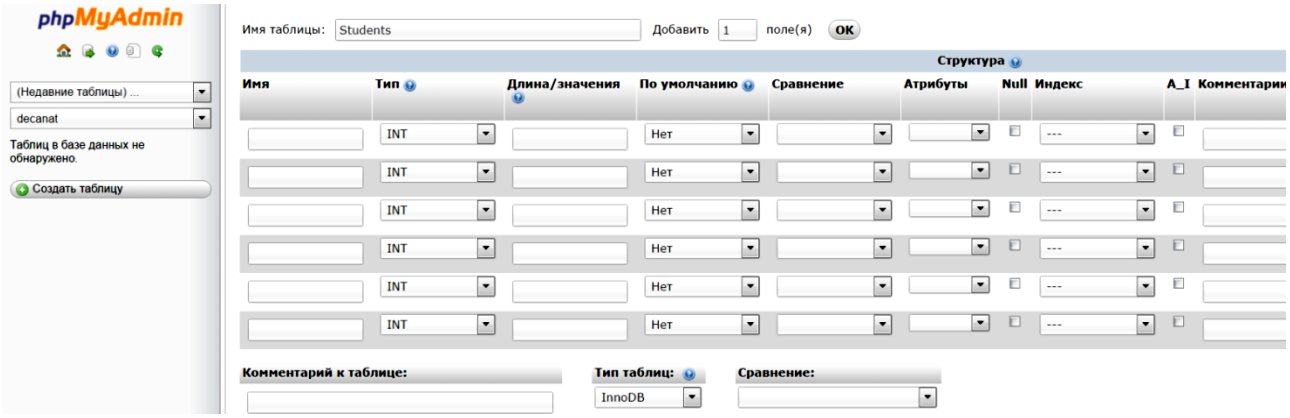
Лабораторна робота 2

Створення таблиць інтернет-бази даних засобами phpMyAdmin

Лабораторне завдання

1. Створити таблицю *students* БД *Деканат* з 11 полів: StudentID, Gradebook, Surname, Name, Patronymic, Sex, Birth_Date, Enter_Date, GroupID, Form_Study, Phone, Address, Student_Info, для кожного з яких задати властивості:

– ім’я поля;
– тип даних;
– довжина;
– усталене значення;
– порівняння (як буде здійснюватися пошук даних);
– атрибути;
– Null (чи може стовпець бути порожнім);
– індекс поля;
– A_I (auto_increment) – чи має стовпець автоматичне прирощення;
– коментар.



Під час створення таблиць варто звертати увагу на такі властивості полів:

– первинний ключ (PRIMARY) з автоінкрементом;

– ширина рядків (якщо вона виявиться недостатньою, то при введенні рядки будуть урізатися);

– можливість набувати значення NULL (для тих полів, де значення може бути відсутнім, наприклад, додаткова інформація про студента (Student_Info) у таблиці students, науковий ступінь (DegreeID) викладача у таблиці teachers або ID завідувача кафедри (HeadID) у таблиці departments);

– унікальність (властивість UNIQUE для табельного номера (Table_Number) викладача у таблиці teachers).

Структура таблиці students має бути такою:

127.0.0.1 » decanat » students

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	StudentID	int(11)			Нет	*Нем*	AUTO_INCREMENT
2	Gradebook	varchar(10)	utf8_general_ci		Нет	*Нем*	
3	Surname	varchar(50)	utf8_general_ci		Нет	*Нем*	
4	Name	varchar(50)	utf8_general_ci		Нет	*Нем*	
5	Patronymic	varchar(50)	utf8_general_ci		Нет	*Нем*	
6	Sex	varchar(1)	utf8_general_ci		Нет	*Нем*	
7	Birth_Date	date			Нет	*Нем*	
8	Enter_Date	date			Нет	*Нем*	
9	GroupID	int(11)			Нет	*Нем*	
10	Form_Study	varchar(1)	utf8_general_ci		Нет	*Нем*	
11	Phone	varchar(10)	utf8_general_ci		Да	*NULL*	
12	Address	varchar(100)	utf8_general_ci		Да	*NULL*	
13	Student_Info	text	utf8_general_ci		Да	*NULL*	

Усі створені поля можна редагувати, видаляти, змінювати їхні властивості (первинний ключ, унікальне значення, індекс тощо) або переглядати унікальні значення полів. Такі самі операції можна здійснювати над групою виділених (відмічених галочкою) полів таблиці. У таблицю можна додавати нові поля в кінець, початок або після вказаного поля. Зокрема, щоб додати поле GroupID, слід на сторінці *Структура* під переліком полів вказати, що поле додається після поля Enter_Date.

Коли формування структури і властивостей полів таблиці завершено, треба натиснути кнопку *Зберегти*.

2. Створити у БД *Деканат* таблицю disciplines (дисципліни) такої структури:

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс	A_I	Комментарии
DisciplineID	INT		Нет			☐	PRIMARY	☑	Код дисципліни
Discipline	VARCHAR	50	Нет			☐	---	☐	Назва дисципліни
Hours	INT		Нет			☐	---	☐	Обсяг годин
Result	CHAR	1	Нет			☐	---	☐	Екзамен (е) або з.

3. Створити таблицю departments (кафедри):

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс	A_I	Комментарии
DepartentID	INT		Нет			☐	PRIMARY	☑	Код дисципліни
Department	VARCHAR	50	Нет			☐	---	☐	Назва дисципліни
Phone	VARCHAR	10	Нет			☐	---	☐	Обсяг годин
HeadID	INT		Нет			☐	INDEX	☐	Екзамен (е) або з

4. Аналогічно створити решту таблиць БД Деканат (відповідно до схеми, показаної на рис. 1).

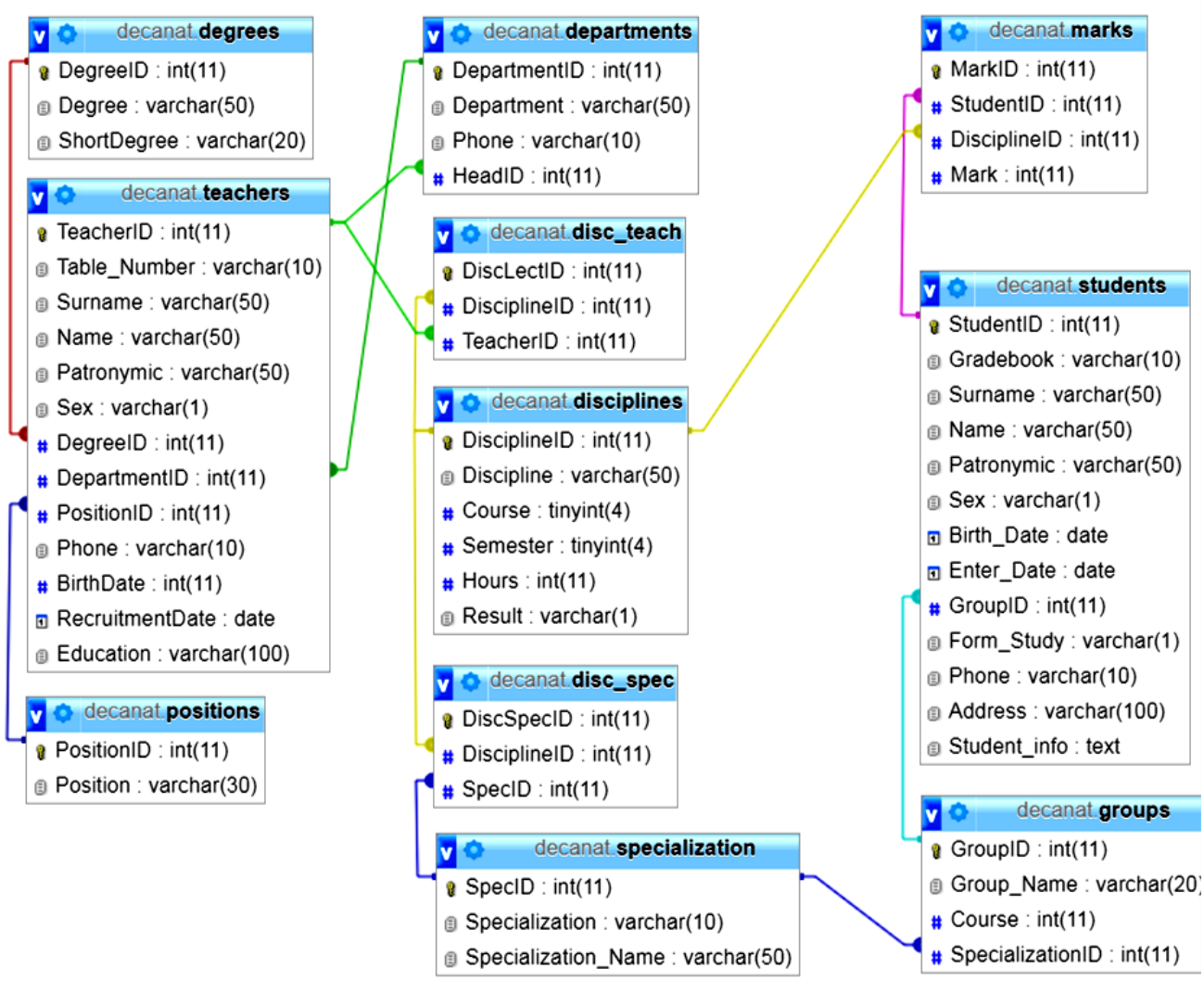


Рисунок 1 – Структура (схема зв'язків) бази даних *Деканат* у phpMyAdmin

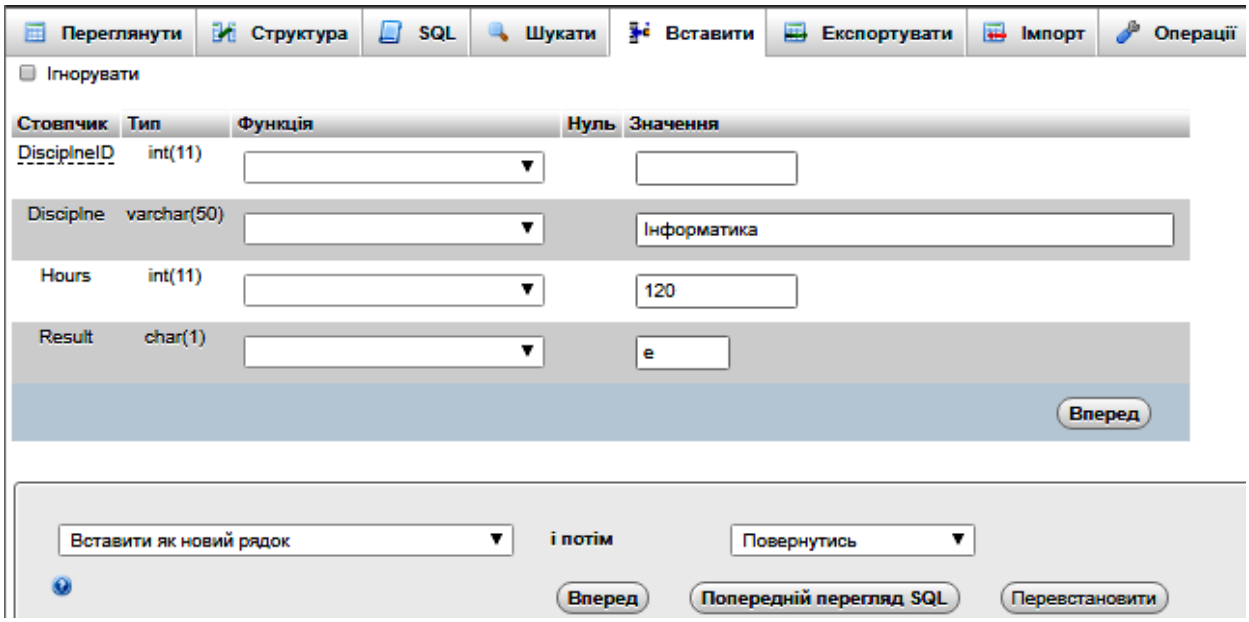
Лабораторна робота 3

Заповнення даними таблиць БД

Теоретичні відомості

Після створення таблиці students і встановлення властивостей її полів потрібно заповнити таблицю даними. Зробити це можна кількома способами: вставленням окремих записів, імпортуванням даних або через SQL-запит.

Якщо треба додати в таблицю кілька записів, то найпростіше скористатися меню *Вставити*. На екрані з'явиться форма для внесення нових записів.



Якщо скористатися вкладкою SQL, то для додавання в таблицю записів можна скористатись запитом, подібним до такого:

```
INSERT INTO `disciplines` (`DisciplneID`, `Discipline`, `Hours`, `Result`)
VALUES ('', 'Історія України', '90', 'з'), ('', 'Економіка', '90', 'з')
```

Найпоширеніший спосіб додавання даних в таблиці – імпортування, приміром, із таблиць Microsoft Excel.

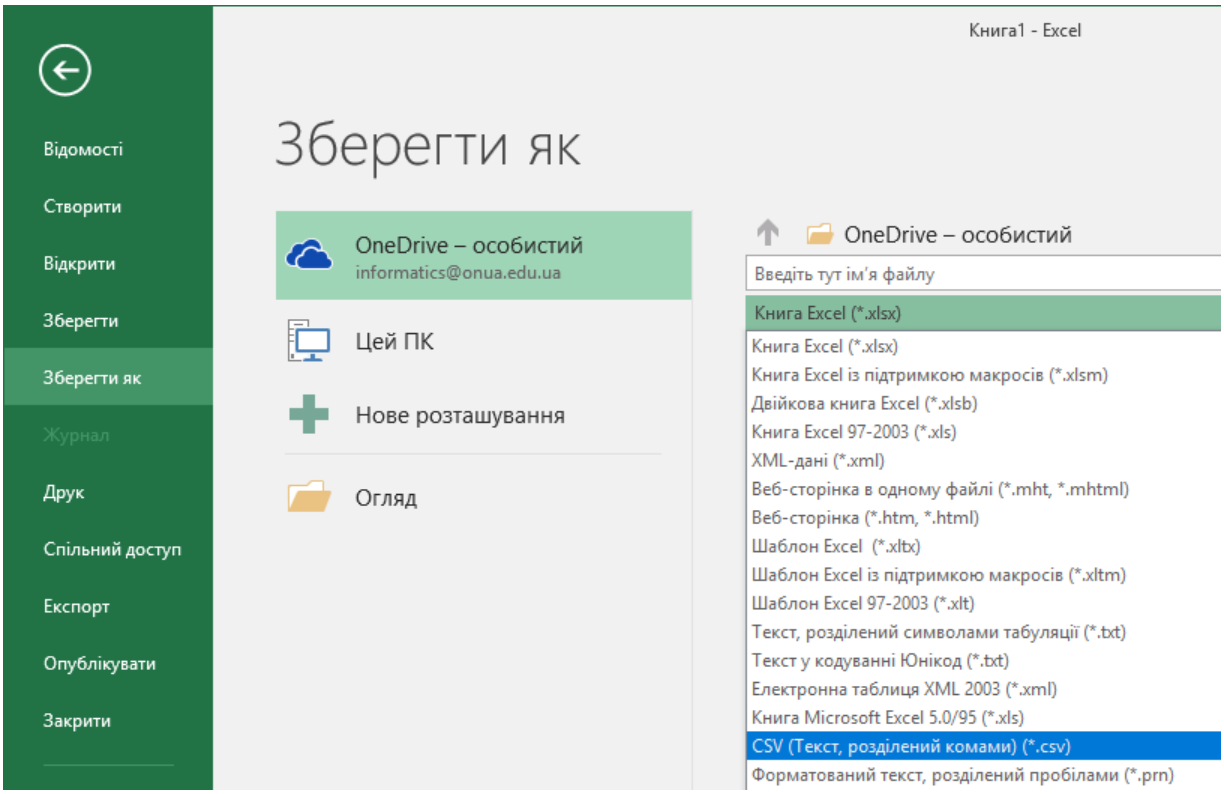
Лабораторне завдання

1. Створити в Microsoft Excel таблицю з даними про студентів, при цьому для полів із датами задати текстовий формат, щоб уникнути помилок невідповідності формату під час імпорту. Поля StudentID та Student_Info заповнити значеннями NULL.

Таблиця *students* (поле *Gradebook* слід поки що залишити порожнім)

StudentID	Gradebook	Surname	Name	Patronymic	Sex	Birth_Date	Enter_Date	GroupID	Form_Study	Phone	Address	Student_Info
1	001-14	Петров	Микола	Сергійович	ч	1996-11-19	2014-08-22	5	б	0937651234	Одеса, вул. Буніна, 5, кв. 7	NULL
2	002-14	Шевчук	Іван	Федорович	ч	1997-01-14	2014-08-22	5	б	0945671287	Одеса, вул. Корольова, 75, кв. 108	NULL
3	003-15	Марченко	Тетяна	Олегівна	ж	1998-03-10	2015-08-22	3	к	0632347658	Одеса, вул. Доброволь-ського 123, кв. 53	NULL
4	004-15	Сердюк	Борис	Іванович	ч	1998-05-06	2015-08-22	4	б	0674562345	Одеса, вул. Старопор-тофранківська, 71а	NULL
5	005-14	Федоренко	Ольга	Павлівна	ж	1996-12-13	2014-08-22	13	к	0664321654	Одеса, вул. Інглезі, 15, кв. 4	NULL
6	006-16	Романчук	Зоя	Петрівна	ж	1999-03-15	2016-08-22	9	б	067453621	Одеса, вул. Манежна, 42	NULL
7	007-16	Погоріла	Лариса	Лазарівна	ж	1999-06-17	2016-08-22	9	б	0503456798	Одеса, вул. Польська, 4	NULL
8	008-15	Рябчук	Михайло	Вадимович	ч	1998-09-22	2015-08-22	1	б	0917652349	Одеса, вул. Шклярука, 8	батько бере участь в ООС
9	009-16	Стоколос	Ірина	Петрівна	ж	1999-01-10	2016-08-22	2	к	0507781265	Одеса, вул. Конна, 68а	NULL
10	010-15	Усенко	Марія	Сергіївна	ж	1998-11-07	2015-08-22	1	б	0674458273	Одеса, вул. Садова, 71а	NULL
11	011-14	Кошмак	Раїса	Григорівна	ж	1997-05-12	2014-08-28	13	б	0638762351	Одеса, вул. І.Рабіна,2, кв.3	NULL
12	012-14	Шульга	Тетяна	Петрівна	ж	1996-12-03	2014-08-28	14	к	0674389245	Одеса, вул. Ніщинського, 4	NULL
13	013-16	Волошенюк	Любов	Романівна	ж	1999-07-08	2016-08-30	9	к	0487432745	Одеса, вул. Садова, 35, кв. 2	NULL
14	014-15	Чава	Олександр	Леонідович	ч	1997-10-23	2015-08-28	3	б	0947162536	Одеса, вул. Манежна, 42	NULL
15	015-15	Мінченко	Станіслав	Миколайович	ч	1998-07-05	2015-08-28	4	б	0937694387	Одеса, вул. Старопор-тофранківська, 20, кв. 3	батько бере участь в ООС
16	016-15	Хітрук	Микола	Петрович	ч	1997-11-28	2015-08-28	3	б	0637547612	Одеса, вул. Пастера, 68а	NULL
17	017-14	Пінчук	Афанасій	Миколайович	ч	1996-02-24	2014-08-28	13	б	0947654321	Одеса, вул. Садова, 3, кв. 7	NULL
18	018-15	Сергієнко	Василь	Дмитрович	ч	1997-09-18	2015-08-28	10	к	0487431576	Одеса, вул. Старопор-тофранківська, 71а	дитина з багато-дітної родини
19	019-16	Кучма	Світлана	Андріївна	ж	1999-05-23	2016-08-28	9	б	0675432160	Одеса, вул. Садова, 21, кв. 5	NULL
20	020-14	Шевченко	Олеся	Василівна	ж	1996-12-20	2014-08-28	5	б	0675438765	Одеса, вул. Гаванна, 20	NULL

2. Зберегти файл у форматі *CSV (Текст, розділений комами)(*.csv)*.



3. Після цього треба найти цей файл у *Провіднику* і відкрити його за допомогою *Блокнота* (*Notepad*). Далі зберегти цей файл з **кодуванням UTF-8** командою *Файл* / *Зберегти як* / *Кодування*. Якщо ж пропустити цей крок, то літери кирилиці в базу імпортовано не буде.

4. Для імпортування щойно створеної таблиці повернутися до phpMyAdmin, вибрати таблицю students, натиснувши на вкладці *Імпорт* кнопку *Огляд* та вибравши формат CSV. Далі скористатися *LOAD DATA* і кнопкою *ОК*. Після цього перейти на вкладку *Обзор* (*Огляд*) для перегляданння вмісту таблиці students.

Більш складний варіант імпортування передбачає ситуацію, коли Excel-таблиця не відповідає структурі таблиці бази даних й інформація потребує розподілу за декількома таблицями і/або вставленню ID замість текстових значень. Це, а також особливості заповнення таблиць за допомогою SQL-запитів розглянемо дещо пізніше.

5. Подібним чином створити та імпортувати таблицю **teachers**.

6. Також створити та імпортувати решту таблиць БД *Деканат.*

Таблиця **spec_groups** поки що залишається порожньою, вона буде заповнена значеннями пізніше за допомогою SQL.

Таблиця *teachers*

TeacherID	Table_Number	Surname	Name	Patronymic	Sex	DegreeID	DepartmentID	PositionID	Phone	BirthDate	RecruitmentDate	Education
1	1234	Ткаченко	Василь	Петрович	ч	1	1	2	3225566	1965-04-10	1990-09-01	ОНПУ, ФАОТ, 1987
2	5678	Макарова	Оксана	Миколаївна	ж	6	2	1	3334455	1960-01-11	2000-09-01	ОНУ, ІМЕМ, 1982
3	5436	Зайцев	Іван	Павлович	ч	3	3	1	7651234	1966-03-23	1996-09-01	ОНУ, історичний факультет, 1998
4	9876	Матвієнко	Олексій	Михайлович	ч	5	1	2	7543245	1976-12-04	2006-02-01	ОНПУ, ФАОТ, 1999
5	3874	Васильєв	Сергій	Іванович	ч	7	1	3	7771234	1970-11-05	2010-09-01	ОНПУ, ФАОТ, 1992
6	7543	Львівський	Олександр	Михайлович	ч	4	4	2	7159933	1944-12-24	1994-09-01	ОНУ, ІМЕМ, 1982
7	8765	Пивовар	Ірина	Юхимівна	ж	2	2	1	7056644	1965-07-08	2015-09-01	ОНУ, ІМЕМ, 1982
8	7765	Караваєва	Тетяна	Євгенівна	ж	2	1	1	7031265	1954-02-12	1994-09-01	ОНУ, ІМЕМ, 1976
9	4918	Іванов	Василь	Миколайович	ч	1	1	1	7013456	1972-10-12	2001-08-25	КНПУ, мех-мат., 1994
10	7104	Шевченко	Тетяна	Іванівна	ж	7	3	4	7157842	1983-02-27	2005-09-01	ОНУ, історичний факультет, 2005
11	6098	Щукіна	Олена	Михайлівна	ж	7	1	4	7771203	1980-07-20	2002-08-27	ОНПУ, ІКС, 1987
12	1543	Петренко	Олена	Петрівна	ж	2	2	3	2223344	1958-02-21	1998-09-01	ОНУ, ІМЕМ, 1982
13	7654	Сидорчук	Микола	Васильович	ч	1	1	1	7154367	1971-08-15	2001-03-01	ОНПУ, ФАОТ, 1993

Таблиця *groups*

GroupID	Group_Name	Course	SpecializationID
1	ІПЗ-1.01	1	121
2	ІПЗ-1.02	1	121
3	ІПЗ-2.01	2	121
4	ІПЗ-2.02	2	121
5	ІПЗ-3.01	3	121
6	ІПЗ-3.02	3	121
7	ІК-1.01	1	172
8	ІК-1.02	1	172
9	ІК-1.03	1	172
10	ІК-2.01	2	172
11	ІК-2.02	2	172
12	ІК-2.03	2	172
13	ІК-3.01	3	172
14	ІК-3.02	3	172
15	ІК-3.03	3	172
16	ІК-4.01	4	172
17	ІК-4.02	4	172
18	ІК-4.03	4	172

Таблиця *disc_teach*

DiscLectID	DisciplineID	TeacherID
1	1	1
2	3	1
3	9	2
4	4	2
5	5	3
6	3	4
7	10	5
8	8	5
9	4	12
10	3	13
11	10	13
12	6	6
13	2	7
14	1	8
15	3	8
16	10	9
17	7	9
18	5	10
22	1	11
23	10	11
24	7	11

Таблиця *disc_spec*

DiscSpecID	DisciplineID	SpecID
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	2
7	6	1
8	6	2
9	7	1
10	8	1
11	8	2
12	9	2
13	8	2
14	9	2
31	17	2
32	18	2
33	10	2

Таблиця *specialization*

SpecID	Specialization	Specialization_Name
1	ІПЗ	Інженерія програмного забезпечення
2	ІК	Телекомунікації

Таблиця *positions*

PositionID	Position
1	доцент
2	професор
3	старший викладач
4	викладач
5	лаборант
6	інженер

Таблиця *departments*

Depart-mentID	Department	Phone	HeadID
1	Інформаційних технологій	7775544	4
2	Вищої математики	7775533	2
3	Історії	7775511	3
4	Економіки	7775588	8

Таблиця *disciplines*

DisciplineID	Discipline	Course	Semester	Hours	Result
1	Алгоритмізація і програмування	1	1	120	е
2	Математичний аналіз	1	2	60	е
3	Об'єктно-орієнтоване програмування	1	2	100	е
4	Лінійна алгебра і геометрія	1	1	80	е
5	Історія України	1	1	50	з
6	Економіка	2	1	60	з
7	Комп'ютерна графіка	2	2	60	з
8	Бази даних	3	1	120	е
9	Вища математика	1	1	100	е
10	Інформатика	1	1	100	е

Таблиця *degrees*

DegreeID	Degree	ShortDegree
1	Кандидат технічних наук	к.т.н.
2	Кандидат фізико-математичних наук	к.ф.-м.н.
3	Кандидат історичних наук	к.і.н.
4	Кандидат економічних наук	к.е.н.
5	Доктор технічних наук	д.т.н.
6	Доктор фізико-математичних наук	д.ф.-м.н.

Таблиця marks

MarkID	StudentID	DisciplineID	Mark
1	1	1	98
2	1	2	84
3	1	3	95
4	1	4	70
5	1	5	97
6	1	6	64
7	1	7	68
8	1	8	74
9	2	1	88
10	2	2	76
11	2	3	100
12	2	4	88
13	2	5	72
14	2	6	88
15	2	7	81
16	2	8	88
17	3	1	72
18	3	2	61
19	3	3	83
20	3	4	63
21	3	5	60
22	3	6	62
23	3	7	69
24	3	8	74
25	4	1	98
26	4	2	71
27	4	3	74
28	4	4	97
29	4	5	69
30	4	6	99
31	4	7	67
32	4	8	63
33	5	6	72
34	5	6	64
35	5	8	65
36	5	9	71
37	5	8	77
38	5	9	77
39	5	17	72
40	5	18	99
41	6	6	98
42	6	6	86
43	6	8	65
44	6	9	95
45	6	8	66
46	6	9	68
47	6	17	63
48	6	18	61
49	7	6	75
50	7	6	72
51	7	8	67
52	7	9	100
53	7	8	73
54	7	9	71
55	7	17	61
56	7	18	84
57	8	1	85
58	8	2	99
59	8	3	98
60	8	4	63
61	8	5	99
62	8	6	72
63	8	7	61
64	8	8	63
65	9	1	89
66	9	2	66
67	9	3	73
68	9	4	69
69	9	5	97
70	9	6	72
71	9	7	63
72	9	8	85
73	10	1	98
74	10	2	63
75	10	3	99
76	10	4	76
77	10	5	61
78	10	6	60
79	10	7	97
80	10	8	80
81	11	6	61

82	11	6	71
83	11	8	62
84	11	9	98
85	11	8	73
86	11	9	91
87	11	17	64
88	11	18	92
89	12	6	86
90	12	6	90
91	12	8	100
92	12	9	80
93	12	8	80
94	12	9	66
95	12	17	96
96	12	18	67
97	13	6	65
98	13	6	79
99	13	8	89
100	13	9	72
101	13	8	99
102	13	9	69
103	13	17	65
104	13	18	71
105	14	1	86
106	14	2	95
107	14	3	62
108	14	4	67
109	14	5	90
110	14	6	85
111	14	7	71
112	14	8	98
113	15	1	85
114	15	2	82
115	15	3	61
116	15	4	93
117	15	5	99
118	15	6	79
119	15	7	73
120	15	8	84
121	16	1	98
122	16	2	80
123	16	3	67
124	16	4	94
125	16	5	82
126	16	6	86
127	16	7	65
128	16	8	78
129	17	6	63
130	17	6	82
131	17	8	74
132	17	9	93
133	17	8	79
134	17	9	61
135	17	17	88
136	17	18	100
137	18	6	75
138	18	6	84
139	18	8	93
140	18	9	67
141	18	8	97
142	18	9	78
143	18	17	76
144	18	18	73
145	19	6	77
146	19	6	74
147	19	8	89
148	19	9	75
149	19	8	71
150	19	9	82
151	19	17	74
152	19	18	78
153	20	1	92
154	20	2	86
155	20	3	84
156	20	4	74
157	20	5	95
158	20	6	72
159	20	7	81
160	20	8	69

Лабораторна робота 4

Зв'язування таблиць в інтернет-БД

Лабораторне завдання

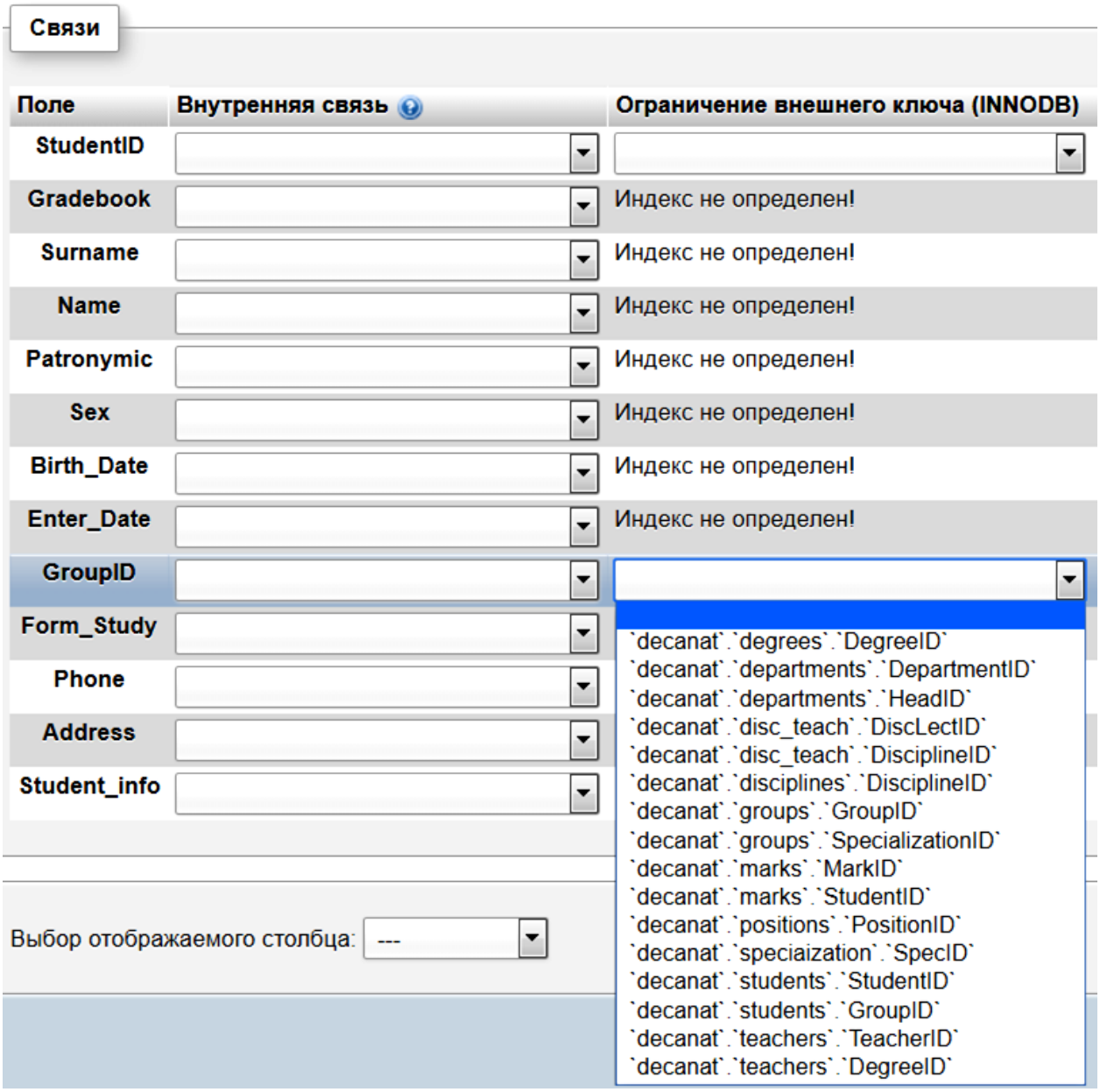
1. Встановити зв'язки між таблицею з даними про студентів і таблицею з відомостями про групи. Зв'язувати ці таблиці за полем GroupID. Для цього, працюючи з таблицею students, перейти на вкладку *Структура*, клацнути на *Связи* під списком полів.

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
☐	1	**StudentID**	int(11)			Нет	*Нет*	AUTO_INCREMENT
☐	2	**Gradebook**	varchar(10)	utf8_general_ci		Нет	*Нет*	
☐	3	**Surname**	varchar(50)	utf8_general_ci		Нет	*Нет*	
☐	4	**Name**	varchar(50)	utf8_general_ci		Нет	*Нет*	
☐	5	**Patronymic**	varchar(50)	utf8_general_ci		Нет	*Нет*	
☐	6	**Sex**	varchar(1)	utf8_general_ci		Нет	*Нет*	
☐	7	**Birth_Date**	date			Нет	*Нет*	
☐	8	**Enter_Date**	date			Нет	*Нет*	
☐	9	**GroupID**	int(11)			Нет	*Нет*	
☐	10	**Form_Study**	varchar(1)	utf8_general_ci		Нет	*Нет*	
☐	11	**Phone**	varchar(10)	utf8_general_ci		Да	*NULL*	
☐	12	**Address**	varchar(100)	utf8_general_ci		Да	*NULL*	
☐	13	**Student_info**	text	utf8_general_ci		Да	*NULL*	

Отметить все / Снять выделение *С отмеченными:* Обзор Изменить Удалить

Версия для печати Связи Анализ структуры таблицы Отслеживать таблицу

Для зв'язування таблиць треба встановити обмеження зовнішнього ключа. Зв'язувати можна поля, для яких встановлено індекс. У нашому прикладі для зв'язування за полем GroupID треба з випадного списку вибрати відповідне поле таблиці Groups:



Далі треба вказати, що відбуватиметься при видаленні запису (певної групи) з таблиці Groups, тобто задати обмеження цілісності даних. Можливі варіанти:

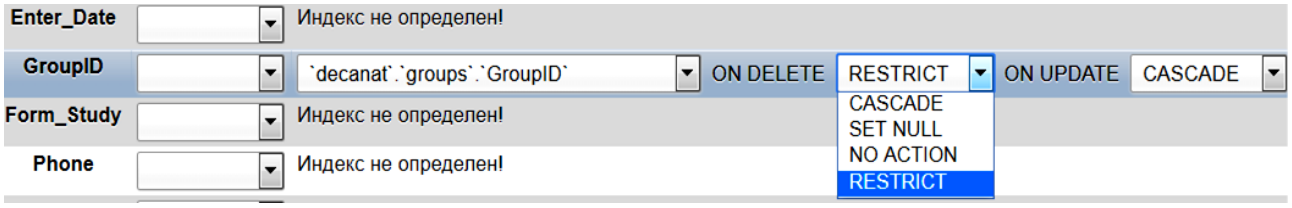
– CASCADE – каскадне видалення всіх пов'язаних з цією групою записів;

– SET NULL – встановлення значення NULL замість ID видаленої групи;

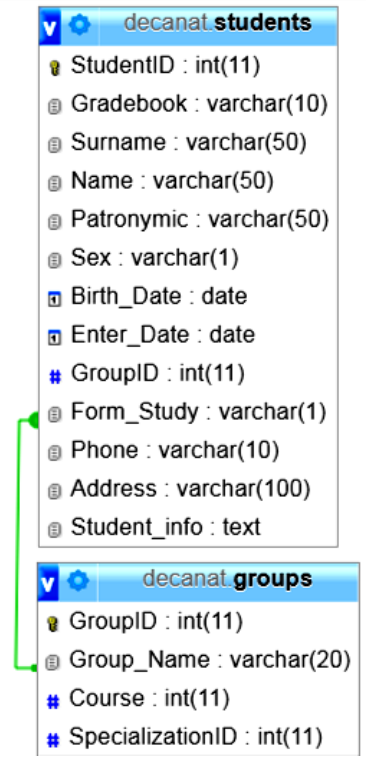
– NO ACTION – нічого не змінювати;

– RESTRICT – заборонити видалення групи, якщо у таблиці students є пов'язані з нею записи.

Оскільки неможливо видалити групу, якщо в ній навчаються студенти, то виберемо RESTRICT. Так само вибирається дія у разі оновлення значення ID групи, однак тоді потрібне каскадне оновлення даних (CASCADE).



Далі вгорі вікна (над рядком меню) треба натиснути на назву бази decanat. У меню праворуч у пункті *Ще* вибрати *Дизайнер*, після чого у вікні з'являться всі таблиці, дві з яких (students і groups) будуть з'єднані зв'язком.

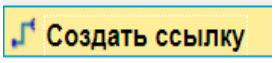


Варто розташувати таблиці у вікні так, щоб вони не накладалися одна на одну (наприклад, як на рис. 1 у лабораторній роботі 7), і натиснути кнопку з дискетою *Зберегти*.

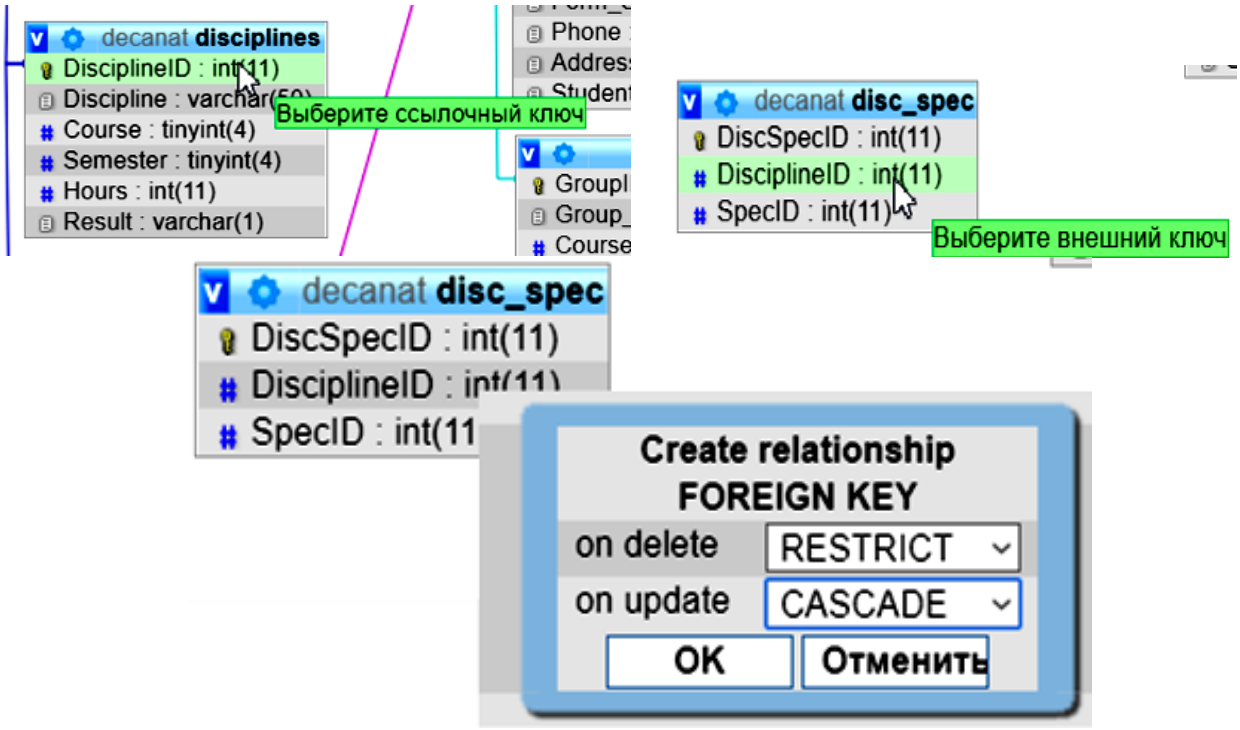
Подібно до зазначеного порядку треба встановити зв'язки між іншими таблицями. Поля, за якими відбувається зв'язування, зазвичай мають у назві складник ID. Наприклад, таблицю teachers треба пов'язати з трьома таблицями: з таблицею degrees за полем DegreeID, з таблицею departments за полем DepartmentID і з таблицею positions за полем PositionID.

Перевірити правильність зв'язування таблиць (відповідно до рис. 1 у лабораторній роботі 7) можна у *Дизайнері*.

Поле	Внутренняя связь	Ограничение внешнего ключа (INNODB)
TeacherID		`decanat`.`disc_teach`.`TeacherID` ON DELETE RESTRICT ON UPDATE CASCADE
Table_Number		Индекс не определен!
Surname		Индекс не определен!
Name		Индекс не определен!
Patronymic		Индекс не определен!
Sex		Индекс не определен!
DegreeID		`decanat`.`degrees`.`DegreeID` ON DELETE RESTRICT ON UPDATE CASCADE
DepartmentID		`decanat`.`departments`.`DepartmentID` ON DELETE RESTRICT ON UPDATE CASCADE
PositionID		`decanat`.`positions`.`PositionID` ON DELETE RESTRICT ON UPDATE CASCADE
Phone		Индекс не определен!
BirthDate		Индекс не определен!
RecruitmentDate		Индекс не определен!
Education		Индекс не определен!

Існує й інший спосіб встановлення зв'язків між таблицями – безпосередньо у *Дизайнері*. Для цього треба натиснути 

, вибрати спочатку поле з первинним (PRIMARY) ключем, а потім – поле із зовнішнім ключем. Зокрема, щоб пов'язати таблиці disciplines та disc_spec за полем DisciplineID, слід спочатку натиснути на поле DisciplineID таблиці disciplines, а потім – на однойменне поле таблиці disc_spec і встановити обмеження для зовнішнього ключа у віконці, яке з'явиться.



Для подальшої роботи зі створеною БД *Деканат* треба заповнити даними таблиці groups, departments, degrees, positions, teachers, disciplines. Зверніть увагу на послідовність заповнення таблиць та відповідність ID. Так, перш ніж заповнювати таблицю teachers, треба заповнити таблиці positions та departments (останній стовпець HeadID спочатку залишити порожнім – його можна буде заповнити після таблиці teachers).

Лабораторна робота 5
Формування запитів на відбір даних

Теоретичні відомості

1 Пошук даних у таблиці

Специфіку пошуку певних даних у таблицях БД розглянемо на прикладі пошуку студентів, які навчаються за контрактом. Для цього треба вибрати таблицю **students** і перейти на вкладку *Пошук*. Для поля **Form_Study** вставити оператор **LIKE** і праворуч ввести літеру 'к', після чого натиснути *ОК*.

Унаслідок на екран у вигляді таблиці буде виведено список студентів-контрактників. Вгорі над результатами пошуку сформовано повідомлення про кількість найдених записів і час виконання запиту. Нижче – код запиту мовою SQL (phpMyAdmin згенерує його автоматично):

```
SELECT *
FROM ‘students’
WHERE ‘Form_Study’ LIKE 'к'
LIMIT 0 , 30;
```

Означає цей оператор SELECT таке:

"ВИБРАТИ всі поля
з ТАБЛИЦІ students,
ДЕ поле Form_Study дорівнює 'к'.
ОБМЕЖИТИ кількість найдених записів значенням 30".

Імена таблиць і полів в одинарних лапках вставляються автоматично. Вони не є обов’язковими елементами синтаксису і можуть бути видалені.

Результат пошуку:

StudentID Код студента	Gradebook Номер залікової книжки	Surname Прізвище	Name Ім'я	Patronymic По батькові	Sex	Birth_Date Дата народження	Enter_Date Дата поступлення	GroupID Код групи	Form_Study Бюджет (б) чи контракт (к)	Phone	Address Адреса реєстрації	Student_Info Інформація
9	9-16	Стокопос	Ірина	Петрівна	ж	1999-01-10	2016-08-22	2	к	0507781265	Одеса, вул. Новосельського, 68а	NULL
5	5-14	Федоренко	Ольга	Павлівна	ж	1996-12-13	2014-08-22	13	к	0664321654	Одеса, вул. Інглезі, 15, кв. 46	NULL
3	3-15	Марченко	Тетяна	Олегівна	ж	1998-03-10	2015-08-22	3	к	0632347658	Одеса, вул. Добровольського 123, кв. 53	NULL
18	18-15	Сергієнко	Василь	Дмитрович	ч	1997-09-18	2015-08-28	10	к	0487431576	Одеса, вул. Старопортофранківська, 71а	NULL
13	13-16	Вопощенюк	Любов	Романівна	ж	1999-07-08	2016-08-30	9	к	0487432745	Одеса, вул. Малиновського, 35, кв.2	Батько бере участь в ООС
12	12-14	Шульга	Тетяна	Глібівна	ж	1996-12-03	2014-08-28	14	к	0674389245	Одеса, вул. Композитора Ніщинського, 4	NULL

Розглянемо пошук студентів чоловічої статі, які навчаються на бюджеті. Для цього треба знову скористатися вкладкою *Пошук*, де задати дві умови: для поля Sex ввести значення 'ч', а для поля Form_Study – значення 'б'.

Унаслідок пошуку сформується такий SQL-запит:

```
SELECT * FROM ‘students’
WHERE ‘Sex’ LIKE 'ч' AND ‘Form_Study’ LIKE 'б'
LIMIT 0 , 30;
```

Тут обидві умови пов’язані за допомогою AND (логічне "І").

Результат пошуку:

StudentID Код студента	Gradebook Номер залікової книжки	Surname Прізвище	Name Ім'я	Patronymic По батькові	Sex	Birth_Date Дата народження	Enter_Date Дата поступлення	GroupID Код групи	Form_Study Бюджет (б) чи контракт (к)	Phone	Address Адреса реєстрації	Student_Info Інформація
8	8-15	Рябчук	Михайло	Вадимович	ч	1998-09-22	2015-08-22	1	б	0917652349	Одеса, вул. Піонерська, 20	Батько бере участь в ООС
4	4-15	Сердюк	Борис	Іванович	ч	1998-05-06	2015-08-22	4	б	0674562345	Одеса, вул. Старопортофранківська, 71а	NULL
2	2-14	Шевчук	Іван	Федорович	ч	1997-01-14	2014-08-22	5	б	0945671287	Одеса, вул. Корольова, 75, кв. 108	NULL
17	17-14	Пінчук	Афанасій	Миколайович	ч	1996-02-24	2014-08-28	13	б	0947654321	Одеса, вул. Пантелеймонівська, 23, кв. 7	дитина з багатодітної родини
16	16-15	Хітрук	Микопай	Володимирович	ч	1997-11-28	2015-08-28	3	б	0637547612	Одеса, вул. Новосельського, 68а	NULL
15	15-15	Мінченко	Станіслав	Миколайович	ч	1998-07-05	2015-08-28	4	б	0937694387	Одеса, вул. Старопортофранківська, 20, кв. 3	NULL
14	14-15	Чава	Олександр	Леонідович	ч	1997-10-23	2015-08-28	3	б	0947162536	Одеса, вул. Манежна, 42	NULL
1	1-14	Петров	Микопа	Сергійович	ч	1996-11-19	2014-08-22	5	б	0937651234	Одеса, вул. Пушкінська 25, кв. 17	NULL

Третім прикладом пошуку розглянемо відбір у таблиці degrees усіх вчених, наукові ступені яких розпочинаються зі слова "доктор". Для цього у полі Degree треба задати оператор LIKE і ввести значення доктор%. Тут знак % означає довільний текст після слова доктор.

SQL-запит матиме такий вигляд:

```
SELECT  * FROM  ‘degrees’
WHERE  ‘Degree’ LIKE 'доктор%';
```

Результат пошуку:

DegreeID	Degree науковий ступень	ShortDegree Коротка форма запису наукового ступеня
5	Доктор технічних наук	д.т.н.
6	Доктор фізико-математичних наук	д.ф.м.н.

Отже, відбір даних з однієї таблиці можна здійснювати за допомогою пошуку в phpMyAdmin. Хоча того ж результату можна досягти написанням SQL-запиту вручну, що і здебільшого використовується на практиці.

2 Відбір даних засобами MySQL

Написання SQL-запитів дозволяє формувати не лише прості вибірки, а й складні запити для відбору з кількох зв’язаних таблиць. Як приклад, виберемо прізвища та імена студентів 1998 року народження за допомогою SQL-запиту. Для цього перейдемо на вкладку SQL і у вікні запиту введемо такий код:

```
SELECT  Surname, Name, Birth_Date
FROM  students
WHERE  YEAR(Birth_Date)=1998;
```

Зверніть увагу, що у вікні SQL-запиту вже написана заготовка, яку треба лише доповнити, а назви полів можна вибрати з правої частини вікна подвійним клацанням.

Використана у запиті функція YEAR виділятиме рік з дати народження.

Surname Прізвище	Name Ім'я	Birth_Date Дата народження
Марченко	Тетяна	1998-03-10
Сердюк	Борис	1998-05-06
Рябчук	Михайло	1998-09-22
Усенко	Марія	1998-11-07
Мінченко	Станіслав	1998-07-05

Відсортуємо здобуті результати за абеткою. Для цього треба натиснути на *Швидка правка* або *Змінити* під кодом запиту і дописати ORDER BY:

```
SELECT Surname, Name, Birth_Date
FROM Students
WHERE YEAR(Birth_Date)=1998
ORDER BY Surname;
```

Surname Прізвище	Name Ім'я	Birth_Date Дата народження
Мінченко	Станіслав	1998-07-05
Марченко	Тетяна	1998-03-10
Рябчук	Михайло	1998-09-22
Сердюк	Борис	1998-05-06
Усенко	Марія	1998-11-07

Запит із сортуванням результатів за датою народження виглядає так:

```
SELECT Surname, Name, Birth_Date
FROM Students
WHERE YEAR(Birth_Date)=1998
ORDER BY Birth_Date;
```

Surname Прізвище	Name Ім'я	Birth_Date Дата народження
Марченко	Тетяна	1998-03-10
Сердюк	Борис	1998-05-06
Мінченко	Станіслав	1998-07-05
Рябчук	Михайло	1998-09-22
Усенко	Марія	1998-11-07

Визначимо кількість студентів жіночої статі. Для цього скористаємося функцією підрахунку кількості COUNT:

```
SELECT COUNT(*)
FROM students
WHERE Sex = 'ж';
```

Результат: 11.

Як вже зазначалось, здебільшого на практиці виникає потреба формування запитів для відбору даних з декількох таблиць БД водночас. Це можуть бути зв'язані або не зв'язані між собою таблиці. У таких запитах дуже важливо уникати неоднозначності, а тому треба вказувати повні імена полів, які фактично складаються з імені таблиці та імені стовпця, розділених крапкою. Приклади імен: students.Surname, teachers.DegreeID, groups.Group_Name.

Наприклад, запит на виведення даних про студентів (прізвище, ім'я, курс, назва групи) може мати вигляд.

```
SELECT students.Surname, students.Name,
    groups.Course, groups.Group_Name
FROM students, groups
WHERE students.GroupID =
    groups.GroupID;
```

Surname	Name	Course	Group_Name
Петров	Микола	3	ІПЗ-3.01
Шевчук	Іван	3	ІПЗ-3.01
Марченко	Тетяна	2	ІПЗ-2.01
Сердюк	Борис	2	ІПЗ-2.02
Федоренко	Ольга	3	ІК-3.01
Романчук	Зоя	1	ІК-1.03
Погорі л а	Лариса	1	ІК-1.03
Рябчук	Михайло	1	ІПЗ-1.01
Стоколос	Ірина	1	ІПЗ-1.02
Усенко	Марія	1	ІПЗ-1.01
Кошмак	Раїса	3	ІК-3.01
Шульга	Тетяна	3	ІК-3.02
Волошенюк	Любов	1	ІК-1.03
Чава	Олександр	2	ІПЗ-2.01
Мінченко	Станіслав	2	ІПЗ-2.02
Хітрук	Микола	2	ІПЗ-2.01
Пінчук	Афанасій	3	ІК-3.01
Сергієнко	Василь	2	ІК-2.01
Кучма	Світлана	1	ІК-1.03
Шевченко	Олеся	3	ІПЗ-3.01

При виконанні запиту для кожної комбінації двох рядків таблиць перевірятиметься умова, зазначена у WHERE. Якщо ця комбінація задовольнятиме умову, то вона вибиратиметься.

Для скорочення запису запиту імена таблиць позначають скорочено псевдонімами:

```
SELECT s.Surname, s.Name, g.Course, g.Group_Name
FROM students AS s, groups AS g WHERE s.GroupID = g.GroupID;
```

У цьому прикладі таблицю students позначено псевдонімом s, а таблицю groups – псевдонімом g. До речі, слово AS перед псевдонімом можна опускати:

```
SELECT s.Surname, s.Name, g.Course, g.Group_Name
FROM students s, groups g
WHERE s.GroupID = g.GroupID;
```

Ще одним способом звертання до декількох таблиць у запиті є використання конструкції JOIN ON. Як приклад, відберемо всіх викладачів із зазначенням наукового ступеня, створивши такий запит:

```
SELECT t.TeacherID, t.Surname, t.Name, t.Patronymic, d.ShortDegree
FROM teachers AS t
JOIN degrees d ON t.DegreeID=d.DegreeID;
```

TeacherID	Surname	Name	Patronymic	ShortDegree
1	Ткаченко	Василь	Петрович	к.т.н.
2	Макарова	Оксана	Миколаївна	д.ф-м.н.
3	Зайцев	Іван	Павлович	к.і.н.
4	Матвієнко	Олексій	Михайлович	д.т.н.
6	Львівський	Олександр	Михайлович	к.е.н.
7	Пивовар	Ірина	Юхимівна	к.ф-м.н.
8	Караваєва	Тетяна	Євгеновна	к.ф-м.н.
9	Іванов	Василь	Миколайович	к.т.н.
12	Петренко	Олена	Петрівна	к.ф-м.н.
13	Сидорчук	Микола	Васильович	к.т.н.

У цьому прикладі дані вибираються з двох таблиць – teachers і degrees, які об'єднані між собою за допомогою індексу DegreeID. Унаслідок запиту були втрачені записи щодо викладачів, які не мають наукового ступеня.

Якщо мета запиту – викладачі, зокрема без ступеня, то запит має виглядати так:

```
SELECT t.TeacherID, t.Surname, t.Name, t.Patronymic, d.ShortDegree
FROM teachers AS t
LEFT JOIN degrees d ON t.DegreeID = d.DegreeID;
```

LEFT JOIN дозволяє вивести записи з таблиці teachers, для яких немає відповідності в таблиці degrees.

Об'єднувати можна і більше двох таблиць в одному запиті. Наприклад, виведемо відомості про викладачів із зазначенням наукового ступеня та посади:

```
SELECT t.TeacherID, t.Surname, t.Name, t.Patronymic, d.ShortDegree, p.Position
FROM teachers AS t
LEFT JOIN degrees d ON t.DegreeID = d.DegreeID
LEFT JOIN positions p ON t.PositionID = p.PositionID;
```

У цьому запиті беруть участь три таблиці: teachers, degrees та positions. Дві останні пов'язані з таблицею викладачів через поля DegreeID і PositionID.

TeacherID	Surname	Name	Patronymic	ShortDegree	Position
1	Ткаченко	Василь	Петрович	к.т.н.	професор
2	Макарова	Оксана	Миколаївна	д.ф-м.н.	доцент
3	Зайцев	Іван	Павлович	к.і.н.	доцент
4	Матвієнко	Олексій	Михайлович	д.т.н.	професор
5	Васильєв	Сергій	Іванович		старший викладач
6	Львівський	Олександр	Михайлович	к.е.н.	професор
7	Пивовар	Ірина	Юхимівна	к.ф-м.н.	доцент
8	Караваєва	Тетяна	Євген івна	к.ф-м.н.	доцент
9	Іванов	Василь	Миколайович	к.т.н.	доцент
10	Шевченко	Тетяна	Іванівна		викладач
11	Щукіна	Олена	Михайлівна		викладач
12	Петренко	Олена	Петрівна	к.ф-м.н.	старший викладач
13	Сидорчук	Микола	Васильович	к.т.н.	доцент

3 Робота над рядками в MySQL

MySQL надає широкий спектр функцій для роботи з рядками, типовий для багатьох мов програмування. Опис цих функцій можна знайти у документації MySQL. Зупинимося лише на деяких з них.

CHAR_LENGTH(рядок) – повертає довжину рядка (кількість символів).

CONCAT(рядок1, рядок2, ...) – повертає рядок, що складається зі зчеплених аргументів. Повертає NULL, якщо будь-який з аргументів дорівнює NULL. Числовий аргумент перетворюється на еквівалентну рядкову форму. Наприклад,

```
CONCAT('My', 'S', 'QL');
```

поверне 'MySQL'.

LEFT(рядок, кількість символів) – повертає вказану кількість символів від початку рядка (зліва). Наприклад, оператор

```
SELECT LEFT('MySQL', 3);
```

поверне 'MyS'.

Як приклад роботи з рядками сформуємо запит для відбору даних про докторів наук. Для цього вибиратимемо поля з двох таблиць teachers і degrees, а саме: Surname, Name, Patronymic, Degree. Тут можливі різні підходи до роботи з рядками. Розглянемо три з них.

1 спосіб: для порівняння наукових ступенів викладачів візьмемо перші шість символів поля Degree зі словом "доктор":

```
SELECT t.Surname, t.Name, t.Patronymic, d.Degree
FROM teachers t
JOIN degrees d ON t.DegreeID=d.DegreeID
WHERE LEFT(d.Degree, 6) LIKE "доктор";
```

Surname	Name	Patronymic	Degree
Матвієнко	Олексій	Михайлович	Доктор технічних наук
Макаров	Оксана	Миколаївна	Доктор фізико-математичних наук

2 спосіб: порівняємо не перші 6 літер, а початок назви ступеня, що є більш універсальним варіантом:

```
SELECT t.Surname, t.Name, t.Patronymic, d.Degree
FROM teachers t
JOIN degrees d ON t.DegreeID=d.DegreeID
WHERE d.Degree LIKE CONCAT("доктор", '%');
```

3 спосіб: порівняємо від початку ступеня стільки символів, скільки є у слові "доктор":

```
SELECT t.Surname, t.Name, t.Patronymic, d.Degree
FROM teachers t
JOIN degrees d ON t.DegreeID=d.DegreeID
WHERE LEFT(d.Degree, CHAR_LENGTH("доктор")) LIKE CONCAT("доктор");
```

До речі, відбір даних оператором SELECT можна спрямувати у нову таблицю. Синтаксис такого запиту:

```
CREATE TABLE <ім'я нової таблиці> [AS] SELECT * FROM <ім'я таблиці>;
```

Як приклад спрямуємо результати відбору попереднього запиту у створювану таблицю doctors:

```
CREATE TABLE doctors
   SELECT t.Surname, t.Name, t.Patronymic, d.Degree FROM teachers t
   JOIN degrees d ON t.DegreeID=d.DegreeID
   WHERE LEFT(d.Degree, CHAR_LENGTH("доктор")) LIKE CONCAT("доктор");
```

Подібно створимо таблицю degree_selection. Крім того, оголосимо змінну і задамо її значення – слово "доктор", щоб не писати його двічі в умові:

```
SET @s="доктор";
CREATE TABLE degree_selection
   SELECT t.Surname, t.Name, t.Patronymic, d.Degree
   FROM teachers t
   JOIN degrees d ON t.DegreeID=d.DegreeID
   WHERE LEFT(d.Degree, CHAR_LENGTH(@s)) LIKE CONCAT(@s) ;
```

Додамо тепер у наявну таблицю degree_selection відомості про кандидатів технічних наук. Для цього скористаємось INSERT INTO SELECT:

```
SET @s="кандидат технічних наук";
INSERT INTO degree_selection
   SELECT t.Surname, t.Name, t.Patronymic, d.Degree
   FROM teachers t
   JOIN degrees d ON t.DegreeID=d.DegreeID
   WHERE d.Degree LIKE @s;
```

4 Використання групових операцій при формуванні запитів

Агрегатні функції COUNT, SUM, AVG, MAX, MIN при формуванні запитів виконують обчислення над набором значень і повертають обчислене значення, тобто формують певні сукупні характеристики, обраховані за всіма або за вка-

заними записами таблиці. Використаємо їх для групування записів. Наприклад, визначимо кількість студентів у кожній групі:

```
SELECT g.Group_Name,
          COUNT( s.StudentID )
FROM students s
JOIN groups g ON s.GroupID = g.GroupID
GROUP BY s.GroupID;
```

Group_Name	COUNT(s.StudentID)
ІПЗ-1.01	2
ІПЗ-1.02	1
ІПЗ-2.01	3
ІПЗ-2.02	2
ІПЗ-3.01	3
ІК-1.03	4
ІК-2.01	1
ІК-3.01	3
ІК-3.02	1

Унаслідок запиту записи згрупуються за групами і підрахується кількість записів у кожній з груп. Недоліком цього запиту є те, що не виводитимуться ті групи, в яких немає студентів. Щоб виправити це, треба зробити праве об'єднання:

```
SELECT g.Group_Name, COUNT(s.StudentID)
FROM students s
RIGHT JOIN groups g ON s.GroupID = g.GroupID
GROUP BY s.GroupID;
```

Group_Name	COUNT(s.StudentID)
ІПЗ-3.02	0
ІПЗ-1.01	2
ІПЗ-1.02	1
ІПЗ-2.01	3
ІПЗ-2.02	2
ІПЗ-3.01	3
ІК-1.03	4
ІК-2.01	1

Додамо у цей запит сортування груп за назвою:

```
SELECT g.Group_Name, COUNT(s.StudentID)
FROM students s
RIGHT JOIN groups g ON s.GroupID=g.GroupID
GROUP BY s.GroupID
ORDER BY g.Group_Name;
```

Group_Name ▲ 1	COUNT(s.StudentID)
ІК-1.03	4
ІК-2.01	1
ІК-3.01	3
ІК-3.02	1
ІПЗ-1.01	2
ІПЗ-1.02	1
ІПЗ-2.01	3
ІПЗ-2.02	2
ІПЗ-3.01	3
ІПЗ-3.02	0

Визначимо найвищу оцінку з дисципліни "Алгоритмізація і програмування":

```
SELECT  MAX( m.Mark )  FROM  marks m
    JOIN  disciplines d ON d.DisciplineID = m.DisciplineID
WHERE  d.Discipline LIKE 'Алгоритмізація і програмування';
```

MAX(m.Mark)
98

Визначимо студента (або студентів) з найвищою оцінкою з алгоритмізації і програмування. Для цього створимо **вкладений запит**:

```
SELECT s.Surname, s.Name, g.Group_Name, m.Mark
  FROM students s
    JOIN groups g
      ON g.GroupID = s.GroupID
    JOIN marks m
      ON m.StudentID = s.StudentID
    JOIN disciplines d
      ON d.DisciplineID = m.DisciplineID
WHERE d.Discipline LIKE 'Алгоритмізація і програмування'
    AND m.Mark = (SELECT MAX( m.Mark )
                  FROM marks m
                    JOIN disciplines d
                      ON d.DisciplineID = m.DisciplineID
                  WHERE d.Discipline LIKE 'Алгоритмізація і програмування');
```

Surname	Name	Group_Name	Mark
Петров	Микола	ПІ-3.01	98
Сердюк	Борис	ПІ-2.02	98
Усенко	Марія	ПІ-1.01	98
Хітрук	Миколай	ПІ-2.01	98

Лабораторне завдання

1. В окремій вкладці сформувати запит на відбір студентів, які навчаються за контрактом.

2. В окремій вкладці сформувати запит на відбір студентів чоловічої статі, які навчаються на бюджеті.

3. Сформувати запит на відбір усіх даних з таблиці degrees про вчених, наукові ступені яких розпочинаються зі слова "доктор".

4. За допомогою SQL-запиту вибрати прізвища та імена студентів 1998 року народження із сортуванням прізвищ за абеткою.

5. За допомогою SQL-запиту визначити кількість студентів жіночої статі.

6. В окремій вкладці сформувати SQL-запит на відбір даних про студентів (прізвище, ім’я, курс, назва групи) з таблиць students та groups.

7. Відібрати з двох таблиць teachers і degrees дані про викладачів з науковим ступенем.

8. Засобами SQL-запиту відібрати дані про викладачів без ступеня.

9. Вибрати відомості про викладачів із зазначенням наукового ступеня та посади, використовуючи дані з трьох таблиць: teachers, degrees та positions.

10. Сформувати запит для відбору даних: Surname, Name, Patronymic, Degree з двох таблиць teachers і degrees про докторів наук.

11. Спрямувати результати відбору попереднього запиту у створювану таблицю doctors.

12. Подібно попередньому запиту створити таблицю degree_selection, в яку спочатку засобами SQL-запиту відібрати дані про докторів наук, а ще одним запитом туди додати відомості про кандидатів технічних наук.

13. Сформувати SQL-запит для визначення кількості студентів у кожній групі, використовуючи відповідну агрегатну функцію.

14. Сформувати SQL-запит для визначення найвищої оцінки з дисципліни "Алгоритмізація і програмування".

15. Засобами вкладеного SQL-запиту визначити студента (або студентів) з найвищою оцінкою з алгоритмізації і програмування.

Лабораторна робота 6
Маніпулювання даними

Теоретичні відомості

1 Вставлення і видалення записів із таблиці

Оператори маніпулювання даними в таблицях БД (SELECT, INSERT, UPDATE, DELETE) не змінюють структуру таблиць, а змінюють дані всередині цих таблиць. Крім того, оператори INSERT, UPDATE, DELETE не повертають набір рядків – вони просто виконуються.

Основним призначенням оператора **INSERT** є внесення в таблиці даних. Загальний формат INSERT:

```
INSERT INTO <ім'я таблиці> (<поле1>, <поле2>, ...)
VALUES (<значення1>, <значення2>, ...);
```

де: ім'я таблиці – ім'я таблиці, в яку треба внести дані;
поле1, поле2, ... – імена стовпців;
значення1, значення2, … – значення для цих стовпців.

Наприклад, для вставлення нового запису в таблицю marks оператор INSERT матиме вигляд:

```
INSERT INTO 'marks' ('StudentID', 'DisciplineID', 'Mark')
VALUES (6, 9, 95);
```

Цей запит вставить запис про студента з ID = 6 (Романчук Зоя Петрівна), що отримав із дисципліни з ID = 9 (вища математика) оцінку 95.

Оператор INSERT може формувати із результатів відбору запитів нові щойно створені таблиці. Наприклад:

```
CREATE TABLE IF NOT EXISTS ik_students
SELECT  s.* FROM 'students' s
    JOIN   groups g
      ON   s.GroupID=g.GroupID
 WHERE  g.Group_Name LIKE "ІК-%";
```

Після цього запиту буде створено таблицю ik_students із даними таблиці students про студентів напряму ІК. Зверніть увагу на фрагмент SELECT s.* – Він означає, що у нову таблицю копіюватимуться всі поля таблиці students.

Для видалення записів із таблиці треба скористатися оператором **DELETE**. Він вимагає зазначення імені таблиці, а також можуть задані певні умови. Якщо жодної умови не задано, то видаляються всі дані з таблиці.

```
DELETE FROM <ім'я таблиці> [ WHERE <умова> ];
```

Як приклад, сформуємо запит для видалення з таблиці students студентів, що вступили до вишу у 2012 році:

```
DELETE FROM students
WHERE Gradebook LIKE "%-12";
```

Для очищення таблиці використовується оператор **TRUNCATE**:

```
TRUNCATE TABLE <ім'я таблиці>;
```

Оператор **DROP TABLE** видаляє одну або декілька таблиць, при цьому усі дані таблиць видаляються. Слід дуже обережно використовувати цю команду.

```
DROP TABLE [IF EXISTS] <ім'я таблиці> [, <ім'я таблиці>, ...] [RESTRICT | CASCADE]
```

Ключові слова IF EXISTS попереджають про помилку при спробі видалити таблицю, якої немає. Приміром, запит для видалення таблиці ik_students матиме вигляд:

```
DROP TABLE IF EXISTS ik_students;
```

Для оновлення даних в таблицях використовується оператор **UPDATE**. Заповнимо стовпець Gradebook у таблиці students (він досі залишався порожнім). Номер залікової книжки складається з ID студента, дефісу та двох останніх цифр року вступу:

```
UPDATE students
SET Gradebook=CONCAT(StudentID, '-', RIGHT(YEAR(Enter_Date),2));
```

У цьому прикладі використовується функція CONCAT для з'єднання кількох значень в один рядок і функція RIGHT для виділення останніх двох символів із рядка.

Після виконання цього запиту таблиця students набуде вигляду (тут показано фрагмент таблиці):

StudentID Код студента	Gradebook Номер залікової книжки	Surname Прізвище	Name Ім'я	Patronymic По батькові	Sex	Birth_Date Дата народження	Enter_Date Дата вступу
1	1-14	Петров	Микола	Сергійович	ч	1996-11-19	2014-08-22
2	2-14	Шевчук	Іван	Федорович	ч	1997-01-14	2014-08-22
3	3-15	Марченко	Тетяна	Олегівна	ж	1998-03-10	2015-08-22
4	4-15	Сердюк	Борис	Іванович	ч	1998-05-06	2015-08-22
5	5-14	Федоренко	Ольга	Павлівна	ж	1996-12-13	2014-08-22
6	6-16	Романчук	Зоя	Петрівна	ж	1999-03-15	2016-08-22
7	7-16	Погоріла	Лариса	Лазарівна	ж	1999-06-17	2016-08-22
8	8-15	Рябчук	Михайло	Вадимович	ч	1998-09-22	2015-08-22

Це ще не остаточне значення поля Gradebook, оскільки не вистачає нулів напочатку номерів залікових книжок. Треба обчислити кількість символів у кожному значенні й додати на початок поля нулі так, щоб загальна кількість символів дорівнювала 6. Скористаємося для цього рядковою функцією REPEAT:

REPEAT(*рядок, кількість повторів*) – створює рядок, до якого входить вказаний у першому параметрі рядок зазначену у другому параметрі кількість разів. Наприклад, REPEAT('MySQL',3) поверне рядок "MySQLMySQLMySQL".

Код SQL-запиту для додавання нулів на початок поля Gradebook:

```
UPDATE students
SET Gradebook= CONCAT(REPEAT('0', 6-CHAR_LENGTH(Gradebook)), Gradebook);
```

Символ '0' буде повторено 6-CHAR_LENGTH(Gradebook) разів і результат буде приєднано до поточного значення поля Gradebook.

Фрагмент таблиці students після цього запиту:

StudentID Код студента	Gradebook Номер залікової книжки	Surname Прізвище	Name Ім'я	Patronymic По батькові	Sex	Birth_Date Дата народження	Enter_Date Дата вступу
1	001-14	Петров	Микола	Сергійович	ч	1996-11-19	2014-08-22
2	002-14	Шевчук	Іван	Федорович	ч	1997-01-14	2014-08-22
3	003-15	Марченко	Тетяна	Олегівна	ж	1998-03-10	2015-08-22
4	004-15	Сердюк	Борис	Іванович	ч	1998-05-06	2015-08-22

2 Експорт даних із БД

Для експортування даних є функція phpMyAdmin *Експорт*, яка дозволяє експортувати як окремі записи, так і цілі таблиці. Крім того, можливо експортувати результати запитів.

Наприклад, треба скопіювати інформацію про всіх викладачів. Таблиця teachers містить замість назви кафедри, наукового ступеня та посад їхні ID у відповідних таблицях. Але в експортованих даних потрібні не ID, а самі назви. Сформуємо запит, який формуватиме таблицю з даними про вчителів із розшифруванням усіх ID:

```
SELECT * FROM `teachers` t
LEFT JOIN departments d ON t.DepartmentID=d.DepartmentID
LEFT JOIN degrees dg ON t.DegreeID=dg.DegreeID
LEFT JOIN positions p ON t.PositionID=p.PositionID;
```

Цей запит можна відкоригувати так, щоб коди, дублікати та сторонні поля не виводилися:

```
SELECT t.TeacherID, t.Table_Number, t.Surname, t.Name, t.Patronymic, t.Sex,
        dg.Degree, d.Department, p.Position, t.Phone, t.RecruitmentDate
FROM `teachers` t
LEFT JOIN departments d ON t.DepartmentID = d.DepartmentID
LEFT JOIN degrees dg ON t.DegreeID = dg.DegreeID
LEFT JOIN positions p ON t.PositionID = p.PositionID;
```

Після того як дані для експорту готові, внизу вікна у розділі *Використання результатів запиту* слід вибрати *Експорт* і задати такі параметри експортування:

Спосіб експорту – Звичайний;

Рядки – Вивантажити всі рядки;

Виведення – Зберегти виведення у файл;

Формат – Microsoft Word 2000;

Параметри формату;

Збереження таблиці – Структура й дані;

Параметри збереження даних – Замінити NULL на порожнє значення (видалити NULL з віконця) та зазначити *Помістити назви полів у першому рядку*.

Таблицю буде експортовано у документ Word, звідки її легко роздрукувати або скопіювати до іншого файлу:

TeacherID	Table_Number	Surname	Name	Patronymic	Sex	Degree	Department	Position	Phone	Recruitment-Date
1	1234	Ткаченко	Василь	Петрович	ч	кандидат технічних наук	Інформаційних технологій	професор	3225566	1990-09-01
2	5678	Макарова	Оксана	Миколаївна	ж	доктор фізико-математичних наук	Вищої математики	доцент	3334455	2000-09-01
3	5436	Зайцев	Іван	Павлович	ч	кандидат історичних наук	Історії	доцент	7651234	1996-09-01
4	9876	Матвієнко	Олексій	Михайлович	ч	доктор технічних наук	Інформаційних технологій	професор	7543245	2006-02-01
5	3874	Васильєв	Сергій	Іванович	ч		Інформаційних технологій	старший викладач	7771234	2010-09-01
6	1543	Петренко	Олена	Петрівна	ж	кандидат економічних наук	Вищої математики	викладач	2223344	1998-09-01
7	7654	Сидорчук	Микола	Васильович	ч	кандидат технічних наук	Інформаційних технологій	викладач	7154367	2001-03-01
8	7543	Львівський	Олександр	Михайлович	ч	кандидат економічних наук	Економіки	професор	7159933	1994-09-01
9	8765	Пивовар	Ірина	Юхимівна	ж	кандидат технічних наук	Вищої математики	доцент	7056644	2015-09-01
10	7765	Караваєва	Тетяна	Євгенівна	ж	кандидат фізико-математичних наук	Інформаційних технологій	доцент	7031265	2004-09-01

Лабораторне завдання

1. Засобами оператора INSERT вставити нові записи в таблицю marks. Цей та подальші запити маніпулювання даними зберігати в окремих вкладках задля зручності їх перевірки викладачем.

2. Створити запит на створення таблиці ik_students, в яку відібрати дані з таблиці students про студентів напряму ІК.

3. Сформувати запит для видалення з таблиці students студентів, що вступили до вишу у 2012 році.

4. Сформувати запит для видалення таблиці ik_students.

5. Сформувати запит на оновлення даних у таблиці students шляхом заповнення поки що порожнього стовпця Gradebook (номер залікової книжки) в такий спосіб: доєднати до ID студента, дефіс та дві останні цифри року вступу, наприклад: 1-17, 2-16 тощо.

6. Сформувати ще один запит на оновлення даних стовпця Gradebook у таблиці students, додавши на початок поля нулі так, щоб загальна кількість символів дорівнювала 6.

7. Сформувати запит на відбір даних про викладачів із розшифруванням усіх ID: усіх відомостей із таблиці teachers, а також назви кафедри, де працює викладач, його наукового ступеня та посади, вибравши їх із відповідних таблиць. Але в експортованих даних потрібні не ID, а самі назви.

8. Відкоригувати попередній запит так, щоб коди, дублікати та сторонні поля не виводилися.

9. Експортувати дані, здобуті внаслідок роботи попереднього запиту у документ Word, звідки їх легко роздрукувати або скопіювати до іншого файлу.

Лабораторна робота 7
Техніки програмування SQL

Теоретичні відомості

1 Збережені процедури і функції (stored procedures and functions)

Збережені підпрограми (процедури і функції) являють собою набір команд SQL, які можуть компілюватися і зберігатися на сервері. Часто використовувані запити має сенс оформляти у вигляді збережених процедур та викликати за потреби. Це забезпечуватиме кращу продуктивність, оскільки такий запит аналізується лише один раз, і при цьому зменшується трафік між сервером і клієнтом. У збережених процедурах і функціях можна оголошувати змінні, керувати потоками даних, а також застосовувати інші техніки програмування.

Набір команд, які використовують разом зі збереженими підпрограмами:

Назва	Опис
CREATE PROCEDURE	створення процедури
CREATE FUNCTION	створення функції
ALTER PROCEDURE	змінення процедури
ALTER FUNCTION	змінення функції
DROP PROCEDURE	видалення процедури
DROP FUNCTION	видалення функції
CALL proc_name()	виклик процедури proc_name
DECLARE	оголошення локальної змінної
SET	змінення значень локальних і глобальних змінних
SELECT … INTO	збереження результату у змінну
IF	умовний оператор IF-THEN-ELSE-END IF
CASE … WHEN	оператор вибору
LOOP, REPEAT, WHILE	цикли
RETURNS	повернення значення з функції

Створення збереженої процедури:

```
CREATE FUNCTION <ім'я процедури> ([<параметр> [,...]]) RETURNS type
[characteristic ...] routine_body
```

Тут ім'я_процедури – ім'я збереженої процедури, у дужках за ім'ям, передається необов'язковий список параметрів (через кому). Кожен параметр дозволяє передати до процедури вхідні дані або з неї – результат роботи процедури і має такий синтаксис:

```
[ IN | OUT | INOUT ] <ім'я параметра>  <тип>
```

Перед іменем параметра зазначається одне з ключових слів in, out, inout, які дозволяють задати напрямок передавання даних:

– IN – дані передаються всередину до збереженої процедури, але якщо параметру з даним модифікатором усередині надається нове значення, після виходу з неї він не зберігається і параметр набуває значення, яке він мав до виклику процедури;
– OUT – дані передаються зі збереженої процедури, навіть якщо параметр має якесь початкове значення, всередині збереженої процедури це значення не береться до уваги. З іншого боку, якщо параметр змінюється всередині процедури, після виклику процедури параметр має значення, надане йому всередині процедури;
– INOUT – значення цього параметра береться до уваги всередині процедури і це значення зберігається після виходу з неї.

Після імені параметра зазначається його тип. Якщо жоден з модифікаторів не вказаний, вважається, що параметр оголошений з ключовим словом IN.

Важливо, що список аргументів, вкладених у круглі дужки, повинен бути присутнім завжди. Якщо аргументи відсутні, треба використовувати порожній список аргументів ().

Текст збереженої процедури починається зі встановлення обмежувача – символу або рядка символів, який використовується для повідомлення про завершення написання SQL-виразу. Тут використовується рядок "//" як обмежувач.

```
DELIMITER //
```

Далі йде команда створення збереженої процедури (у нашому прикладі її ім’я p2) з цілим параметром x:

```
CREATE PROCEDURE ‘p2’ (x INT)
```

Початок і кінець процедури позначаються BEGIN та END.

У збережених підпрограмах часто використовують змінні. Вони повинні бути оголошені на початку блока BEGIN-END разом з їхніми типами даних:

```
DECLARE <ім’я змінної> <тип> [DEFAULT <усталене значення>];
```

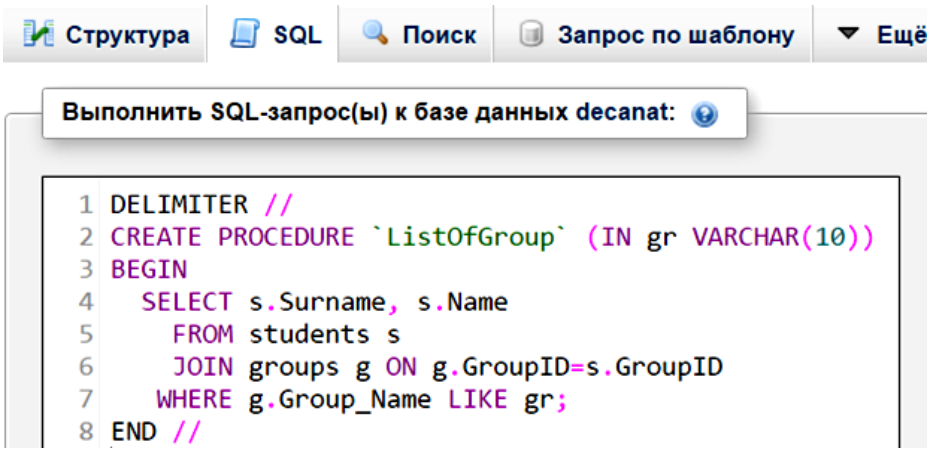
Приклади оголошень змінних:

```
DECLARE val INT DEFAULT 0;
DECLARE a, b INT DEFAULT 5;
DECLARE str VARCHAR(50);
DECLARE v1, v2, v3 TINYINT;
```

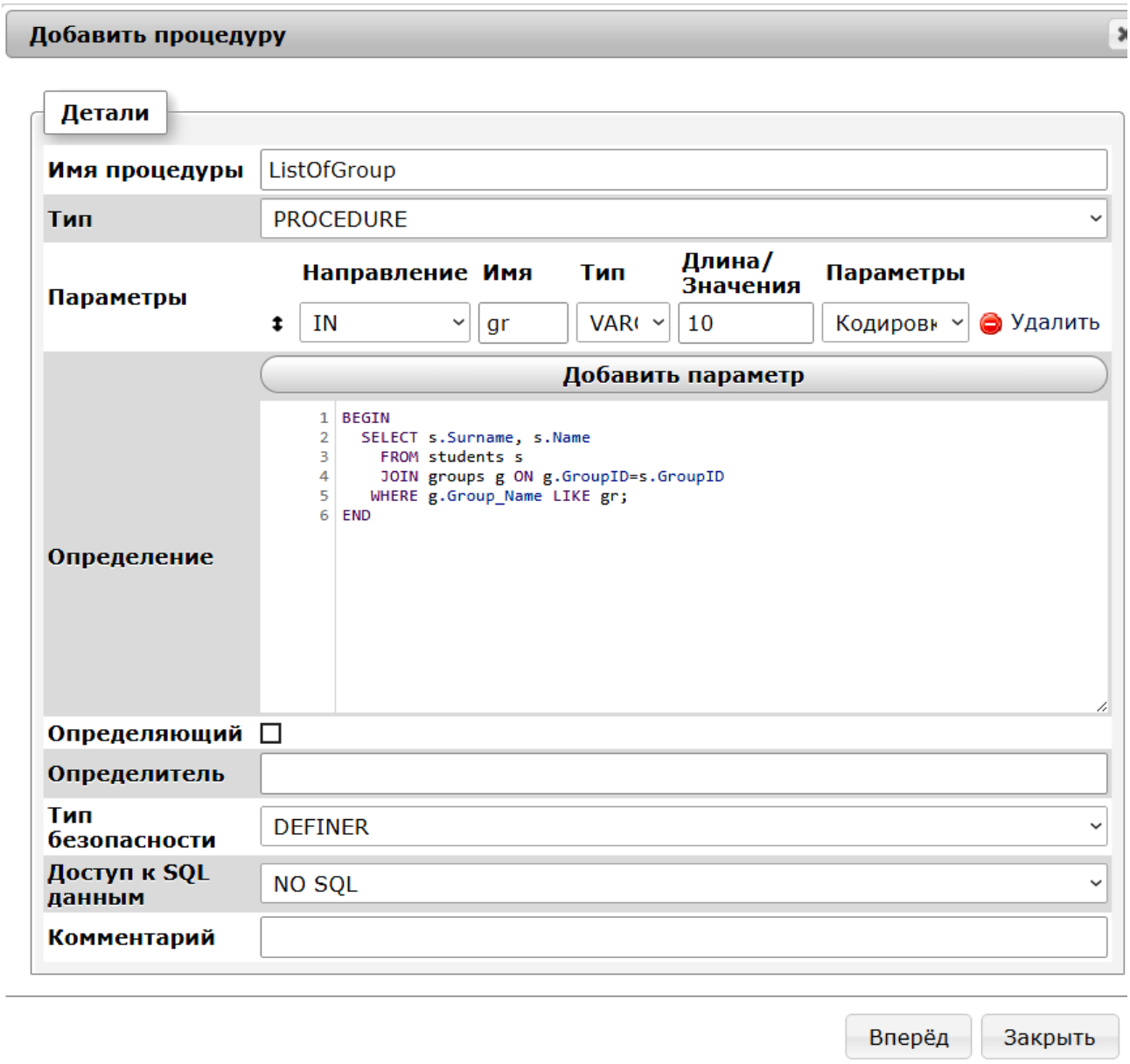
Надати значення змінній можна оператором SET. Наприклад:

```
SET val = 5;
```

Наступна збережена процедура сформує список групи, назва якої передається до неї як параметр. Її назву потрібно набрати у вікні для введення SQL-код і натиснути *ОК*. Результатом стане створення процедури (не виконання).



Інший спосіб створення процедури: на вкладці *Процедуры* – натиснути *Добавить процедуру* і заповнити форму, яка з'явиться на екрані.



Щоб запустити створену процедуру на виконання, потрібно написати запит:

```
CALL ListOfGroup ("ІПЗ-2.01")
```

або на вкладці *Процедуры* вибрати процедуру ListOfGroup, натиснути *Выполнить* і ввести значення параметра ІП3-2.01. Результат виконання процедури:

Surname	Name
Марченко	Тетяна
Чава	Олександр
Хітрук	Микола

Примітка. Якщо процедура не виконується і виникає помилка: "Thread stack overrun: 6436 bytes of used a 131072 byte stack, and 128000 bytes needed. Use 'mysqld -O thread_stack=#' to specify a bigger stack", потрібно змінити значення змінної thread_stack у конфігураційному файлі *my.ini* на 256 Кб.

Створимо за допомогою збереженої процедури телефонний довідник студентів заданого курсу:

```
DELIMITER //
CREATE PROCEDURE `ListOfPhones` ( IN `kurs` INT )
BEGIN
  SELECT  s.StudentID, s.Surname, s.Name, s.Phone
    FROM  students s
      JOIN  groups g
        ON  s.GroupID = g.GroupID
  WHERE  g.Course = kurs;
END
```

Наприклад, телефонний довідник студентів першого курсу:

```
call ListOfPhones(1)
```

☐ Показать все | Количество строк: 25

+ Параметры

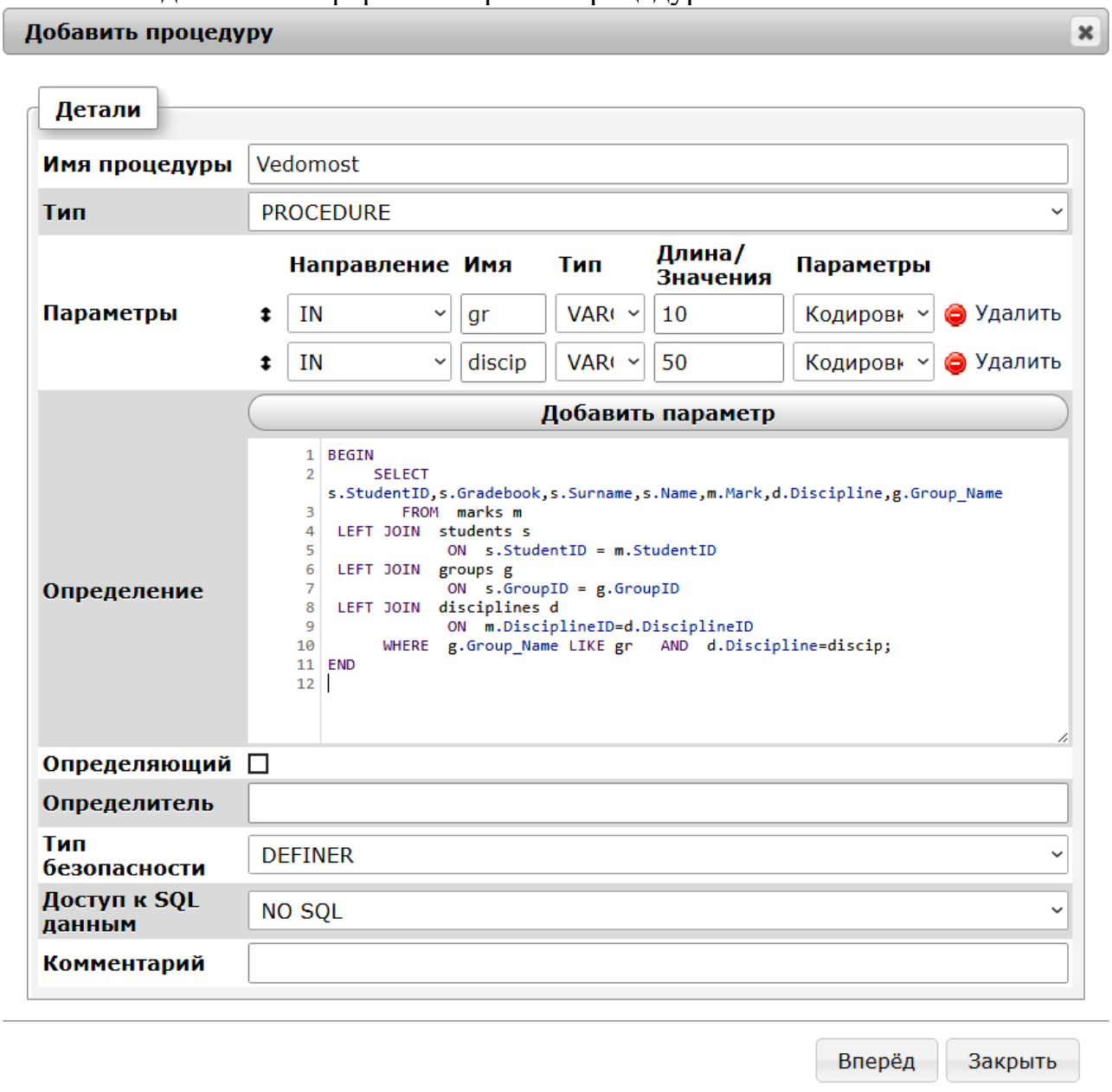
StudentID	Surname	Name	Phone
8	Рябчук	Михайло	0917652349
10	Усенко	Марія	0674458273
9	Стоколос	Ірина	0507781265
6	Романчук	Зоя	067453621
7	Погоріла	Лариса	0503456798
13	Волошенюк	Любов	0487432745
19	Кучма	Світлана	0675432160

Створимо відомість про складання екзамену студентів певної групи по заданому предмету. SQL-запит для створення:

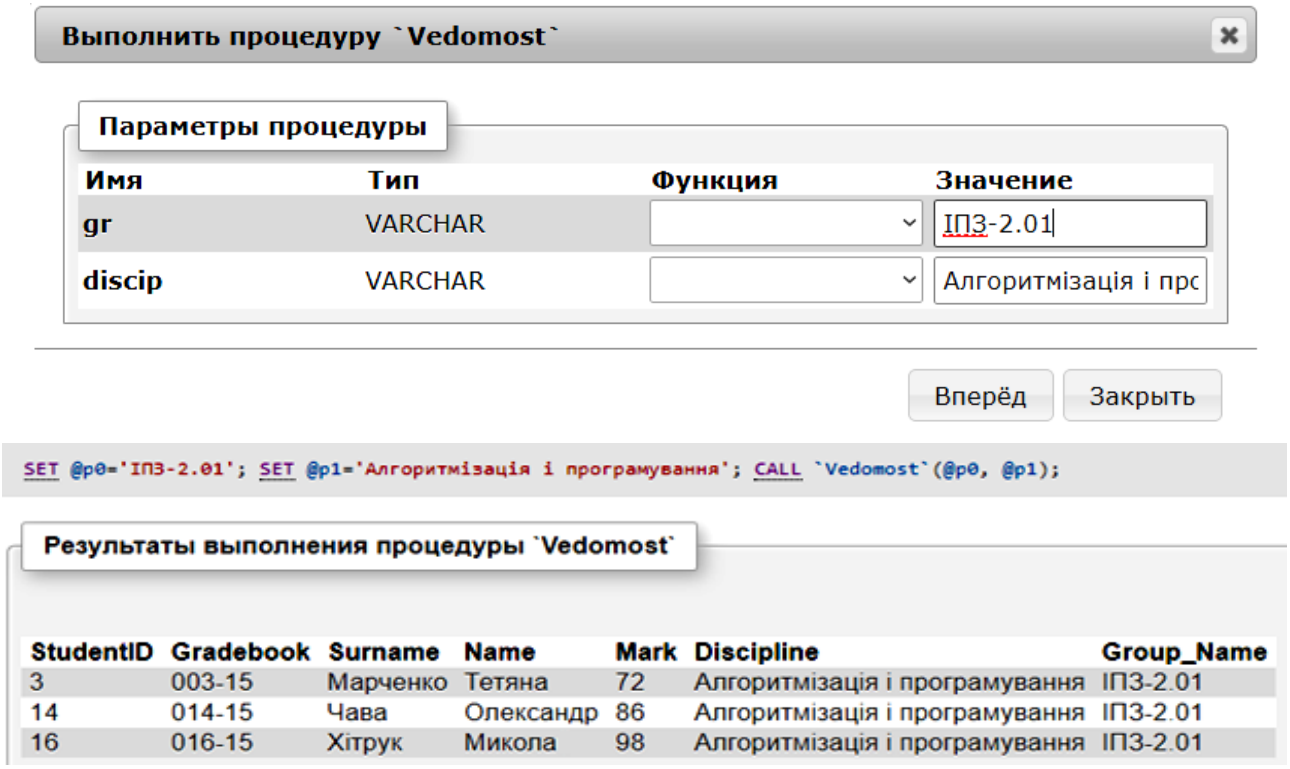
```
DELIMITER //
CREATE PROCEDURE `Vedomost` ( gr VARCHAR(10), discip VARCHAR (50) )
BEGIN
```

```
    SELECT s.StudentID,s.Gradebook,s.Surname,s.Name,m.Mark,d.Discipline,g.Group_Name
        FROM  marks m
   LEFT JOIN  students s
           ON  s.StudentID = m.StudentID
   LEFT JOIN  groups g
           ON  s.GroupID = g.GroupID
   LEFT JOIN  disciplines d
           ON  m.DisciplineID=d.DisciplineID
      WHERE  g.Group_Name LIKE gr   AND  d.Discipline=discip;
  END
```

Або за допомогою форми створення процедури:



Відомість про складання екзамену з дисципліни "Алгоритмізація і програмування" ІПЗ-2.01:



Параметри OUT використовують при виклику збережених процедур з php-коду. Наступна процедура отримує назву дисципліни і визначає кількість студентів, які одержали з цієї дисципліни оцінки вищі за середньоарифметичну. Шукана кількість записується у змінну cnt. Виклик збережених процедур MySQL з php-коду є поза межами нашого розгляду. Протестувати підрахунок кількості можна, додавши наприкінці процедури SELECT cnt:

```
DELIMITER //
CREATE PROCEDURE `AvgMarks` ( IN discipl VARCHAR(20), OUT cnt int )
BEGIN
  SELECT COUNT(m.MarkID)
      INTO cnt
    FROM marks m
      JOIN disciplines d
        ON d.DisciplineID=m.DisciplineID
   WHERE d.Discipline LIKE discipl
       AND m.Mark>=(SELECT AVG(mr.Mark) FROM marks mr
                          JOIN  disciplines dc  ON dc.DisciplineID=mr.DisciplineID
                       WHERE  dc.Discipline LIKE discipl) ;
  SELECT cnt;
END
```

У наведеному прикладі результат команди SELECT записується у змінну cnt. Синтаксис запиту для запису результату у змінну такий:

```
SELECT ...
INTO <змінна>
FROM ...
WHERE ...
```

Збережена функція відрізняється від процедури тим, що обов'язково повертає значення. Приклад функції:

```
DELIMITER //
CREATE FUNCTION func() RETURNS INTEGER
BEGIN
  DECLARE val INTEGER;
  SELECT  id   INTO  val  FROM  table;
  RETURN  IFNULL(val, 0);
END
```

На відміну від процедури, під час створення функції треба явно задавати тип очікуваного результату:

```
RETURNS <тип>
```

Виклик створеної функції:

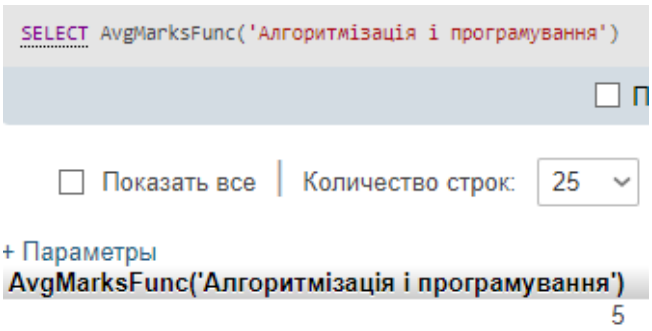
```
SELECT func();
```

Видалення створеної функції:

```
DROP FUNCTION IF EXISTS func;
```

Останню збережену процедуру можна записати у вигляді функції:

```
DELIMITER //
CREATE FUNCTION `AvgMarksFunc` (discipl VARCHAR(50))
RETURNS INT
BEGIN
  DECLARE cnt INT;
SELECT COUNT(m.MarkID)
     INTO  cnt
   FROM  marks m
     JOIN  disciplines d
       ON  d.DisciplineID=m.DisciplineID
WHERE  d.Discipline LIKE discipl
     AND  m.Mark>=(SELECT AVG(mr.Mark)
                     FROM  marks mr
                       JOIN  disciplines dc
                         ON  dc.DisciplineID=mr.DisciplineID
                    WHERE  dc.Discipline LIKE discipl);
  RETURN IFNULL(cnt, 0);
END
```



Створимо процедуру, яка за назвою кафедри виводитиме список викладачів кафедри із зазначенням вченого ступеня, посади і переліку дисциплін, що викладаються ними.

Для спрощення коду створимо спочатку допоміжні функції для визначення текстових назв кафедри, вченого ступеня, посади та дисципліни за їхніми ID.

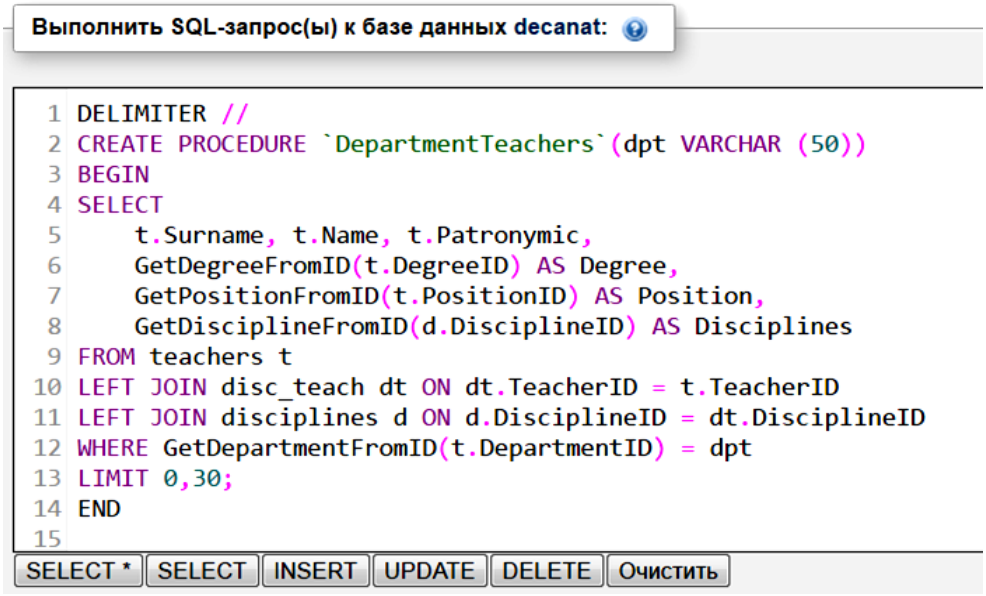
Функція GetDepartmentFromID визначає назву кафедри за ID:

```
CREATE FUNCTION `GetDepartmentFromID` (dptID INT)
RETURNS VARCHAR(50)
BEGIN
  DECLARE dpt VARCHAR(50);
```

```
 SELECT  Department INTO dpt    FROM  departments
 WHERE  DepartmentID=dptID;
RETURN  IFNULL(dpt, "");
END
```

Аналогічно створюються функції GetDegreeFromID, GetPositionFromID та GetDisciplineFromID.

Для розв'язання задачі спочатку спробуємо обмежитися командами та функціями, які ми вже застосовували раніше:



У наведеному коді задано імена полів (псевдоніми): Degree замість GetDegreeFromID (t.DegreeID), Position замість GetPositionFromID (t.PositionID) тощо. Але результат нас не влаштовує, оскільки викладачі виводяться декілька разів (зокрема, для параметра Інформаційних технологій):

Surname	Name	Patronymic	Degree	Position	Disciplines
Ткаченко	Василь	Петрович	Кандидат технічних наук	професор	Алгоритмізація і програмування
Ткаченко	Василь	Петрович	Кандидат технічних наук	професор	Об'єктно-орієнтоване програмування
Матвієнко	Олексій	Михайлович	Доктор технічних наук	професор	Об'єктно-орієнтоване програмування
Васильєв	Сергій	Іванович		старший викладач	Інформатика
Васильєв	Сергій	Іванович		старший викладач	Бази даних
Караваєва	Тетяна	Євгенівна	Кандидат фізико-математичних наук	доцент	Алгоритмізація і програмування
Караваєва	Тетяна	Євгенівна	Кандидат фізико-математичних наук	доцент	Об'єктно-орієнтоване програмування
Іванов	Василь	Миколайович	Кандидат технічних наук	доцент	Інформатика
Іванов	Василь	Миколайович	Кандидат технічних наук	доцент	Комп'ютерна графіка
Щукіна	Олена	Михайлівна		викладач	Алгоритмізація і програмування
Щукіна	Олена	Михайлівна		викладач	Інформатика
Щукіна	Олена	Михайлівна		викладач	Комп'ютерна графіка
Сидорчук	Микола	Васильович	Кандидат технічних наук	доцент	Об'єктно-орієнтоване програмування
Сидорчук	Микола	Васильович	Кандидат технічних наук	доцент	Інформатика

Спроба згрупувати записи за викладачами призводить до того, що виводиться лише одна з дисциплін:

Surname	Name	Patronymic	Degree	Position	Disciplines
Ткаченко	Василь	Петрович	Кандидат технічних наук	професор	Алгоритмізація і програмування
Матвієнко	Олексій	Михайлович	Доктор технічних наук	професор	Об'єктно-орієнтоване програмування
Васильєв	Сергій	Іванович		старший викладач	Інформатика
Караваєва	Тетяна	Євгенівна	Кандидат фізико-математичних наук	доцент	Алгоритмізація і програмування
Іванов	Василь	Миколайович	Кандидат технічних наук	доцент	Інформатика
Щукіна	Олена	Михайлівна		викладач	Алгоритмізація і програмування
Сидорчук	Микола	Васильович	Кандидат технічних наук	доцент	Об'єктно-орієнтоване програмування

Розв’язанням проблеми є використання функції GROUP_CONCAT() для об’єднання в один рядок назв дисциплін, розділених комами, для кожного викладача. Синтаксис її такий:

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
[ORDER BY {unsigned_integer | col_name | expr}
[ASC | DESC] [, col_name ...]]
[SEPARATOR str_val])
```

Щодо нашої задачі функція GROUP_CONCAT() буде такою:

```
GROUP_CONCAT(GetDisciplineFromID(d.DisciplineID)
ORDER BY Discipline
SEPARATOR ", " )
```

Її можна прочитати так:
"ОБ’ЄДНАТИ назви дисциплін,
ВІДСОРТУВАТИ їх за назвами,
РОЗДІЛИТИ їх комами".

Остаточний код процедури:

```
DELIMITER //
CREATE PROCEDURE `DepartmentTeachers`(dpt VARCHAR (50))
BEGIN
  SELECT t.Surname, t.Name, t.Patronymic, GetDegreeFromID(t.DegreeID) AS Degree,
              GetPositionFromID(t.PositionID) AS Position,
              GROUP_CONCAT( GetDisciplineFromID(d.DisciplineID)
  ORDER BY Discipline
  SEPARATOR ", " ) AS Disciplines
  FROM teachers t
  LEFT JOIN disc_teach dt ON dt.TeacherID = t.TeacherID
  LEFT JOIN disciplines d ON d.DisciplineID = dt.DisciplineID
  WHERE GetDepartmentFromID(t.DepartmentID) = dpt
  GROUP BY t.TeacherID
END
```

Результат виконання процедури:

Surname	Name	Patronymic	Degree	Position	Disciplines
Ткаченко	Василь	Петрович	Кандидат технічних наук	професор	Алгоритмізація і програмування, Об'єктно-орієнтоване програмування
Матвієнко	Олексій	Михайлович	Доктор технічних наук	професор	Об'єктно-орієнтоване програмування
Васильєв	Сергій	Іванович		старший викладач	Інформатика, Бази даних
Караваєва	Тетяна	Євгенівна	Кандидат фізико-математичних наук	доцент	Алгоритмізація і програмування, Об'єктно-орієнтоване програмування
Іванов	Василь	Миколайович	Кандидат технічних наук	доцент	Інформатика, Комп'ютерна графіка
Щукіна	Олена	Михайлівна		викладач	Інформатика, Алгоритмізація і програмування, Комп'ютерна графіка
Сидорчук	Микола	Васильович	Кандидат технічних наук	доцент	Інформатика, Об'єктно-орієнтоване програмування

2 Умови та цикли в MySQL

2.1 Засоби MySQL для перевірки умов

MySQL підтримує конструкції IF, CASE, ITERATE, LEAVE LOOP, WHILE і REPEAT та курсори для керування потоками в межах збереженої процедури.

Конструкція IF дозволяє виконувати задачі, які містять умови:

```
IF <умова>
THEN <вираз>;
ELSE <вираз>;
END IF
```

Приклад:

```
DELIMITER //
CREATE PROCEDURE `proc_IF` (IN param1 INT)
BEGIN
 DECLARE variable1 INT;
 SET variable1 = param1 + 1;
 IF variable1 = 0 THEN
    SELECT variable1;
 END IF;
 IF param1 = 0 THEN
    SELECT 'Parameter value = 0';
 ELSE
    SELECT 'Parameter value <> 0';
 END IF;
END
```

Конструкція CASE – це ще один спосіб перевірки умов і вибору відповідного рішення. CASE дозволяє замінити множину конструкцій IF. Синтаксис:

```
CASE <змінна>
WHEN <умова> THEN
        <вираз>;
WHEN <умова> THEN
        <вираз>;
 ELSE <вираз>;
END CASE
```

Приклад:

```
CREATE PROCEDURE `proc_CASE` (IN param1 INT)
BEGIN
  DECLARE variable1 INT;
  SET variable1 = param1 + 1;
  CASE variable1
    WHEN 0 THEN
```

```
        INSERT INTO table1 VALUES (param1);
      WHEN 1 THEN
        INSERT INTO table1 VALUES(variable1);
      ELSE
        INSERT INTO table1 VALUES (99);
    END CASE;
  END
```

або:

```
  CREATE PROCEDURE `proc_CASE` (IN param1 INT)
  BEGIN
    DECLARE variable1 INT;
    SET variable1 = param1 + 1;
    CASE
      WHEN variable1 = 0 THEN
        INSERT INTO table1 VALUES (param1);
      WHEN variable1 = 1 THEN
        INSERT INTO table1 VALUES(variable1);
      ELSE
        INSERT INTO table1 VALUES (99);
    END CASE;
  END
```

Крім типових для програмування команд для перевірки умов (IF, CASE), в MySQL є й специфічні ISNULL, IFNULL і NULLIF:

ISNULL (EXP1, EXP2) – якщо EXP1<>NULL, ISNULL замінює значення EXP1 значенням EXP2;

IFNULL (EXP1, EXP2) – якщо EXP1<>NULL, IFNULL повертає EXP1, інакше повертає EXP2;

NULLIF (EXP1, EXP2) – якщо EXP1=EXP2, то повертає NULL, інакше – EXP1.

2.2 Циклічні конструкції MySQL

Цикл WHILE повторює блок операторів до тих пір, доки умова істинна:

```
WHILE <умова> DO <вираз>;
END WHILE
```

Приклад:

```
CREATE PROCEDURE test_while(IN a INT)
BEGIN
 WHILE (a < 20) DO
  SELECT a;
  SET a=a + 1;
 END WHILE;
END
```

Цикл REPEAT повторює вираз допоки умова UNTIL виконується. Відмінність з циклом WHILE полягає в тому, що REPEAT виконується хоча б один раз, а WHILE може не виконатися жодного разу, якщо умова спочатку є хибною.

```
REPEAT <вираз>
```

```
UNTIL <умова>
END REPEAT
```

Наприклад:

```
CREATE PROCEDURE test_repeat(IN a INT)
  BEGIN
  REPEAT
    SELECT a;
    SET a = a + 1;
  UNTIL a > 5
  END REPEAT;
  END
```

Приклад процедури, яка містить цикл WHILE:

```
DELIMITER //
CREATE PROCEDURE `proc_WHILE` (IN param1 INT)
BEGIN
DECLARE variable1, variable2 INT;
SET variable1 = 0;
WHILE variable1 < param1 DO
  INSERT INTO table1 VALUES (param1);
  SELECT COUNT(*) INTO variable2 FROM table1;
  SET variable1 = variable1 + 1;
END WHILE;
END
```

Курсори є специфічною циклічною конструкцією. Їх використовують для проходження набором рядків й опрацювання кожного рядка.

MySQL підтримує курсори у збережених процедурах.

Короткий синтаксис створення і використання курсора:

```
/*Оголошення курсора та його заповнення */
DECLARE cursor-name CURSOR FOR SELECT ...;
/*Що робити, якщо більше немає записів */
DECLARE CONTINUE HANDLER FOR NOT FOUND
OPEN cursor-name;                          /*Відкрити курсор*/
/*Зчитати у змінну variable значення поточного стовпчика */
FETCH cursor-name INTO variable [, variable];
CLOSE cursor-name;                          /*Закрити курсор*/
```

Приклад процедури з використанням курсора:

```
DELIMITER //
CREATE PROCEDURE `proc_CURSOR` (OUT param1 INT)
BEGIN
  DECLARE a, b, c INT;
  DECLARE cur1 CURSOR FOR SELECT col1 FROMtable1;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET b = 1;
  OPEN cur1;
    SET b = 0;
    SET c = 0;
    WHILE b = 0 DO
       FETCH cur1 INTO a;
```

```
        IF b = 0 THEN
          SET c = c + a;
        END IF;
      END WHILE;
    CLOSE cur1;
  SET param1 = c;
END
```

У курсорів є такі властивості, які необхідно зрозуміти, щоб уникнути несподіваних результатів:

– відкритий курсор не враховуватиме змінення в таблиці, що сталися після відкриття цього курсора;
– під час роботи курсора не передбачено зворотного опрацювання і неможливо пропускати рядки, тобто всі записи опрацьовують послідовно один за одним.

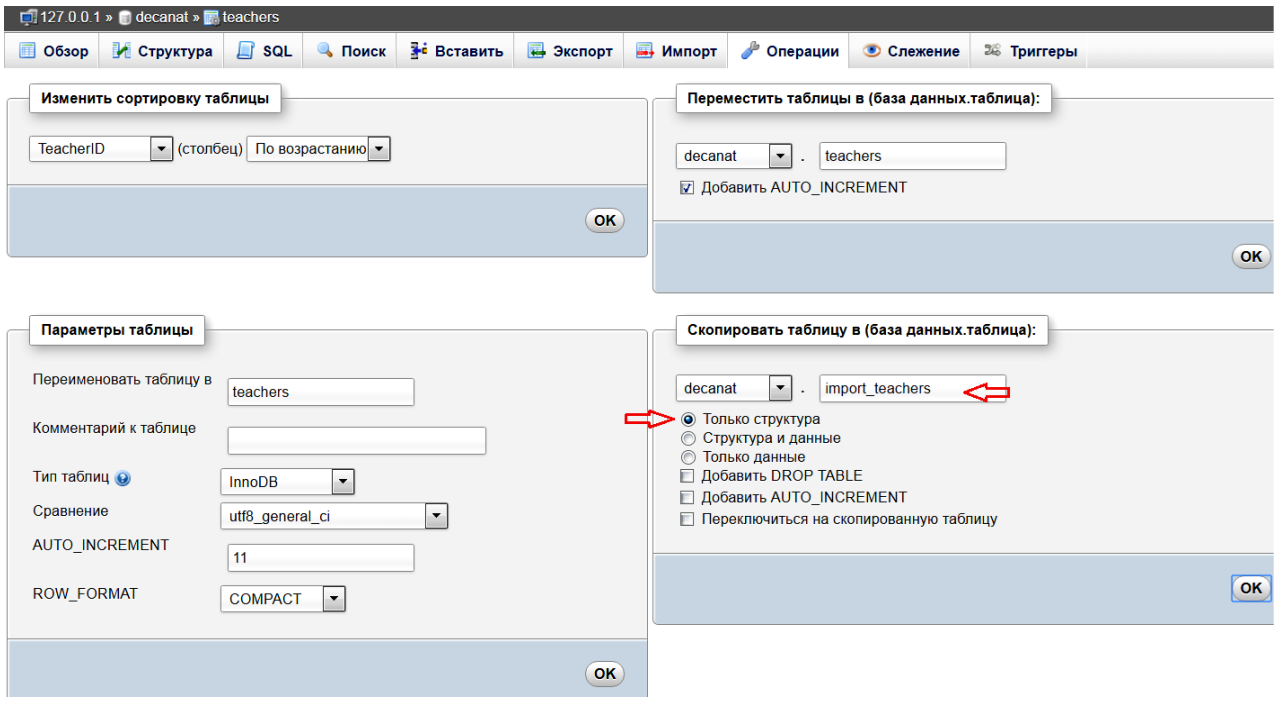
Курсори вважаються громіздким інструментом, а тому не рекомендується використовувати їх у часто виконуваних процедурах (наприклад, пошук у базі) без нагальної потреби. Але вони є незамінними, наприклад, при імпорті інформації до бази. Для демонстрації роботи з курсором створимо процедуру імпортування даних про викладачів. Підготуємо Excel-таблицю (див. рис. 2 далі) на основі тої, що ми експортували раніше.

Процедура імпортування повинна виконати таке:

1) знайти і замінити текстові назви вчених ступенів, посад, кафедр і дисциплін на відповідні ID;
2) розподілити дані по таблицях.

У разі, якщо дані про викладача вже є в базі, то вони оновлюватимуться.

Створимо спочатку в БД decanat таблицю import_teachers такої самої структури, як підготовлена таблиця в Excel, й імпортуємо до неї дані про викладачів. Для створення вказаної таблиці можна використати структуру таблиці teachers:



Потім до структури таблиці треба внести змінення:

Изменить

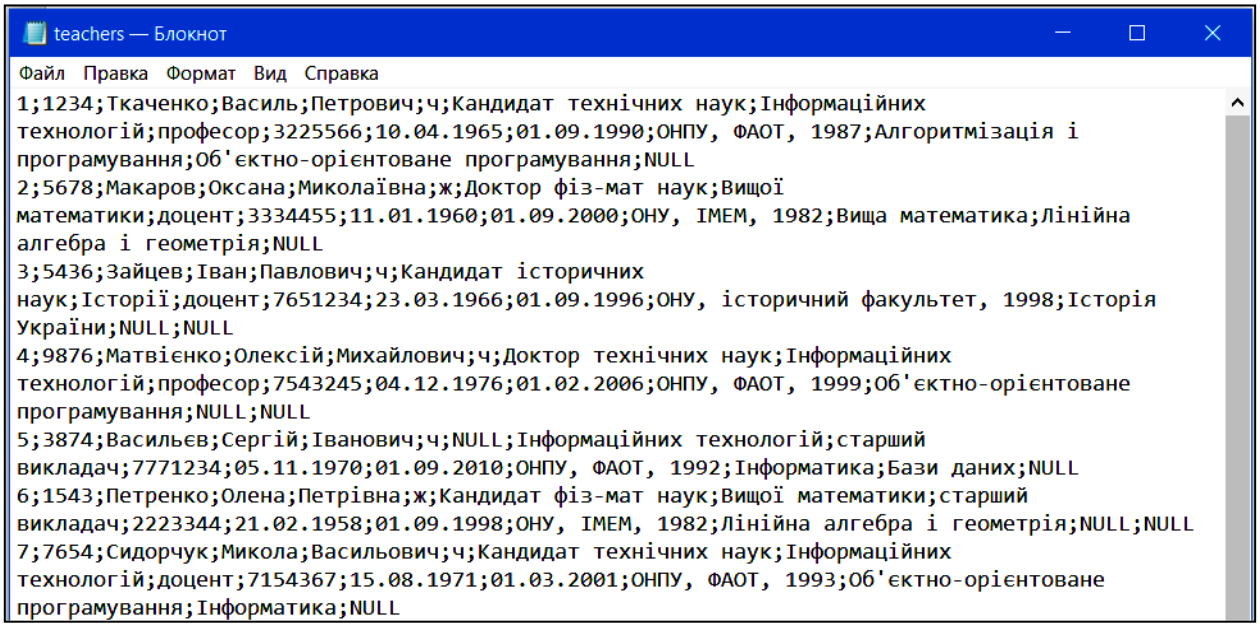
Структура

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты	Null	A_I
Degree	VARCHAR	50	NULL			☑	☐
Department	VARCHAR	50	Нет			☐	☐
Position	VARCHAR	30	Нет			☐	☐

Зверніть увагу, що поля Discipline2 і Discipline3 можуть мати значення NULL.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
1	**TeacherID**	int(11)			Нет	*Нет*	AUTO_INCREMENT
2	**Table_Number**	varchar(10)	utf8_general_ci		Нет	*Нет*	
3	**Surname**	varchar(50)	utf8_general_ci		Нет	*Нет*	
4	**Name**	varchar(50)	utf8_general_ci		Нет	*Нет*	
5	**Patronymic**	varchar(50)	utf8_general_ci		Да	*NULL*	
6	**Sex**	varchar(1)	utf8_general_ci		Нет	*Нет*	
7	**Degree**	varchar(50)	utf8_general_ci		Да	*NULL*	
8	**Department**	varchar(50)	utf8_general_ci		Нет	*Нет*	
9	**Position**	varchar(30)	utf8_general_ci		Нет	*Нет*	
10	**Phone**	varchar(10)	utf8_general_ci		Да	*NULL*	
11	**BirthDate**	date			Нет	*Нет*	
12	**RecruitmentDate**	date			Нет	*Нет*	
13	**Education**	varchar(100)	utf8_general_ci		Нет	*Нет*	
14	**Discipline1**	varchar(50)	utf8_general_ci		Нет	*Нет*	
15	**Discipline2**	varchar(50)	utf8_general_ci		Да	*NULL*	
16	**Discipline3**	varchar(50)	utf8_general_ci		Да	*NULL*	

CSV-файл teachers у Блокноті:



Імпортуємо його до таблиці import_teachers.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	ID	Tab Num	Surname	Name	Patronymic	Sex	Degree	Department	Position	Phone	BirthDate	RecruitmentDate	Education	Discipline1	Discipline2	Discipline3
2	1	1234	Ткаченко	Василь	Петрович	ч	Кандидат технічних наук	Інформаційних технологій	професор	3225566	10.04.1965	01.09.1990	ОНПУ, ФАОТ, 1987	Алгоритмізація і програмування	Об'єктно-орієнтоване програмування	
3	2	5678	Макарова	Оксана	Миколаївна	ж	Доктор фіз.-мат. наук	Вищої математики	доцент	3334455	11.01.1960	01.09.2000	ОНУ, ІМЕМ, 1982	Вища математика	Лінійна алгебра і геометрія	
4	3	5436	Зайцев	Іван	Павлович	ч	Кандидат історичних наук	Історії	доцент	7651234	23.03.1966	01.09.1996	ОНУ, історичний факультет, 1998	Історія України		
5	4	9876	Матвієнко	Олексій	Михайлович	ч	Доктор технічних наук	Інформаційних технологій	професор	7543245	04.12.1976	01.02.2006	ОНПУ, ФАОТ, 1999	Об'єктно-орієнтоване програмування		
6	5	3874	Васильєв	Сергій	Іванович	ч		Інформаційних технологій	старший викладач	7771234	05.11.1970	01.09.2010	ОНПУ, ФАОТ, 1992	Інформатика	Бази даних	
7	6	1543	Петренко	Олена	Петрівна	ж	Кандидат фіз.-мат. наук	Вищої математики	старший викладач	2223344	21.02.1958	01.09.1998	ОНУ, ІМЕМ, 1982	Лінійна алгебра і геометрія		
8	7	7654	Сидорчук	Микола	Васильович	ч	Кандидат технічних наук	Інформаційних технологій	доцент	7154367	15.08.1971	01.03.2001	ОНПУ, ФАОТ, 1993	Об'єктно-орієнтоване програмування	Інформатика	
9	8	7543	Львівський	Олександр	Михайлович	ч	Кандидат економічних наук	Економіки	професор	7159933	24.12.1944	01.09.1994	ОНУ, ІМЕМ, 1982	Економіка		
10	9	8765	Півовар	Ірина	Юхимівна	ж	Кандидат фіз.-мат. наук	Вищої математики	доцент	7056644	08.07.1965	01.09.2015	ОНУ, ІМЕМ, 1982	Математичний аналіз		
11	10	7765	Караваєва	Тетяна	Євгенівна	ж	Кандидат фіз.-мат. наук	Інформаційних технологій	доцент	7031265	12.02.1954	01.09.1994	ОНУ, ІМЕМ, 1976	Алгоритмізація і програмування	Об'єктно-орієнтоване програмування	
12	11	4918	Іванов	Василь	Миколайович	ч	Кандидат технічних наук	Інформаційних технологій	доцент	7013456	12.10.1972	25.08.2001	КНПУ, мех-мат, 1994	Інформатика	Комп'ютерна графіка	
13	12	7104	Шевченко	Тетяна	Іванівна	ж		Історії	викладач	7157842	27.02.1983	01.09.2005	ОНУ, історичний факультет, 2005	Історія України		
14	13	6098	Щукіна	Олена	Михайлівна	ж		Інформаційних технологій	викладач	7771203	20.07.1980	27.08.2002	ОНПУ, ІКС, 1987	Алгоритмізація і програмування	Інформатика	Комп'ютерна графіка

Рисунок 2 – Дані про викладачів, підготовлені в таблиці Microsoft Excel для імпортування

Оскільки в процесі виконання коду процедури можуть виникати помилки, повідомлення про які не виводяться, то має сенс вести журнал помилок. Створимо таблицю error_log з полями: ErrID (INT), RecID (INT), Description (TEXT).

Код процедури ImportTeachers матиме вигляд:

```
DELIMITER //
CREATE PROCEDURE `ImportTeachers`() /*процедура не має параметрів*/
BEGIN
/*Оголошення змінних, які будуть використовуватися у коді процедури*/
 DECLARE vTeacherID, teachID, deptID, posID, degrID, discID1,  discID2, discID3, err INT;
 DECLARE vTable_Number, vPhone VARCHAR (10);
 DECLARE vSex VARCHAR(1);
 DECLARE vBirthDate, vRecruitmentDate DATE;
 DECLARE vSurname, vName, vPatronymic, vDegree, vDepartment, vPosition, vDiscipline1,
vDiscipline2, vDiscipline3 VARCHAR(50);
 DECLARE vEducation VARCHAR(100);
 DECLARE DONE INT DEFAULT 0;
/*Оголошення курсора */
DECLARE Cur CURSOR FOR
 SELECT DISTINCT TeacherID, Table_Number, Surname, Name, Patronymic, Sex, Degree,
Department, Position, Phone, BirthDate, RecruitmentDate, Education, Discipline1, Discipline2,
Discipline3
 FROM import_teachers;
 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET DONE = 1;
  OPEN Cur; /*Відкриття курсора */
 /* Почергове зчитування записів із таблиці, доки вони не закінчаться*/
 WHILE not DONE DO
   FETCH Cur INTO vTeacherID, vTable_Number, vSurname, vName, vPatronymic, vSex,
vDegree, vDepartment, vPosition, vPhone, vBirthDate, vRecruitmentDate, vEducation,
vDiscipline1, vDiscipline2, vDiscipline3;
   SET deptID=NULL;
   SET degrID =NULL;
   SET posID =NULL;
   SET discID1=NULL;
   SET discID2=NULL;
   SET discID3=NULL;
   SET err=0;
/* Якщо вчене звання NULL, замінити його порожнім рядком*/
IF (ISNULL(vDegree)) THEN SET vDegree="";END IF;
/*Визначення ID кафедри, посади, ступеня, дисциплін відповідно до їх текстових назв.
Якщо ID встановити не вдається, у журнал помилок запишеться повідомлення про це й
прапору помилки присвоїться 1*/
IF (ISNULL(vDepartment) OR vDepartment ="") THEN
    INSERT INTO error_log (RecID, Description)
    VALUES (vTeacherID,"Назва кафедри відсутня");
```

```
    SET err=1;
ELSE
    SELECT DepartmentID INTO deptID  FROM departments
     WHERE Department= vDepartment;
    IF (ISNULL(deptID)) THEN
     INSERT INTO error_log (RecID, Description)
     VALUES (vTeacherID,"Кафедру не знайдено");
     SET err=1;
    END IF;
END IF;
IF (ISNULL(vPosition) OR vPosition ="") THEN
  INSERT INTO error_log (RecID, Description)
   VALUES (vTeacherID,"Назва посади відсутня");
   SET err=1;
ELSE
  SELECT PositionID INTO posID  FROM positions
    WHERE Position= vPosition;
   IF (ISNULL(posID)) THEN
    INSERT INTO error_log (RecID, Description)
    VALUES (vTeacherID,"Посаду не знайдено");
    SET err=1;
   END IF;
END IF;
  SELECT DegreeID INTO degrID
    FROM degrees
   WHERE Degree= vDegree;
  IF (ISNULL(degrID)) THEN
     INSERT INTO error_log (RecID, Description)
     VALUES (vTeacherID,"Ступінь не знайдено");
     SET err=1;
  END IF;
  IF (ISNULL(vDiscipline1) OR vDiscipline1="") THEN
   INSERT INTO error_log (RecID, Description)
   VALUES (vTeacherID,"Назва дисципліни відсутня");
   SET err=1;
  ELSE
   SELECT DisciplineID INTO discID1   FROM disciplines
    WHERE Discipline= vDiscipline1;
   IF (ISNULL(discID1)) THEN
    INSERT INTO error_log (RecID, Description)
    VALUES (vTeacherID,"Дисципліну 1 не знайдено");
    SET err=1;
   END IF;
  END IF;
```

```
  IF (ISNULL(vDiscipline2)=0 AND vDiscipline2<>"") THEN
   SELECT DisciplineID INTO discID2
    FROM disciplines
   WHERE Discipline= vDiscipline2;
   IF (ISNULL(discID2)) THEN
    INSERT INTO error_log (RecID, Description)
    VALUES (vTeacherID,"Дисципліну 2 не знайдено");
    SET err=1;
   END IF;
  END IF;
  IF (ISNULL(vDiscipline3)=0 AND vDiscipline3<>"") THEN
   SELECT DisciplineID INTO discID3
    FROM disciplines
   WHERE Discipline= vDiscipline3;
   IF (ISNULL(discID3)) THEN
    INSERT INTO error_log (RecID, Description)
    VALUES (vTeacherID,"Дисципліну 3 не знайдено");
    SET err=1;
   END IF;
  END IF;
 /*Якщо помилок не було*/
 IF(err=0) THEN
 /*Якщо викладача з табельним номером у таблиці teachers ще немає*/
   IF NOT EXISTS (SELECT * FROM teachers
                WHERE Table_Number = vTable_Number)
   THEN
 /*Вставити запис у таблицю teachers*/
     INSERT INTO teachers (Table_Number, Surname, Name, Patronymic, Sex, DegreeID,
DepartmentID, PositionID, Phone, BirthDate, RecruitmentDate, Education)
     VALUES (vTable_Number, vSurname, vName, vPatronymic, vSex, degrID, deptID, posID,
vPhone, vBirthDate, vRecruitmentDate, vEducation);
   ELSE
 /*Інакше оновити запис*/
    UPDATE teachers SET Surname= vSurname, Name =vName, Patronymic= vPatronymic,
Sex= vSex, DegreeID= degrID, DepartmentID=deptID, PositionID=posID, Phone= vPhone,
BirthDate =vBirthDate, RecruitmentDate= vRecruitmentDate, Education= vEducation
    WHERE Table_Number=vTable_Number;
   END IF;
 /*Знайти ID викладача, який було вставлено або оновлено*/
SELECT TeacherID INTO teachID   FROM teachers
WHERE Table_Number = vTable_Number;
IF(ISNULL(teachID))THEN
 INSERT INTO error_log (RecID, Description) VALUES (vTeacherID,"TeacherID не знайдено");
END IF;
```

```
 /*Видалити з таблиці disc_teach записи про викладача*/
 DELETE FROM disc_teach  WHERE TeacherID= teachID;
  /*Вставити у таблицю disc_teach нові записи про викладача*/
    IF (ISNULL(discID1)=0 AND discID1<>0) THEN
     INSERT INTO disc_teach (DisciplineID, TeacherID)  VALUES (discID1, teachID);
    END IF;
    IF (ISNULL(discID2)=0 AND discID2<>0) THEN
     INSERT INTO disc_teach (DisciplineID, TeacherID)  VALUES (discID2, teachID);
    END IF;
    IF (ISNULL(discID3)=0 AND discID3<>0)  THEN
     INSERT INTO disc_teach (DisciplineID, TeacherID)  VALUES (discID3, teachID);
    END IF;
   END IF;
  END WHILE;
 CLOSE Cur;  /*Закрити курсор*/
END
```

Лабораторне завдання

1. Створити збережену процедуру, яка формуватиме список групи за назвою групи, щоб передавати до неї як параметр. Викликати (виконати) цю процедуру для групи (параметра) ІПЗ-2.01.

2. Створити за допомогою збереженої процедури телефонний довідник студентів заданого курсу, наприклад, телефонний довідник студентів першого курсу.

3. Створити збережену процедуру, яка формуватиме відомість про складання екзамену студентів певної групи за заданим предметом. Викликати (виконати) цю процедуру для групи ІПЗ-2.01 з дисципліни "Алгоритмізація і програмування" ІПЗ-2.01.

4. Створити збережену процедуру, яка отримує назву дисципліни і визначає кількість студентів, які мають з цієї дисципліни оцінки вищі за середньоарифметичну.

5. Організувати останню збережену процедуру і записати у вигляді функції.

6. Створити процедуру, яка за назвою кафедри виводитиме список викладачів кафедри із зазначенням вченого ступеня, посади і переліку дисциплін, що викладаються ними. Для спрощення коду спочатку можна створити допоміжні функції для визначення текстових назв кафедри, вченого ступеня, посади та дисципліни за їхніми ID. При відборі даних доречно об’єднати в один рядок назви дисциплін для кожного викладача, щоб дані не дублювалися.

7. За допомогою курсора створити процедуру імпортування даних про викладачів. Для цього спочатку створити відповідну Excel-таблицю такої самої структури та реалізувати процедуру імпортування.

Лабораторна робота 8
Перенесення БД на хостинг

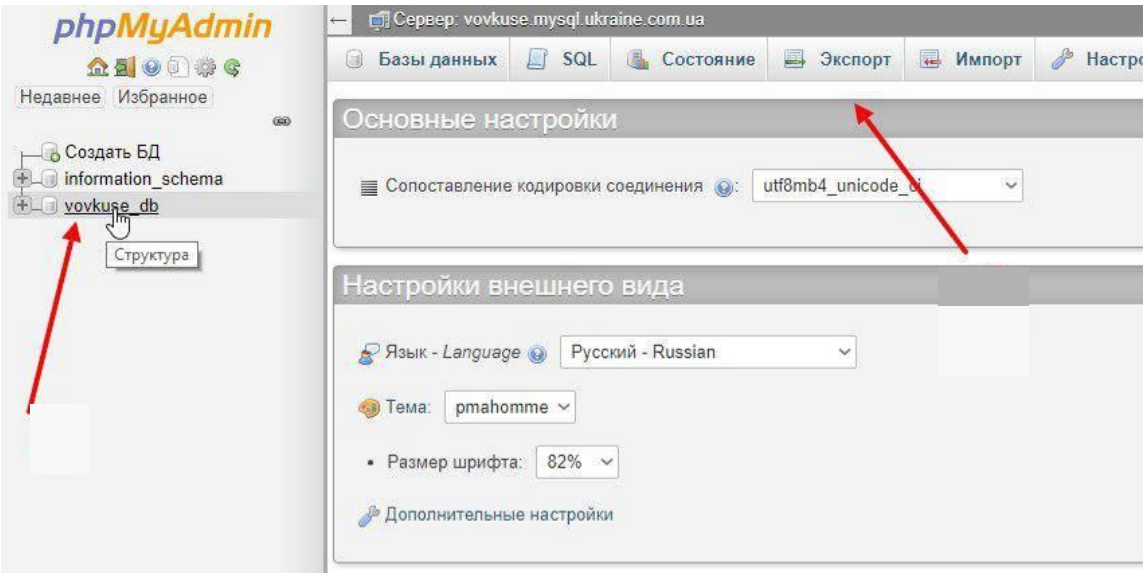
Теоретичні відомості

За потреби перенесення бази даних з одного хостингу на інший або з локального сервера на віддалений, або з одного комп'ютера на інший загальний алгоритм дій може бути таким:

1) експортувати базу даних у .sql файл;
2) перенести цей файл на інший комп'ютер, якщо це потрібно;
3) створити нову базу даних на новому хостингу або локальному сервері;
4) імпортувати у створену базу дані з .sql файлу.

Далі розглянемо докладно ці дії:

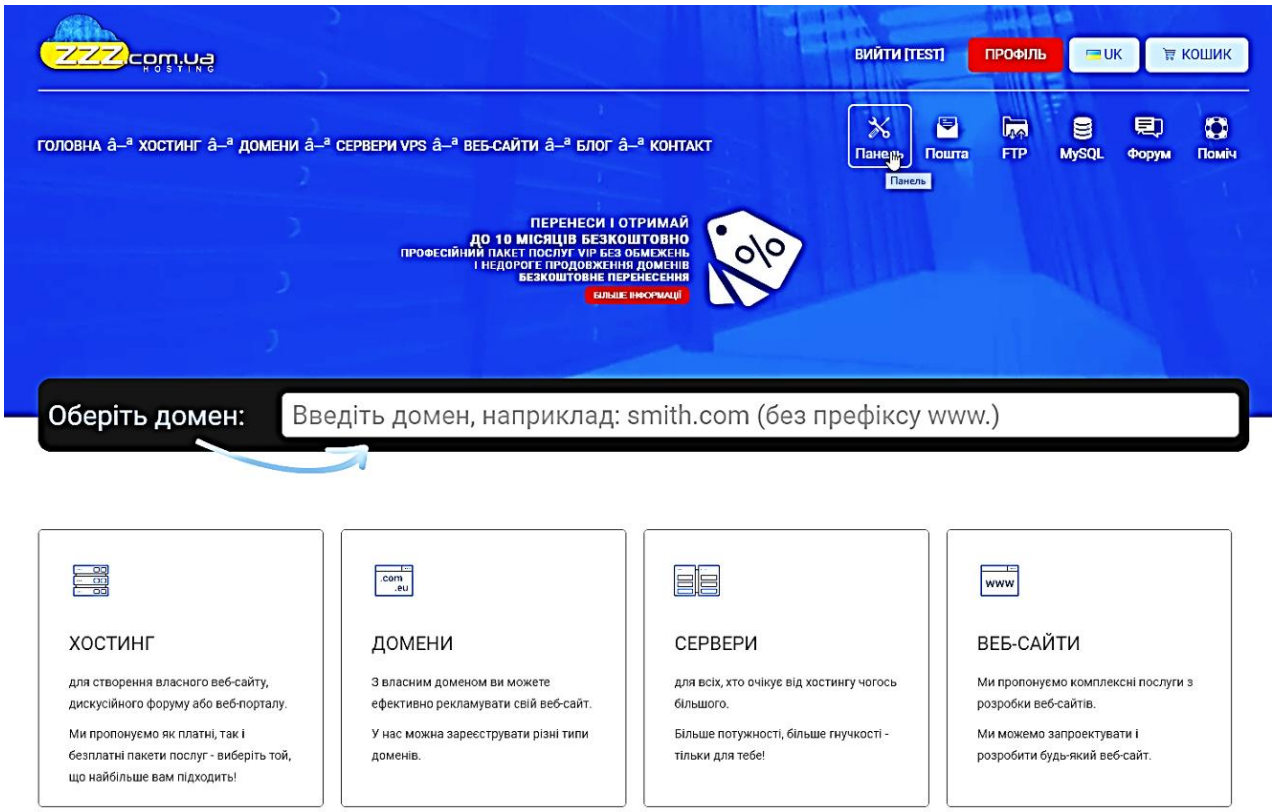
1. Для експортування БД у .sql файл треба у вікні phpMyAdmin на лівій панелі вибрати базу даних, яку треба перенести. У правій частині вікна натиснути команду *Экспорт*.



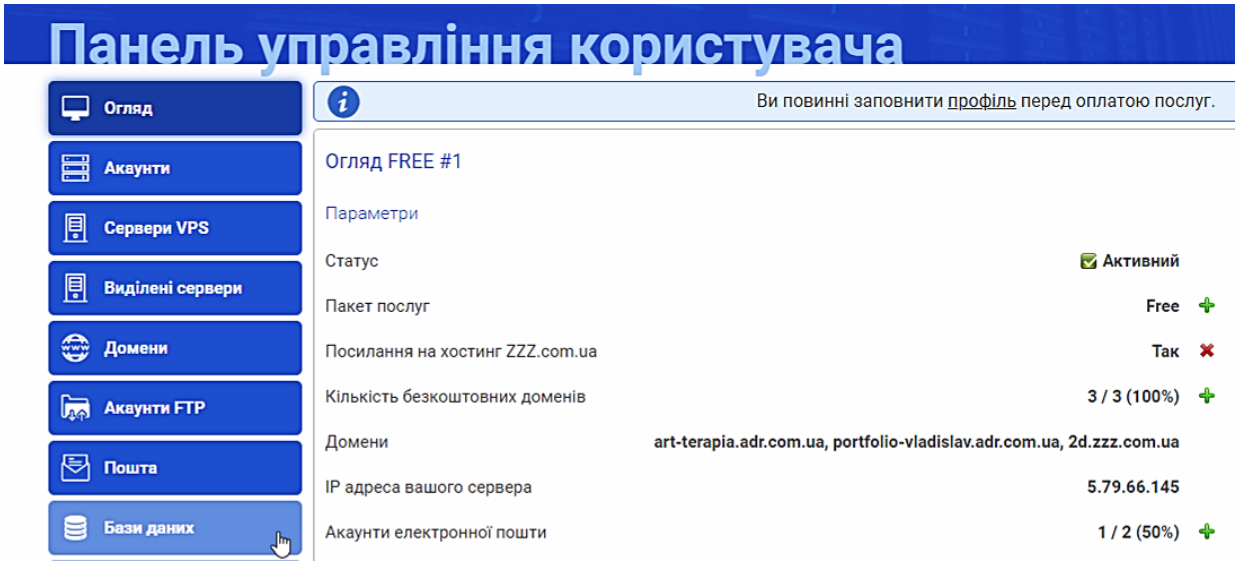
На вкладці *Экспорт* треба натиснути кнопку *Вперёд*.

Внаслідок зазначених дій буде створено .sql файл з базою даних, готовий для перенесення.

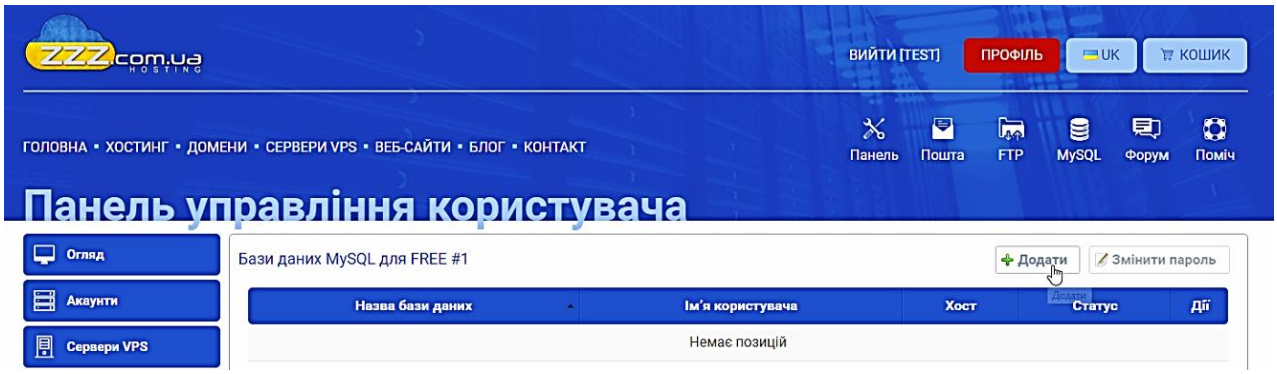
2. Далі треба зайти на новий хостинг або локальний сервер на іншому комп'ютері, куди планується перенести базу. Як приклад замовимо безкоштовні хостингові послуги від zzz.com.ua, де підтримується робота з базами даних MySQL і SQLite та широкий вибір версій PHP.



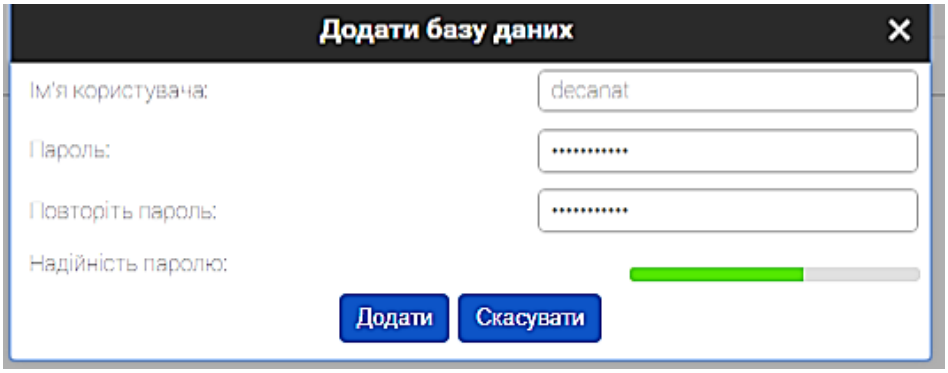
Після вибору імені домену та заповнення анкети для створення хостингового акаунта можна перейти до панелі управління користувача. У меню найти і клацнути на команду *Бази даних*. Оскільки інтерфейс на різних хостингах відрізняється, тому місце розташування цієї команди може різнитися.



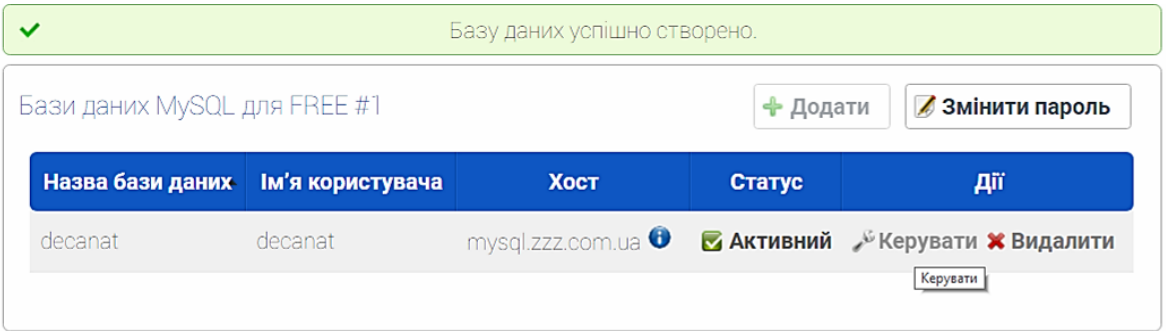
Клацання команди *Бази даних* відкриє список баз даних, якщо такі є на хостингу. У цьому вікні треба натиснути кнопку *Додати*.



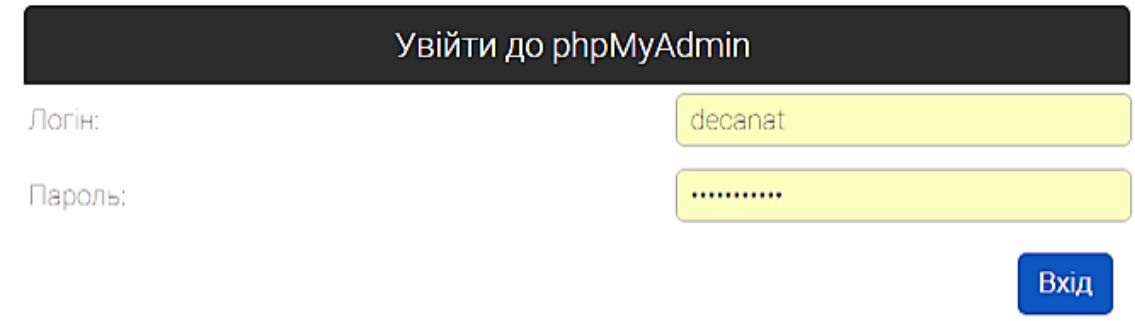
Далі у діалоговому вікні треба ввести інформацію: ім’я та пароль для доступу до БД. На різних хостингах набір таких даних різниться.



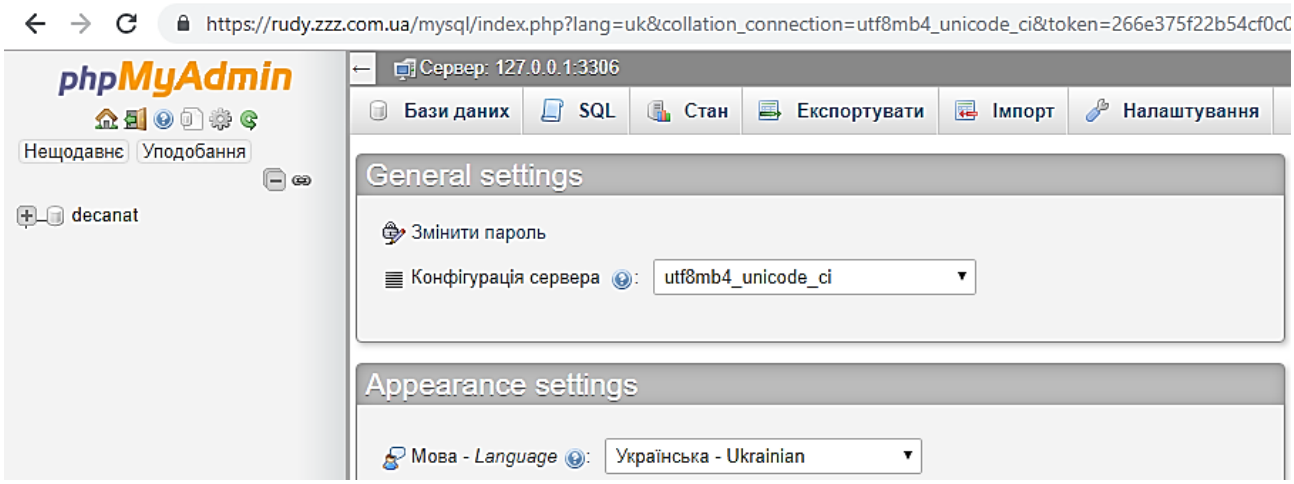
Після того як база даних створена, в неї треба зайти. На одних хостингах у списку баз даних є кнопка для входу, на інших треба знайти в меню phpMyAdmin і запустити його. В нашому прикладі роботи з zzz.com.ua треба натиснути опцію *Керувати*.



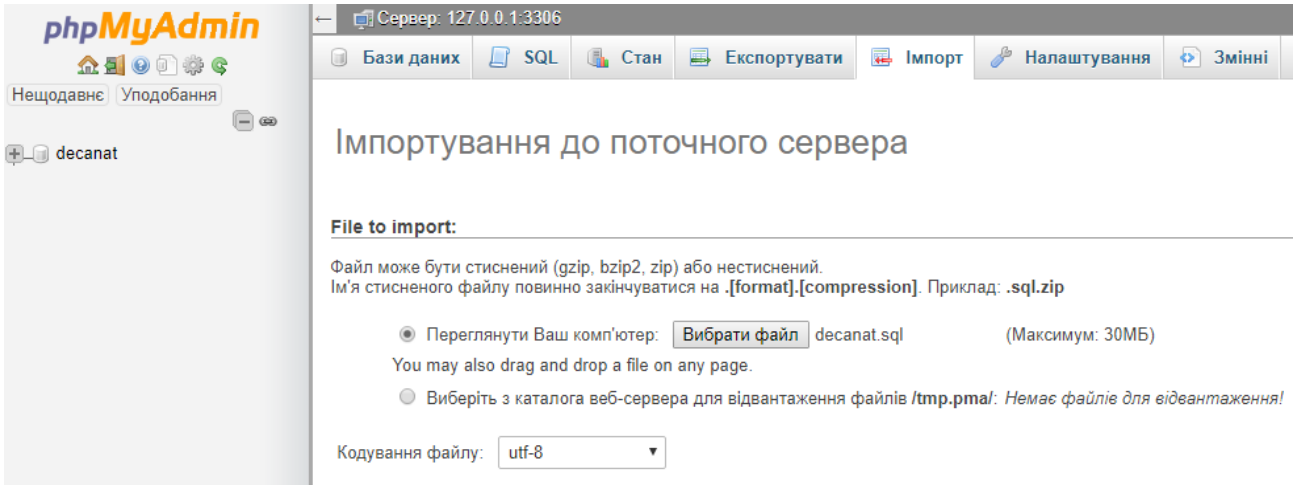
Для подальшого переходу до вікна phpMyAdmin треба підтвердити пароль.



Після цього відкриється вікно phpMyAdmin зі створеною, але поки що порожньою БД decanat.



Треба перейти на вкладку *Імпорт*, вибрати .sql файл з базою даних і натиснути кнопку *Вперед*.



Унаслідок цих дій база даних буде розгорнута на новому хостингу.

Слід зауважити, що багато безкоштовних хостингів не підтримують бази даних InnoDB (тобто із зовнішніми ключами). Якщо імпортувати до такого хостингу базу даних зі зв'язаними таблицями, зв'язки будуть втрачені.

Лабораторне завдання

1. Для перенесення бази даних експортувати базу даних у .sql файл.
2. Замовити безкоштовні хостингові послуги, де підтримується робота з базами даних MySQL, наприклад, від zzz.com.ua чи то інші.
3. Після вибору імені домену та заповнення анкети для створення хостингового акаунта перейти до панелі управління користувача. У меню найти і клацнути команду *Бази даних*. Оскільки інтерфейс на різних хостингах відрізняється, тому місце розташування цієї команди може різнитися.
4. Додати базу даних на хостинг, задавши ім'я та пароль для доступу до БД. На різних хостингах набір таких даних різниться.
5. Після того як БД створена, в неї треба зайти та імпортувати sql-файл зі своєю базою даних, тобто розгорнути свою БД на новому хостингу.

Навчально-методичне видання

Трофименко Олена Григорівна,
Прокоп Юлія Віталіївна,
Буката Людмила Миколаївна

СКБД MySQL і робота з інтернет базами даних засобами phpMyAdmin

Методичні вказівки
для виконання лабораторних робіт
з дисципліни
"Створення та опрацювання баз даних"

Комп'ютерне верстання Гардиман Ж.А.

Коректор Кодрул Л.А.

ДУІТЗ
65023, м. Одеса, вул. Кузнечна, 1
Обсяг 3.03 др.арк., елект. версія