КЫРГЫЗ РЕСПУБЛИКАСЫНЫН БИЛИМ БЕРҮҮ ЖАНА ИЛИМ МИНИСТРЛИГИ
ОШ МАМЛЕКЕТТИК УНИВЕРСИТЕТИ
МАТЕМАТИКИ ЖАНА ИНФОРМАЦИЯЛЫК ТЕХНОЛОГИЯЛАР ФАКУЛЬТЕТИ
«АССТ» КАФЕДРАСЫ

ОКУУ-МЕТОДИКАЛЫК КОМПЛЕКС

Дисциплина: «Берилгендер базасы »

Багыты: “ССИБИ”

Түзгөн: ИТАС кафедрасынын улук окутуучусу Г. Сейитказыева

Ош-2020

КЫРГЫЗ РЕСПУБЛИКАСЫНЫН БИЛИМ БЕРҮҮ ЖАНА ИЛИМ МИНИСТРЛИГИ
Ош мамлекеттик университети
Математика жана информациялык технологиялар факультети
АССТ кафедрасы

«Макулдашылды»
МИТ факультетинин Методикалык кеңешинин төрайымы
Ф.-м.и.к., доцент: Д. Зулпукарова
«___» _____ 2020-ж.

«Бекитилди»
АССТ кафедрасынын
2020-жылдын 29-августунда
өткөрүлгөн
№1-протоколунда
Каф. башч., ф.-и.к..: Молдояров У.

ЖУМУШЧУ ПРОГРАММА

Дисциплина: Берилгендер базасы

Багыты: «ССИБИ»

Окутуунун формасы: Күндүзгү

Окуу жылы: 2020-2021

Окуу планы боюнча сааттардын эсеби

Берилгендер базасы	Сааттардын саны					СӨ АИ	Отчеттуулук
	Баары	Аудиториялык сабактар					
		ауд. Сабак	*Лекц*	*прак*	*Лаб*		
2-курс, 3-сем.	120	60	*30*	*10*	*20*	60	Экзамен

Силлабус мамлекеттик билим берүү стандартынын негизги билим берүү программасынын жана ОшМУнун №19 бюллетенин негизинде түзүлдү.

Түзгөн: улук окутуучу _____ Г.И. Сейитказыева

Ош – 2020

1. . Дисциплинанын максаты

Курсту окутуунун максаты- Берилгендер базасы жөнүндө жалпы түшүнүк алуу. Реляциондук берилгендер базасынын үстүндө иштөөгө үйрөнүү.Студенттер MySql тилинде запросторду түзүү методдору менен иштөөнү үйрөнүшөт, берилгендердин базасын башкаруу системасын (программасын) түзүү билгичтиги калыптанат. Медицина тармагында берилгендер базасы менен иштей алат.

2. Окутууда калыптандырылуучу компетенциялар

ЖК-3: Заманбап билим берүүчүлүк жана маалыматтык технологияларын пайдаланып өз алдынча жаңы билимдерге ээ болууга жөндөмдүү.

КК-5:Медициналык маалыматтар системаларынын жана берилгендер базасынын абалын жана өнүгүүсүн анализдөөгө жөндөмдүү

ИК-1: Максатты коюу, ага жетүү жолдорун, маалыматты анализдөөгө, кабылдоого жөндөмдүү.

ПК-12:Медициналык маалыматтар системасына жетүүнү башкарууну билет

Дисциплинаны окуп-үйрөнүп, өздөштүрүүнүн натыйжасында студент төмөнкү көрсөткүчтөрдүн деңгээлине ээ болуусу керек:

Билүүсү керек:

- Түрдүү маалыматтар системасы жана үчүн берилгендер базасын жана ББББСын;
- Берилгендерди берүү деңгээлдерин, берилгендердин моделин, берилгендерди берүү моделдерин кайра иштетүү методдору;
- MySql тилинин командаларын.

Билгичтиктер:

- Маалыматты башкаруу каражаты катары компьютерде иштей алуу;
- Берилгендер базасы жана ББББСлары менен иштөө;
- Берилгендер базасын жана моделди иштеп чыгуу;
- MySql тилинде берилгендер базасын түзүү.

ЭЭ болуу:

- Берилген предметтик областтын моделин тургузуп анын берилгендер базасын түзүү жана запросторду формировкалоо, жыйынтыктарды алуу.
- Берилгендер базасын түзүүнүн принциптерин, берилгендер базасыны жашоо циклын, берилгендер базасында берилгендерди кайра иштетүү процессин уюштуруу;

3. НББПнын структурасындагы дисциплинанын орду

«Берилгендер базасы» дисциплинасы окуу планынын профессионалдык циклинин базалык бөлүмүндө жайгашкан. Дисциплина адисти калыптандыруучу негизги дисциплиналардын бири болуп эсептелип, 3-семестрде окутулат.

4. Дисциплинанын технологиялык картасы

Баары	Ауд. Саат	1-модул (58 с., 30 б.)				2-модул (58 с., 30 б.)				Жыйынт. текш. (ЖТ) (40 б.)					**Жалп**
		Ауд. саат		СӨАИ	**1-аралыктагы текш. (АТ1)**	Ауд. Саат		СӨАИ	**2-аралыктагы текш.(АТ2)**	Лекция	Лаборатория	СӨАИ	Сыйлык балл	**ЖТ (ИК)**	
		Лекция	Лаборатория			Лекция	Лаборатория								
108	54	14	14	28		14	12	26							
Баллдар		18б.	12 б.	30 б		18	12 б.	30 б		36	24		**10**	**40б.**	**100**
УТ=(Лек+Лаб+ +СӨАИ)/3, М1=(УТ1+УТ2+ +АТ1)/3				УТ=(Лек+Лаб+ +СӨАИ)/3, М2=(УТ3+УТ4+ +АТ2)/3				ЖТ=(Лек+Лаб+ +СӨАИ)/3, Экз=(М1+М2+ +ЖТ)/3+10							100

Ауд. – аудиториялык, УТ – учурдагы текшерүү, АТ – аралык текшерүү, М – модулдар, СӨАИ – студенттин өз алдынча иши, ЖТ – жыйынтыктоочу текшерүү.

5. Дисциплинанын компетенциялар картасынын модулдарда жана бөлүмдөрдө берилиши

6. Дисциплина боюнча баллдарды топтоонун картасы

Баллдарды топтоонун картасы – сабактардын бардык түрлөрүндөгү текшерүү боюнча канча балл (максималдуу) ала тургандыгы жөнүндө студенттерге жеткирилүүчү маалымат.

Студенттер баллдарды модулдарда төмөнкүдөй топтошот:

1-модулда эки учурдагы текшерүү (УТ1, УТ2) жана бир аралыктагы текшерүү (АТ1) уюштурулат. Ар бир текшерүү үчүн 30 баллдык баалоо системасы колдонулат. Баллдар тапшырмалар менен кошо тааныштырылат.

УТ1 текшерүүсү 4-жумада, УТ2 текшерүүсү 8-жумада уюштурулат, ал эми аралыктагы текшерүү дагы 8-жумада уюштурулат.

УТ1 деп 4-жумага чейин өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $УТ1 = \frac{Лек + Лаб + СӨАИ}{3}$.

УТ2 деп сабак башталгандан баштап 4-жумадан 8-жумага чейин өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $УТ2 = \frac{Лек + Лаб + СӨАИ}{3}$.

Ведомостко жана журналга УТ1, УТ2 лердин жыйынтыктары коюлат.

Модулдар, бөлүмдөрдүн номери жана аталышы	**Компотенциялар**					
	ЖК-3	**ИК-1**	**КК-5**	**КК-10**	**КК-12**	**Компетен-циялар-дын жалпы саны**
1-бөлүм. СУБД жөнүндө түшүнүк.MySql программалоо чөйрөсү			+			1
2-бөлүм. MySql тилинин операторлору	+					2
3- бөлүм. MySql тилинде байланыштар.Индекстер.	+			+		2
4-бөлүм. My sql тилинде функиялардын колдонулуштары						1

8-жумада 1-модулдун материалдары боюнча 1-аралыктагы текшерүү уюштурулат. Мында 1-модулда өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $АТ1 = \frac{Лек + Лаб + СӨАИ}{3}$.

1-модулда баалоо учурдагы текшерүүлөрдүн жана 1-аралыктагы текшерүүнүн арифметикалык орточосу менен аныкталат: $M1 = \frac{УТ1 + УТ2 + АТ1}{3}$.

2-модулдагы баалоо 1-модулдагы баалоо сыяктуу эле аткарылат.

Жыйынтыктоочу текшерүүдө семестрде ичинде өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз:

$$ЖТ = \frac{Лек + Лаб + СӨАИ}{3}$$

Экзамендеги баалоо модулдардын жана жыйынтыктоочу текшерүүнүн арифметикалык орточосу менен сыйлык (C) баллдардын суммасы менен аныкталат:

$$Экз = \frac{M1 + M2 + ЖТ}{3} + C.$$

Баллдар тапшырмаларды берүүдө кошо көрсөтүлөт. C – сыйлык баллдар «Билимди баалоо системасы» жөнүндөгү жободо көрсөтүлгөн.

7. Дисциплинанын программасы

СУБД жөнүндө жалпы түшүнүк. Берилгендер базасы. Тургузуу принциптери. Берилгендердин моделдери. Реляциондук берилгендер базасы. Mysql структураланган запростор тили.Берилгендер базасын түзүү. Таблицалар түзүү, маалыматтарды кийирүү.Таблицалардын структурасын өзгөртүү. Берилгендердин типтери. Таблицалардын ортосундагы байланыштар.

8. Лекциялык жана лабораториялык сабактардын календардык тематикалык планы

№	Темалар	Лекц. саат	Лаб .саат	Прак. саат
1.	Киришүү. СУБД жөнүндө жалпы түшүнүк.	2	1	1
2.	Mysql тили менен таанышуу. Mysqlде берилгендер менен иштөө.	2	1	1
3.	Берилгендер базасын түзүү жана тандоо. MySQL тилинин колдонулушу жана синтаксиси.	4	2	1
4.	Таблицалар. Таблицалардын структурасын өзгөртүү. Берилгендердин типтери	2	2	1
5.	SELECT, INSERT, UPDATE, WHERE операторлору, мисалдар	4	2	1
6.	ORDER BY, GROUP BY, DEFAULT, ALTER TABLE операторлорунун колдонулуштары	4	2	1
7.	VIEW, UNION операторлору	2	2	1

8.	Таблицалардын ортосундагы байланыштар. Индекстер. Сырткы жана ички ключтөр	4	2	1
9.	Таблицаларды бириктирүү. INNER JOIN, LEFT JOIN, RIGHT JOIN операторлору.	2	2	1
10.	FULL JOIN, CROSS JOIN операторлору. Кесилиш биригүү жөнүндө түшүнүк.	2	2	1
11.	My sql тилинде функиялардын колдонулуштары	2	2	1
	Баары:	30	20	10

9. Сабактардын түрлөрү боюнча календардык-тематикалык план

9.1. Лекциялык сабактар

1-бөлүм. MySql программалоо чөйрөсү

1-2-лекция. СУБД жөнүндө жалпы түшүнүк. Берилгендер базасы.Берилгендердин моделдери. Иерархиялык берилгендер. Тармактык берилгендер. Реляциондук берилгендер базасы. Биринчи программа (1-ЛИ).

Лекциянын планы:

1). Берилгендер базасынын классификациясы;
2). MySql программалоо чөйрөсү;
3). Биринчи программа (1-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [1, 2,3]. Кошумча: [3, 4]

3-лекция. Mysql тили менен таанышуу. Mysqlде берилгендер менен иштөө.

Лекциянын планы:

1) MySqlни орнотуу.
2) Программаны жүктөө;
3) Mysqlде берилгендер менен иштөө.
4). Биринчи программа (1-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [1, 3]. Кошумча: [3, 4]

4-5-лекция Берилгендер базасын түзүү жана тандоо. MySQL тилинин колдонулушу жана синтаксиси. Экинчи программа (1-ЛИ).

Лекциянын планы:

1). Берилгендер базасын түзүү жана өчүрүү;
2). MySql тилинде берилгендер базасын тандоо;
3) MySQL тилинин синтаксиси.

4). Экинчи программа (2-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [3, 2]. Кошумча: [3, 4]

6-лекция. Таблицалар. Таблицалардын структурасын өзгөртүү. Берилгендердин типтери.

Лекциянын планы:
1). Таблица түзүү жана өчүрүү;
2). Берилгендердин типтери
3). үчүнчү программа (5-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [1, 2,3]. Кошумча: [3, 4]

7-лекция. SELECT, INSERT, UPDATE, WHERE операторлору, мисалдар

Лекциянын планы:
1). SELECT операторунын кызматы
2). INSERT оператору
3). UPDATE, WHERE операторлору
4). (6-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [1, 2,3]. Кошумча: [3, 4]

8-лекция. ORDER BY, GROUP BY, DEFAULT, ALTER TABLE операторлорунун колдонулуштары

Лекциянын планы:
1). ORDER BY, GROUP BY, операторлорунун колдонулуштары
2). DEFAULT, ALTER TABLE операторлорунун колдонулуштары
3). (7-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [1, 2,3] Кошумча: [3, 4]

9-лекция. VIEW, UNION операторлору

Лекциянын планы:
1). VIEW операторунын кызматы
2). UNION операторлору
3). (8-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:
Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.
Адабияттар Негизги: [2,3]. Кошумча: [3, 4]

10-11-лекция. Таблицалардын ортосундагы байланыштар. Индекстер. Сырткы жана ички ключтөр

Лекциянын планы:

1). Байланыштардын түрлөрү

2). PRIMARY KEY ички , FOREYGN KEY сырткы ачкыч жөнүндө түшүнүк;

3). (10-11-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:

Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.

Адабияттар Негизги: [1, 2]. Кошумча: [3, 4]

12-лекция. Таблицаларды бириктирүү. INNER JOIN, LEFT JOIN, RIGHT JOIN операторлору.

Лекциянын планы:

1). INNER JOIN, операторлору.

2). LEFT JOIN операторлору.

3). RIGHT JOIN операторлору.

4). (12-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:

Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.

Адабияттар Негизги: [1, 2,3]. Кошумча: [3, 4]

13-лекция. FULL JOIN, CROSS JOIN операторлору. Кесилиш биригүү жөнүндө түшүнүк.

Лекциянын планы:

1). FULL JOIN операторлору..

2). CROSS JOIN операторлору.

3). (13-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:

Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.

Адабияттар Негизги: [1, 3]. Кошумча: [3, 4]

14-лекция. My sql тилинде функиялардын колдонулуштары

Лекциянын планы:

1). Математикалык функциялар

2). Дата жана время функциялары

3). Жолчолук функциялар

4). (14-ЛИ).

Өтүлгөн материалды бышыктоо жана билимди текшерүүнүн формалары:

Негизги түшүнүктөр боюнча сурамжылоо, экспресс-суроо, текшерүү иш.

Адабияттар Негизги: [1, 2,3]. Кошумча: [3, 4]

9.2. Лабораториялык сабактар

Лабораториялык сабактарда ишти аткаруу төмөнкү этаптардан турат:

1). Маселенин моделин (макетин) түзүү;
2). Программанын кодун түзүү;
3). Каталыктарды алдын алуу;
4). Программаны сыноо;
5). Жыйынтыкты коргоо.

Колдонулуучу адабияттар: Негизги: [1, 2]. Кошумча: [3, 4]

1-ЛИ. MySql тилин орнотуу.
2-ЛИ.Берилгендер базасын жана таблица түзүү.
3-ЛИ. Берилгендердин типтери.
4-ЛИ.Таблицаларды түзүү жана маалыматтар менен толтуруу
5-ЛИ.Берилгендерди тандоо **SELECT-операторү**
6-ЛИ. Камтылган запростор
7-ЛИ. Таблицаларды бириктирүү (ички жана сырткы бириктирүү)
8-ЛИ.Жазууларды группировкалоо жана COUNT() функциясы
9-ЛИ.Берилгендерди жаңылоо, өчүрүү жана редактирлөө

9.3. Студенттердин өз алдынча иштери

Студенттердин өз алдынча иштери пландоодо жана тапшырмаларды даярдоодо Блумдун 6 деңгээлдүү таксономиясы (билүү, түшүнүү, колдонуу, анализ, синтез, баалоо) колдонулат.

9.3.1. Билүү, түшүнүү жана колдонуу үчүн берилген тапшырмалар

1. Берилгендердин иерархиялык моделинин өзгөчөлүктөрү
2. Берилгендердин тармактык моделинин өзгөчөлүктөрү
3. Таблица түзүү.
4. Таблицанын ичин толтуруу, толтуруунун жолдору
5. MySQLдин тилинин структурасы
6. Клиент-сервердик архитектура
7. Структура дистрибутивдердин структурасы
8. Официалдуу документация

9.3.2. Анализдөө жана синтездөө үчүн берилген тапшырмалар

9. MySQLди орнотуу
10. MySQLдин иш жөндөмдүүлүгүн текшерүү
11. MySQLди Linux ОСна орнотуу
12. Конфигурациялык файл
13. Берилгендердин каталогун жылдыруу
14. Учрдагы версиясын жаңылоо
15. Обзор утилит MySQL утилитасынын обзору
16. *mysql* утилитасы
17. Командалык жолчо

18.Кодировканы ылайыктоо
19. *SELECT* и *LOAD DATA операторлорунун колдонулушу*
20.BACKUP TABLE *и* RESTORE TABLE

9.3.3. Баалоо үчүн берилген тапшырмалар

21.Таблицаларды түзүү жана маалыматтар менен толтуруу жолдору
22. Таблицалардын ортосундагы байланыштарды түзүү
23.Бирден –бирге байланышы
24.Бирден – көпкө байланышы
25.Көптөн –көпкө байланышы
26.SELECT-операторун колдонулушу
27.Функциялардын колдонулушу
28.IF… THEN… ELSE…операторлору
29.Математикалык функциялар
30. Дата жана время функциялары
31. Жолчолук функциялар

Студенттер өз алдынча тапшырмаларды төмөндөгү методдордун бири менен коргойт (кафедрадагы кезекчилик мезгилинде, сабактан кийин, ишемби күнү, модулдук жумада):

- Презентация;
- Реферат;
- Оозэки баяндоо;

Бланкалык же компьютердик тестирлөө.

10. Жыйынтыктоочу экзамендеги тапшырмалар (үлгү)

1. Берилгендер базасын түзүү
2. Берилгендер базасын тандоо
3. Таблица түзүү
4. Берилгендердин сандык тиби
5. Берилгендердин жолчолук тиби
6. Берилгендердин календардык тиби
7. NULL берилгендер тиби
8. Таблицанын структурасын көрүү
9. Мамычалардын параметлери
10. Берилгендер базасынын, таблицалардын талаалардын мүмкүн болгон аттары
11. WHERE операторлору, мисалдар
12. ORDER BY, GROUP BY операторлору, мисалдар
13. DEFAULT, ALTER TABLE операторлорунун колдонулуштары
14. VIEW, UNION операторлору
15. Таблицанын параметрлери
16. Убактылуу таблицалар
17. Таблицанын көчүрмөсүн түзүү
18. Таблицанын оперативдик эске жайгаштыруу
19. Бир нече таблицаларды бир таблицага бириктирүү
20. Таблицаларды өчүрүү

21. Таблицаны өзгөртүү
22. Талааларды кошу
23. Талааны өчүрүү
24. Бар талаалардын атын өзгөртүү
25. Таблицанын атын өзгөртүү
26. Таблицанын параметрлерин өзгөртүү
27. Таблицаны калыбына келтирүү
28. REPAIR TABLE оператору
29. Таблицанын текшерүүчү суммасы
30. Индекстер
31. Ички жана сырткы ключтөр
32. AUTO_INCREMENT атрибутунун жардамында уникалдуу индексти түзүү
33. Кадимки жана уникалдуу индекстер
34. Таблицага индекстерди кошуу жана өчүрүү
35. Индекстерди калыбына келтирүү
36. *INSERT* опреаторнунун колдонулушу
37. TIMESTAMP берилгендер тиби
38. Сандык маанилерди толтуруу
39. Жолчолук маанилерди толтуруу
40. Календардык маанилерди толтуруу
41. *UNIXSTAMP* форматында маанилерди толтуруу
42. AUTO_INCREMENT механизми
43. Эсептөөчү маанилерди толтуруу
44. *INSERT... SELECT* операторунунун колдонулушу
45. Берилгендерди өчүрүү
46. DELETE оператору
47. TRUNCATE оператору
48. Бир нече таблицалардан өчүрүү
49. Бир нече таблицалардан каскаддык өчүрүү
50. Жазууларды жаңылоо
51. CASCADE оператору
52. UPDATE көп таблицалуу оператору

11. Окуу-методикалык камсыздалышы

11.1. Негизги адабияттар

1. Дейт К. Дж. Введение в системы баз данных. – Киев: Диалектика, 1998.
2. Мартин Дж. Организация баз данных в вычислительных системах – М.: Мир, 1980. – 662 с.
3. Дейт К. Дж. Руководство по реляционной СУБД DB2. – М.: Финансы и статистика, 1988. – 320 с.

11.2. Кошумча адабияттар

4. Максим Кузнецов, Игорь Симьдянов MySQL на примерах , Санкт-Петербург -2007.

Дополнительная литература

5. Хансен Г., Хансен Д. Базы данных и управление. – М.: Бином, 1999.
Конноли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. – М.– С./П.– К., 2000.

11.3. а) Толук тексттүү берилгендер базасы

1) ЭБС «Руконт». Режим доступа [http://www.rucont.ru/].
2) Ресурс Цифровые учебные материалы. Режим доступа [http://abc.vvsu.ru/].

б) Интернет-ресурстар

1. http://www.sql.ru/ - информационных ресурс для программистов SQL
2. http://programmersforum.ru - форум программистов

12. Балл коюу саясаты

Учурдагы, аралыктагы жана жыйынтыктоочу текшерүүлөр «Билимди баалоо» жөнүндөгү жобо менен аныкталат.

Студенттин билим деңгээли 100 баллдык системада төмөнкү эрежеге ылайык коюлат:

Рейтинг (балл)	Тамгалык система боюнча баа	GPA боюнча баалоонун цифралык эквиваленти	Традициялык системе боюнча баа
87 – 100	A	4,0	Эң жакшы
80 – 86	B	3,33	Жакшы
74 – 79	C	3,0	
68 – 73	Д	2,33	Канааттандыраарлык
61 – 67	E	2,0	
31 -60	FX	0	Канааттандыраарлык эмес
0 – 30	F	0	

Экзаменде бааны коюуда объективдүүлүк жана акыйкаттуулук принциптеринин негизинде студенттин билиминин сапаты бардык тараптан анализделип, модулдук-рейтингдик системанын жобосуна ылайык коюлат.

КЫРГЫЗ РЕСПУБЛИКАСЫНЫН БИЛИМ БЕРҮҮ ЖАНА ИЛИМ МИНИСТРЛИГИ

Ош мамлекеттик университети

Математика жана информациялык технологиялар факультети

АССТ кафедрасы

«Макулдашылды»
МИТ факультетинин Методикалык кеңешинин төрайымы
Ф.-м.и.к., доц.: Д.Зулпукарова
«___» _____ 2020-ж.

«Бекитилди»
АССТ кафедрасынын
2020-жылдын 29-августунда
өткөрүлгөн №1-протоколунда
Каф. башч.,ф.-м.и.к.: У.Молдояров

студенттин окуу программасы

СИЛЛАБУС (syllabus)

Дисциплина: Берилгендер базасы

Багыты: «ССБМИ»

Окутуунун формасы: Күндүзгү

Окуу жылы: 2020-2021

Окуу планы боюнча сааттардын эсеби

Берилгендер базасы		Сааттардын саны				СӨАИ	Отчеттуулук
	Баары		Аудиториялык сабактар				
		ауд. Сабак	*Лекц*	*Прак.*	*Лабор.*		
2-курс, 3-сем.	120	60	*30*	*10*	*20*	60	Экзамен

Силлабус мамлекеттик билим берүү стандартынын негизги билим берүү программасынын жана ОшМУнун №19 бюллетенин негизинде түзүлдү.

Түзгөн: улук окутуучу _____ Г.И. Сейитказыева

Ош – 2020

1. Окутуучу жөнүндө маалымат

Лектор-окутуучу: Сейитказыева Гульнара Имамалиевна –ИТАС каф. улук окутуучусу, МИТ факультети, ОшМУ

Эмгек стажы – 18 лет.

Билими – жогорку, ОшМУ, МИТ факультетин 2002-жылы бүтүргөн.

Жум. тел: 0322221185, жумуш орду: Ленин көчөсү 331,ОшМУнун башкы корпусу, 205-каб.

Моб. телефон : 0773 71-40-76,

E-mail: seitkazyeva@mail.ru

2. Дисциплинанын максаты

Берилгендер базасы жөнүндө жалпы түшүнүк алуу. Реляциондук берилгендер базасынын үстүндө иштөөгө үйрөнүү.Студенттер MySql тилинде запросторду түзүү методдору менен иштөөнү үйрөнүшөт, берилгендердин базасын башкаруу системасын (программасын) түзүү билгичтиги калыптанат. Экономикалык, математикалык, IT технология багытындагы жана ишмердүүлүк чөйрөдөгү кесиптик маселелерди иш жүзүндө чече алууга даяр болот.

Дисциплинаны өздөштүрүүдө студент төмөнкү окутуу натыйжаларына жетишет:

3. Диципликанаы өздөштүрүүнүн натыйжалары

Дисциплинаны окуп-үйрөнүп, өздөштүрүүнүн натыйжасында студент төмөнкү көрсөткүчтөрдүн деңгээлине ээ болуусу керек:

Билүүсү керек:

- Түрдүү маалыматтар системасы жана үчүн берилгендер базасын жана ББББСын;

- Берилгендерди берүү деңгээлдерин, берилгендердин моделин, берилгендерди берүү моделдерин кайра иштетүү методдору;

- MySql тилинин командаларын.

Билгичтиктер:

- Маалыматты башкаруу каражаты катары компьютерде иштей алуу;
- Берилгендер базасы жана ББББСлары менен иштөө;
- Берилгендер базасын жана моделди иштеп чыгуу;
- MySql тилинде берилгендер базасын түзүү.

ЭЭ болуу:

- Берилген предметтик областтын моделин тургузуп анын берилгендер базасын түзүү жана запросторду формировкалоо, жыйынтыктарды алуу.
- Берилгендер базасын түзүүнүн принциптерин, берилгендер базасыны жашоо циклын, берилгендер базасында берилгендерди кайра иштетүү процессин уюштуруу;

.

4. Пререквизиттер: Информатика, *программалоонун негиздери*

5. Постреквизиттер: Информационные системы в здравоохранении.

6. Технологиялык карта

Баары	Ауд. Саат		1-модул (24 с., 30 б.)					2-модул (22 с., 30 б.)					Жыйынт. текш. (ЖТ) (40 б.)					Жалпы балл
		Ауд. Саат			СӨАИ	**1-аралыктагы**		Ауд. саат			СӨАИ	**2-аралыктагы**						
		Лекция	Лабора	Прак.				Лекция	Лабора	Прак.			Лекция	Лаборатория	СӨАИ	Сыйлык балл	**ЖТ (ИК)**	

223	151	15	10	5	23		15	10	5	23							
Баллдар		18 б.	10б.	2б.	30 б		18	10 б.	2 б.	30 б		24	22	**46**	**10**	**40б.**	**100**
УТ=(Лек+Лаб+ +СӨАИ)/3, М1=(УТ1+УТ2+ +АТ1)/3					УТ=(Лек+Лаб+ +СӨАИ)/3, М2=(УТ3+УТ4+ +АТ2)/3				ЖТ=(Лек+Лаб+ +СӨАИ)/3, Экз=(М1+М2+ +ЖТ)/3+10							100	

Ауд. – аудиториялык, УТ – учурдагы текшерүү, АТ – аралык текшерүү, М – модулдар, СӨАИ – студенттин өз алдынча иши, ЖТ – жыйынтыктоочу текшерүү.

7. Баллдарды топтоонун картасы

Баллдарды топтоонун картасы – сабактардын бардык түрлөрүндөгү текшерүү боюнча канча балл (максималдуу) ала тургандыгы жөнүндө студенттерге жеткирилүүчү маалымат.

Студенттер баллдарды модулдарда төмөнкүдөй топтошот:

1-модулда эки учурдагы текшерүү (УТ1, УТ2) жана бир аралыктагы текшерүү (АТ1) уюштурулат. Ар бир текшерүү үчүн 30 баллдык баалоо системасы колдонулат. Баллдар тапшырмалар менен кошо таанышты рылат.

УТ1 текшерүүсү 4-жумада, УТ2 текшерүүсү 8-жумада уюштурулат, ал эми аралыктагы текшерүү дагы 8-жумада уюштурулат.

УТ1 деп 4-жумага чейин өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $УТ1 = \frac{Лек + Лаб + СӨАИ}{3}$.

УТ2 деп сабак башталгандан баштап 4-жумадан 8-жумага чейин өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз:

$УТ2 = \frac{Лек + Лаб + СӨАИ}{3}$.

Ведомостко жана журналга УТ1, УТ2 лердин жыйынтыктары коюлат.

8-жумада 1-модулдун материалдары боюнча 1-аралыктагы текшерүү уюштурулат. Мында 1-модулда өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $АТ1 = \frac{Лек + Лаб + СӨАИ}{З}$.

1-модулда баалоо учурдагы текшерүүлөрдүн жана 1-аралыктагы текшерүүнүн арифметикалык орточосу менен аныкталат: $М1 = \frac{УТ1 + УТ2 + АТ1}{З}$.

2-модулдагы баалоо 1-модулдагы баалоо сыяктуу эле аткарылат.

Жыйынтыктоочу текшерүүдө семестрде ичинде өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз:

$$ЖТ = \frac{Лек + Лаб + СӨАИ}{З}$$

Экзамендеги баалоо модулдардын жана жыйынтыктоочу текшерүүнүн арифметикалык орточосу менен сыйлык ($С$) баллдардын суммасы менен аныкталат:

$$Экз = \frac{М1 + М2 + ЖТ}{З} + С.$$

Баллдар тапшырмаларды берүүдө кошо көрсөтүлөт. $С$ – сыйлык баллдар «Билимди баалоо системасы» жөнүндөгү жободо көрсөтүлгөн.

8. Дисциплинанын кыскача мазмуну

СУБД жөнүндө жалпы түшүнүк. Берилгендер базасы. Берилгендердин моделдери. Иерархиялык берилгендер. Тармактык берилгендер. Реляциондук берилгендер базасы. Mysql структураланган запростор тили. Берилгендер базасын түзүү. Таблицалар.Таблицалардын структурасын өзгөртүү. Берилгендердин типтери. Таблицалардын ортосундагы байланыштар. Запростор. Жөнөкөй запростор. Кесилишүүчү запростор. Формалар. Отчеттор.

9. Лекциялык жана лабораториялык сабактардын календардык тематикалык планы

№	Темалар	Лекц. саат	Лаб .саат	Прак. саат
1.	Киришүү. СУБД жөнүндө жалпы түшүнүк.	2	1	1
2.	Mysql тили менен таанышуу. Mysqlде берилгендер менен иштөө.	2	1	1
3.	Берилгендер базасын түзүү жана тандоо. MySQL тилинин колдонулушу жана синтаксиси.	4	2	1
4.	Таблицалар. Таблицалардын структурасын өзгөртүү. Берилгендердин типтери	2	2	1
5.	SELECT, INSERT, UPDATE, WHERE операторлору, мисалдар	4	2	1
6.	ORDER BY, GROUP BY, DEFAULT, ALTER TABLE операторлорунун колдонулуштары	4	2	1
7.	VIEW, UNION операторлору	2	2	1
8.	Таблицалардын ортосундагы байланыштар. Индекстер. Сырткы жана ички ключтөр	4	2	1
9.	Таблицаларды бириктирүү. INNER JOIN, LEFT JOIN, RIGHT JOIN операторлору.	2	2	1
10.	FULL JOIN, CROSS JOIN операторлору. Кесилиш биригүү жөнүндө түшүнүк.	2	2	1
11.	My sql тилинде функиялардын колдонулуштары	2	2	1
	Баары:	30	20	10

10. Окуу-методикалык камсыздалышы

1. Дейт К. Дж. Введение в системы баз данных. – Киев: Диалектика, 1998.

2. Мартин Дж. Организация баз данных в вычислительных системах – М.: Мир, 1980. – 662 с.

3. Дейт К. Дж. Руководство по реляционной СУБД DB2. – М.: Финансы и статистика, 1988. – 320 с.

4. Максим Кузнецов, Игорь Симьдянов MySQL на примерах , Санкт-Петербург -2007.

Дополнительная литература

Хансен Г., Хансен Д. Базы данных и управление. – М.: Бином, 1999.

Конноли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. – М.– С./П.– К., 2000.

11. Баалар боюнча маалымат

Учурдагы, аралыктагы жана жыйынтыктоочу текшерүүлөр «Билимди баалоо» жөнүндөгү жобо менен аныкталат.

Студенттин билим деңгээли 100 баллдык системада төмөнкү эрежеге ылайык коюлат:

Рейтинг (балл)	Тамгалык система боюнча баа	GPA боюнча баалоонун цифралык эквиваленти	Традициялык системе боюнча баа
87 – 100	A	4,0	Эң жакшы
80 – 86	B	3,33	Жакшы
74 – 79	C	3,0	
68 – 73	Д	2,33	Канааттандыраарлык
61 – 67	E	2,0	
31 -60	FX	0	Канааттандыраарлык эмес
0 - 30	F	0	

Экзаменде бааны коюуда обөективдүүлүк жана акыйкаттуулук принциптеринин негизинде студенттин билиминин сапаты бардык тараптан анализделип, модулдук-рейтингдик системанын жобосуна ылайык коюлат.

12. Курстун саясаты

Студенттерге коюлуучу талаптар:

а) сабактарга сөзсүз катышуу;

б) практикалык (семинардык) сабактардагы активдүүлүгү;

в) сабактарга, үй тапшырмасын жана өз алдынча иштерди аткарууга даярдыгы ж.б.

г) калтырган сабактардын конспектисин өз алдынча даярдап келип, окутуучуга баяндап бериши.

Төмөнкүлөргө жол берилбейт:

а) сабактарга кечигүү жана сабактан кетип калуу;

б) сабак учурунда уюлдук телефонду пайдалануу;

в) жалганчылык жана көчүрүп алуу (плагиат);

г) тапшырмаларды өз убагында тапшырбоо ж.б.

13. Калтырылган сабакты толуктоо (отработка)

Студент калтырылган сабакта өтүлгөн теманы өз алдынча өздөштүрүп, деканаттын уруксат кагазы менен кафедрага келип, предметникке (предметник талап кылган формада) теманы кайрадан тапшырат. Калтырылган сабакты толуктоо аралык текшерүүгө чейин кабыл алынат жана модулдук баллдарга таасир этет. Калтырылган сабак толукталбаса, ар бир калтырылган сабак үчүн предметник модулдан 2 балл кемитет. Предметник кайра тапшырууну атайын журналга каттап, деканаттын уруксат кагазына «калтырылган сабак толукталды» деген белгини коюп берет.

14. Студенттердин өз алдынча иштери үчүн тапшырмалар (СӨАИ)

Берилгендердин иерархиялык моделинин өзгөчөлүктөрү

Берилгендердин тармактык моделинин өзгөчөлүктөрү

Таблица түзүү.

Таблицанын ичин толтуруу, толтуруунун жолдору

MySQLдин структурасы

Клиент-сервердик архитектура

Структура дистрибутивдердин структурасы

Официалдуу документация

MySQLди орнотуу

MySQLдин иш жөндөмдүүлүгүн текшерүү 23

MySQLди Linux ОСна орнотуу

Конфигурациялык файл

Берилгендердин каталогун жылдыруу

Учрдагы версиясын жаңылоо

Обзор утилит MySQL утилитасынын обзору

mysql утилитасы

Командалык жолчо

Кодировканы ылайыктоо

SELECT и *LOAD DATA операторлорунун колдонулушу*

BACKUP TABLE *и* RESTORE TABLE

15. Жыйынтыктоочу экзамендин программасы

1. Берилгендер базасын түзүү
2. Берилгендер базасын тандоо
3. Таблица түзүү
4. Берилгендердин сандык тиби
5. Берилгендердин жолчолук тиби
6. Берилгендердин календардык тиби
7. NULL берилгендер тиби
8. Таблицанын структурасын көрүү
9. Мамычалардын параметлери
10. Берилгендер базасынын, таблицалардын талаалардын мүмкүн болгон аттары
11. **SELECT, INSERT,** UPDATE, WHERE операторлору, мисалдар
12. ORDER BY, GROUP BY,
13. **DEFAULT,** ALTER TABLE операторлорунун колдонулуштары
14. VIEW, UNION операторлору
15. Таблицанын параметрлери
16. Убактылуу таблицалар
17. Таблицанын көчүрмөсүн түзүү
18. Таблицанын оперативдик эске жайгаштыруу
19. Бир нече таблицаларды бир таблицага бириктирүү
20. Таблицаларды өчүрүү
21. Таблицаны өзгөртүү

22. Талааларды кошу
23. Талааны өчүрүү
24. Бар талаалардын атын өзгөртүү
25. Таблицанын атын өзгөртүү
26. Таблицанын параметрлерин өзгөртүү
27. Таблицаны калыбына келтирүү
28. REPAIR TABLE оператору
29. Таблицанын текшерүүчү суммасы
30. Индекстер
31. Ички жана сырткы ключтөр
32. AUTO_INCREMENT атрибутунун жардамында уникалдуу индексти түзүү
33. Кадимки жана уникалдуу индекстер
34. Таблицага индекстерди кошу жана өчүрүү
35. Индекстерди калыбына келтирүү
36. Таблицаларды бириктирүү.
37. INNER JOIN, LEFT JOIN, RIGHT JOIN операторлору.
38. FULL JOIN, CROSS JOIN операторлору. Кесилиш биригүү жөнүндө түшүнүк.
39. My sql тилинде функиялардын колдонулуштары
40. TIMESTAMP берилгендер тиби
41. Сандык маанилерди толтуруу
42. Жолчолук маанилерди толтуруу
43. Календардык маанилерди толтуруу
44. *UNIXSTAMP* форматында маанилерди толтуруу
45. AUTO_INCREMENT механизми
46. Эсептөөчү маанилерди толтуруу
47. *INSERT... SELECT* операторунунун колдонулушу
48. Берилгендерди өчүрүү
49. DELETE оператору
50. TRUNCATE оператору
51. Бир нече таблицалардан өчүрүү
52. Бир нече таблицалардан каскаддык өчүрүү

53.Жазууларды жаңылоо

54.UPDATE оператору

55.UPDATE көп таблицалуу оператору

ЖАЛПЫ ТЕОРИЯЛЫК МАТЕРИАЛДАР

1-2-лекция

Лектор:Г.И.Сейитказыева

Тема: **СУБД жөнүндө жалпы түшүнүк**

Сабактын планы:

1. Берилгендер базасын башкаруу системинин жана берилегендер базасынын аныктамасы
2. Берилгендер базасын башкаруу системасынын типтери
3. Берилгендер базасын берүү жолдору

1.Сабактын максаты:

Билим берүүчүлүк: Студенттерге берилгендер базасы жөнүндө жалпы түшүнүк берүү, СУБД жана берилгендер базасынын аныктамаларын,типтерин, негизги обөекттерин үйрөтүү.

Өнүктүрүүчүлүк: Аналогдорун тургузууну үйрөтүү, маселени коюп аны чечүүнү үйрөтүү.

Тарбиялык: Тыкандыкка, тактыкка, сылыктыкка тарбиялоо.

2.Сабактын жабдылышы: Проектр, компьютер.

3. Окутууда колдонулган усулдары: Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.

4. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

5. Жаңы теманы түшүндүрүү

Берилгендер базасын башкаруу системи (ББББС) (англ. Database Management System-Система управления базой данных (СУБД)) деп берилгендер жана файлдарды түзүүгө, сактоого мүмкүндүк берген жана байланыштар жөнүндө билдирип турган программдык жабдык аталат. Бир типтеги файлдарды берилгендер базасы катары кароого болот, себеби алар берилгендердин жыйындысы, бирок бир типтеги файлдарда байланышуу уюштурулган эмес. Байланыштар берилгендер базасында так аныкталышы керек. ББББслер берилгендердин түзүлүшүн орнотуп, алардын калыптоо, оңдоо жана башка ушул сыяктуу иштерди аткарууга мүмкүндүк берет.

ББББнын типтери.

Берилгендер базасында маалыматтарды уюштуруунун з ыкмасы бар.

1. ***Иерархиялык*** (Дарак түрүндө)
2. ***Тармактык***
3. ***Реляциондук***

• ***Иерархиялык берилгендер базасында*** элеметтердинортосунда катуу тартиптеги көз карандылык болот. Бирөө негизги калганы баш ийүүчү. Мисалы: Дисктеги системалардын каталогу

• ***Тармактык берилгендер базасы бир кыйла ийкемдүү***: анык бир белгиленген негизги элемент жок жана горизонталдык байланыш тургузуу мүмкүнчүлүгү бар. Мисалы Интернеттеги маалыматтардын уюштурулушу.

• Эң кеңири таралган р***еляциондук берилгендер базасы.***

Реляциондук англис тилинен алынган « relation» - байланыш дегенди түшүндүрөт. Релациондук берилгендер базасы бул берилгендердин таблица түрүндө берилиши.Таблицанын ар бир жолчосу конкреттү бир объект жөнүндөгү маалыматты кармайт.Ар бирмамычасы булл обөекьттин мүнөздөмөсүн кармайт. Жолчолор «жазуу» деп, мамычалар «талаа» деп аталат.

Мисалы:

Таб_№	***ФИО***	***Дата_рожд***	***Дата_приема***	***Должность***	***Оклад***
001	Эшенкулов Т.	12.05.65	1.02.80	директор	1000
002	Асанова А.Г.	30.10.75	2.03.95	бугалтер	500
003	Саматов С.С	4.01.81	4.06.00	Аткаруучу	100

Берилгендер базасы (Database-База данных) - колдонуучулардын маалыматтык керектөөлөрүн камсыздоо үчүн, бирге сакталынган жана өз ара байланышкан файлдардын тобу. Берилгендер базасы бир учурда бир бирине байланышкан файлдардын каалаганына кайрылууга мүмкүндүк берет. Мисалы, кайсы бир окуу жайынын берилгендер базасын карай турган болсок, ал бир нече файлдардан турат, аларга илимий-иштер, мугалимдердин эмгектери, окуу-методикалык колдонмолор, факультеттин файлдары, курстук файлдар, группа студенттеринин файлдары жана

башка ушул сыяктуу көптөгөн файлдар кириши мүмкүн. Бул файлдар биригип, окуу жайынын берилгендер базасын түзүшөт.

Берилгендер базасы фактографиялык жана документалдык болуп бөлүнөт.

Фактографиялык берилгендер базасы объект жөнүндө так формата берилген кыскача маалыматтарды кармайт.

В ***документалдык берилгендер базасы*** түрдүү типтеги:тексттик, үндүк, графикалык,мультимедиялык маалыматтарды кармайт

3-лекция **Лектор:Г.И.Сейитказыева**

Тема: **Mysql тили менен таанышуу. Mysqlде берилгендер менен иштөө.**

Сабактын планы:

4. Mysql тили менен таанышуу
5. **SQL тилинин командаларынын тиби**
6. Жөнөкөй запросторду кийирүү.

1.Сабактын максаты:

Билим берүүчүлүк: *Студенттерге Mysql тили жөнүндө жалпы түшүнүк берүү. Mysql тилинин өзгөчөлүктөрү, жетишкендиктери жана командалары жөнүндө билимдерди үйрөтүү.*

Өнүктүрүүчүлүк: *Аанализдөөнү үйрөтүү жана маселени коюп аны чечүүнү үйрөтүү.*

Тарбиялык: *Тыкандыкка, тактыкка, сылыктыкка тарбиялоо.*

2. Калыптандыруучу компотенция: ***ОК-1, ОК-2***

3.Сабактын жабдылышы: *Проектр, компьютер, доска, бор.*

4. Окутууда колдонулган усулдары: *Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.*

5. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

Б)Өтүлгөн теманы кайталоо

6. Жаңы теманы түшүндүрүү

Бүгүнкү күндө берилгендер базасын башкаруунун реляциондук системаларын колдонуу түрдүү областтарда кеңири тараган.

Ошондой эле азыр көп колдонулган СУБДлардан MySQLди айтууга болот. СУБД-колдонуучу тарабынан берилгендер базасын түзүү, өзгөртүү, аныктоо жана ага жетүүнү башкарууга мүмкүндүк берүүчү программалык камсыздоо.

MySQL SQLтилин колдонот.

SQL-Structured Query Language- запростордун структураланган тили деп которулат. Запростордун стандарртык тили деп да айтууга болот.Реляциондук типтеги моделдерди иштеп чыгуунун жардамында пайда болгон тилдердин бири болуп эсептелет.Азыркы күндө СУБДнын бардык компоненттерин камтыган берилгендер базасын иштетүүчү негизги стандарттык тил катары каралат. SQL- тилинин стандарты 1986-жылы Америкалык улуттук стандарттар Институту (ANSI) тарабынан кабыл алынган жана 1987-жылы Эл аралык стандарттык уюм (ISO) тарабынан эл аралык деңгээлде кабыл алынган.

СУБДлардын ичинен MySQLдин көптөгөн жетишкен жактары бар.

- **Иштин тездиги.** MySQL — өтө тез иштеген СУБДларды толуктайт. Өндүрүмдүүлүгү жогору.
- **Колдонуунун жөнөкөйлүгү.** MySQLде администрлөө жөнөкөй.

- **Запростордун тилин колдоосу.** MySQL SQLтилинин командаларын колдонот. Бул тил бардык заманбап СУБДларда колдонулат.
- **Мүмкүнчүлүктөрү.** Сервер бир эле учурда чектелбеген сандагы колдонуучуларга

туташа алат. СУБД MySQL серверине түрдүү интерфейстердин жардамында туташууга болот.

- **Өз ара аракеттешүүсү.** MySQL тармакта иштөөгө арналган жана интернет аркылуу дагы байланышууга болот.

SQL тилинин командаларынын тиби

SQL- тили берилгендереге запрос жүргүзүүнү аткаруу менен бирге колдонмо программаларды да түзүүгө жөндөмдүү. SQL тилинин негизги команадаларынын категорияларынын жардамында берилгендер базасынын обьектилерин түзүү жана манипуляциялоо, таблицага баштапкы берилгендерди жүктөө, таблицада бар берилгендерди жаңылоо, өчүрүү, берилгендер базасына жүргүзүлгөн запросторду аткаруу иштерин аткарууга болот.

SQL тилинин негизги командаларынын категориялары

1. **DDL**-Data Definition Language-берилгендерди аныктоо тили. DDLдин жардамында берилгендер базасынын обьектилеринин структурасын түзүп, өзгөртүп, өчүрүүгө болот.

   CREATE TABLE-таблица түзүү

   ALTER TABLE-таблицаны өзгөртүү

   DROP TABLE-таблицаны өчүрүү

   CREATE INDEX-индексти түзүү

   ALTER INDEX-индексти өзгөртүү

   DROP INDEX-индексти өчүрүү

MySQL клиент/сервер архитектурасын колдонот. Бул архитектурага ылайык сервер берилгендер базасы менен иштейт ал эми клиенттер серверге тармак аркылуу туташат. Бирок биз негизинен mysql дин клиенттик программасы менен иштөөгө киришебиз. mysql клиенттик программасы sql-запросторду колдонуучудан кабыл алып серверге жиберип жыйынтыгын кодонуучуга жиберет.

Жөнөкөй запросторду кийирүү.

mysql ге запросту кийирүү үчүн ошол запросту печатап «;» менен жыйынтыктап <Enter> басуу жетиштүү. Запросту эки символ менен жыйынтыктоого болот. «;» же «\g» символдору менен.

Запросту кийиргенден кийин mysql аны аткарууга запроско жиберет. Сервер запросту кайра иштеп чыгып жыйынтыгын mysql клиентине жиберет.

Мисалы учурдагы датаны жана убакытты билүү үчүн эң жөнөкөй запрос жөнөнтүп көрөлү. Ал үчүн NOW () функциясын колдонобуз.

Wysql> **SELECT** NOW() ;

Жыйынтыгы төмөнкүдөй көрсөтүлөт.

+ ----------+

I NOW() |

4 ---------- +

I2018-02-04 11:02:36 |

- ----------

row m set (0.00 sec)

mysql программасы үчүн көпчүлүк учурда команда кийирилип жаткан регистр маанеге ээ эмес. Төмөндөгү запростор бири-бирине эквивалентүү.

SELECT DSERO select user ()

SeLeCt UsEr()

Негизинен ачкычтык сөздөрдү жана функцияларды чон тамгалар менен берилип берилгендер базасынын, таблицалардын ататры кичине тамагалар менен берилгени ылайыктуу.

7. Билимдерди текшерүү үчүн суроолор:

1. **Запростордун тилин колдоосу эмнени түшүндүн?**
2. MySQL тили архитектураны колдонот?
3. SELECT операторнун кызматы?

4-лекция **Лектор:Г.И.Сейитказыева**

Тема: Берилгендер базасын түзүү жана тандоо. MySQL тилинин колдонулушу жана синтаксиси.

1.Сабактын максаты:

Билим берүүчүлүк: Студенттерге Mysql тилинде берилгендер базасын түзүүдө аны тандоодо, түзүлгөн берилгендер базасын өчүрүүдө колдонулган операторлор жөнүндө түшүнүк берүү.

Өнүктүрүүчүлүк: Mysql тилинде берилгендер базасын түзүүнү аны тандоону, түзүлгөн берилгендер базасын өчүрүүнү үйрөтүү.

Тарбиялык: Тыкандыкка, тактыкка, сылыктыкка тарбиялоо.

2. Калыптандыруучу компотенция: ОК-2, ПК-2.3.

3.Сабактын жабдылышы: Проектр, компьютер, доска, бор.

4. Окутууда колдонулган усулдары: Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.

5. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

Б)Өтүлгөн теманы кайталоо

6. Жаңы теманы түшүндүрүү

Берилгендер базасын түзүү үчүн **CREATE DATABASE операторун колдонобуз.**

Жазылуу синтаксиси *CREATE DATABASE Берилгендер_ базасыны_аты;*

Gruppa_db аталышындагы берилгендер базасын түзүү үчүн:

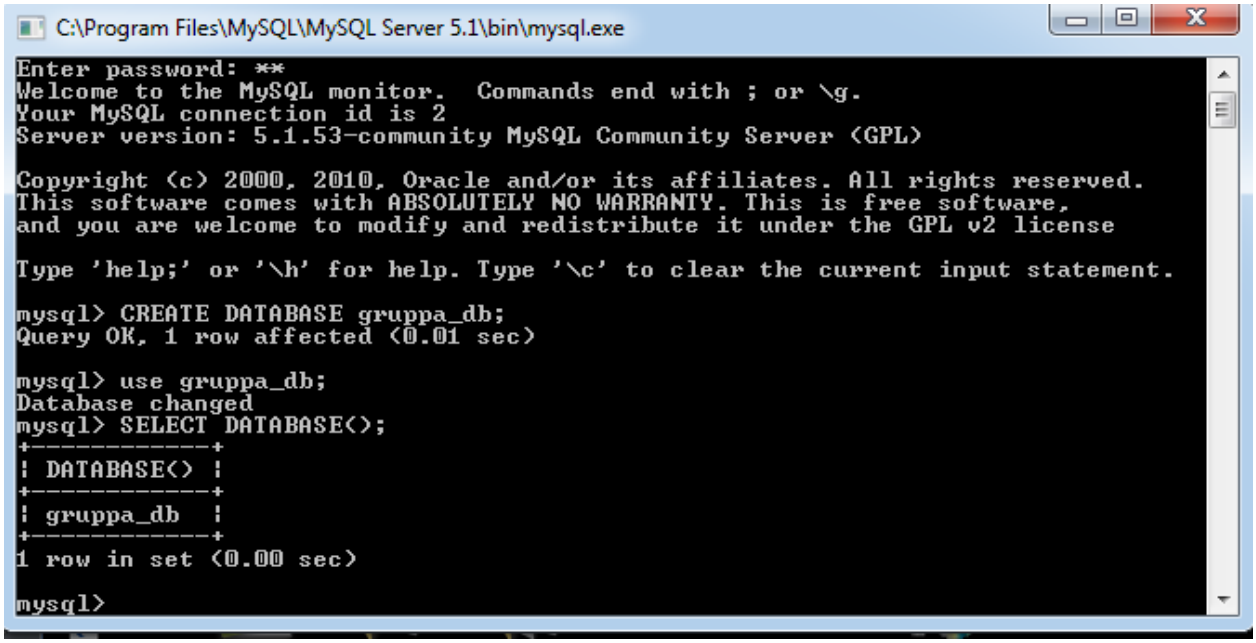
mysql> CREATE DATABASE gruppa_db; кодун жазабыз.

Берилгендер базасын тандоо үчүн use командасын колдонобуз.

Синтаксиси: ***USE*** *берилгендер_базасынын_аты;*

mysql> USE gruppa_db;

Учурда активдүү берилгендер базасын билүү үчүн **SELECT DATABASE** операторун колдонобуз. Синтаксиси: *SELECT DATABASE();*



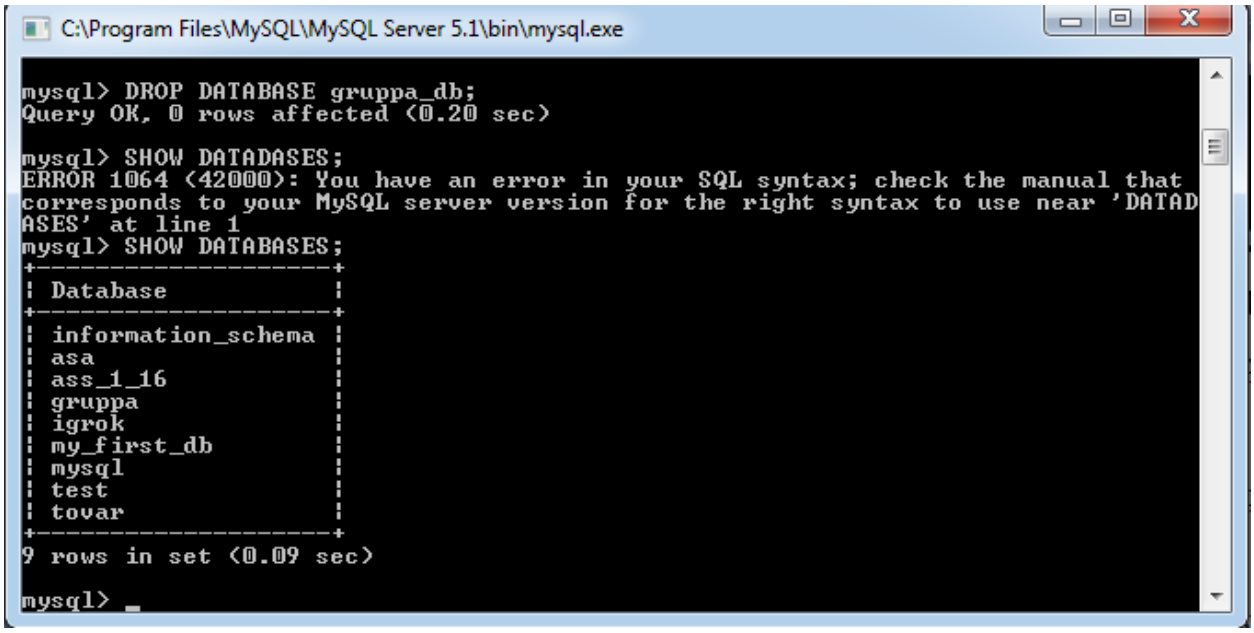
DROP операторунун жардамында өчүрүүнү аткарабыз.

Синтаксиси: *DROP DATABASE берилгендер_базасынын_аты;*

Мисалы: DROP DATABASE gruppa;

Show операторун колдонуп берилгендер базасынын же таблицалардын тизмесин көрүүгө болот. Мисалы:

show databases; запросун жазсак анда бардык берилгендер базасынын тизмесин чыгарат.



Жаңы теманы бышыктоо үчүн суроолор:

1. CREATE оператору эмне үчүн колдонулат?
2. Учурдагы активдүү берилгендер базасын аныктоо үчүн кандай запрос жазабыз?
3. Өчүрүү үчүн кайсы операторду колдонобуз?

5-лекция

Тема: Таблицаларды түзүү. Таблицалардын структурасын өзгөртүү. Берилгендердин типтери

1.Сабактын максаты:

Билим берүүчүлүк: Студенттерге Mysql тилинде таблицаларды түзүүдө, структурасын өзгөртүүдө колдонулган операторлор берилгендердин типтери жөнүндө түшүнүк берүү.

Өнүктүрүүчүлүк: Mysql тилинде таблицаларды түзүүнү структурасын өзгөртүүнү, үйрөтүү.

Тарбиялык: Тыкандыкка, тактыкка, сылыктыкка тарбиялоо.

2. Калыптандыруучу компотенция: ОК-2, ПК-2.3.

3.Сабактын жабдылышы: Проектр, компьютер, доска, бор.

4. Окутууда колдонулган усулдары: Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.

5. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

Б)Өтүлгөн теманы кайталоо

6. Жаңы теманы түшүндүрүү:

Таблица деген бир типтүү объекттер жөнүндө маалыматтардын топтому. Ал жолчолордон жана мамычалардан (талаалардан) турат.
Жолчолор- жазуулар деп аталып бир обьект жөнүндө маалыматты кармайт.

Мамычалар- атрибут деп аталып бардык обьекттердин конкреттүү бир белгиси жөнүндө маалыматты кармайт.

MySql тилинде таблица түзүү үчүн CREATE TABLE оператору колдонулат.

- Синтаксиси: **CREATE TABLE** таблицанын_аты;

(биринчи_талаанын_аты тиби, экинчи_талаанын_аты тиби,

…, акыркы_ талаанын_аты тиби);

- **МИСАЛЫ:**

```
CREATE TABLE   IF NOT EXISTS  table_1 (kod  int (10)
AUTO_INCREMENT ,
fam  varchar (25) NOT  NULL DEFAULT ‘ ‘ COMMENT
‘Студенттер’ ,
 imia  varchar (15) NOT  NULL,
raion  varchar (20) NOT  NULL,
kontrakt  decimal (8,2) NOT NULL DEFAULT ’0.00’,
PRIMARY KEY (kod)
);
```

Биз түзүлгөн таблицабыздын структурасын (талааларын) көрүү үчүн DESCRIBE командасын колдонобуз.

Синтаксиси: **DESCRIBE** таблицанын_аты; же

DESC таблицанын_аты; деп жазууга болот.

Ушул таблицадагы типтерди карап көрөлү.

- **INT**-талаа бүтүн сандарды кабыл алат..
- **NOT NULL**- тиби берилгенде талаанын ар бир жолчосу толтурулууга тийиш, жолчолор бош маанини кабыл албайт.
- **AUTO_INCREMENT**- СУБД MySQLдин атайын атрибуту.Ал таблицага берилгендерди кошуу үчүн кайрылган учурда мамычанын (талаанын) максималдык маанисине 1 ди кошуп мамычага чыгарат б.а. автоматтык түрдө уникалдуу номер берет.
- **CHAR** тиби- бул VARCHAR тиби менен окшош болгону менен CHAR тибинде талаанын узундугу

фиксирленген болуп таблица түзүлүп жатканда берилет. Узундугу 1 ден 255 ке чейинки каалаган маанини алат. Бул типтин чоңдугу сактоо учурунда берилген узундукка чейин пробелдер менен толукталат.

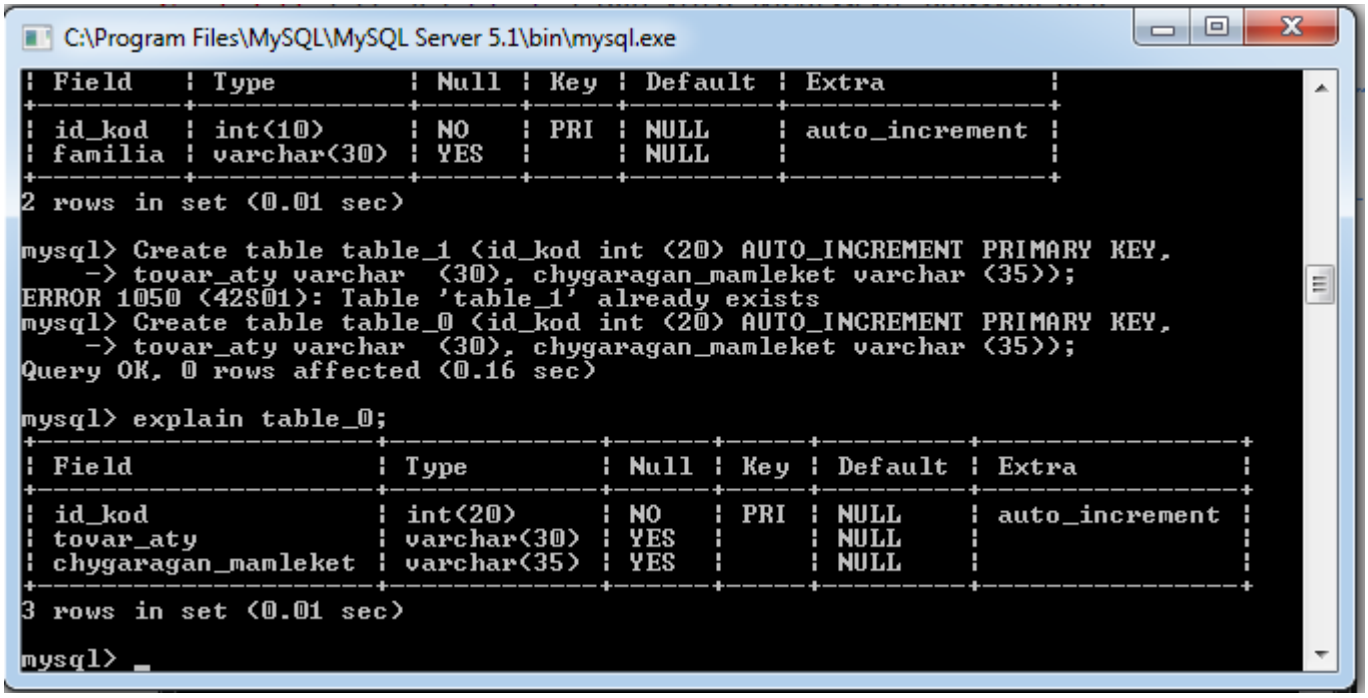
- **VARCHAR**- бул тип өзгөрүлмө узундукка ээ. CHAR тибиндегидей эле узундугу 1 ден 255 ке чейинки каалаган маанини алат бирок сактоо учурунда керектүү болгон символдордун санын жана ага кошумча узундук жазуу үчүн 1 байт алат. Пробелдер менен толукталбайт тескерисинче акыркы жыйынтыктоочу пробелдер сактоо учурунда өчүрүлөт.
- PRIMARY KEY- ТАЛААНЫН ИНДЕКСИ БОЛУП ЭСЕПТЕЛЕТ ЖАНА АР БИР МААНИСИ УНИКАЛДУУ. БУЛ БИР ИДЕНТИФИКАТОРДУ КАЙТАЛАП КОЛДОНУУГА МУМКУНДУК БЕРБЕЙТ.
- СИНТАКСИСИ: **PRIMARY KEY**(ТАЛААНЫН_АТЫ);
- **FOREIGN KEY-** сырткы ключту көрсөтүү үчүн колдонулат.
- Синтаксиси:
- **FOREIGN KEY** (сырткы_ключ_болгон_талаанын_аты) **REFERENCES** родительский_таблицанын_аты (родителдик_таблицанын_талаасынын_аты);

Таблицаны өзгөртүү.

Таблицаны өзгөртүү үчүн **ALTER TABLE** оператору колдонулат.

Таблицага талаа кошуу үчүн төмөндөгү мисалды карайлы:

```
Create table table_0 (
id_kod int (20) AUTO_INCREMENT PRIMARY KEY,
tovar_aty varchar (30), chygaragan_mamleket varchar (35)
);
```

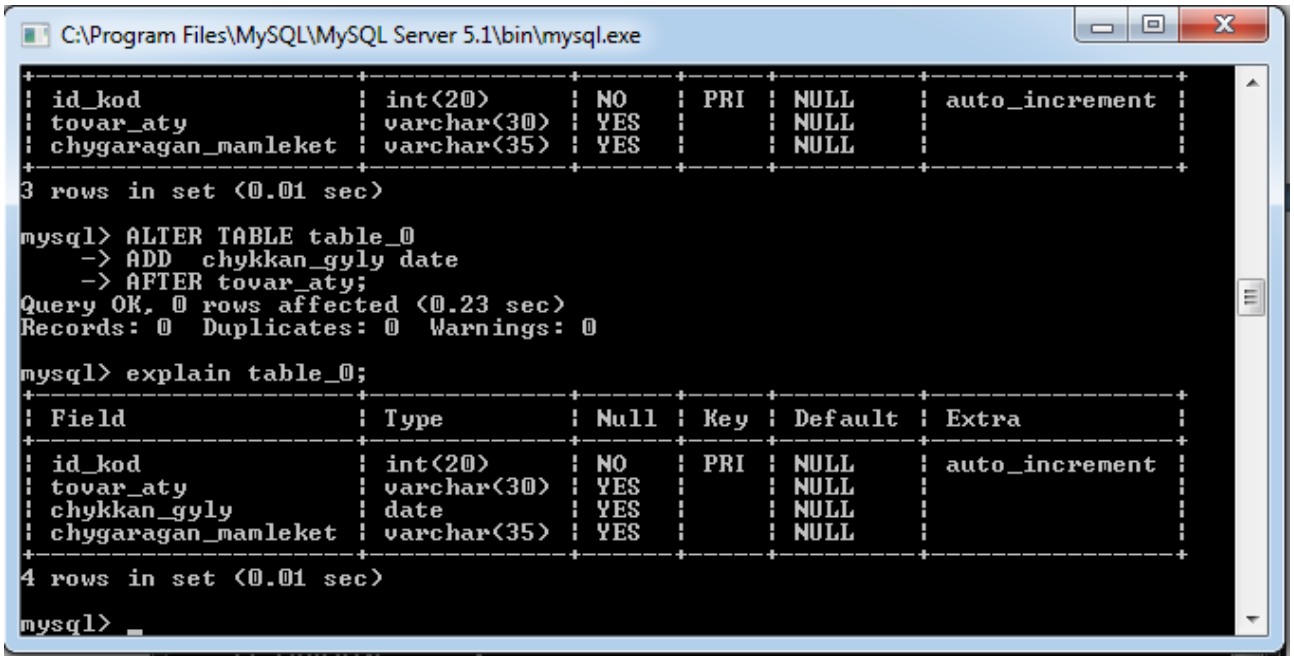


Ушул Table_0 таблицабызга чыккан_жылы талаасын кошолу.

1 **ALTER TABLE** table_0

2 **ADD chykkan_gyly** date

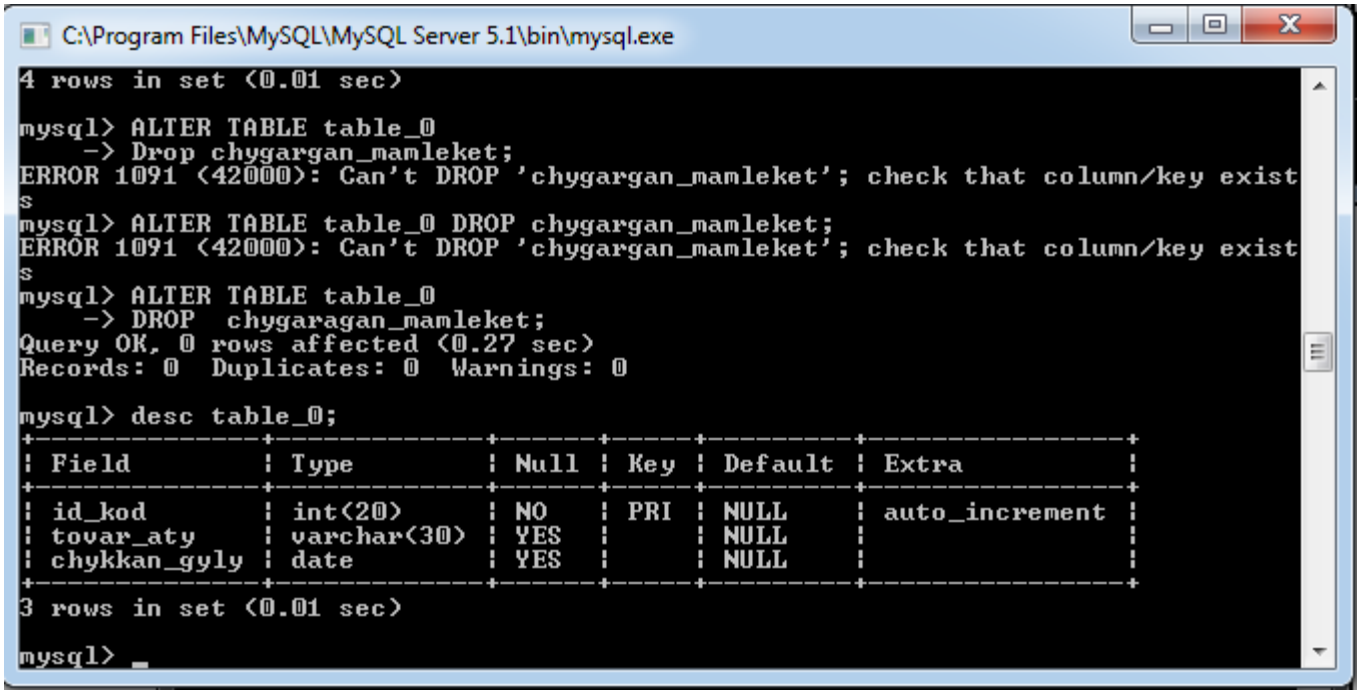
3 **AFTER tovar_aty;**



Таблицадан бизге керек болбогон талааны өчүрүп салуу үчүн:

ALTER TABLE table_0

DROP chygaragan_mamleket;

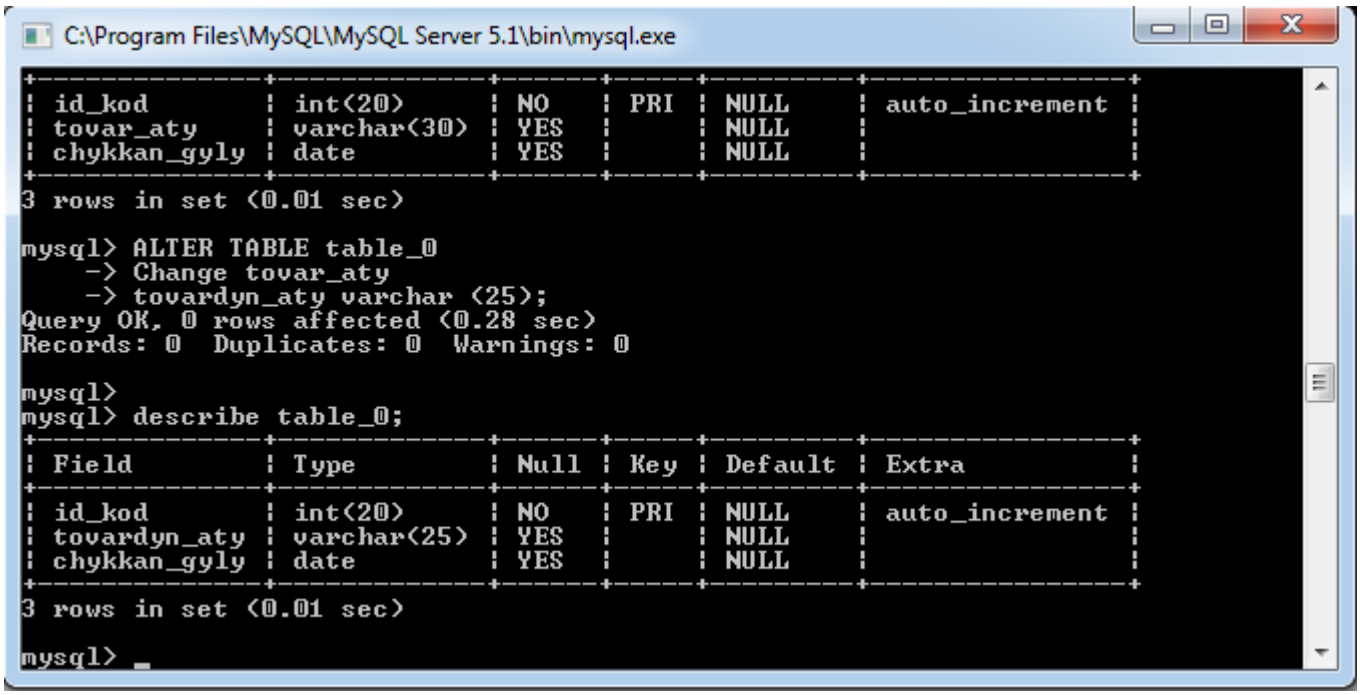


Талаадагы таблицанын атын жана тибин алмаштыруу үчүн:

ALTER TABLE table_0

Change tovar_aty

tovardyn_aty varchar (25);



Тема: Таблицага берилгендерди кийирүү.. SELECT, INSERT, UPDATE, WHERE операторлору, мисалдар **.**

1.Сабактын максаты:

Билим берүүчүлүк: Студенттерге Mysql тилинде таблицалардын ичин толтурууда колдонулган операторлор жөнүндө түшүнүк берүү.

Өнүктүрүүчүлүк: Mysql тилинде таблицаларды толтуруунун жолдорун, маалыматты анализдөөнү үйрөтүү.

Тарбиялык: Тыкандыкка, тактыкка, чынчылдыкка сылыктыкка тарбиялоо.

2. Калыптандыруучу компотенция: ОК-1, ПК-2.3.

3.Сабактын жабдылышы: Проектр, компьютер, доска, бор.

4. Окутууда колдонулган усулдары: Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.

5. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

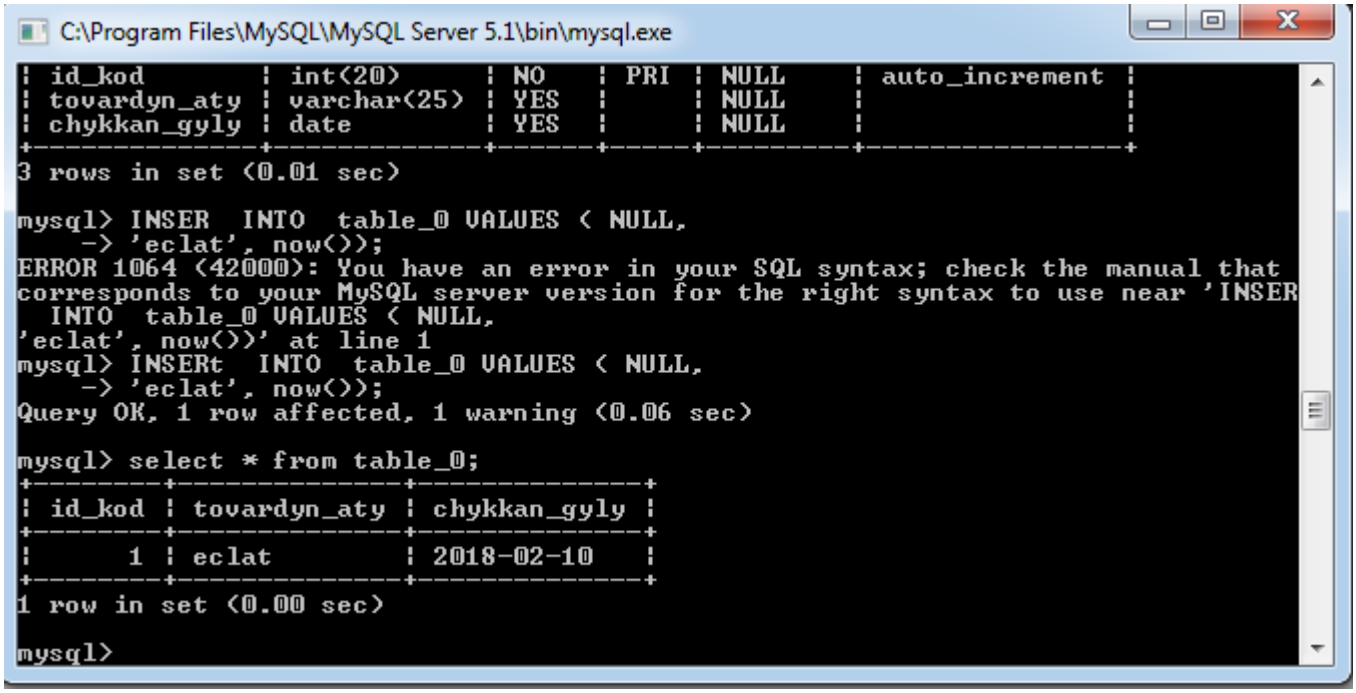
Б)Өтүлгөн теманы кайталоо

6. Жаңы теманы түшүндүрүү:

Таблицага берилгендерди кийирүү үчүн **INSERT** оператору колдонулат.Таблицага берилгендерди кийирүүнүн үч жолу бар.

1-жолу

```
INSERT  INTO  table_0 VALUES ( NULL,
'eclat', now());
```



2-жолу.

Синтаксиси:

INSERT INTO таблицанын_аты SET

Экинчи_талаанын_аты=’chanel’,

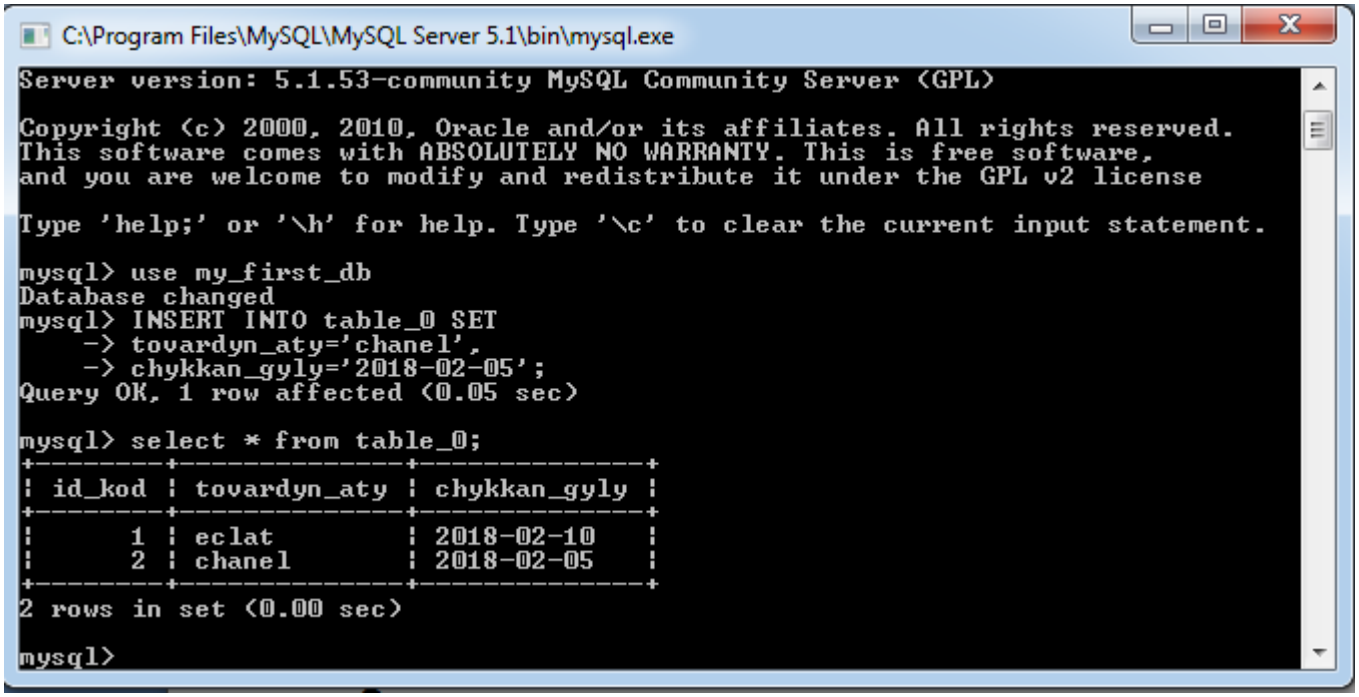
/чүнчү_талаанын_аты=’2018-02-05’**;**

Мисалы:

INSERT INTO table_0 **SET**

tovardyn_aty=’chanel’,

chykkan_gyly=’2018-02-05’;



3-жолу синтаксиси:

INSERT INTO таблицанын_аты ('талаанын_аты', 'талаанын_аты') VALUES ('экинчи_талаанын_мааниси','үчүнчү_талаанын_мааниси');

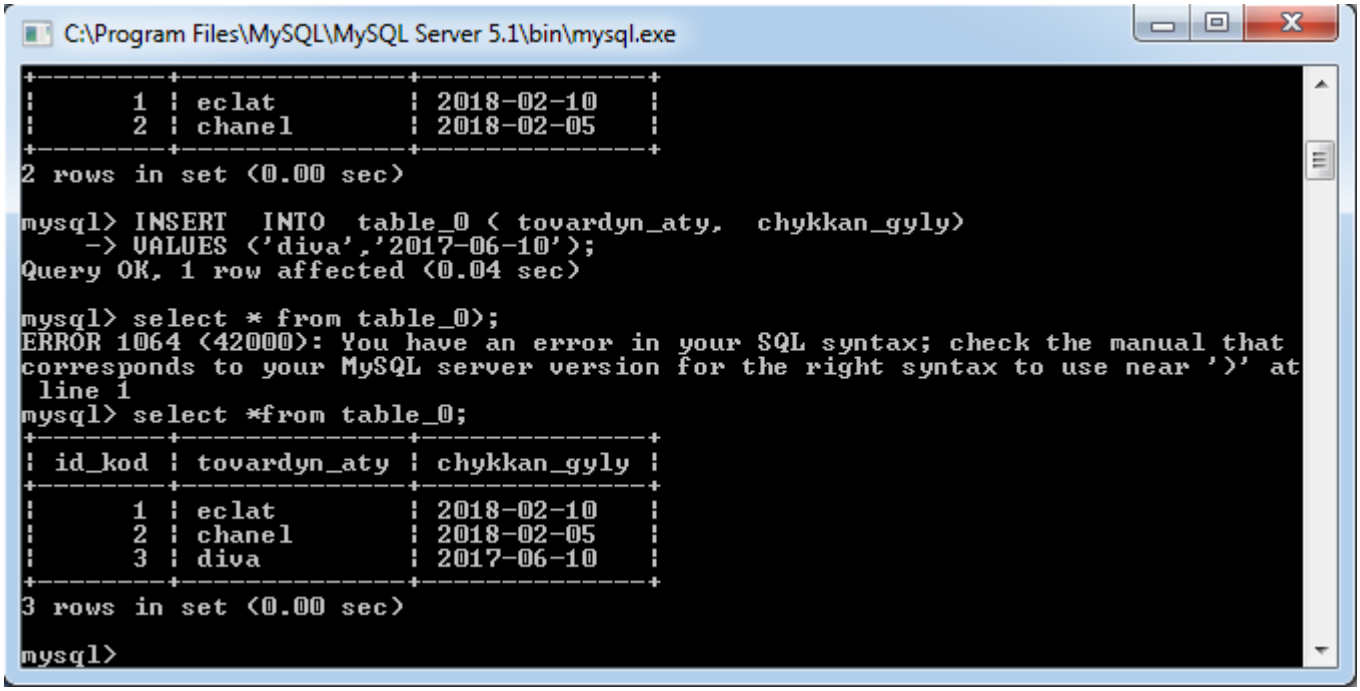
Мисалы:

INSERT INTO table_0 (tovardyn_aty, chykkan_gyly)

VALUES (‘diva’,’2017-06-10’);

Таблицага кийирилген маалыматтарды көрүү үчүн **SELECT** оператору колдонулат.

Синтаксиси: SELECT *FROM таблицанын_аты



UPDATE оператору таблицанын жазууларындагы маанилерди өзгөртүү үчүн колдонулат. UPDATE оператору төмөнкүдөй синтаксиске ээ. UPDATE таблицанын_аты SET талаанын_аты=’маани’

WHERE id= катар номери;

Мисалы:

UPDATE table_0 SET tovardyn_aty=’nostalgy’

WHERE id_kod= 2;

Тема: Маанилерди сортировкалоону, тандоонун жыйынтыктарын бириктирүүнү уюштуруу. ORDER BY, GROUP BY, DEFAULT операторлорунун колдонулуштары

1.Сабактын максаты:

Билим берүүчүлүк: Студенттерге Mysql тилинде таблицалардын ичиндеги маанилерди сортировкалоодо, тандоонун жыйынтыктарын бириктирүүдө колдонулган операторлор жөнүндө түшүнүк берүү.

Өнүктүрүүчүлүк: Mysql тилинде таблицалардын маалыматтарын сортировкалоо жолдорун, маалыматты анализдөөнү үйрөтүү.

Тарбиялык: Тыкандыкка, тактыкка, чынчылдыкка, сылыктыкка тарбиялоо.

2. Калыптандыруучу компотенция: ОК-2, ПК-2.3.

3.Сабактын жабдылышы: Проектр, компьютер, доска, бор.

4. Окутууда колдонулган усулдары: Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.

5. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

Б)Өтүлгөн теманы кайталоо

6. Жаңы теманы түшүндүрүү:

ORDER BY оператору чыгуучу маанилерди сортировкалоодо колдонулат. ORDER BY операторун сандык таалар үчүн да тексттик талаалар үчүн да колдонууга болот. ORDER BY опертаору төмнкүдөй синтаксиске ээ.

ORDER BY талаанын_аты [ASC/DESC];

Сортировканы өсүү жана кемүү тартибинде аткарууга болот.

ASC-параметри кичине мааниден чоңго карай өсүү тартибинде сортировкалайт.

DESC-параметри чоңдон кичине мааниге карай кемүү тартибинде аткарат.

Мисалы группа деген таблица түзөлү.

CREATE TABLE gruppa (id_gruppa int (10) NOT NULL AUTO_INCREMENT PRIMARY KEY, familiacy VARCHAR (30), aty VARCHAR (25), jashy int (10));

Таблицанын ичин толтуралы.

INSERT INTO gruppa VALUES (NULL, ‘Gaparova’, ‘Mahabat’,’17’),

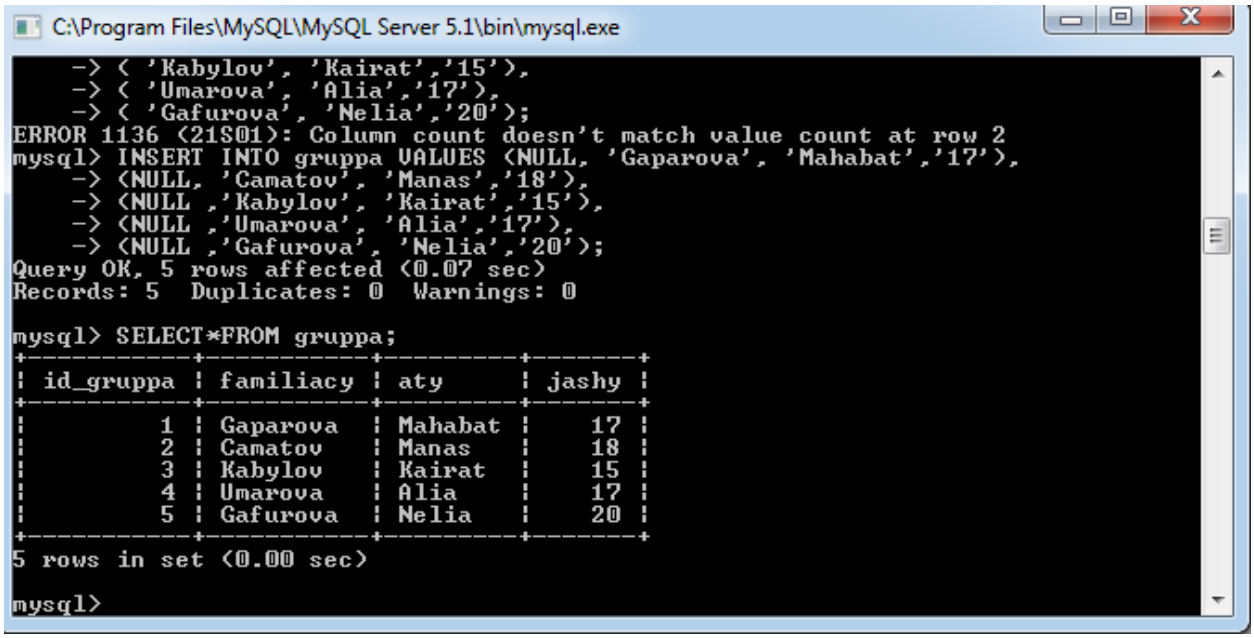
(NULL, ‘Camatov’, ‘Manas’,’18’),

(NULL ,‘Kabylov’, ‘Kairat’,’15’),

(NULL ,‘Umarova’, ‘Alia’,’17’),

(NULL ,‘Gafurova’, ‘Nelia’,’20’);

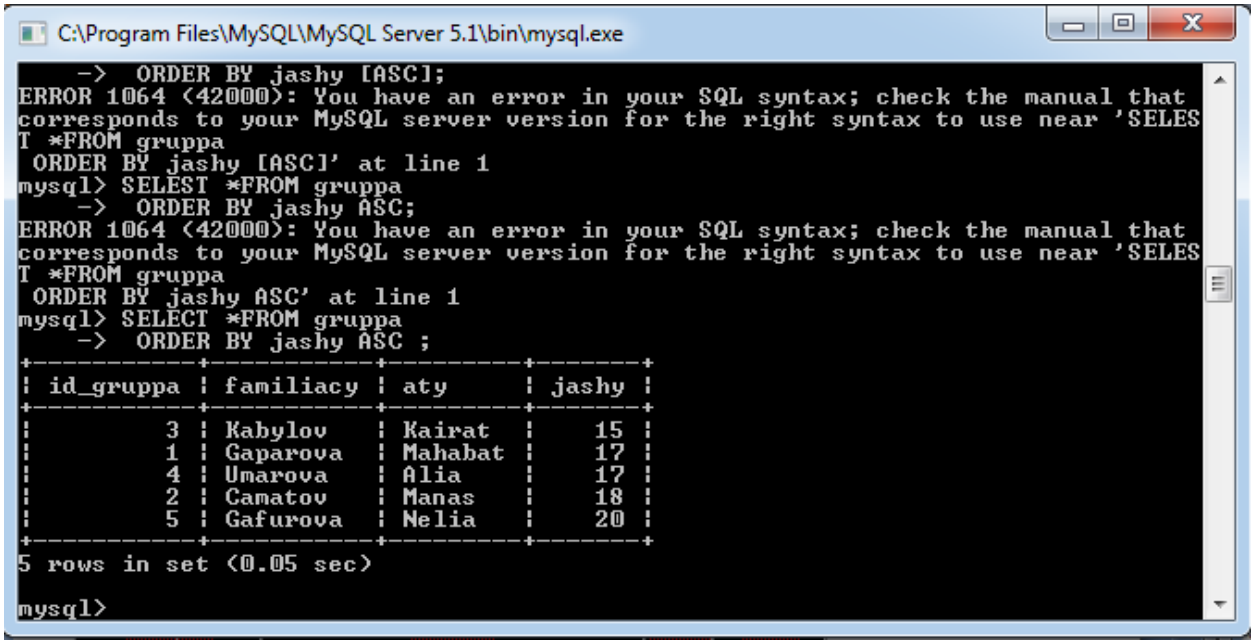
SELECT операторунун жардамында таблицабызды көрөлү.



Эми gruppa таблицасындагы jashy талаасын сортировкалап көрөлү.

SELECT *FROM gruppa

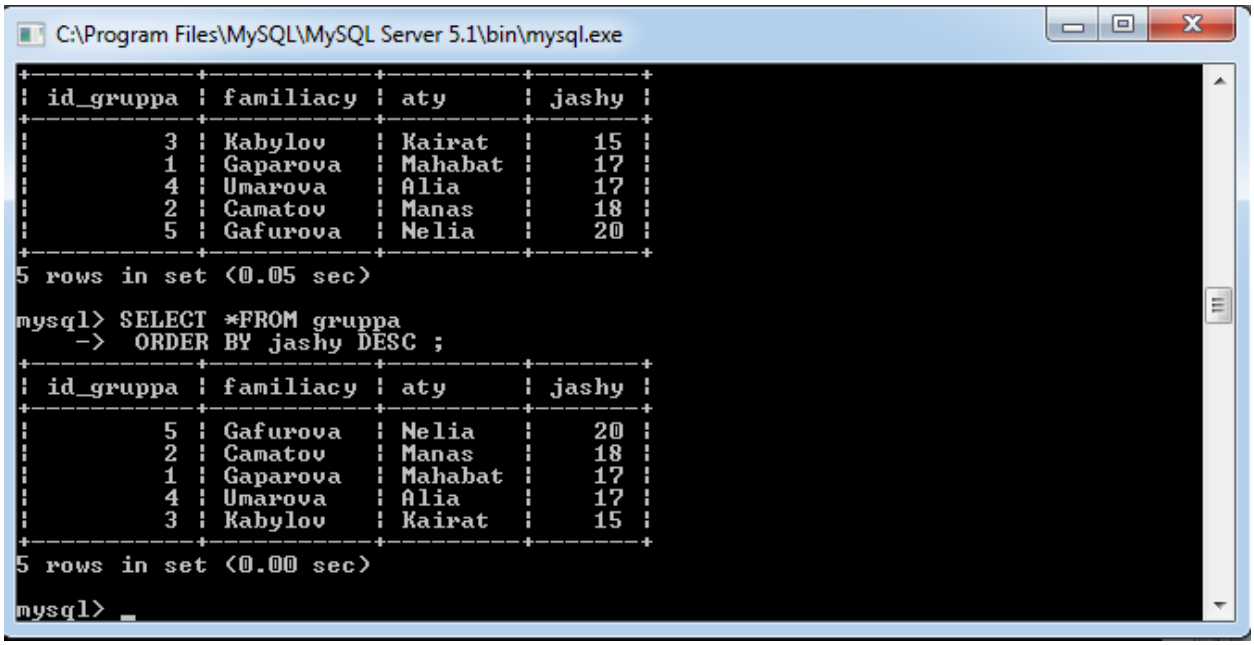
ORDER BY jashy ASC ;



Жыйынтыкта көрсөтүлгөндөй jashy тааласын өсүү тартибинде сортировкаладык. Ушул эле жол менен кемүү тартибинде сортировкаласак болот.

SELECT *FROM gruppa

ORDER BY jashy DESC ;



GROUP BY ОПЕРАТОРУНУН КОЛДОНУЛУШУ.

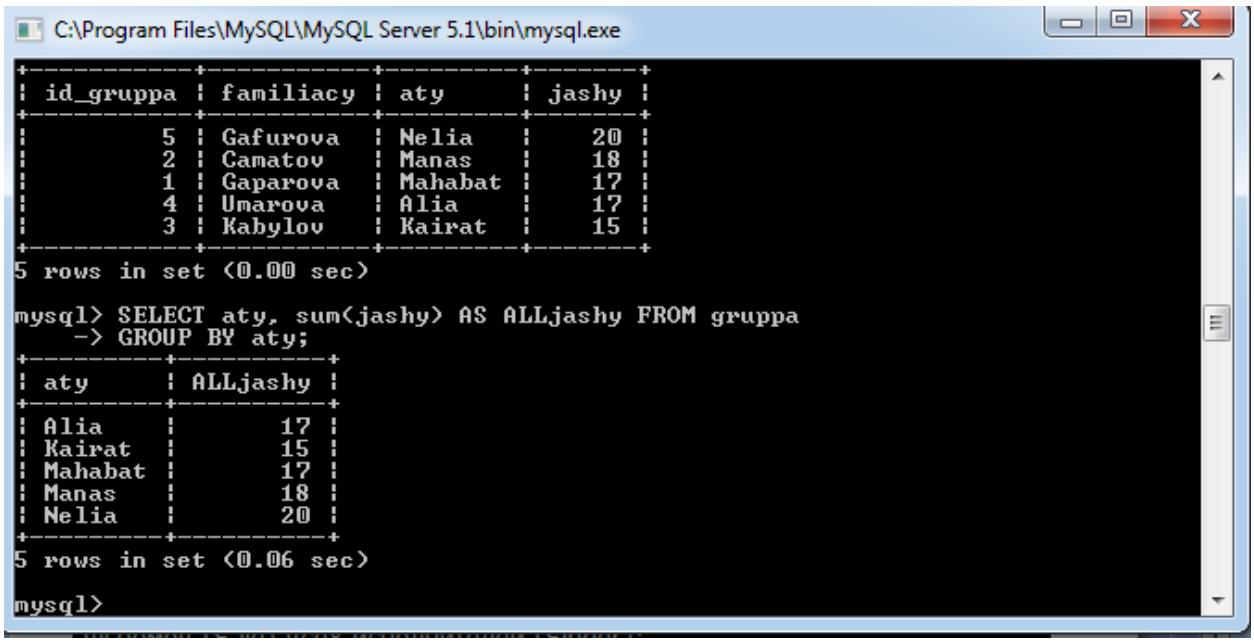
GROUP BY оператору бир же бир нече талаалардагы окшошторунун жыйынтыгын бириктирет. Синтаксиси:

GROUP BY талаанын_аты;

Мисал келтирели.

SELECT aty, SUM(jashy) AS ALLjashys FROM gruppa

GROUP BY aty;



Жогоруда аткарган запросто AS оператору талаага жаңы ат берүүгө мүмкүнчүлүк берди. Бизде таблицабызда aty талаабызда окшош маанилерибиз жок болгондуктан тааланын атын өзгөртүп эле жыйынтыкты чыгарып берди. Эми таблицанын aty талаасына аттары окшош студенттерди кошуп GROUP BY операторун колдонуп көрөлү.

INSERT INTO gruppa VALUES

(NULL ,‘Gafurova’, ‘Nelia’,’28’),

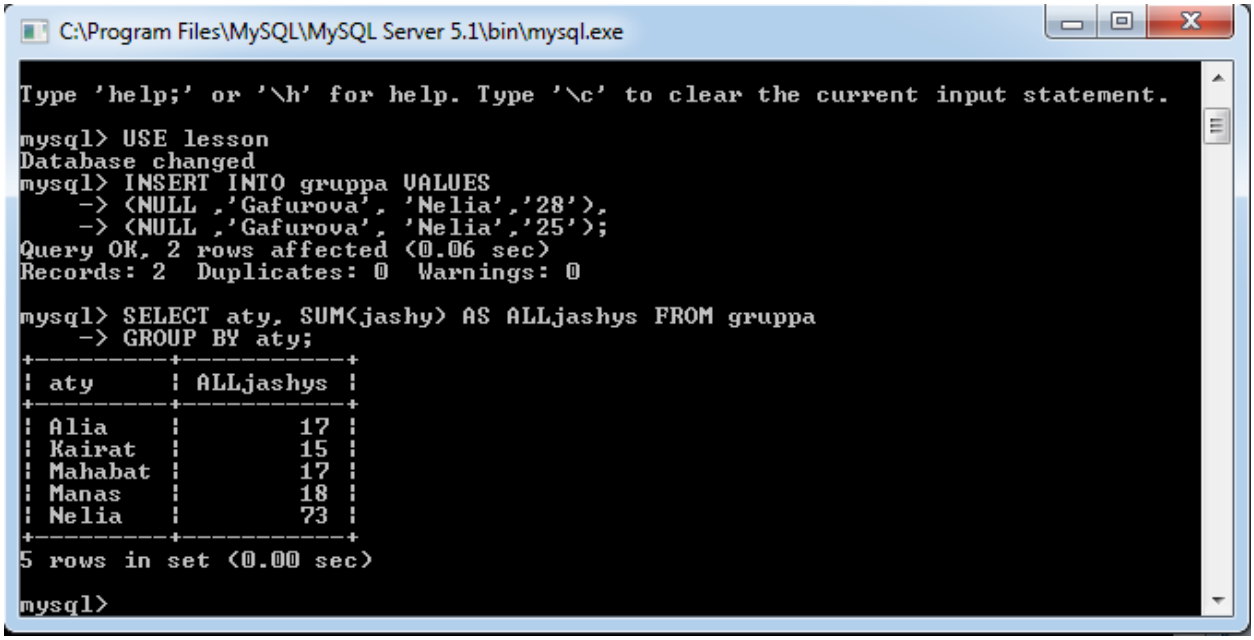
(NULL ,‘Gafurova’, ‘Nelia’,’25’);

Таблицага дагы 2 Nelia деген студент коштук.

SELECT aty, SUM(jashy) AS ALLjashys FROM gruppa

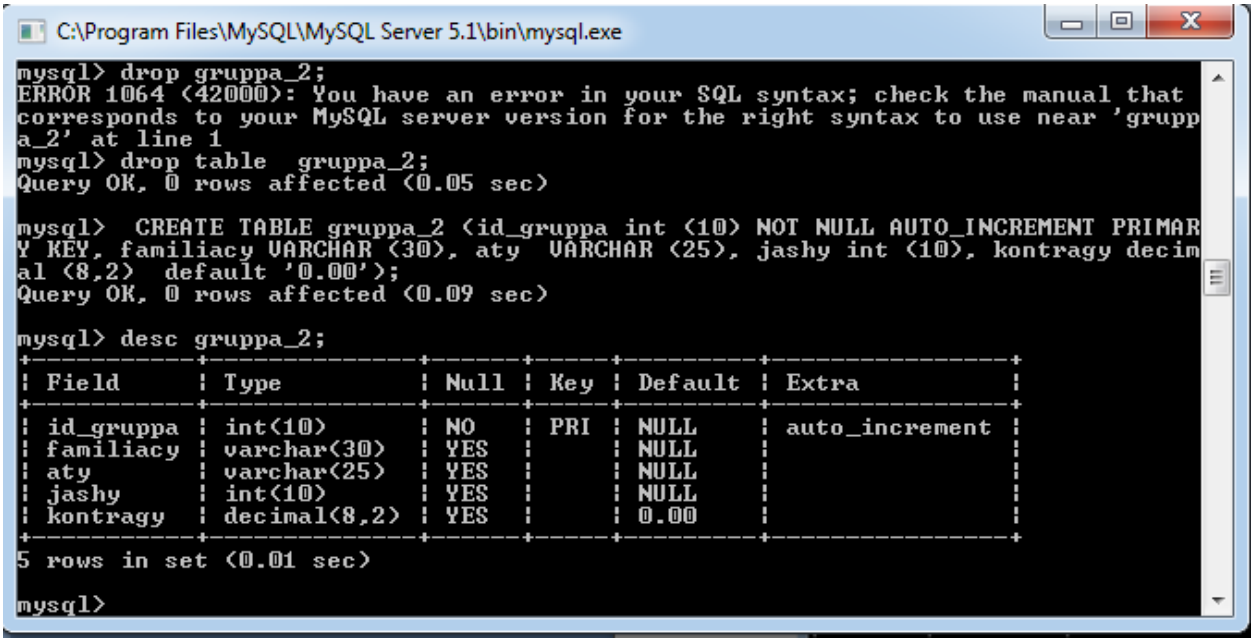
GROUP BY aty;

Ишибиздин жыйынтыгын карайлы. Окшош студенттерди бириктирип жашын суммалап койду.



DEFAULT операторунун жардамында атайын көрсөтүлбөгөн учурда турчу маанини берүүгө болот. Мисалы:

CREATE TABLE gruppa_2 (id_gruppa int (10) NOT NULL AUTO_INCREMENT PRIMARY KEY, familiacy VARCHAR (30), aty VARCHAR (25), jashy int (10), kontragy decimal (8,2) default ‘0.00’);



Жаңы теманы бышыктоочу суроолор:

1. **ORDER BY операторy кандай кызмат аткарат?**
2. **Сортировкалоонун кандай параметрлерин билесин?**
3. **ASC- параметри кандай тартипте сортировкалайт?**
4. **DESC-параметринин сортировкалоо тартиби кандай?**
5. **Группировкалоону кандайча аткарабыз;**

Тема: Виртуладык таблица түзүү жолдору.VIEW, UNION операторлору

1.Сабактын максаты:

Билим берүүчүлүк: Студенттерге Mysql тилинде бар таблицаларды колдонуп виртуалдык таблица түзүү операторлору жөнүндө билимдерди берүү.

Өнүктүрүүчүлүк: Mysql тилинде таблицалардын маалыматтарын колдонуу менен жаңы таблица түзүү жолдорун үйрөтүү.

Тарбиялык: Тыкандыкка, тактыкка, чынчылдыкка, сылыктыкка тарбиялоо.

2. Калыптандыруучу компотенция: ОК-2, ПК-2.3.

3.Сабактын жабдылышы: Проектр, компьютер, доска, бор.

4. Окутууда колдонулган усулдары: Мээ чабуулу (жекече, жуптарда иштөө), ой жүгүртүү.

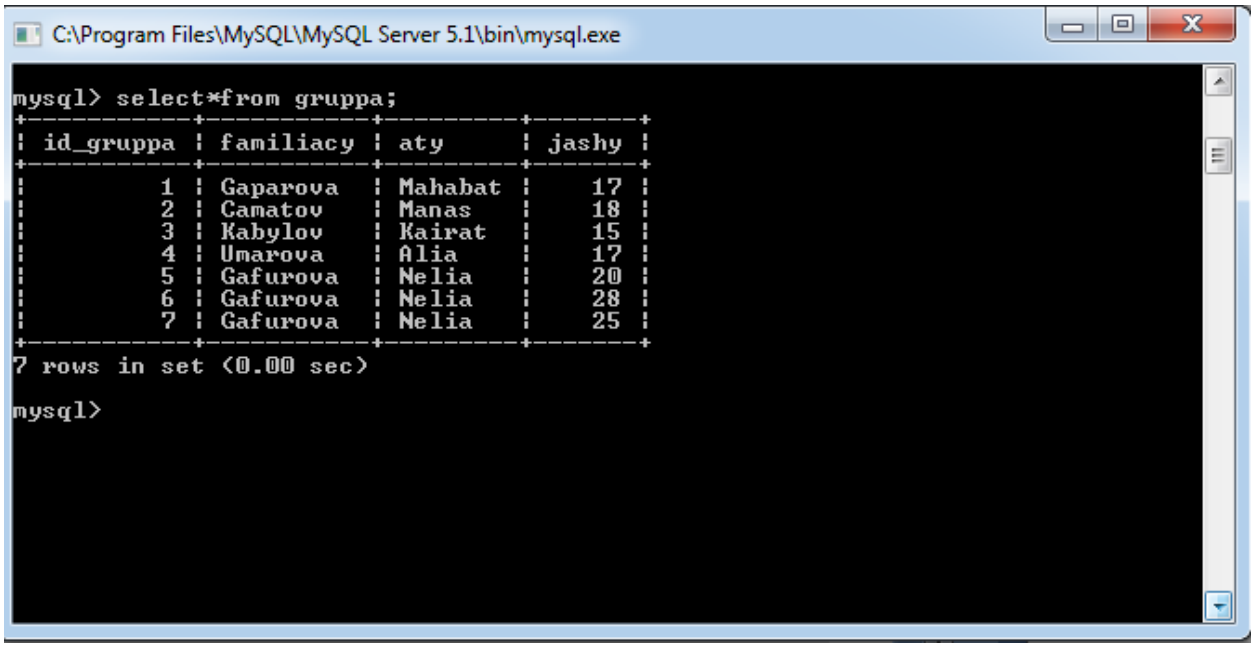
5. Сабактын жүрүшү:

А)Сабакты уюштуруу. Студенттердин катышуусун тактоо, досканын тазалыгына көңүл бөлүү.

Б)Өтүлгөн теманы кайталоо

6. Жаңы теманы түшүндүрүү:

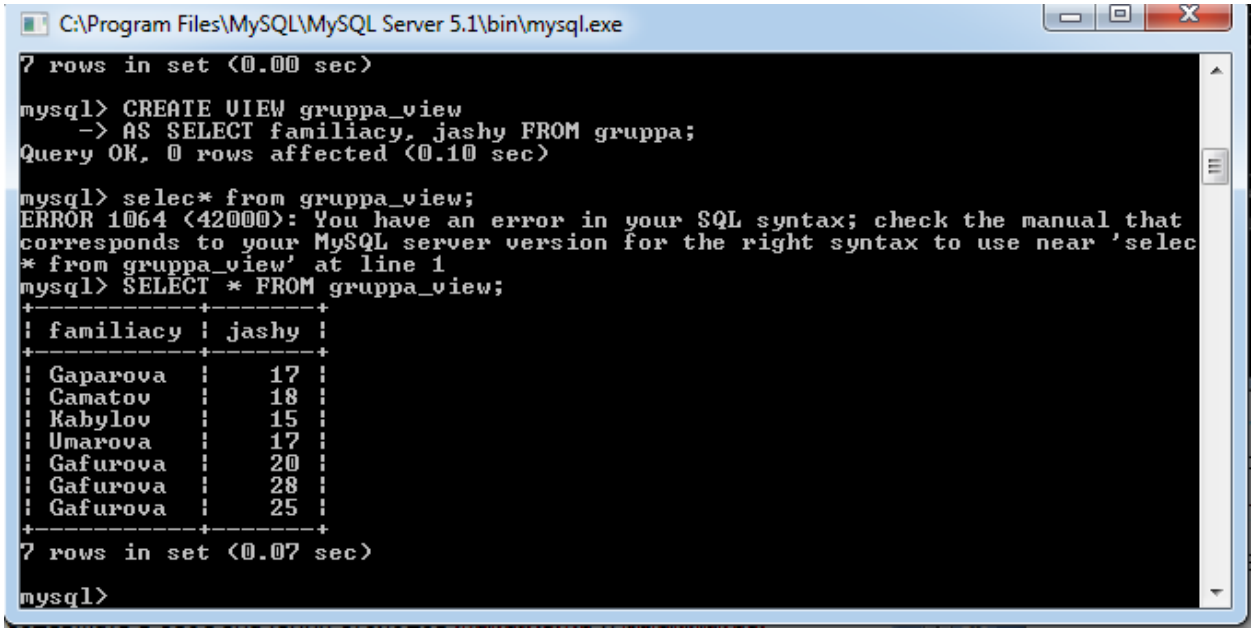
VIEW оператору запростун параметрилерине жараша негизги таблицаны колдонуп виртуалдык таблица түзөт.Виртуалдык таблица берилгендер базасынын структурасын адамдын кабылдоосуна ыңгайлуу кылып көрсөтүүгө мүмкүндүк берет. Бирок негизги таблицадагы маалыматтарга жетүүгө мүмкүндүк бербейт.
Мисалы: Бизде gruppa деген таблицабыз бар.



VIEW операторун колдонуп gruppa таблицасынын эки талаасынан турган таблица түзөлү.
CREATE VIEW gruppa_view
AS SELECT familiacy, jashy FROM gruppa;
деп таблица түздүк. Түзүлгөн таблицаны көрүүнү аткаралы.

SELECT * FROM gruppa_view;

Ишибиздин жыйынтыгы төмөнкүчө:



UNION оператору.

UNION оператору эки же андан көп SELECT операторунун запросторун бириктрүүдө колдонулат. Төмөнкүдөй синтаксиске ээ:

SELECT талаанын_аты FROM 1-таблицанын_аты

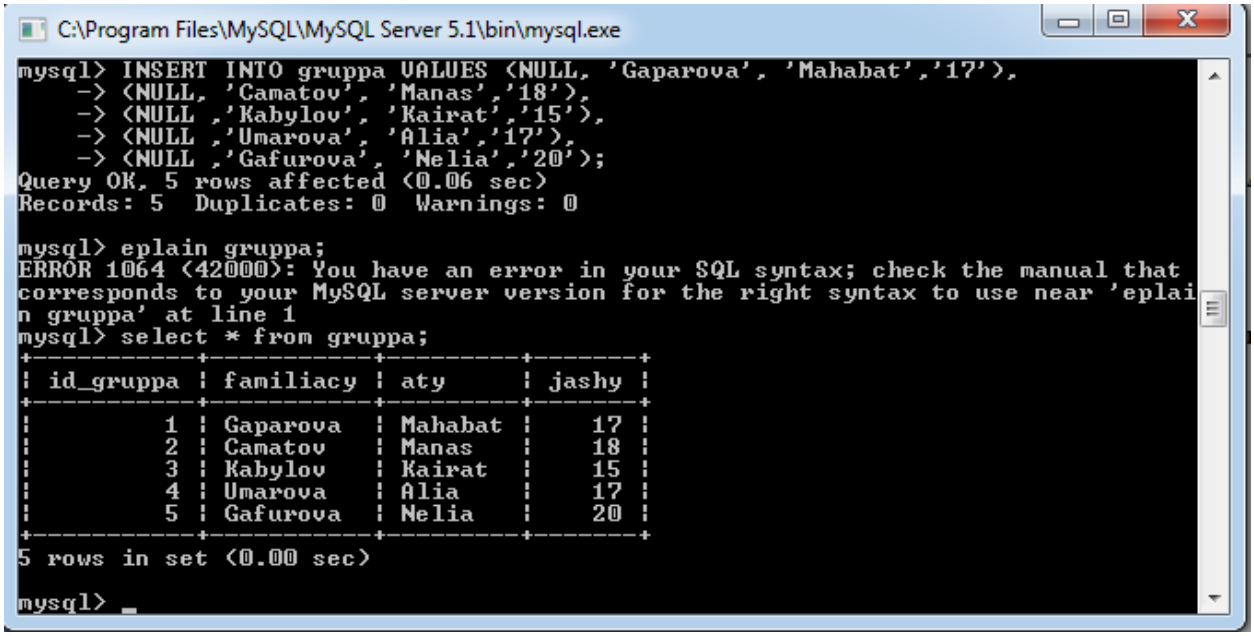
UNION

SELECT талаанын_аты FROM 2- таблицанын_аты

Баса белгилеп айтуу керек SELECT операторунун ар бир запросунда талаалардын саны бир дей жана типтери бирдей болуусу керке. Антпесе жыйынтыктоочу таблицаны формировкалдоодо каталар пайда болот.

Мисалы:

Жаңы gruppa таблицасын колдонолу.



Кошумча дагы бир таблица түзөлү.

CREATE TABLE krujoktor (

id_krujok int (10) AUTO_INCREMENT PRIMARY KEY,

krujoktun_atalyshy VARCHAR (30)

);

Таблицабыздын ичин толтуралы

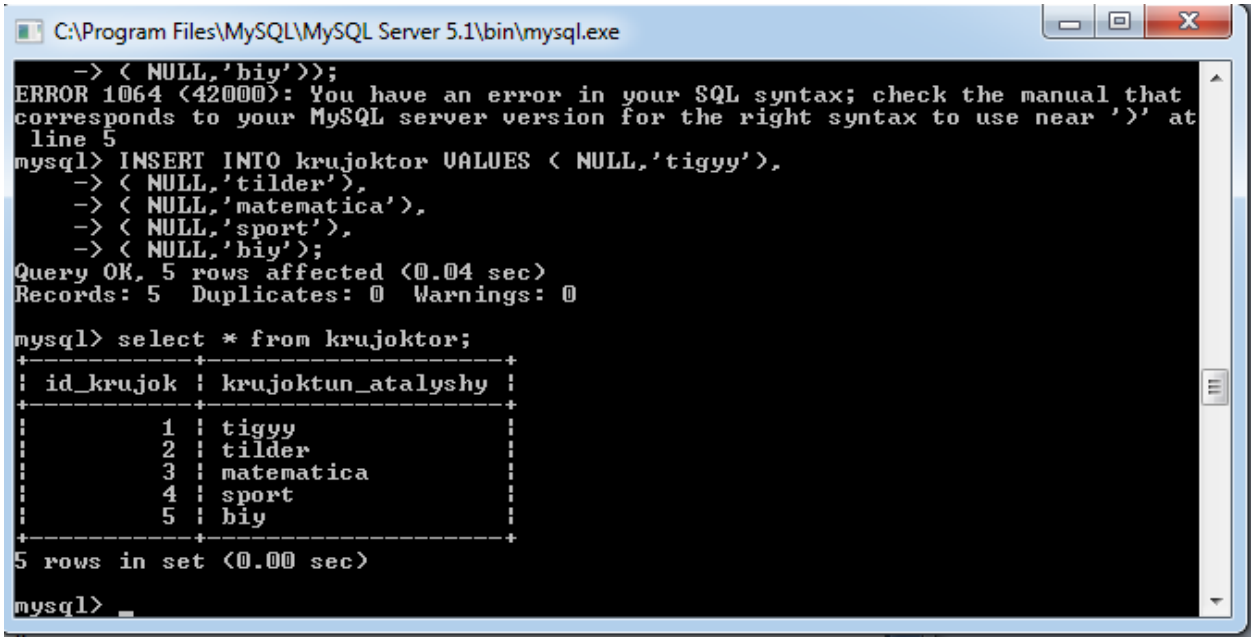
INSERT INTO krujoktor VALUES (NULL,'tigyy'),

(NULL,'tilder'),

(NULL,'matematica'),

(NULL,'sport'),

(NULL,'biy');

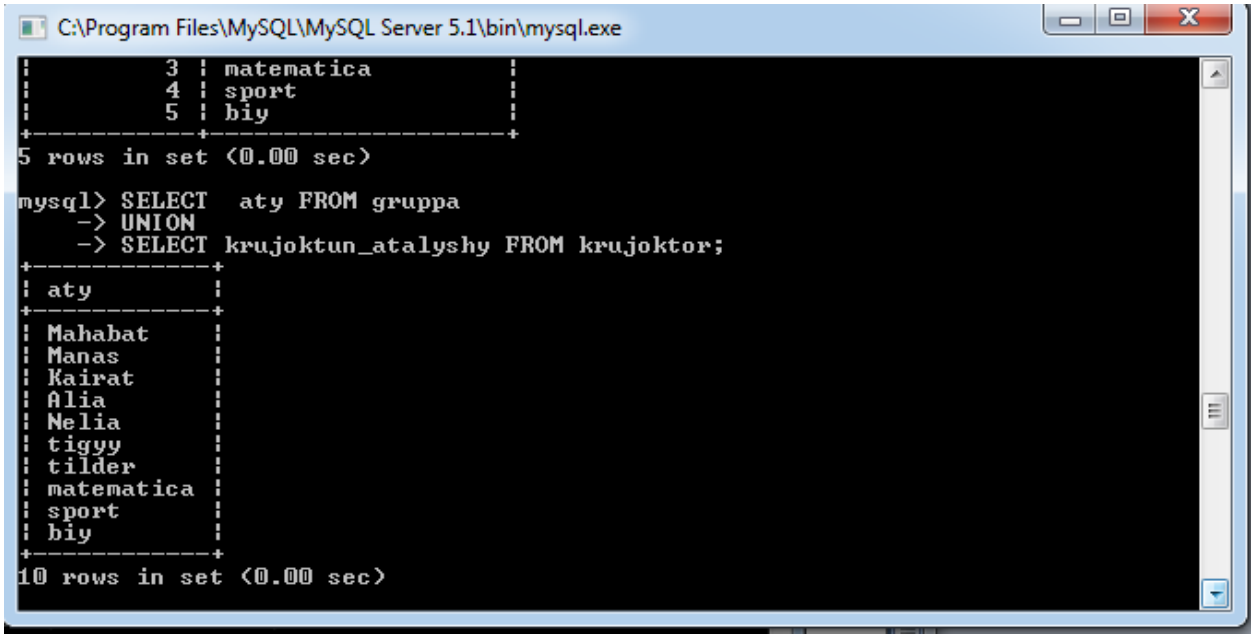


UNION операторун колдонуп запрос жазып көрөлү.

SELECT aty FROM gruppa

UNION

SELECT krujoktun_atalyshy FROM krujoktor;



Жаңы теманы бышыктоо үчүн суроолор:

1.VIEW операторунун колодонулушу

2. UNION операторун колдонууда кандай шарттарды эске алуу керек?

3. Виртуалдык таблица түзүү үчүн кандай иш аракеттерди жасайбыз?

Тема: Таблицалардын ортосундагы байланыштар. Индекстер. Сырткы жана ички ключтөр.

1.Сабактын максаты:

Студенттер Mysql тилинде таблицалардын ортосундагы байланыштар, байланыштардын түрлөрү, индекстер, ички жана сырткы ключтөр жөндүндө билимдерди берет.

2. Калыптандыруучу компотенция: ЖК-1, ЖК-2.

6. Жаңы теманы түшүндүрүү

Берилгендер базасын проектирлөөдө колдонуучу түрдүү берилгендер үчүн т үрдүү таблицаларды колдонушу мүмкүн. Ал таблицалар өз ара байланышы мүмкүн. Төмөндө байланыштын типтерин карайлы.

- Бирден-бирге
- Бирден көпкө, көптөн бирге
- Көптөн-көпкө

Биз байланыш түзүү үчүн 3 таблица түзүп көрөлү.

```
Create table strana (
id_strana INT AUTO_INCREMENT  PRIMARY KEY,
 name_strana varchar (30));

Create table igroki (
id_igroka INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
 id_strana INT (10),
 name_igroka  varchar (30),
FOREIGN KEY (id_strana) REFERENCES  strana (id_strana));

Create table komanda (
id_komanda  INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
 id_strana INT (10),
 id_igroka  INT (10),
```

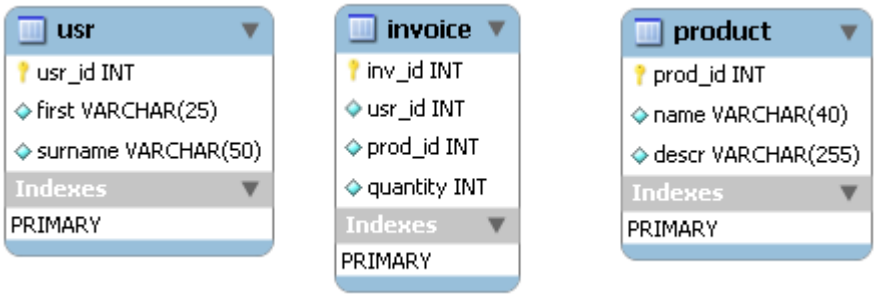
name_komanda varchar (30),
FOREIGN KEY (id_strana) REFERENCES strana (id_strana),
FOREIGN KEY (id_igroka) REFERENCES igroki (id_igroka));

Мында **PRIMARY KEY-** ТАЛААНЫН ИНДЕКСИ БОЛУП ЭСЕПТЕЛЕТ ЖАНА АР БИР МААНИСИ УНИКАЛДУУ. БУЛ БИР ИДЕНТИФИКАТОРДУ КАЙТАЛАП КОЛДОНУУГА МУМКУНДУК БЕРБЕЙТ.
СИНТАКСИСИ: **PRIMARY KEY**(ТАЛААНЫН_АТЫ);
FOREIGN KEY- сырткы ключту көрсөтүү үчүн колдонулат.
Синтаксиси:
FOREIGN KEY (сырткы_ключ_болгон_талаанын_аты) **REFERENCES** родительский_таблицанын_аты (родителдик_таблицанын_талаасынын_аты);
INSERT же UPDATE операторлордун колдонуп запросторду жазып көрөлү.

```
CREATE TABLE usr (
      usr_id  INT AUTO_INCREMENT NOT NULL,
      FIRST  VARCHAR(25) NOT NULL,
        surname  VARCHAR(50) NOT NULL,
        PRIMARY KEY(usr_id)
       );
CREATE TABLE product (
         prod_id  INT AUTO_INCREMENT NOT NULL,
        name  VARCHAR(40) NOT NULL,
        descr  VARCHAR(255) NOT NULL,
        PRIMARY KEY(prod_id)
      ) ;
CREATE TABLE invoice (
        inv_id  INT AUTO_INCREMENT NOT NULL,
        usr_id  INT NOT NULL,
        prod_id  INT NOT NULL,
        quantity INT NOT NULL,
```

```
    PRIMARY KEY(inv_id)
  ) ;
```

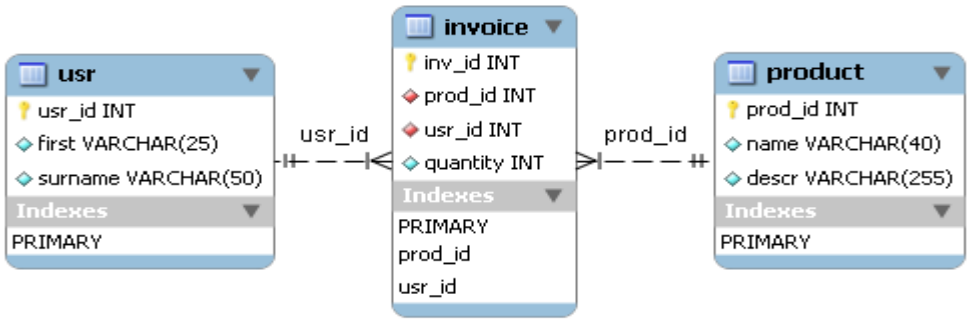
Биздин таблицаларыбыздын структурасы төмөнкүдөй болсун:



Сырткы ключтөрдү кошолу

Жогорудагы таблицаларды байланыштыралы

```
 CREATE TABLE invoice (
   inv_id  INT AUTO_INCREMENT NOT NULL,
   usr_id  INT NOT NULL,
   prod_id  INT NOT NULL,
   quantity INT NOT NULL,
   PRIMARY KEY(inv_id),
   FOREIGN KEY (usr_id) REFERENCES usr(usr_id),
   FOREIGN KEY (prod_id) REFERENCES product(prod_id)
 ) ;
```



Биз сырткы ключтөрдү кошуу менен берилгендердин бүтүндүгүн коргоого жетиштик. Эгерде биз **INSERT** же **UPDATE** запросторун аткаргыбыз келсе анда берилгендер базасы байланышкан таблицаларда изделип жаткан маани бар же жок

экендигин текшерет. Эгер көрсөтүлгөн маани байланышкан таблицаларда жок болсо анда берилгендер базасы запросту аткарабайт.

MySQL тили негизги таблицада берилгендерди жаңылоо же өчүрүү (обновления или удаления данных) убагында байланышкан таблицаларда да жүргүзүлүшүнө мүмкүндүк берет. Таблицада маанилерди жаңылоо жана өчүрүү: **ON UPDATE** и **ON DELETE**нин жардамында жүргүзүлөт**..** MySQL тили **ON UPDATE** же **ON DELETE** маанилерин жардамында 5 иш аракетти жасоо мүмкүнчүлүгүн берет .

- **CASCADE**: эгерде байланышкан негизги таблицадагы жазууларда өзгөрүү же өчүрүлүү болсо анда байанышкан таблицаларда да өчүрүү жаңылоо орун алышы керек.
- **SET NULL**: эгерде негизги таблицадагы жазууларда жаңылану же өчүрүү жүргүзүлсө анда байланышкан таблицалардын айрым жазууларына NULL маанисин ыйгаруу мүмкүн. (таблица мүмкүндүк берген учурда гана)
- **NO ACTION**: RESTRICT менен эквиваленттүү
- **RESTRICT**: эгер негизги таблицадагы байланышкан жазуулар жаңыланса же жазуулары менен өчүрүлсө бирок ошол жазуулар башка байланышкан таблицаларда ошол жазуулар бар болсо анда берилгендер базасы аны өчүрүүгө жол бербейт.
- **SET DEFAULT**:азыркы учурда бул команда парсер тарабынан таанылып жатат төмөнкү мисалда invoice таблицасында , UPDATE (жаңылоо) гана жүргүзүлөт бирок өчүрүүгө жол берилбейт. Ошентип usr жана product таблицасындагы ар бир өзгөрүү автоматтык түрдө invoice таблицасында чагылат.Бирок товарга буюртма берилген болсо же колдонуучуда счету болсо ал өчүрүлбөйт.

Мисалда ON UPDATE жана **ON DELETEнин колдонулушун карап көрөлү**

```
1
2  CREATE TABLE invoice (
3        inv_id  INT AUTO_INCREMENT NOT NULL,
4        usr_id  INT NOT NULL,
5        prod_id  INT NOT NULL,
```

```
6         quantity INT NOT NULL,
7         PRIMARY KEY(inv_id),
8         FOREIGN KEY (usr_id) REFERENCES usr(usr_id)
9           ON UPDATE CASCADE
10          ON DELETE RESTRICT,
11        FOREIGN KEY (prod_id) REFERENCES product(prod_id)
12          ON UPDATE CASCADE
 13         ON DELETE RESTRICT
        ) ;
```

Мугалимдердин тизмесин камтыган таблица түзөлү:

```
CREATE TABLE teacher(
id TINYINT(3) AUTO_INCREMENT,
name VARCHAR(30),
PRIMARY KEY(id)
) ;
```

Таблицаны толтуралы:

```
INSERT INTO teacher (name) VALUES ('Chamashev'), ('Sadanov'), ('Sulaimanov');
+----+--------------+
| id | name         |
+----+--------------+
|  1 | Chamashev    |
|  2 | Sadanov |
|  3 | Sulaimanov   |
+----+--------------+
```

```
3 rows in set (0.00 sec)
```

Студенттердин таблицасын түзөлү:

```
CREATE TABLE student(
id TINYINT (3) AUTO_INCREMENT,
name VARCHAR(30),
PRIMARY KEY(id)
) ;
```

Үчүнчү **student_teacher_consultation таблицасын түзүп байланыш түзүү үчүн запросторду аткаралы**:

```
CREATE TABLE student_teacher_consultation (
  id tinyint NOT NULL,
  student_id tinyint DEFAULT NULL,
  teacher_id tinyint DEFAULT NULL,
  consultation_date datetime DEFAULT NULL
) ;

ALTER TABLE student_teacher_consultation ADD INDEX(student_id);
ALTER TABLE student_teacher_consultation ADD INDEX(teacher_id);
```

```
ALTER TABLE student_teacher_consultation
ADD CONSTRAINT st_student_id
FOREIGN KEY (student_id) REFERENCES student(id)
ON UPDATE CASCADE ON DELETE CASCADE;


ALTER TABLE student_teacher_consultation
ADD CONSTRAINT st_teacher_id
FOREIGN KEY (teacher_id) REFERENCES teacher(id)
ON UPDATE CASCADE ON DELETE CASCADE;
```

Таблицаларды толтуралы:

```
+----+------------+------------+--------------------+
| id | student_id | teacher_id | consultation_date  |
+----+------------+------------+--------------------+
|  1 |          1 |          2 | 2015-07-23 10:40:00 |
|  2 |          2 |          1 | 2015-07-17 12:10:00 |
|  3 |          5 |          1 | 2015-07-17 13:10:00 |
|  4 |          8 |          3 | 2015-07-27 14:15:00 |
|  5 |          6 |          3 | 2015-07-11 10:34:00 |
+----+------------+------------+--------------------
```

Ушинтип биз кошумча таблицанын жардамында мугалимдер жана студенттер таблицасынын ортосуда көптөн-көпкө байланышын тургуздук.

Жаңы теманы бышыктоочу суроолор:

1. Ички ключ менен сырткы ключтун айырмасы кандай?
2. Байланыштын кандай түрлөрү бар?
3. Cascade операторунун колдонулушу?

Тема: JOIN оператору жана анын колдонулушу

Оператор JOIN оператору эки жана андан ашык таблицаларды байланыштырууда колдонулат. Байланыштыруу используется для **соединения двух или нескольких таблиц**. Соединение таблиц может быть внутренним (INNER) или внешним (OUTER), причем внешнее соединение может быть левым (LEFT), правым (RIGHT) или полным (FULL). Далее на примере двух таблиц рассмотрим различные варианты их соединения.

Синтаксис соединения таблиц оператором JOIN имеет вид:

```
FROM <таблица 1>
 [INNER]
 {{LEFT | RIGHT | FULL } [OUTER]} JOIN <таблица 2>
[ON <предикат>]
```

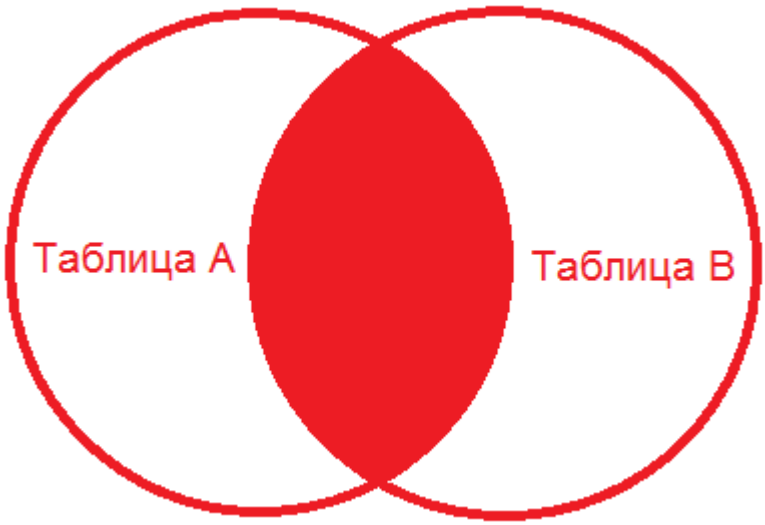
Предикат в этой конструкции определяет условие соединения строк из разных таблиц.

Допустим есть две таблицы (Auto слева и Selling справа), в каждой по четыре записи. Одна таблица содержит названия марок автомобилей (Auto), вторая количество проданных автомобилей (Selling):

```
id   name        id   sum
--   ----        --   ----
1    bmw          1    250
2    opel         5    450
3    kia          3    300
4    audi         6    400
```

Далее соединим эти таблицы по полю id несколькими различными способами. Совпадающие значения выделены красным для лучшего восприятия.

1. **Внутреннее соединение** (**INNER JOIN**) означает, что в результирующий набор попадут только те соединения строк двух таблиц, для которых значение предиката равно TRUE. Обычно используется для **объединения записей, которые есть и в первой и во второй таблице**, т. е. получения **пересечения таблиц**:



Красным выделена область, которую мы должны получить.

Итак, сам запрос:

```
SELECT * FROM 'Auto'
INNER JOIN 'Selling' ON 'Auto'.id = 'Selling'.id
```

И результат:

```
id   name        id   sum
```

```
--  ----      --  ----
1   bmw       1   250
3   kia       3   300
```

Ключевое слово INNER в запросе можно опустить.

В итоге запрос отбирает и соединяет те записи, у которых значение поля id в обоих таблицах совпадает.

2. **Внешнее соединение** (**OUTER JOIN**) бывает нескольких видов. Первым рассмотрим **полное внешнее объединение** (**FULL OUTER JOIN**), которое объединяет записи из обоих таблиц (если условие объединения равно true) и дополняет их всеми записями из обоих таблиц, которые не имеют совпадений. Для записей, которые не имеют совпадений из другой таблицы, недостающее поле будет иметь значение NULL. Граф выборки записей будет иметь вид:

Переходим к запросу:

```
SELECT * FROM 'Auto'
FULL OUTER JOIN 'Selling' ON 'Auto'.id = 'Selling'.id
```

Результат:

```
id  name      id  sum
--  ----      --  ----
```

```
1    bmw          1    250
2    opel       NULL   NULL
3    kia          3    300
4    audi       NULL   NULL
NULL NULL         5    450
NULL NULL         6    400
```

То есть мы получили все записи, которые есть в обоих таблицах. Записи у которых значение поля id совпадает соединяются, а у записей для которых совпадений не найдено недостающие поля заполняются значением NULL.

Ключевое слово OUTER можно опустить.

3. **Левое внешнее объединение** (**LEFT OUTER JOIN**). В этом случае получаем все записи удовлетворяющие условию объединения, плюс все оставшиеся записи из внешней таблицы, которые не удовлетворяют условию объединения. Граф выборки:

Запрос:

```
SELECT * FROM 'Auto'
LEFT OUTER JOIN 'Selling' ON 'Auto'.id = 'Selling'.id
```

Результат:

```
id   name        id   sum
```

```
--  ----      --  ----
1   bmw         1   250
2   opel      NULL  NULL
3   kia         3   300
4   audi      NULL  NULL
```

Запрос также можно писать без ключевого слова OUTER.

В итоге здесь мы получили все записи таблицы Auto. Записи для которых были найдены совпадения по полю id в таблице Selling соединяются, для остальных недостающие поля заполняются значением NULL.

Еще существует **правое внешнее объединение** (**RIGHT OUTER JOIN**). Оно работает точно также как и левое объединение, только в качестве внешней таблицы будет использоваться правая (в нашем случае таблица Selling или таблица Б на графе).

Далее рассмотрим остальные возможные выборки с использованием объединения двух таблиц.

4. Получить все записи из таблицы А, которые не имеют объединения из таблицы Б. Граф:

То есть в нашем случае, нам надо получить все автомобили из таблицы Auto, которые не имеют продаж в таблице Selling.

Запрос:

```
SELECT * FROM 'Auto'
LEFT OUTER JOIN 'Selling' ON 'Auto'.id = 'Selling'.id
WHERE 'Selling'.id IS null
```

Результат:

```
id   name        id   sum
--   ----        --   ----
2    opel        NULL   NULL
4    audi        NULL   NULL
```

5. И последний вариант, получить все записи из таблицы А и Таблицы Б, которые не имеют объединений. Граф:

В нашем случае мы должны получить все записи из таблицы Auto, которые не связаны с таблицей Selling, и все записи из таблицы Selling, которые не имеют сопоставления из таблицы Auto.

Запрос:

```
SELECT * FROM 'Auto'
FULL OUTER JOIN 'Selling'
ON 'Auto'.id = 'Selling'.id
WHERE 'Auto'.id IS null
OR 'Selling'.id IS null
```

Результат:

id	name	id	sum
--	----	--	----
2	opel	NULL	NULL
4	audi	NULL	NULL
NULL	NULL	5	450
NULL	NULL	6	400

Тема: Rename, Modify, Delete, Truncate операторлорунун колдонулушу

Ар кандай эле түзүлгөн таблицанын атын өзгөртүүгө болот.Ал үчүн **Rename** оператору колдонулат.

My sql де таблицанын атын эки жол менен өзгөртүүгө болот:

1. Alter table операторун колдонуу менен. Мисалы user таблицасын users деп өзгөртөлү.

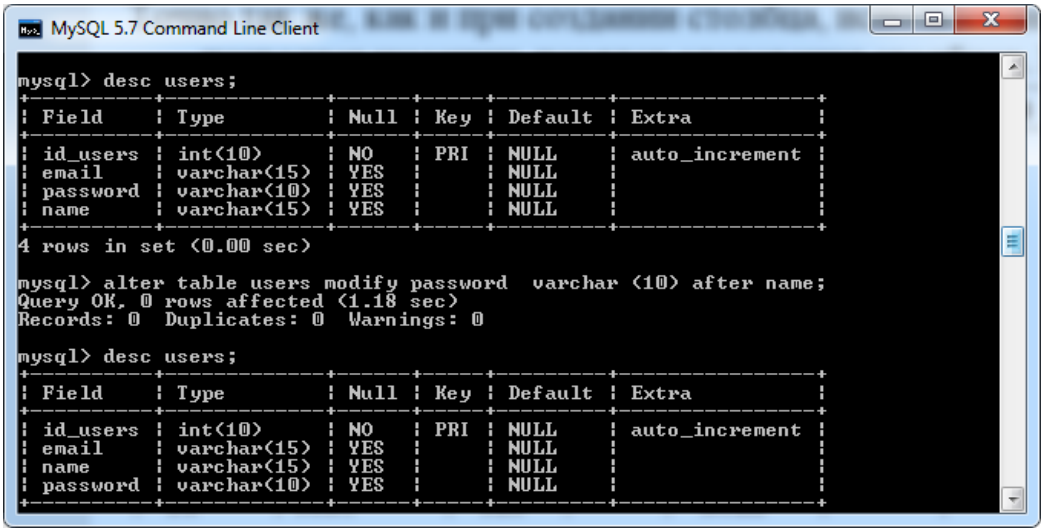
   Alter table user rename to users;

2. Rename table операторунун жардамында. Анда жогорку кодду төмөнкүдөй жазабыз.

   Rename table user to users;

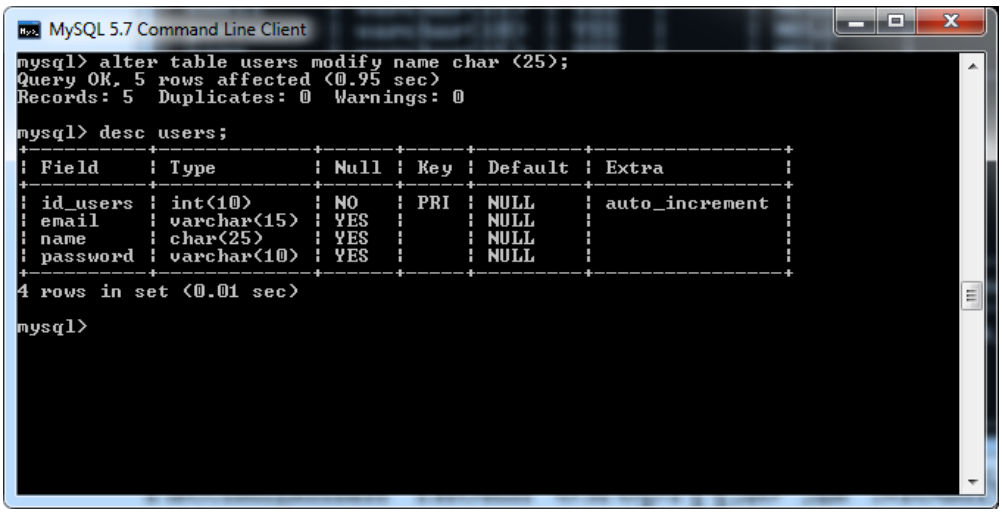
   Modify операторунун жардамында талаалардын ордун алмаштырууга болот. Төмөндөгү Users таблицасындагы password талаасы менен name талаасынын ордун алмаштыралы. Ал үчүн код жазабыз:

   Alter table users modify password varchar (10) after name;



Таблицанын тибин өзгөртүүдө да Modify операторун колдонобуз. Мисалы: Users таблицасындагы name талаасынын varchar (15) тибин char (25) тибине алмаштыралы. Анда коду төмөнкүдөй жазылат.

Alter table users modify name char (25);

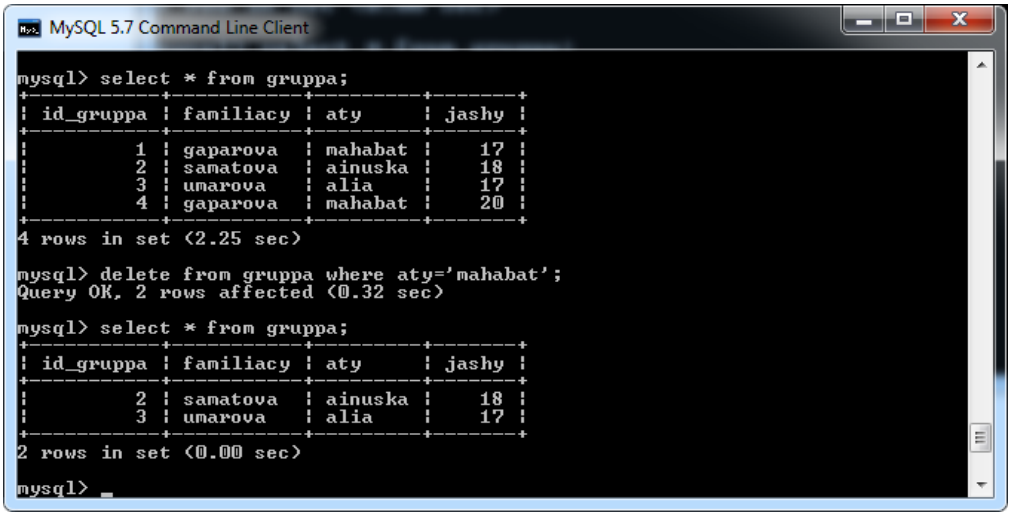


Таблицадан берилгендерди өчүрүү үчүн DELETE оператору колдонулат. Синтаксиси төмөнкүдөй жазылат.

Delete from Таблицанын_аты **where** өчүрүүнүн_шарты;

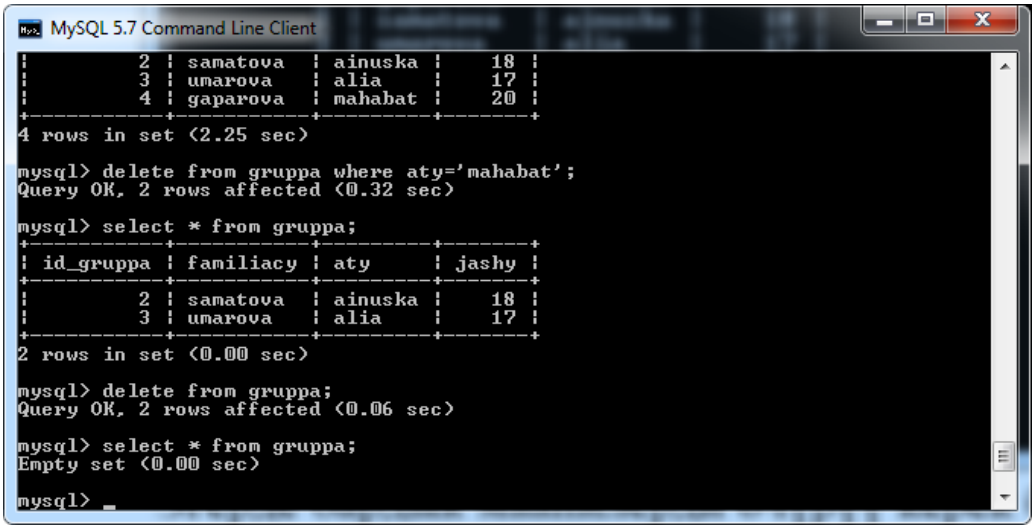
Мисалы gruppa таблицасынан Mahabat аттуу ысымды өчүрөлү.

Delete from gruppa where aty=’mahabat’;



Эгерде бардык маанилерди өчүрүү керек болсо анда эч кандай шарт бербестен кодду төмөнкүдөй жазабыз.

Delete from gruppa;



Delete операторунан айрмаланып Truncate оператору берилгендердин бардык маанилерин эч кандай шарт колдонбой өчүрөт. Синтаксиси: Truncate table Таблицанын_аты;

Мисалы: Truncate table gruppa;

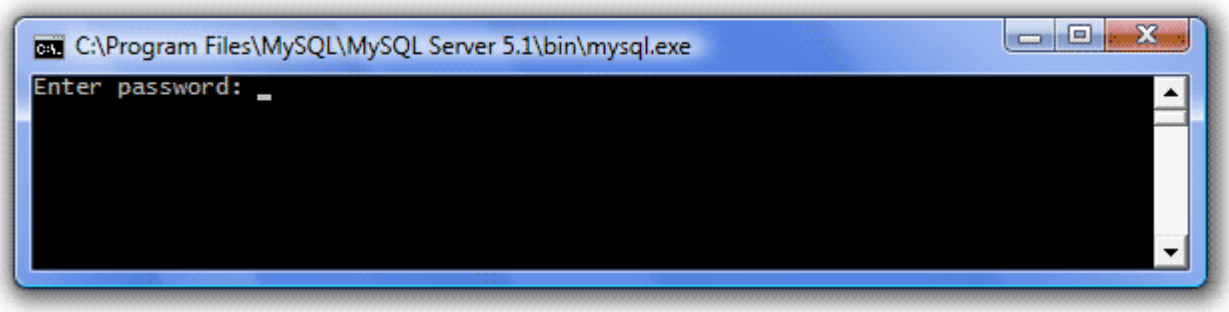
ПРАКТИКАЛЫК МАТЕРИАЛДАР

Лабораторная работа №1

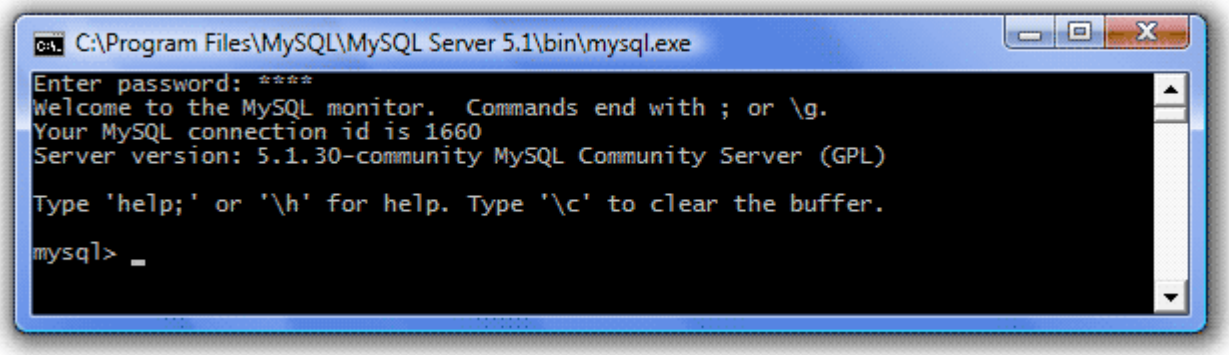
Тема: Создание базы данных и таблиц

Итак, вы установили MySQL, и мы начинаем осваивать язык SQL. В уроке 3 по основам баз данных, мы создали концептуальную модель маленькой БД для форума. Пришло время реализовать ее в СУБД MySQL.

Для этого прежде всего надо запустить сервер MySQL. Идем в системное меню Пуск - Программы - MySQL - MySQL Server 5.1 - MySQL Command Line Client. Откроется окно, предлагающее ввести пароль.



Нажимаем Enter на клавиатуре, если вы не указывали пароль при настройке сервера или указываем пароль, если вы его задавали. Ждем приглашения mysql>.

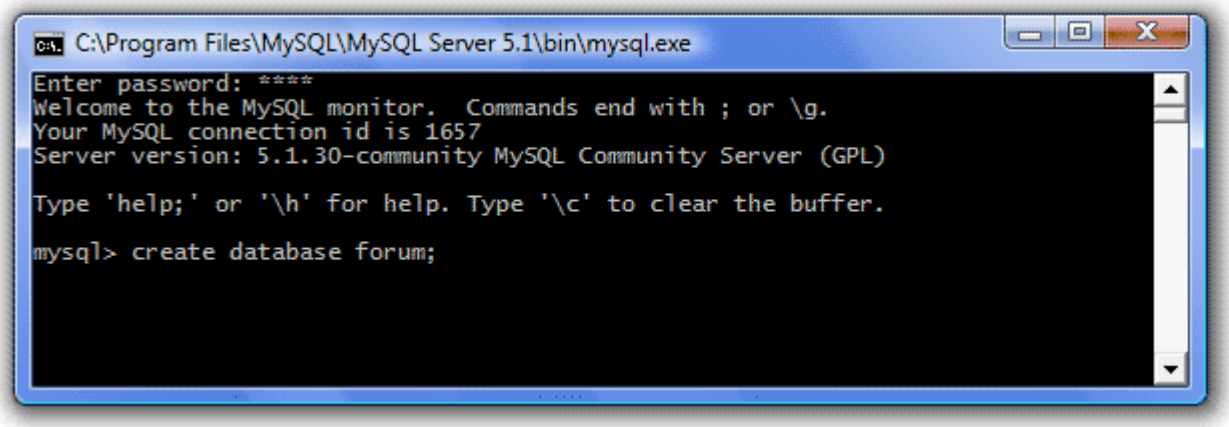


Нам надо создать базу данных, которую мы назовем forum. Для этого в SQL существует оператор *create database*. Создание базы данных имеет следующий синтаксис:

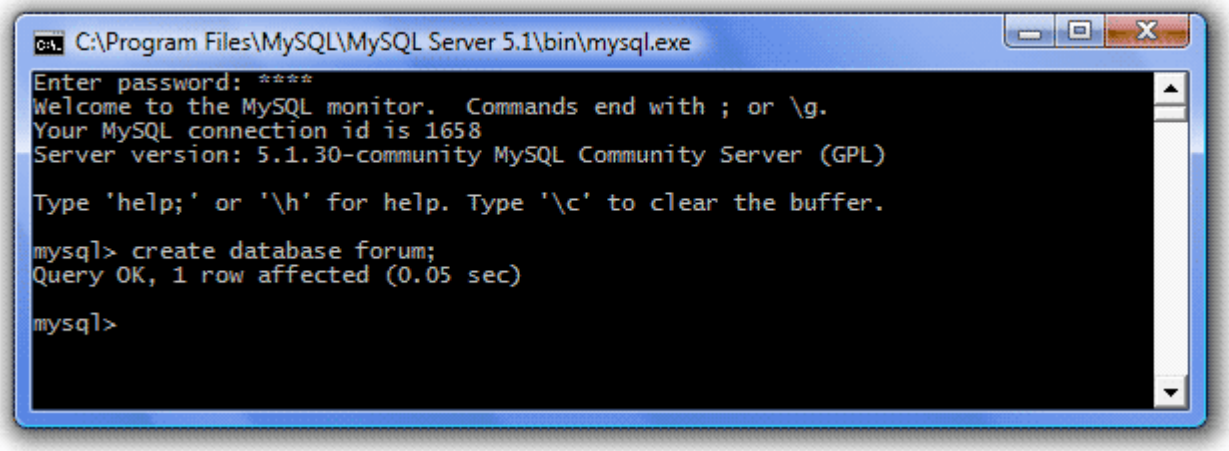
```
create database имя_базы_данных;
```

Максимальная длина имени БД составляет 64 знака и может включать буквы, цифры, символ "_" и символ "$". Имя может начинаться с цифры, но не должно полностью состоять из цифр. Любой запрос к БД заканчивается точкой с запятой (этот символ называется разделителем - delimiter). Получив запрос, сервер выполняет его и в случае успеха выдает сообщение "Query OK ..."

Итак, создадим БД forum:



Нажимаем Enter и видим ответ "Query OK ...", означающий, что БД была создана:

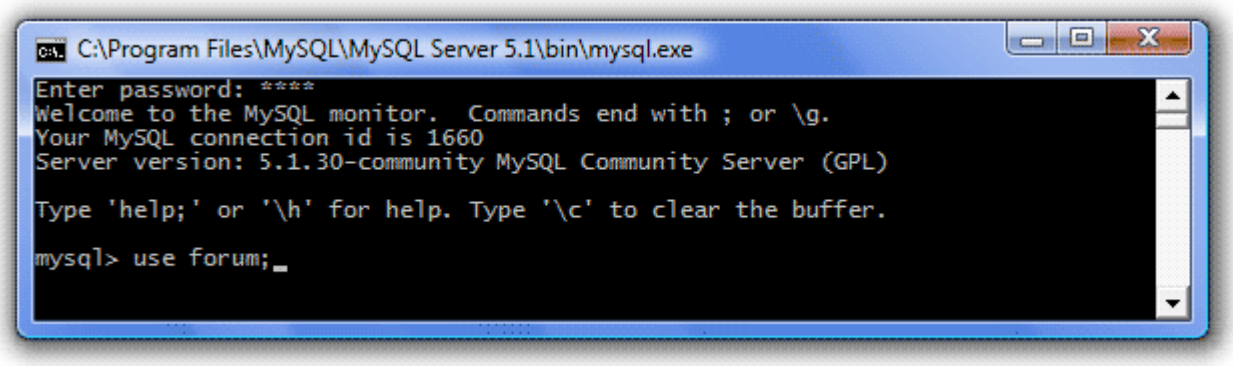


Вот так все просто. Теперь в этой базе данных нам надо создать 3 таблицы: темы, пользователи и сообщения. Но перед тем, как это делать, нам надо указать серверу в

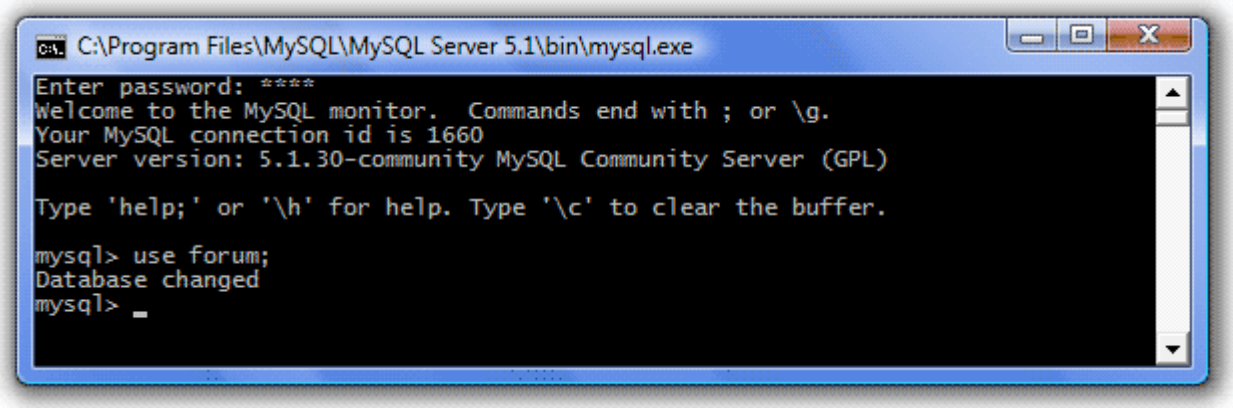
какую именно БД мы создаем таблицы, т.е. надо выбрать БД для работы. Для этого используется оператор *use*. Синтаксис выбора БД для работы следующий:

```
use имя_базы_данных;
```

Итак, выберем для работы нашу БД forum:



Нажимаем Enter и видим ответ "Database changed" - база данных выбрана.

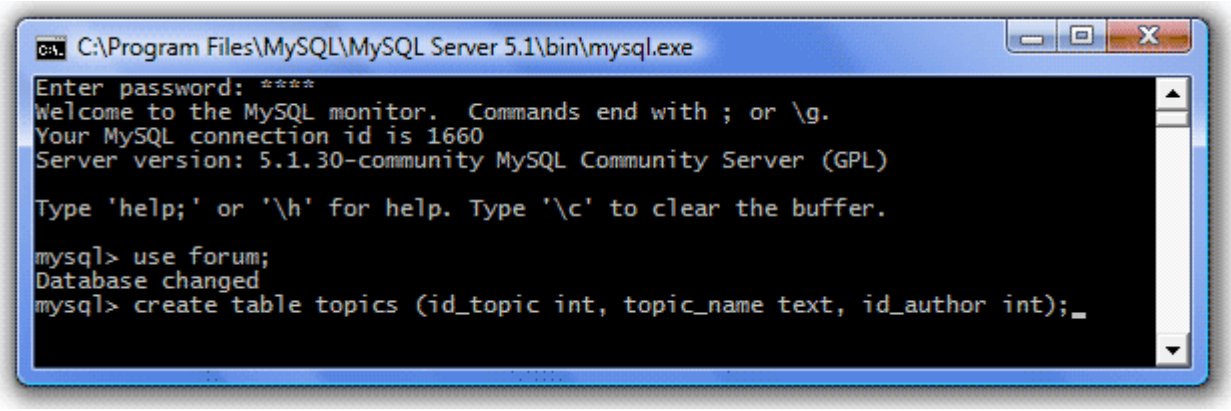


Выбирать БД необходимо в каждом сеансе работы с MySQL.

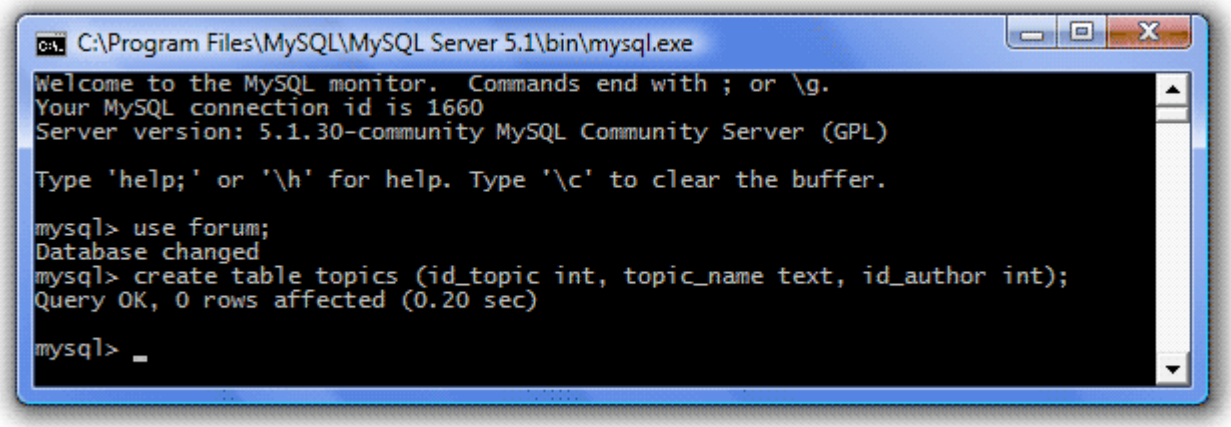
Для создания таблиц в SQL существует оператор *create table*. Создание базы данных имеет следующий синтаксис:

```
create table имя_таблицы (имя_первого_столбца тип, имя_второго_столбца тип, ...,
имя_последнего_столбца тип );
```

Требования к именам таблиц и столбцов такие же, как и для имен БД. К каждому столбцу привязан определенный тип данных, который ограничивает характер информации, которую можно хранить в столбце (например, предотвращает ввод букв в числовое поле). MySQL поддерживает несколько типов данных: числовые, строковые, календарные и специальный тип NULL, обозначающий отсутствие информации. Подробно о типах данных мы будем говорить в следующем уроке, а пока вернемся к нашим таблицам. В них у нас всего два типа данных - целочисленные значения (int) и строки (text). Итак, создадим первую таблицу - Темы:



Нажимаем Enter - таблица создана:

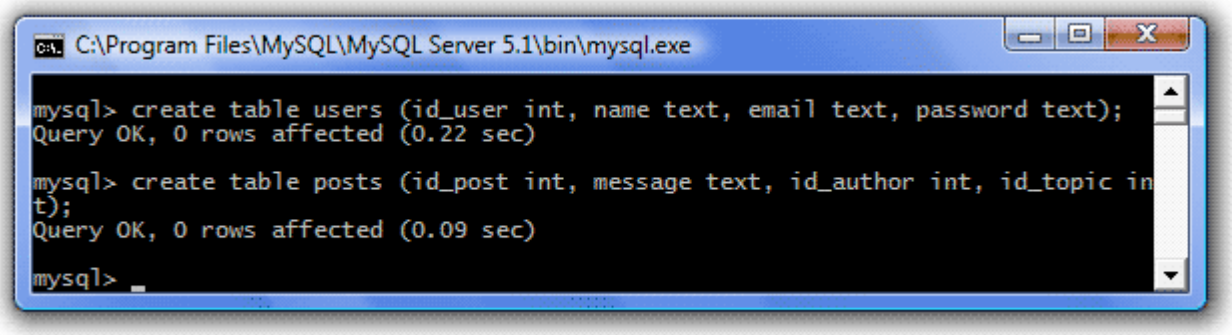


Итак, мы создали таблицу topics (темы) с тремя столбцами:

id_topic int - id темы (целочисленное значение),

topic_name text - имя темы (строка),
id_author int - id автора (целочисленное значение).

Аналогичным образом создадим оставшиеся две таблицы - users (пользователи) и posts (сообщения):



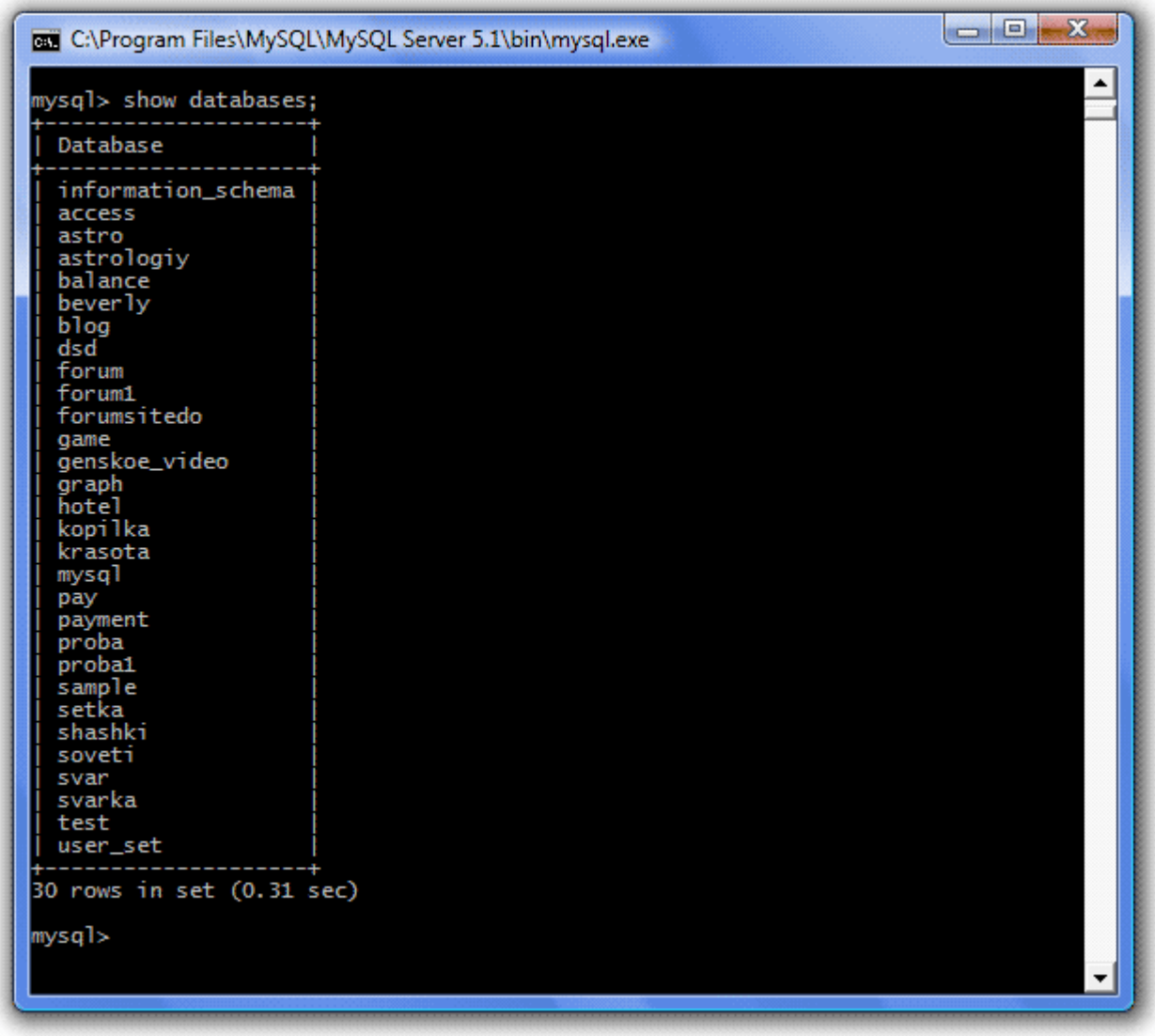
Итак, мы создали БД forum и в ней три таблицы. Сейчас мы об этом помним, но если наша БД будет очень большой, то удержать в голове названия всех таблиц и столбцов просто невозможно. Поэтому надо иметь возможность посмотреть, какие БД у нас существуют, какие таблицы в них присутствуют, и какие столбцы эти таблицы содержат. Для этого в SQL существует несколько операторов:

show databases - показать все имеющиеся БД,

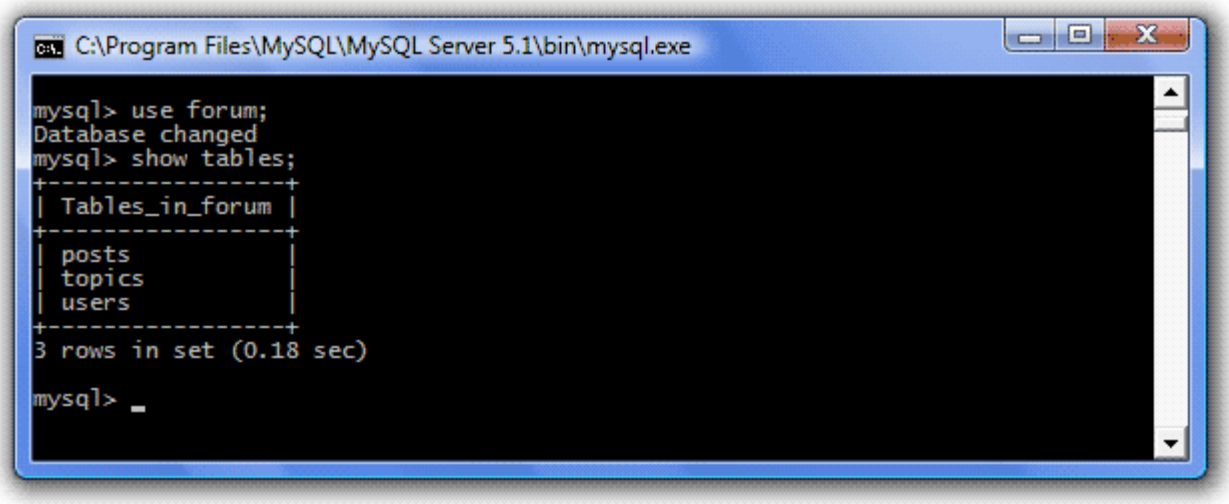
show tables - показать список таблиц текущей БД (предварительно ее надо выбрать с помощью оператора *use*),

describe имя_таблицы - показать описание столбцов указанной таблицы.

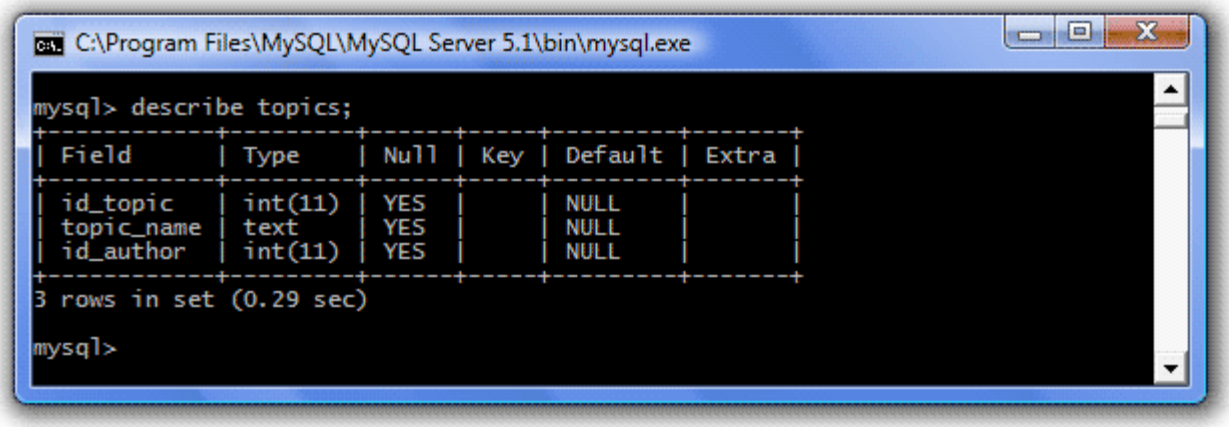
Давайте попробуем. Смотрим все имеющиеся базы данных (у вас она пока одна - forum, у меня 30, и все они перечислены в столбик):



Теперь посмотрим список таблиц БД forum (для этого ее предварительно надо выбрать), не забываем после каждого запроса нажимать Enter:



В ответе видим названия наших трех таблиц. Теперь посмотрим описание столбцов, например, таблицы topics:

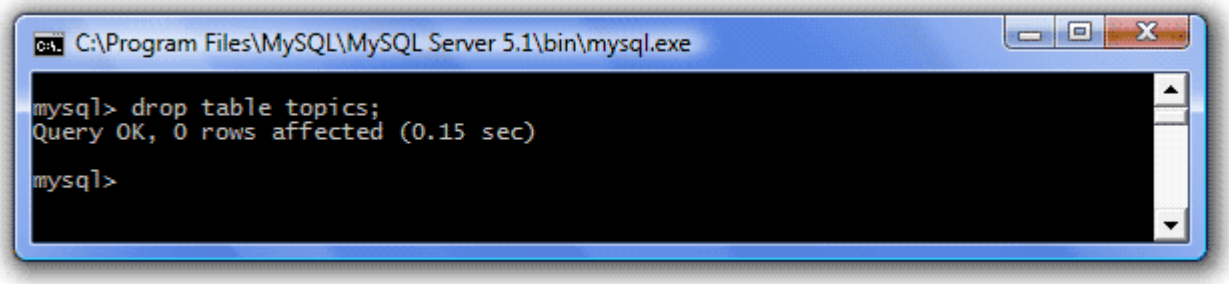


Первые два столбца нам знакомы - это имя и тип данных, значения остальных нам еще предстоит узнать. Но прежде мы все-таки узнаем какие типы данных бывают, какие и когда следует использовать.

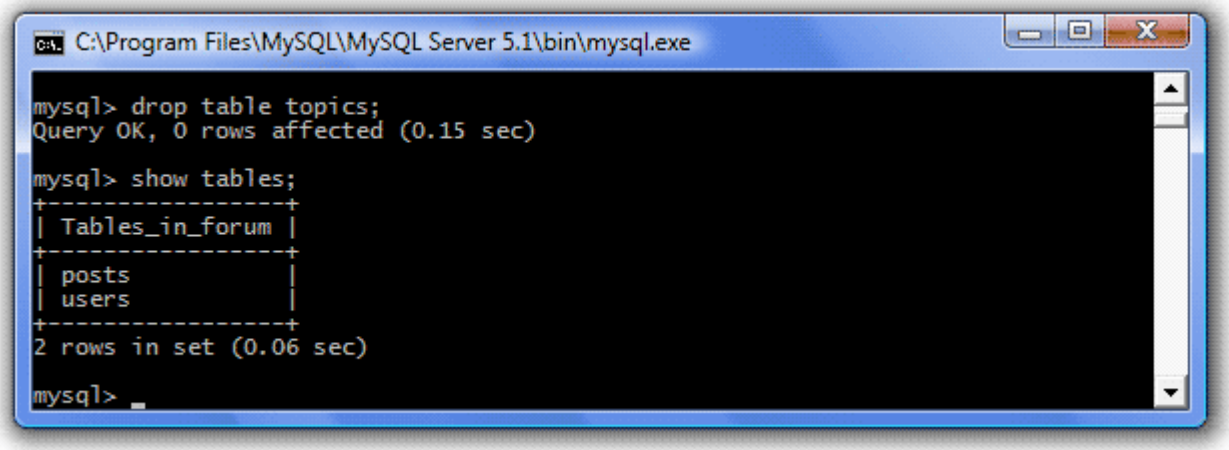
А сегодня мы рассмотрим последний оператор - *drop*, он позволяет удалять таблицы и БД. Например, давайте удалим таблицу topics. Так как мы два шага назад выбирали БД forum для работы, то сейчас ее выбирать не надо, можно просто написать:

```
drop table имя_таблицы;
```

и нажать Enter.



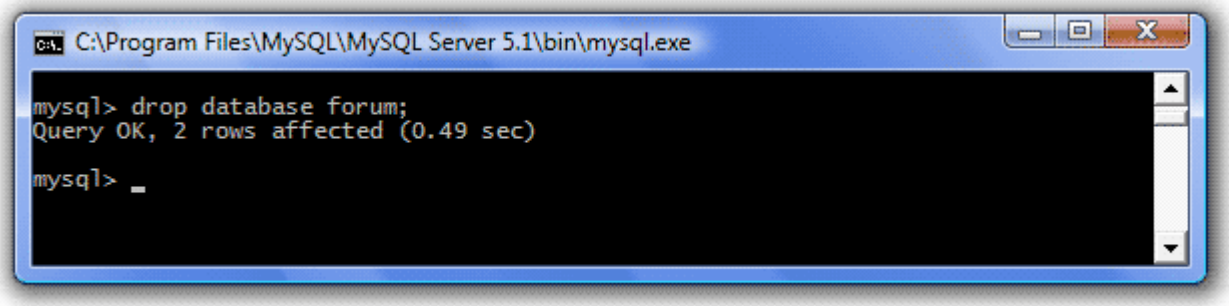
Теперь снова посмотрим список таблиц нашей БД:



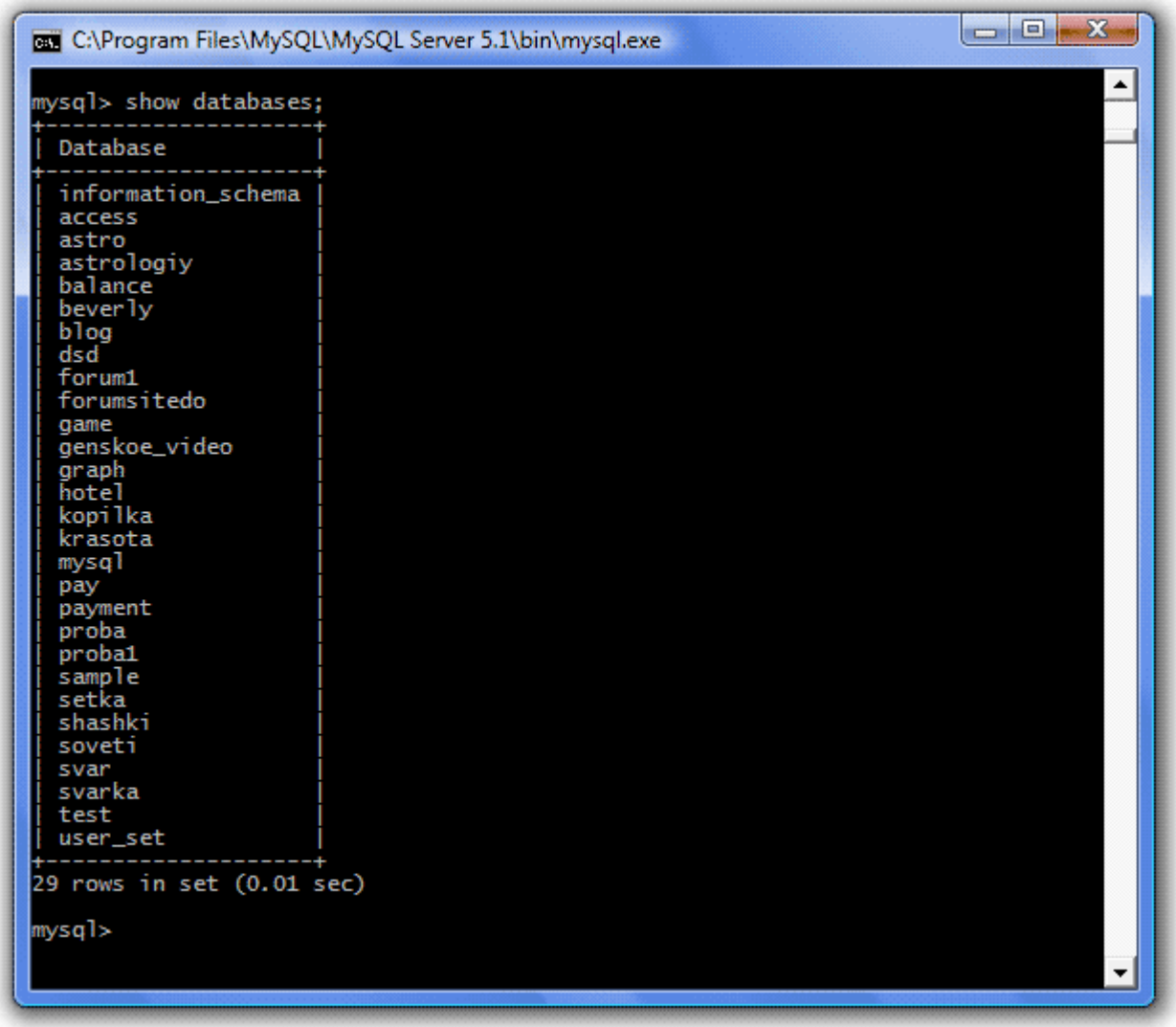
Наша таблица действительно удалена. Теперь давайте удалим и саму БД forum (удаляйте, не жалейте, ее все равно придется переделывать). Для этого напишем:

```
drop database имя_базы данных;
```

и нажмем Enter.



И убедитесь в этом, сделав запрос на все имеющиеся БД:



У вас, наверно, нет ни одной БД, у меня их стало 29 вместо 30.

На сегодня все. Мы научились создавать базы данных и таблицы, удалять их и извлекать информацию об имеющихся базах данных, таблицах и их описаниях.

Лабораторная работа №2

Тема: Типы данных

Этот урок носит больше теоретический характер, но пропустить его нельзя. В дальнейшем вы сможете возвращаться к нему, как к справочному уроку, сейчас же просто ознакомьтесь. В прошлом уроке говорилось, что MySQL поддерживает числовые, строковые, календарные данные и данные типа NULL. Рассмотрим их по очереди.

Числовые типы данных

Тип данных	Объем памяти	Диапазон	Описание
TINYINT (M)	1 байт	от -128 до 127 или от 0 до 255	Целое число. Может быть объявлено положительным с помощью ключевого слова UNSIGNED, тогда элементам столбца нельзя будет присвоить отрицательное значение. Необязательный параметр M - количество отводимых под число символов. Необязательный атрибут ZEROFILL позволяет свободные позиции по умолчанию заполнить нулями. *Примеры:* TINYINT - хранит любое число в диапазоне от -128 до 127.

			TINYINT UNSIGNED - хранит любое число в диапазоне от 0 до 255. TINYINT (2) - предполагается, что значения будут двузначными, но по факту будет хранить и трехзначные. TINYINT (3) ZEROFILL - свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 002.
SMALLINT (M)	2 байта	от -32768 до 32767 или от 0 до 65535	Аналогично предыдущему, но с большим диапазоном. *Примеры:* SMALLINT - хранит любое число в диапазоне от -32768 до 32767. SMALLINT UNSIGNED - хранит любое число в диапазоне от 0 до 65535. SMALLINT (4) - предполагается, что значения будут четырехзначные, но по факту будет хранить и пятизначные. SMALLINT (4) ZEROFILL - свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 0002.
MEDIUMINT (M)	3 байта	от -8388608 до	Аналогично предыдущему, но с большим диапазоном. *Примеры:*

		8388608 или от 0 до 16777215	MEDIUMINT - хранит любое число в диапазоне от -8388608 до 8388608. MEDIUMINT UNSIGNED - хранит любое число в диапазоне от 0 до 16777215. MEDIUMINT (4) - предполагается, что значения будут четырехзначные, но по факту будет хранить и семизначные. MEDIUMINT (5) ZEROFILL - свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 00002.
INT (M) или INTEGER (M)	4 байта	от -2147683648 до 2147683648 или от 0 до 4294967295	Аналогично предыдущему, но с большим диапазоном. *Примеры:* INT - хранит любое число в диапазоне от -2147683648 до 2147683648. INT UNSIGNED - хранит любое число в диапазоне от 0 до 4294967295. INT (4) - предполагается, что значения будут четырехзначные, но по факту будет хранить максимально возможные. INT (5) ZEROFILL - свободные позиции слева заполнит

			нулями. Например, величина 2 будет отображаться, как 00002.
BIGINT (M)	8 байта	от -2^{63} до 2^{63}-1 или от 0 до 2^{64}	Аналогично предыдущему, но с большим диапазоном. *Примеры:* BIGINT - хранит любое число в диапазоне от -2^{63}до 2^{63}-1. BIGINT UNSIGNED - хранит любое число в диапазоне от 0 до 2^{64}. BIGINT (4) - предполагается, что значения будут четырехзначные, но по факту будет хранить максимально возможные. BIGINT (7) ZEROFILL - свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 0000002.
BOOL или BOOLEAN	1 байт	либо 0, либо 1	Булево значение. 0 - ложь (false), 1 - истина (true).
DECIMAL (M,D) или DEC (M,D) или NUMERIC (M,D)	M + 2 байта	зависят от параметров M и D	Используются для величин повышенной точности, например, для денежных данных. M - количество отводимых под число символов (максимальное значение - 64). D - количество знаков после запятой (максимальное значение - 30). *Пример:*

			DECIMAL (5,2) - будет хранить числа от -99,99 до 99,99.
FLOAT (M,D)	4 байта	мин. значение +(-) 1.175494351 * 10^{-39} макс. значение +(-) 3.402823466 * 10^{38}	Вещественное число (с плавающей точкой). Может иметь параметр UNSIGNED, запрещающий отрицательные числа, но диапазон значений от этого не изменится. M - количество отводимых под число символов. D - количество символов дробной части. *Пример:* FLOAT (5,2) - будет хранить числа из 5 символов, 2 из которых будут идти после запятой (например: 46,58).
DOUBLE (M,D)	8 байт	мин. значение +(-) 2.2250738585072015 * 10^{-308} макс. значение +(-) 1.7976931348623157 * 10^{308}	Аналогично предыдущему, но с большим диапазоном. *Пример:* DOUBLE - будет хранить большие дробные числа.

Необходимо понимать, чем больше диапазон значений у типа данных, тем больше памяти он занимает. Поэтому, если предполагается, что значения в столбце не будут превышать 100, то используйте тип TINYINT. Если при этом все значения будут положительными, то используйте атрибут UNSIGNED. Правильный выбор типа данных позволяет сэкономить место для хранения этих данных.

Строковые типы данных

Тип данных	Объем памяти	Максимальный размер	Описание
CHAR (M)	M символов	M символов	Позволяет хранить строку фиксированной длины M. Значение M - от 0 до 65535. *Примеры:* CHAR (8) - хранит строки из 8 символов и занимает 8 байтов. Например, любое из следующих значений: '', 'Иван','Ирина', 'Сергей' будет занимать по 8 байтов памяти. А при попытке ввести значение 'Александра', оно будет усечено до 'Александ', т.е. до 8 символов.
VARCHAR (M)	L+1 символов	M символов	Позволяет хранить переменные строки длиной L. Значение M - от 0 до 65535. *Примеры:*

			VARCHAR (3) - хранит строки максимум из 3 символов, но пустая строка '' занимает 1 байт памяти, строка 'a' - 2 байта, строк 'aa' - 3 байта, строка 'aaa' - 4 байта. Значение более 3 символов будет усечено до 3.
BLOB, TEXT	L+2 символов	2^{16}-1 символов	Позволяют хранить большие объемы текста. Причем тип TEXT используется для хранения именно текста, а BLOB - для хранения изображений, звука, электронных документов и т.д.
MEDIUMBLOB, MEDIUMTEXT	L+3 символов	2^{24}-1 символов	Аналогично предыдущему, но с большим размером.
LONGBLOB, LONGTEXT	L+4 символов	2^{32}-1 символов	Аналогично предыдущему, но с большим размером.
ENUM ('value1', 'value2', ...,'valueN')	1 или 2 байта	65535 элементов	Строки этого типа могут принимать только одно из значений указанного множества. *Пример:* ENUM ('да', 'нет') - в столбце с таким типом может храниться только одно из имеющихся значений. Удобно использовать, если предусмотрено, что в столбце должен храниться ответ на вопрос.

SET ('value1', 'value2', ...,'valueN')	до 8 байт	64 элемента	Строки этого типа могут принимать любой или все элементы из значений указанного множества. *Пример:* SET ('первый', 'второй') - в столбце с таким типом может храниться одно из перечисленных значений, оба сразу или значение может отсутствовать вовсе.

Календарные типы данных

Тип данных	Объем памяти	Диапазон	Описание
DATE	3 байта	от '1000-01-01' до '9999-12-31'	Предназначен для хранения даты. В качестве первого значения указывается год в формате "YYYY", через дефис - месяц в формате "MM", а затем день в формате "DD". В качестве разделителя может выступать не только дефис, а любой символ отличный от цифры.
TIME	3 байта	от '-838:59:59' до '838:59:59'	Предназначен для хранения времени суток. Значение вводится и хранится в привычном формате - hh:mm:ss, где hh - часы, mm - минуты, ss - секунды. В качестве разделителя может выступать любой символ отличный от цифры.

DATATIME	8 байт	от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'	Предназначен для хранения и даты и времени суток. Значение вводится и хранится в формате - YYYY-MM-DD hh:mm:ss. В качестве разделителей могут выступать любые символы отличные от цифры.
TIMESTAMP	4 байта	от '1970-01-01 00:00:00' до '2037-12-31 23:59:59'	Предназначен для хранения даты и времени суток в виде количества секунд, прошедших с полуночи 1 января 1970 года (начало эпохи UNIX).
YEAR (M)	1 байт	от 1970 до 2069 для M=2 и от 1901 до 2155 для M=4	Предназначен для хранения года. M - задает формат года. Например, YEAR (2) - 70, а YEAR (4) - 1970. Если параметр M не указан, то по умолчанию считается, что он равен 4.

Тип данных NULL

Вообще-то это лишь условно можно назвать типом данных. По сути это скорее указатель возможности отсутствия значения. Например, когда вы регистрируетесь на каком-либо сайте, вам предлагается заполнить форму, в которой присутствуют, как обязательные, так и необязательные поля. Понятно, что регистрация пользователя невозможна без указания логина и пароля, а вот дату рождения и пол пользователь может указать по желанию. Для того, чтобы хранить такую информацию в БД и используют два значения:

NOT NULL (значение не может отсутствовать) для полей логин и пароль,

NULL (значение может отсутствовать) для полей дата рождения и пол.

По умолчанию всем столбцам присваивается тип NOT NULL, поэтому его можно явно не указывать.

Пример:

create table users (login varchar(20), password varchar(15), sex enum('man', 'woman') NULL, date_birth date NULL);

Таким образом, мы создаем таблицу с 4 столбцами: логин (не более 20 символов) обязательное, пароль (не более 15 символов) обязательное, пол (мужской или женский) не обязательное, дата рождения (тип дата) необязательное.

Все, на этом урок, посвященный типам данных, закончен. У вас, возможно, остались вопросы, но они исчезнут по мере освоения дальнейшего материала, т.к. на практике все становится более понятно, чем в теории.

Лабораторная работа №3

Тема: Создание таблиц и наполнение их информацией

Итак, мы познакомились с типами данных, теперь будем усовершенствовать таблицы для нашего форума. Сначала разберем их. И начнем с таблицы users (пользователи). В ней у нас 4 столбца:

id_user - целочисленные значения, значит будет тип int, ограничим его 10 символами - int (10).
name - строковое значение varchar, ограничим его 20 символами - varchar(20).
email - строковое значение varchar, ограничим его 50 символами - varchar(50).
password - строковое значение varchar, ограничим его 15 символами - varchar(15).

Все значения полей обязательны для заполнения, значит надо добавить тип NOT NULL.

id_user int (10) NOT NULL
name varchar(20) NOT NULL
email varchar(50) NOT NULL
password varchar(15) NOT NULL

Первый столбец, как вы помните из концептуальной модели нашей БД, является первичным ключом (т.е. его значения уникальны, и они однозначно идентифицируют запись). Следить за уникальностью самостоятельно можно, но не рационально. Для этого в SQL есть специальный атрибут - *AUTO_INCREMENT*, который при обращении к таблице на добавление данных высчитывает максимальное значение этого столбца, полученное значение увеличивает на 1 и заносит его в столбец. Таким образом, в этом столбце автоматически генерируется уникальный номер, а следовательно тип NOT NULL излишен. Итак, присвоим атрибут столбцу с первичным ключом:

id_user int (10) AUTO_INCREMENT
name varchar(20) NOT NULL
email varchar(50) NOT NULL
password varchar(15) NOT NULL

Теперь надо указать, что поле id_user является первичным ключом. Для этого в SQL используется ключевое слово*PRIMARY KEY ()*, в скобочках указывается имя ключевого поля. Внесем изменения:

id_user int (10) AUTO_INCREMENT
name varchar(20) NOT NULL
email varchar(50) NOT NULL
password varchar(15) NOT NULL
PRIMARY KEY (id_user)

Итак, таблица готова, и ее окончательный вариант выглядит так:

```
create table users (
id_user int (10) AUTO_INCREMENT,
name varchar(20) NOT NULL,
email varchar(50) NOT NULL,
password varchar(15) NOT NULL,
PRIMARY KEY (id_user)
);
```

Теперь разберемся со второй таблицей - topics (темы). Рассуждая аналогично, имеем следующие поля:

```
id_topic int (10) AUTO_INCREMENT
topic_name varchar(100) NOT NULL
id_author int (10) NOT NULL
PRIMARY KEY (id_topic)
```

Но в модели нашей БД поле id_author является внешним ключом, т.е. оно может иметь только те значения, которые есть в поле id_user таблицы users. Для того, чтобы указать это в SQL есть ключевое слово *FOREIGN KEY ()*, которое имеет следующий синтаксис:

```
FOREIGN KEY (имя_столбца_которое_является_внешним_ключом) REFERENCES имя_таблицы_родителя (имя_столбца_родителя);
```

Укажем, что id_author - внешний ключ:

```
id_topic int (10) AUTO_INCREMENT
topic_name varchar(100) NOT NULL
```

```
id_author int (10) NOT NULL
PRIMARY KEY (id_topic)
FOREIGN KEY (id_author) REFERENCES users (id_user)
```

Таблица готова, и ее окончательный вариант выглядит так:

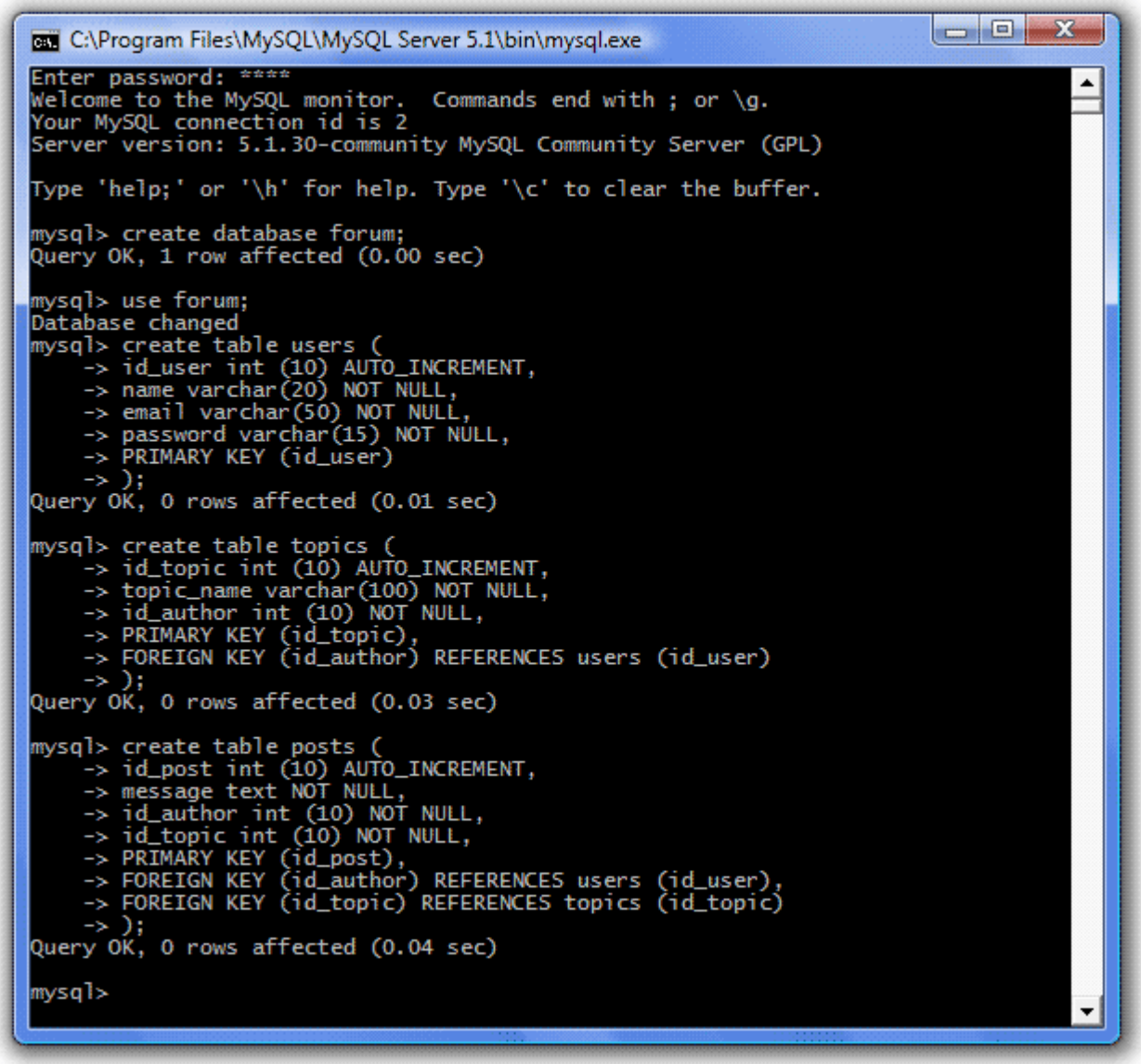
```
create table topics (
id_topic int (10) AUTO_INCREMENT,
topic_name varchar(100) NOT NULL,
id_author int (10) NOT NULL,
PRIMARY KEY (id_topic),
FOREIGN KEY (id_author) REFERENCES users (id_user)
);
```

Осталась последняя таблица - posts (сообщения). Здесь все аналогично, только два внешних ключа:

```
create table posts (
id_post int (10) AUTO_INCREMENT,
message text NOT NULL,
id_author int (10) NOT NULL,
id_topic int (10) NOT NULL,
PRIMARY KEY (id_post),
FOREIGN KEY (id_author) REFERENCES users (id_user),
FOREIGN KEY (id_topic) REFERENCES topics (id_topic)
);
```

Обратите внимание, внешних ключей у таблицы может быть несколько, а первичный ключ в MySQL может быть только один. В первом уроке мы удалили нашу БД forum, пришло время создать ее вновь.

Запускаем сервер MySQL (Пуск - Программы - MySQL - MySQL Server 5.1 - MySQL Command Line Client), вводим пароль, создаем БД forum (create database forum;), выбираем ее для использования (use forum;) и создаем три наших таблицы:

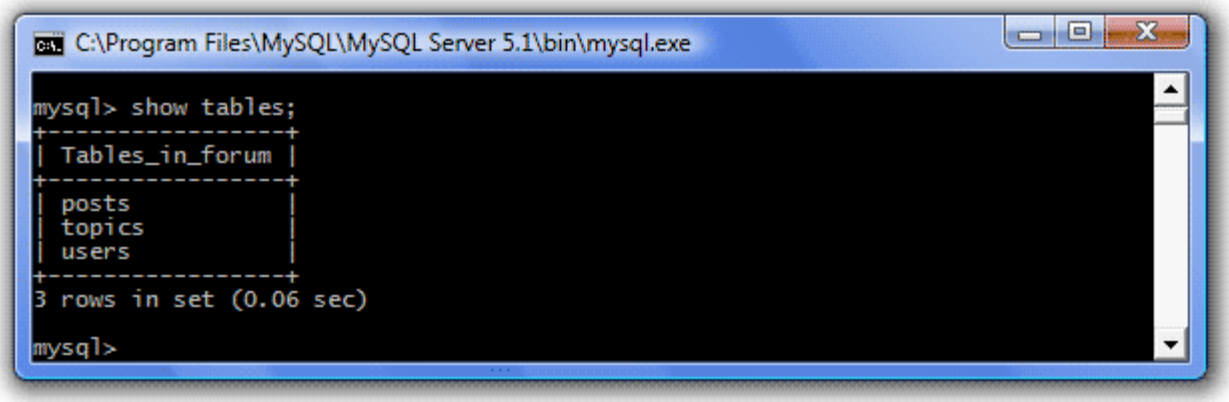


Обратите внимание, одну команду можно писать в несколько строк, используя клавишу Enter (MySQL автоматически подставляет символ новой строки ->), и только после разделителя (точки с запятой) нажатие клавиши Enter приводит к выполнению запроса.

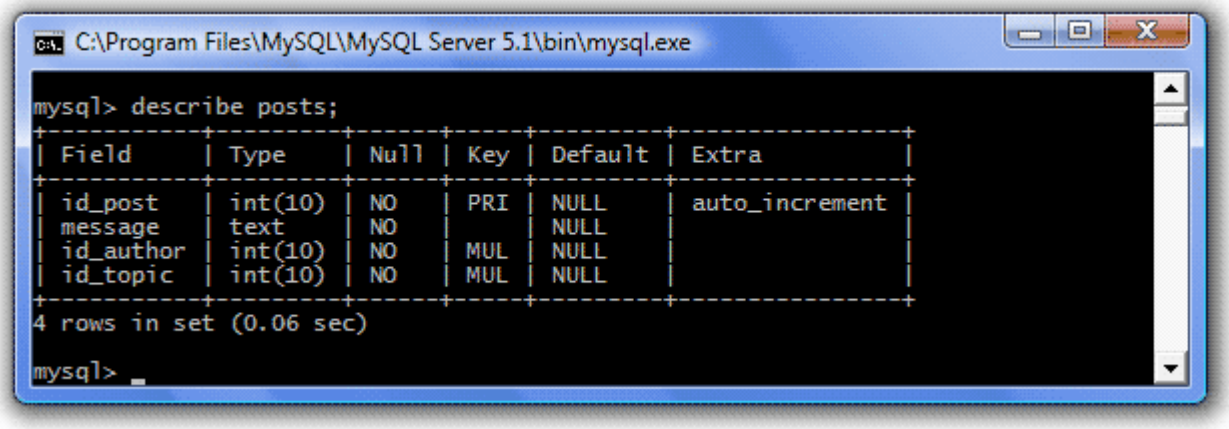
Помните, если вы сделали что-то не так, всегда можно удалить таблицу или всю БД с помощью оператора DROP. Исправлять что-то в командной строке крайне неудобно,

поэтому иногда (особенно на начальном этапе) проще писать запросы в каком-нибудь редакторе, например в Блокноте, а затем копировать и вставлять их в черное окошко.

Итак, таблицы созданы, чтобы убедиться в этом вспомним о команде *show tables*:



И, наконец, посмотрим структуру нашей последней таблицы posts:



Теперь становятся понятны значения всех полей структуры, кроме поля DEFAULT. Это поле значений по умолчанию. Мы могли бы для какого-нибудь столбца (или для всех) указать значение по умолчанию. Например, если бы у нас было поле с названием "Женаты\Замужем" и типом ENUM ('да', 'нет'), то было бы разумно сделать одно из значений значением по умолчанию. Синтаксис был бы следующий:

married enum ('да', 'нет') NOT NULL default('да')

Т.е. это ключевое слово пишется через пробел после указания типа данных, а в скобках указывается значение по умолчанию.

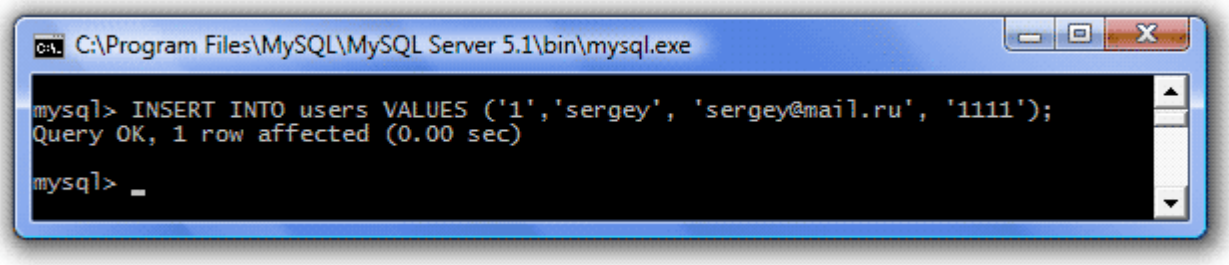
Но вернемся к нашим таблицам. Теперь нам необходимо внести данные в наши таблицы. На сайтах, вы обычно вводите информацию в какие-нибудь html-формы, затем сценарий на каком-либо языке (php, java...) извлекает эти данные из формы и заносит их в БД. Делает он это посредством SQL-запроса на внесение данных в базу. Писать сценарии на php мы пока не умеем, а вот отправлять SQL-запросы на внесение данных сейчас научимся.

Для этого используется оператор *INSERT*. Синтаксис можно использовать двух видов. Первый вариант используется для внесения данных во все поля таблицы:

```
INSERT INTO имя_таблицы VALUES
('значение_первого_столбца','значение_второго_столбца', ...,
'значение_последнего_столбца');
```

Давайте попробуем внести в нашу таблицу users следующие значения:

INSERT INTO users VALUES ('1','sergey', 'sergey@mail.ru', '1111');

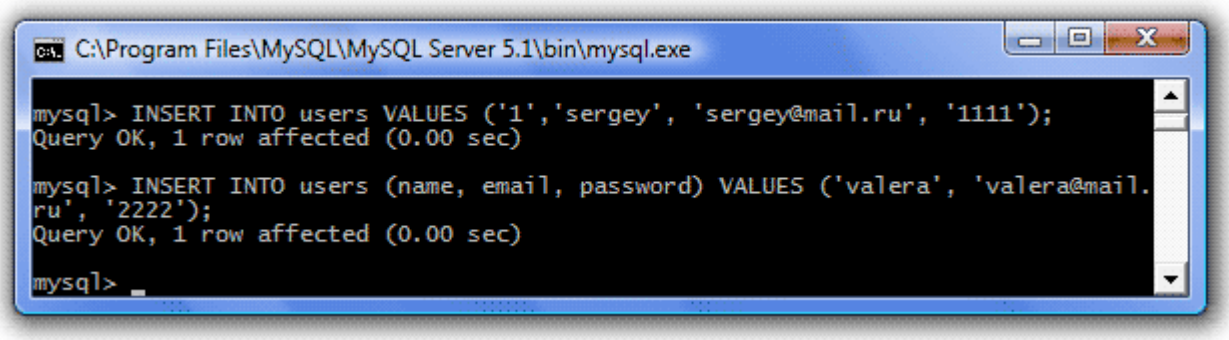


Второй вариант используется для внесения данных в некоторые поля таблицы:

```
INSERT INTO имя_таблицы ('имя_столбца', 'имя_столбца') VALUES
('значение_первого_столбца','значение_второго_столбца');
```

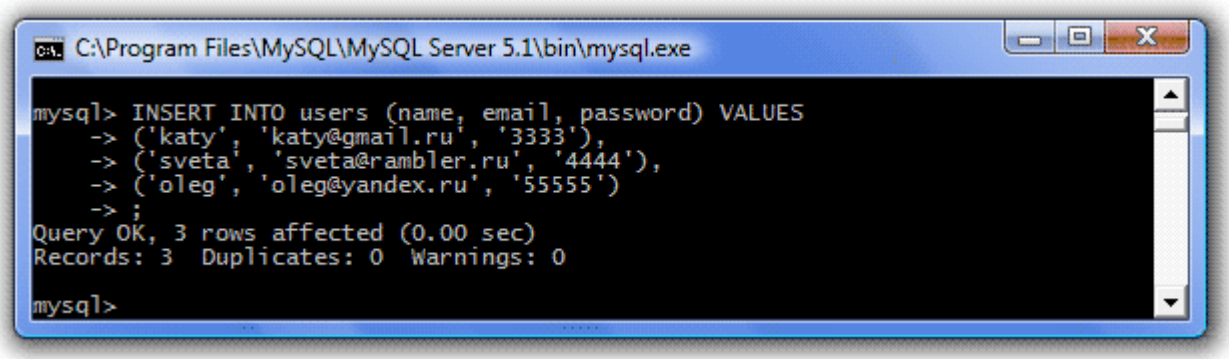
В нашей таблице users все поля обязательны для заполнения, но наше первое поле имеет ключевое слово - AUTO_INCREMENT (т.е. оно заполняется автоматически), поэтому мы можем пропустить этот столбец:

INSERT INTO users (name, email, password) VALUES ('valera', 'valera@mail.ru', '2222');

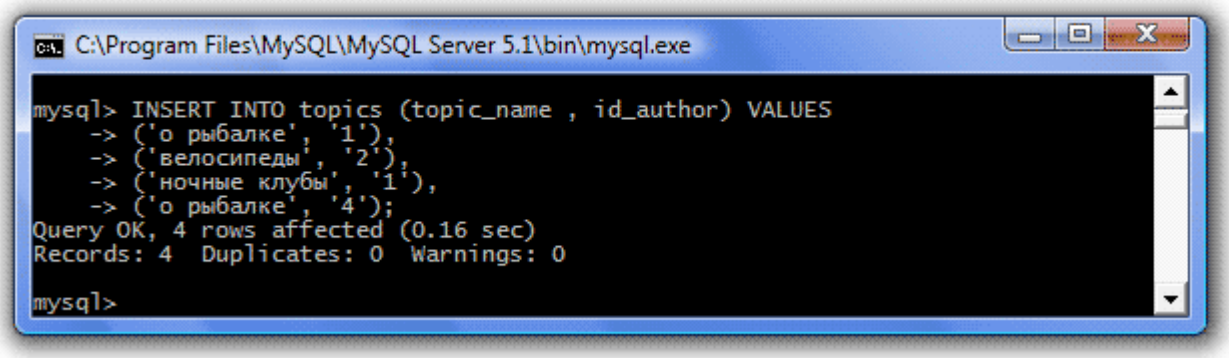


Если бы у нас были поля с типом NULL, т.е. необязательные для заполнения, мы бы тоже могли их проигнорировать. А вот если попытаться оставить пустым поле со значением NOT NULL, то сервер выдаст сообщение об ошибке и не выполнит запрос. Кроме того, при внесении данных сервер проверяет связи между таблицами. Поэтому вам не удастся внести в поле, являющееся внешним ключом, значение, отсутствующее в связанной таблице. В этом вы убедитесь, внося данные в оставшиеся две таблицы.

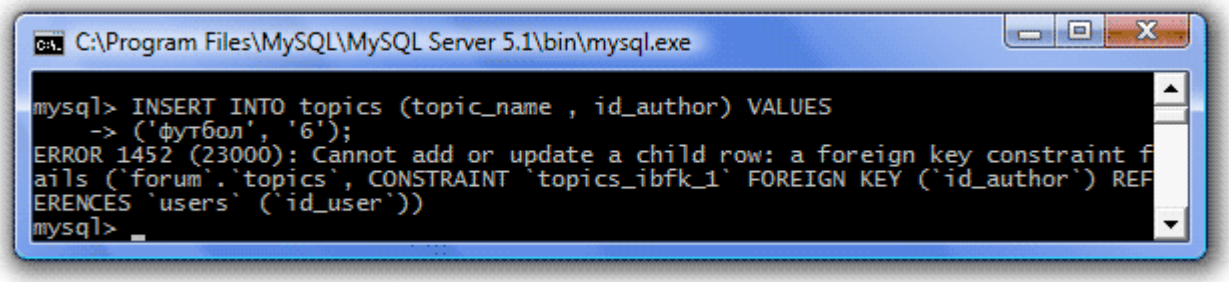
Но прежде внесем информацию еще о нескольких пользователях. Чтобы добавить сразу несколько строк, надо просто перечислять скобки со значениями через запятую:



Теперь внесем данные во вторую таблицу - topics (темы). Все тоже самое, но надо помнить, что значения в поле id_author должны присутствовать в таблице users (пользователи):

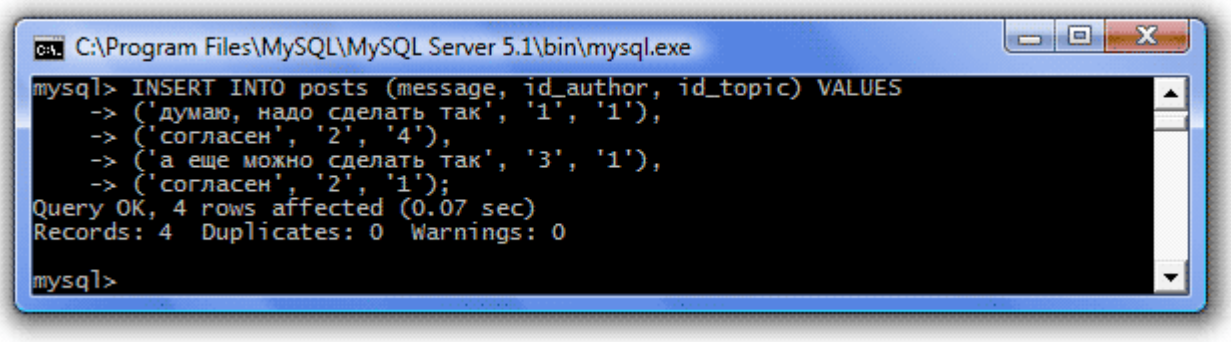


Теперь давайте попробуем внести еще одну тему, но с id_author, которого в таблице users нет (т.к. мы внесли в таблицу users только 5 пользователей, то id=6 не существует):



Сервер выдает ошибку и говорит, что не может внести такую строку, т.к. в поле, являющемся внешним ключом, стоит значение, отсутствующее в связанной таблице users.

Теперь внесем несколько строк в таблицу posts (сообщения), помня, что в ней у нас 2 внешних ключа, т.е. id_author и id_topic, которые мы будем вносить должны присутствовать в связанных с ними таблицах:



Итак, у нас есть 3 таблицы, в которых есть данные. Встает вопрос - как посмотреть, какие данные хранятся в таблицах. Этим мы и займемся на следующем уроке.

Лабораторная работа №4

Тема: Выборка данных - оператор SELECT

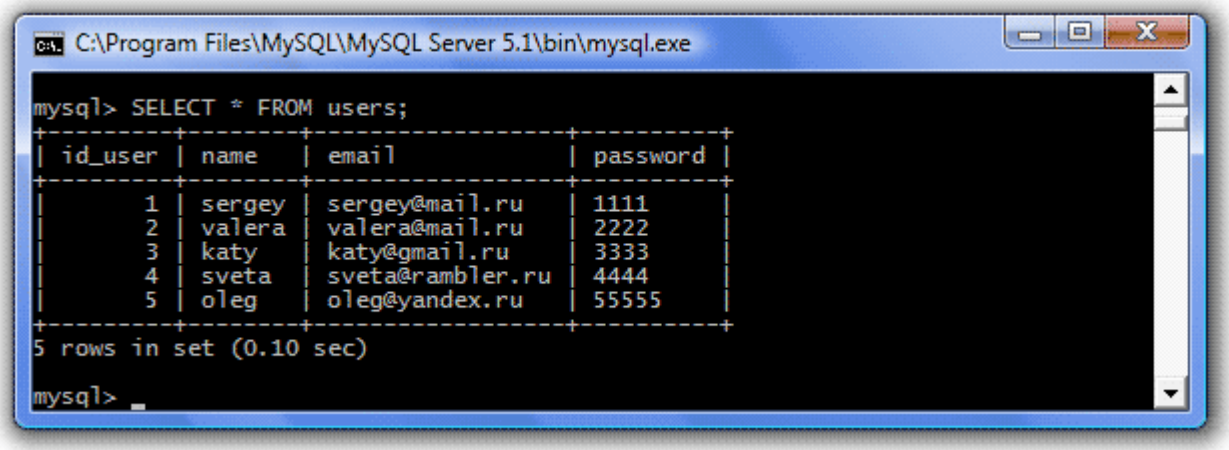
Итак, в нашей БД forum есть три таблицы: users (пользователи), topics (темы) и posts (сообщения). И мы хотим посмотреть, какие данные в них содержатся. Для этого в SQL существует оператор *SELECT*. Синтаксис его использования следующий:

```
SELECT что_выбрать FROM откуда_выбрать;
```

Вместо "что_выбрать" мы должны указать либо имя столбца, значения которого хотим увидеть, либо имена нескольких столбцов через запятую, либо символ звездочки (*), означающий выбор всех столбцов таблицы. Вместо "откуда_выбрать" следует указать имя таблицы.

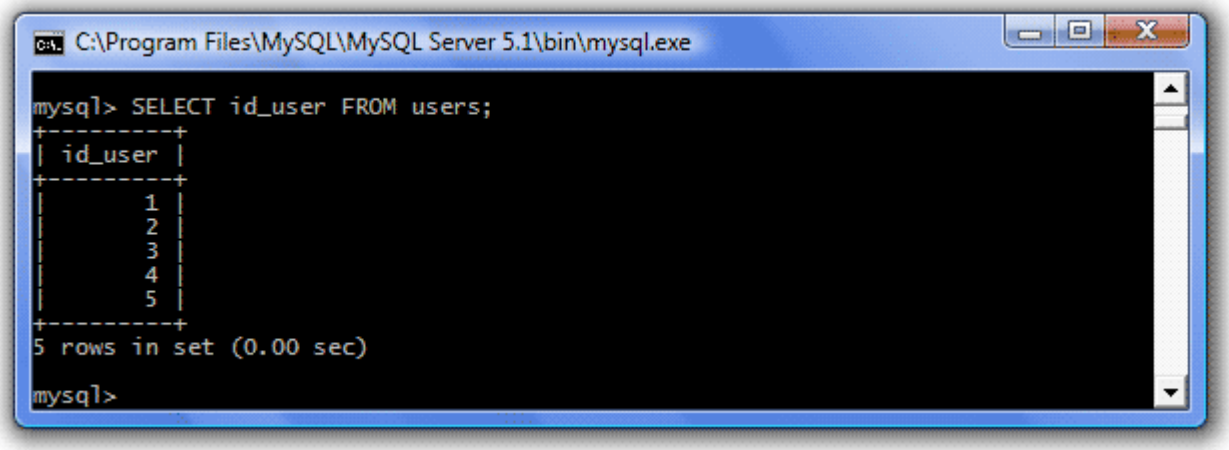
Давайте сначала посмотрим все столбцы из таблицы users:

```
SELECT * FROM users;
```



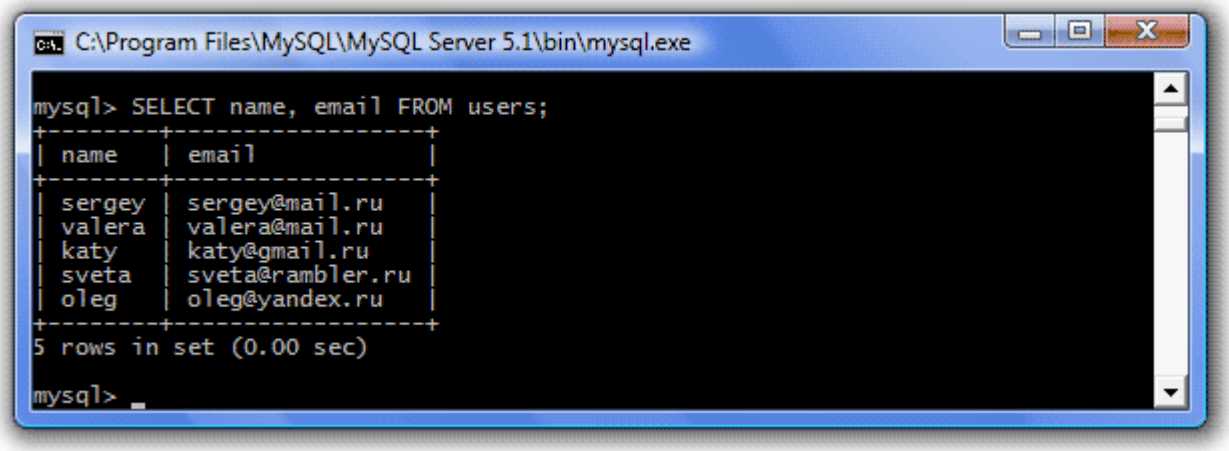
Вот и все наши данные, которые мы вносили в эту таблицу. Но предположим, что мы хотим посмотреть только столбец id_user (например, в прошлом уроке, нам надо было для заполнения таблицы topics (темы) знать, какие id_user есть в таблице users). Для этого в запросе мы укажем имя этого столбца:

SELECT id_user FROM users;



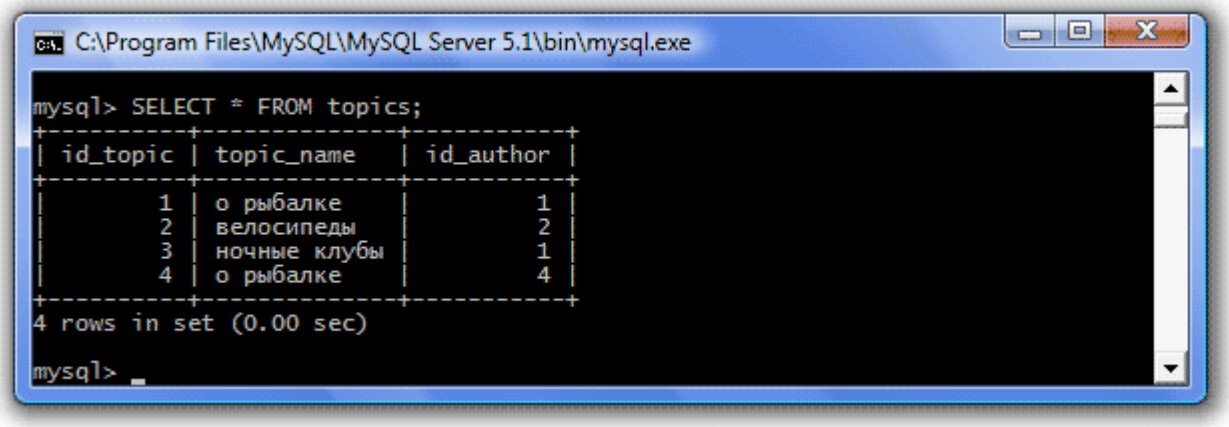
Ну, а если мы захотим посмотреть, например, имена и e-mail наших пользователей, то мы перечислим интересующие столбцы через запятую:

SELECT name, email FROM users;



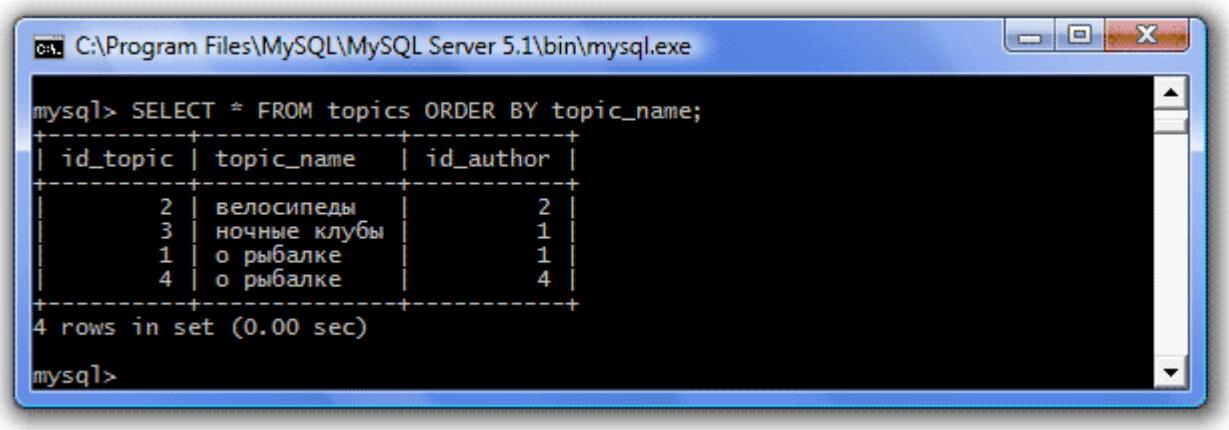
Аналогично, вы можете посмотреть, какие данные содержат и другие наши таблицы. Давайте посмотрим, какие у нас существуют темы:

SELECT * FROM topics;

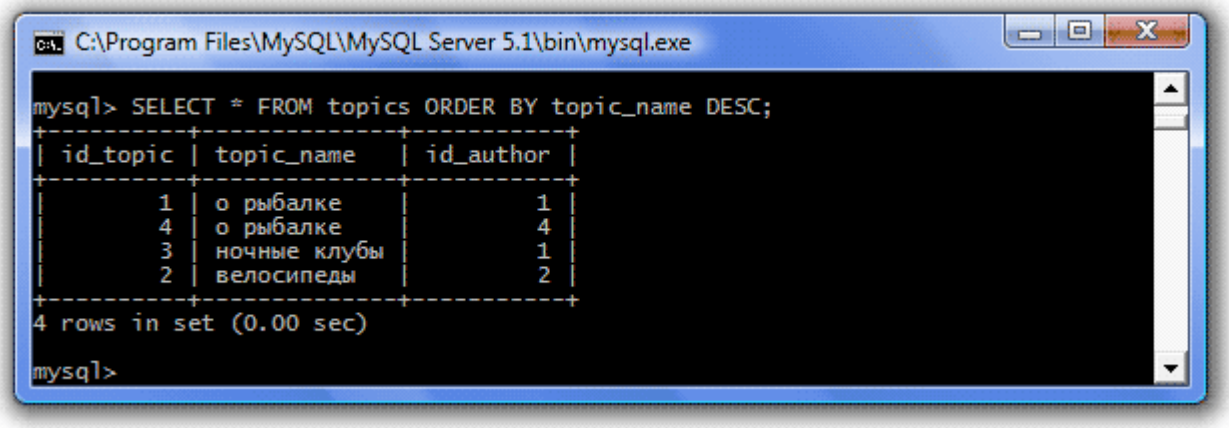


Сейчас у нас всего 4 темы, а если их будет 100? Хотелось бы, чтобы они выводились, например, по алфавиту. Для этого в SQL существует ключевое слово *ORDER BY* после которого указывается имя столбца по которому будет происходить сортировка. Синтаксис следующий:

```
SELECT имя_столбца FROM имя_таблицы ORDER BY имя_столбца_сортировки;
```

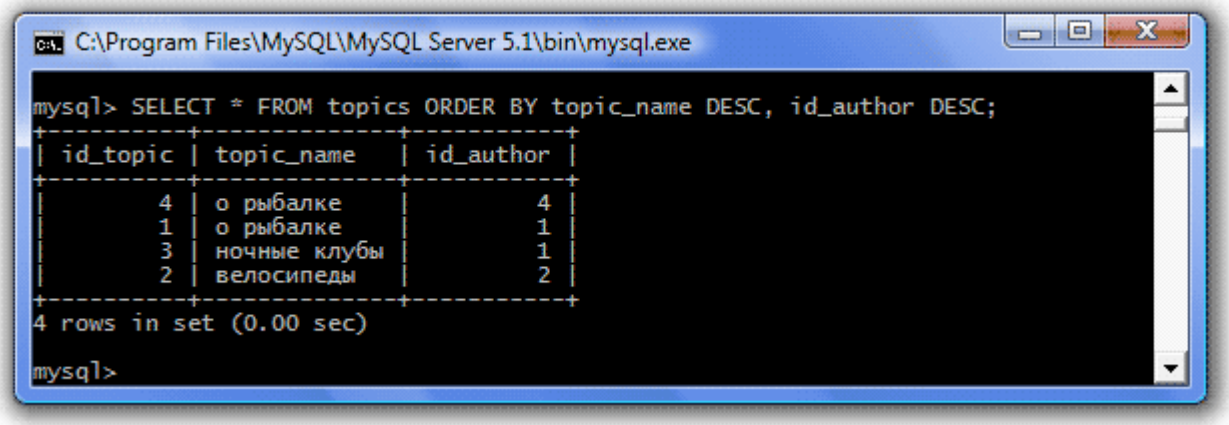


По умолчанию сортировка идет по возрастанию, но это можно изменить, добавив ключевое слово *DESC*



Теперь наши данные отсортированы в порядке по убыванию.

Сортировку можно производить сразу по нескольким столбцам. Например, следующий запрос отсортирует данные по столбцу topic_name, и если в этом столбце будет несколько одинаковых строк, то в столбце id_author будет осуществлена сортировка по убыванию:



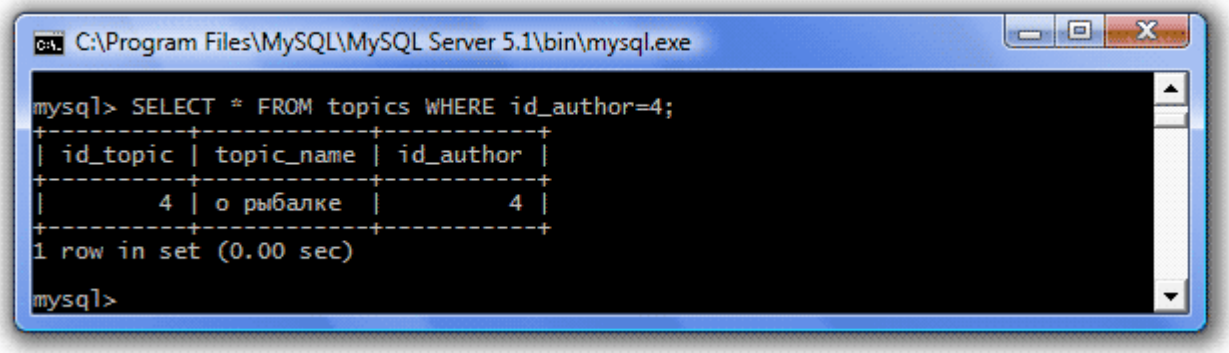
Сравните результат с результатом предыдущего запроса.

Очень часто нам не нужна вся информация из таблицы. Например, мы хотим узнать, какие темы были созданы пользователем sveta (id=4). Для этого в SQL есть ключевое слово *WHERE*, синтаксис у такого запроса следующий:

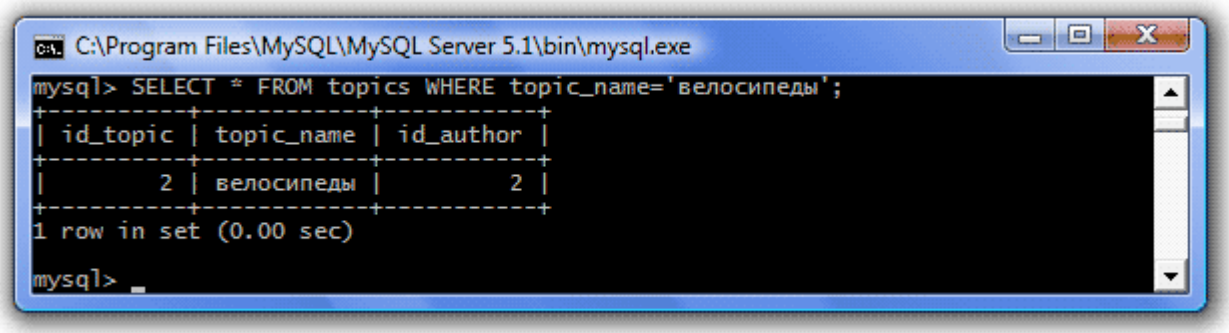
```
SELECT имя_столбца FROM имя_таблицы WHERE условие;
```

Для нашего примера условием является идентификатор пользователя, т.е. нам нужны только те строки, в столбце id_author которых стоит 4 (идентификатор пользователя sveta):

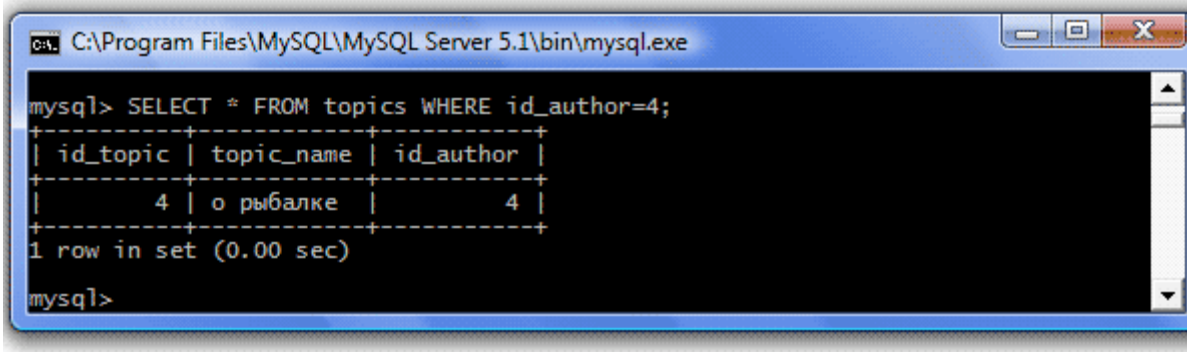
```
SELECT * FROM topics WHERE id_author=4;
```

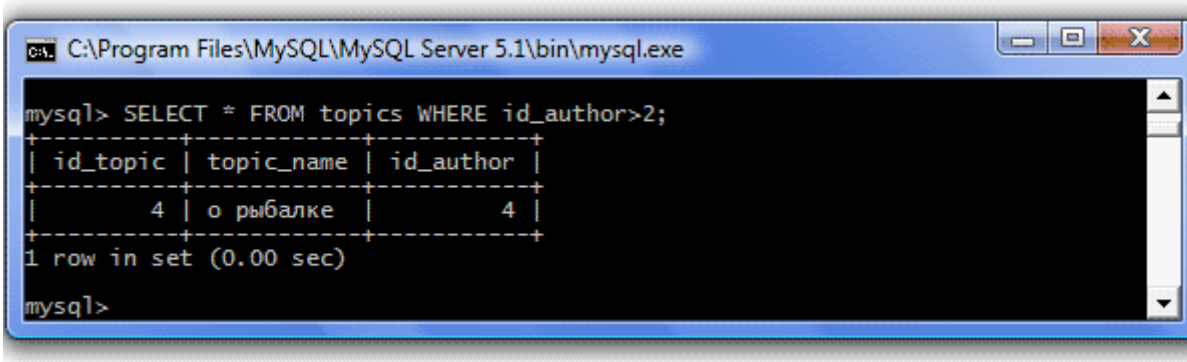
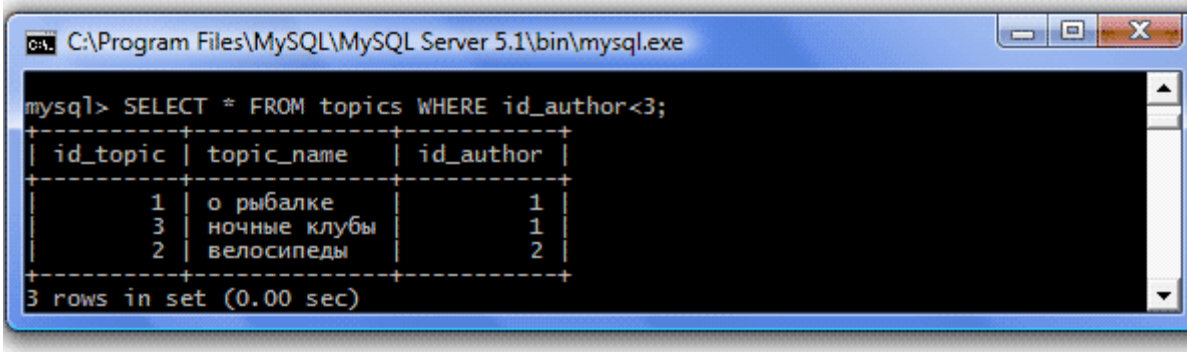


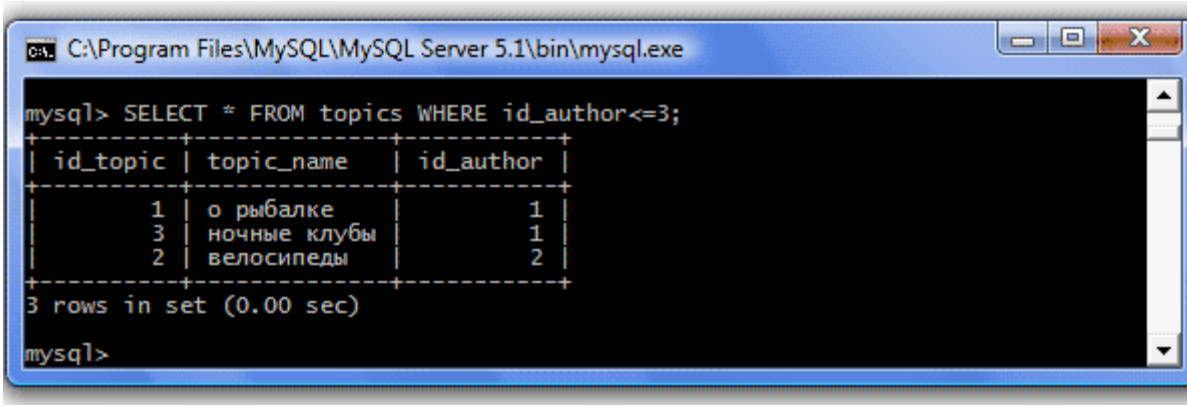
Или мы хотим узнать, кто создал тему "велосипеды":

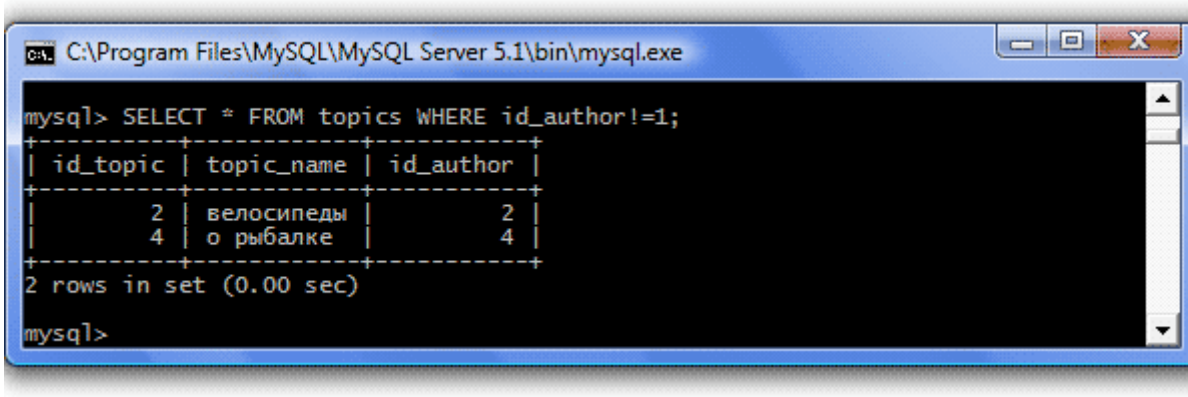
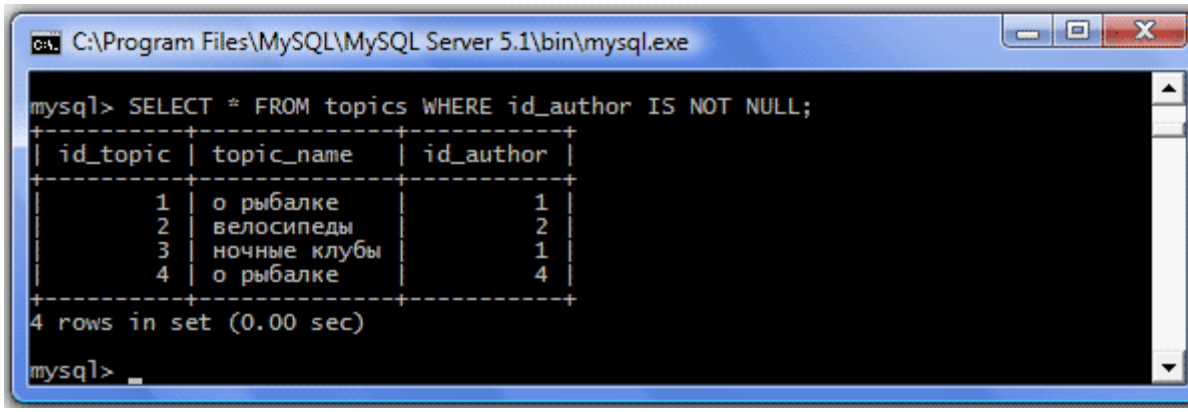


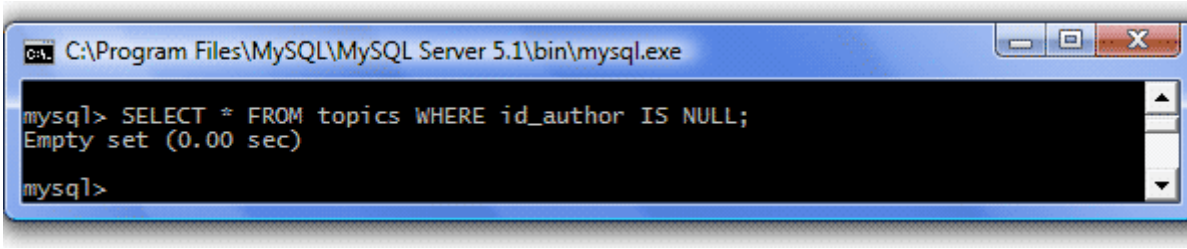
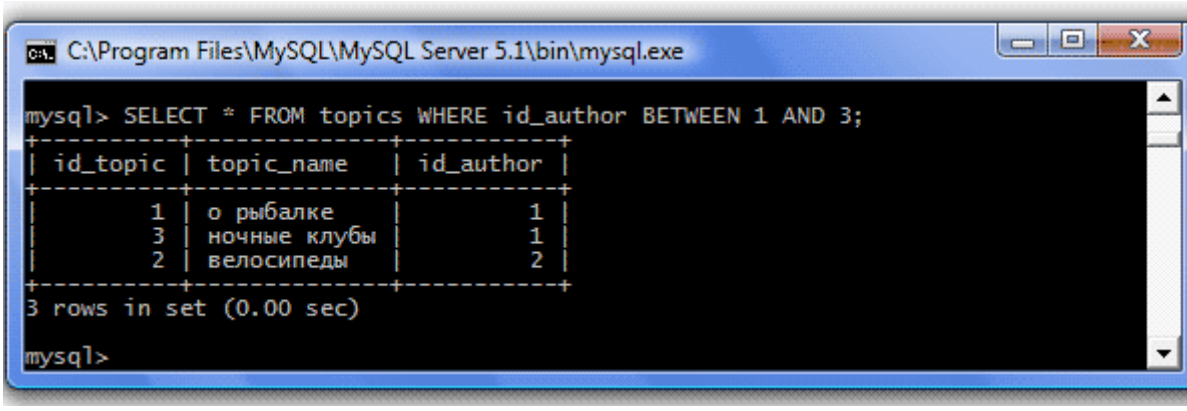
Конечно, было бы удобнее, чтобы вместо id автора, выводилось его имя, но имена хранятся в другой таблице. В последующих уроках мы узнаем, как выбирать данные из нескольких таблиц. А пока узнаем, какие условия можно задавать, используя ключевое слово WHERE.

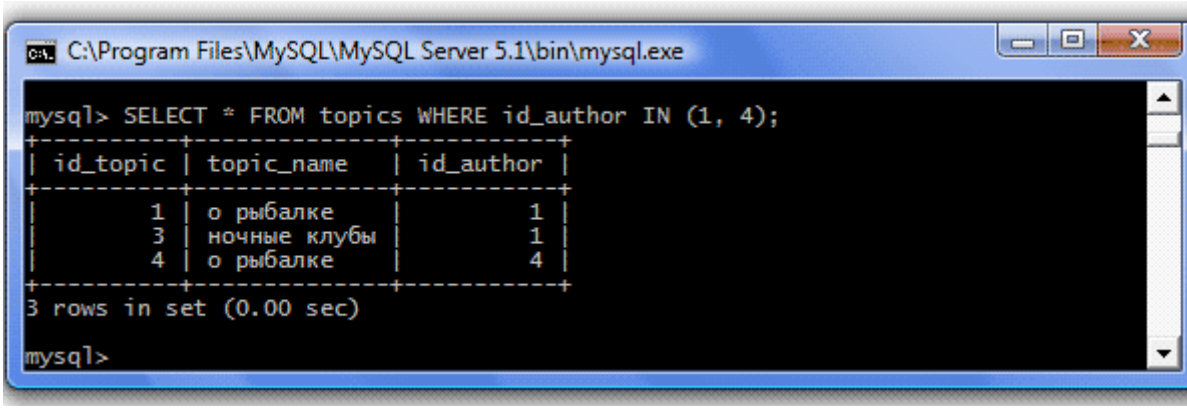
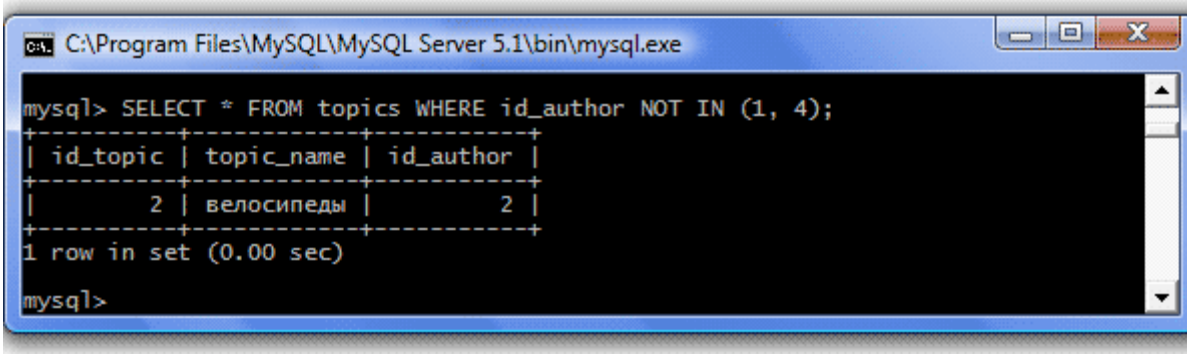
Оператор	Описание
= (равно)	Отбираются значения равные указанному *Пример:* SELECT * FROM topics WHERE id_author=4; *Результат:* 

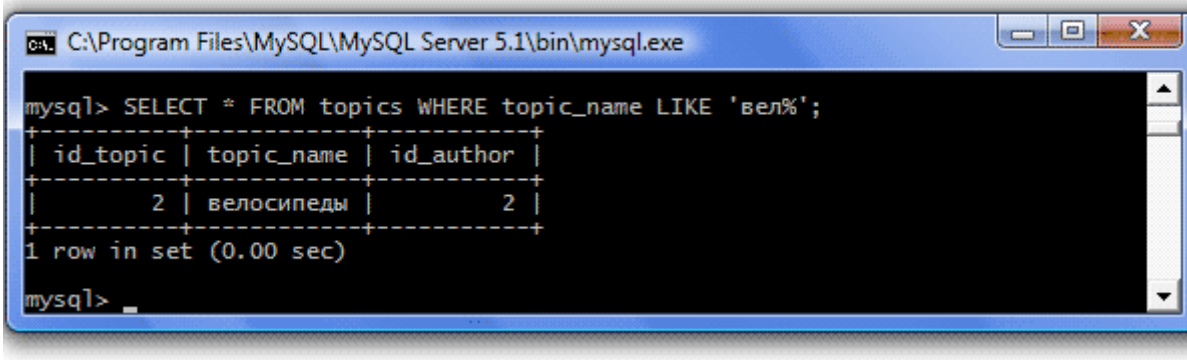
> (больше)	Отбираются значения больше указанного *Пример:* SELECT * FROM topics WHERE id_author>2; *Результат:* 
< (меньше)	Отбираются значения меньше указанного *Пример:* SELECT * FROM topics WHERE id_author<3; *Результат:* 

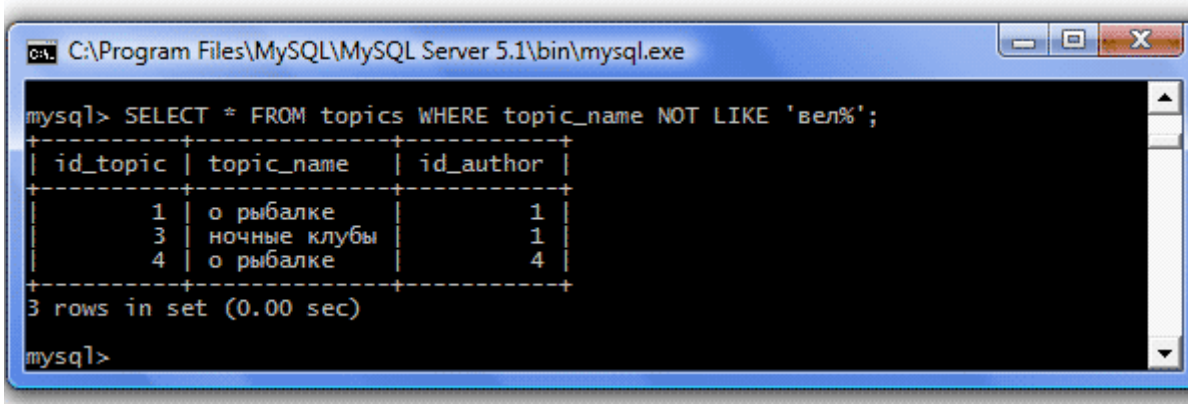
>= (больше или равно)	Отбираются значения большие и равные указанному *Пример:* SELECT * FROM topics WHERE id_author>=2; *Результат:* 
<= (меньше или равно)	Отбираются значения меньшие и равные указанному *Пример:* SELECT * FROM topics WHERE id_author<=3; *Результат:* 

!= (не равно)	Отбираются значения не равные указанному *Пример:* SELECT * FROM topics WHERE id_author!=1; *Результат:* 
IS NOT NULL	Отбираются строки, имеющие значения в указанном поле *Пример:* SELECT * FROM topics WHERE id_author IS NOT NULL; *Результат:* 

IS NULL	Отбираются строки, не имеющие значения в указанном поле *Пример:* SELECT * FROM topics WHERE id_author IS NULL; *Результат:*  Empty set - нет таких строк.
BETWEEN (между)	Отбираются значения, находящиеся между указанными *Пример:* SELECT * FROM topics WHERE id_author BETWEEN 1 AND 3; *Результат:* 

IN (значение содержится)	Отбираются значения, соответствующие указанным *Пример:* SELECT * FROM topics WHERE id_author IN (1, 4); *Результат:* 
NOT IN (значение не содержится)	Отбираются значения, кроме указанных *Пример:* SELECT * FROM topics WHERE id_author NOT IN (1, 4); *Результат:* 

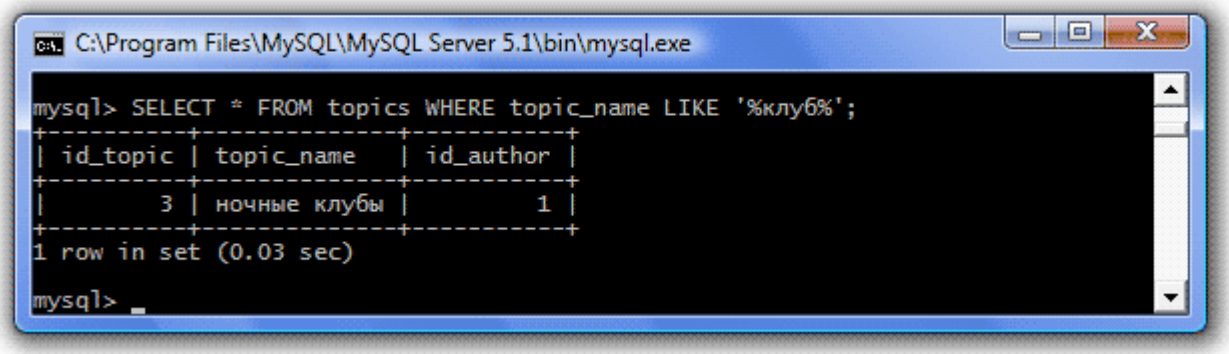
LIKE (соответствие)	Отбираются значения, соответствующие образцу *Пример:* SELECT * FROM topics WHERE topic_name LIKE 'вел%'; *Результат:*  Возможные метасимволы оператора LIKE будут рассмотрены ниже.
NOT LIKE (не соответствие)	Отбираются значения, не соответствующие образцу *Пример:* SELECT * FROM topics WHERE topic_name NOT LIKE 'вел%'; *Результат:*



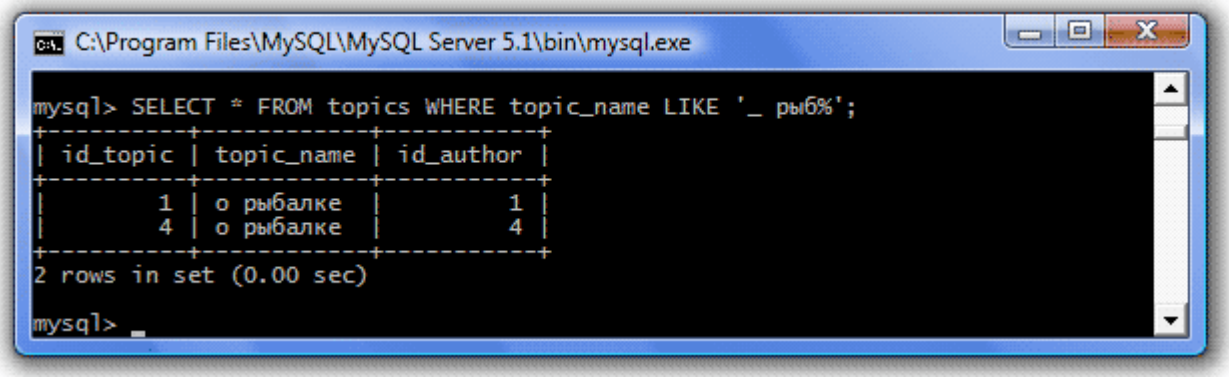
Метасимволы оператора LIKE

Поиск с использованием метасимволов может осуществляться только в текстовых полях.

Самый распространенный метасимвол - %. Он означает любые символы. Например, если нам надо найти слова, начинающиеся с букв "вел", то мы напишем LIKE 'вел%', а если мы хотим найти слова, которые содержат символы "клуб", то мы напишем LIKE '%клуб%'. Например:



Еще один часто используемый метасимвол - _. В отличие от %, который обозначает несколько или ни одного символа, нижнее подчеркивание обозначает ровно один символ. Например:



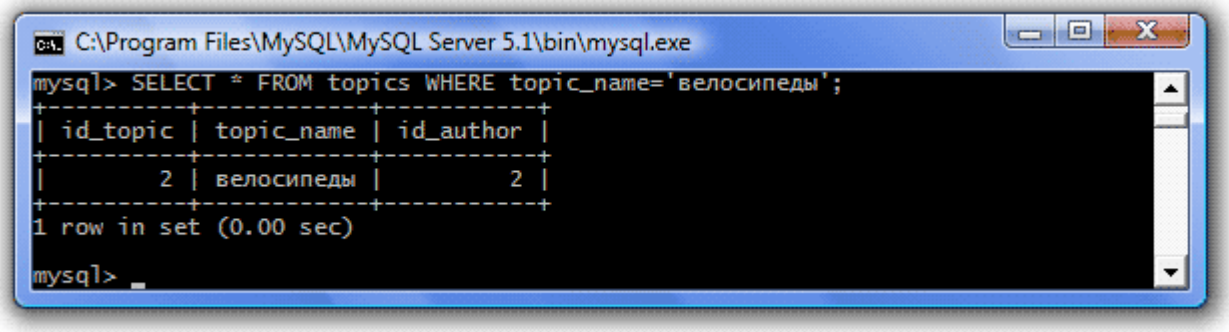
Обратите внимание на пробел между метасимволом и "рыб", если его пропустить, то запрос не сработает, т.к. метасимвол _ обозначает ровно один символ, а пробел - это тоже символ.

На сегодня достаточно. В следующем уроке мы научимся составлять запросы к двум и более таблицам. А пока попробуйте самостоятельно составить запросы к таблице posts (сообщения).

Лабораторная работа №5

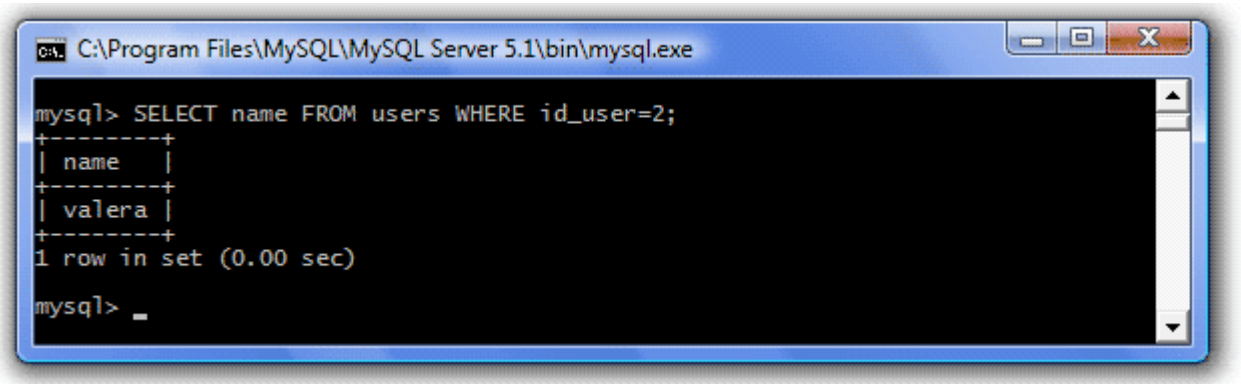
Тема: Вложенные запросы

В прошлом уроке мы столкнулись с одним неудобством. Когда мы хотели узнать, кто создал тему "велосипеды", и делали соответствующий запрос:

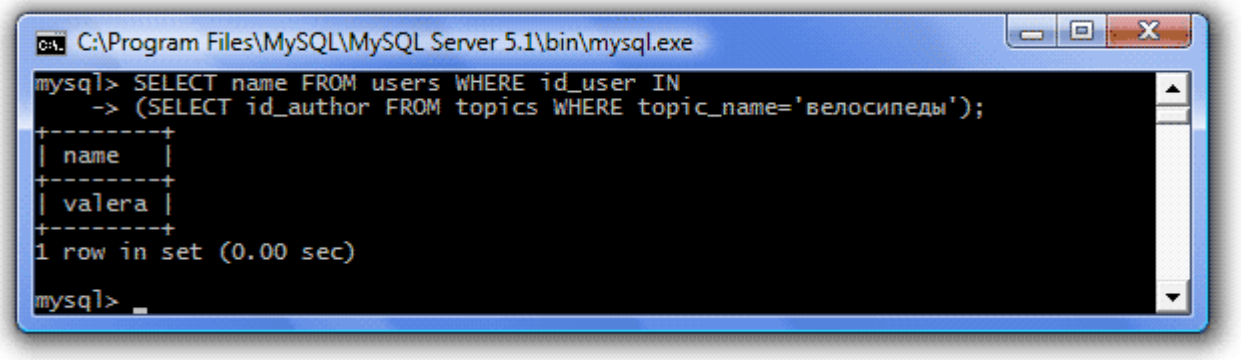


Вместо имени автора, мы получали его идентификатор. Это и понятно, ведь мы делали

запрос к одной таблице - Темы, а имена авторов тем хранятся в другой таблице - Пользователи. Поэтому, узнав идентификатор автора темы, нам надо сделать еще один запрос - к таблице Пользователи, чтобы узнать его имя:



В SQL предусмотрена возможность объединять такие запросы в один путем превращения одного из них в подзапрос (вложенный запрос). Итак, чтобы узнать, кто создал тему "велосипеды", мы сделаем следующий запрос:



То есть, после ключевого слова *WHERE*, в условие мы записываем еще один запрос. MySQL сначала обрабатывает подзапрос, возвращает id_author=2, и это значение передается в предложение *WHERE* внешнего запроса.

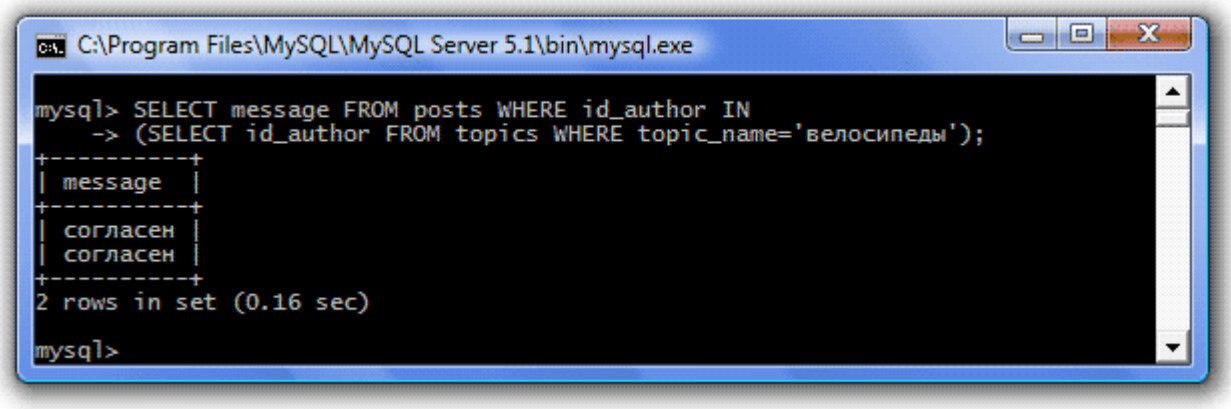
В одном запросе может быть несколько подзапросов, синтаксис у такого запроса следующий:

```
SELECT имя_столбца FROM имя_таблицы WHERE часть условия IN
```

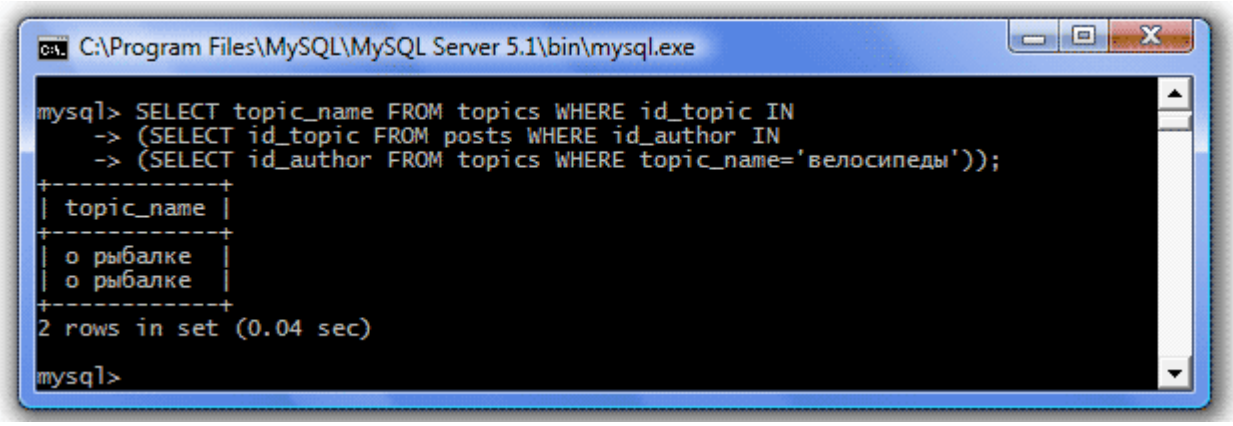
```
    (SELECT имя_столбца FROM имя_таблицы WHERE часть условия IN
      (SELECT имя_столбца FROM имя_таблицы WHERE условие)
    )
  ;
```

Обратите внимание, что подзапросы могут выбирать только один столбец, значения которого они будут возвращать внешнему запросу. Попытка выбрать несколько столбцов приведет к ошибке.

Давайте для закрепления составим еще один запрос, узнаем, какие сообщения на форуме оставлял автор темы "велосипеды":



Теперь усложним задачу, узнаем, в каких темах оставлял сообщения автор темы "велосипеды":



Давайте разберемся, как это работает.

- Сначала MySQL выполнит самый глубокий запрос:

```
SELECT id_author FROM topics WHERE topic_name='велосипеды'
```

- Полученный результат (id_author=2) передаст во внешний запрос, который примет вид:

```
SELECT id_topic FROM posts WHERE id_author IN (2);
```

- Полученный результат (id_topic:4,1) передаст во внешний запрос, который примет вид:

```
SELECT topic_name FROM topics WHERE id_topic IN (4,1);
```

- И выдаст окончательный результат (topic_name: о рыбалке, о рыбалке). Т.е. автор темы "велосипеды" оставлял сообщения в теме "О рыбалке", созданной Сергеем (id=1) и в теме "О рыбалке", созданной Светой (id=4).

Вот собственно и все, что хотелось сказать о вложенных запросах. Хотя, есть два момента, на которые стоит обратить внимание:

- Не рекомендуется создавать запросы со степенью вложения больше трех. Это приводит к увеличению времени выполнения и к сложности восприятия кода.

- Приведенный синтаксис вложенных запросов, скорее наиболее употребительный, но вовсе не единственный. Например, мы могли бы вместо запроса

```
SELECT name FROM users WHERE id_user IN
(SELECT id_author FROM topics WHERE topic_name='велосипеды');
```

написать

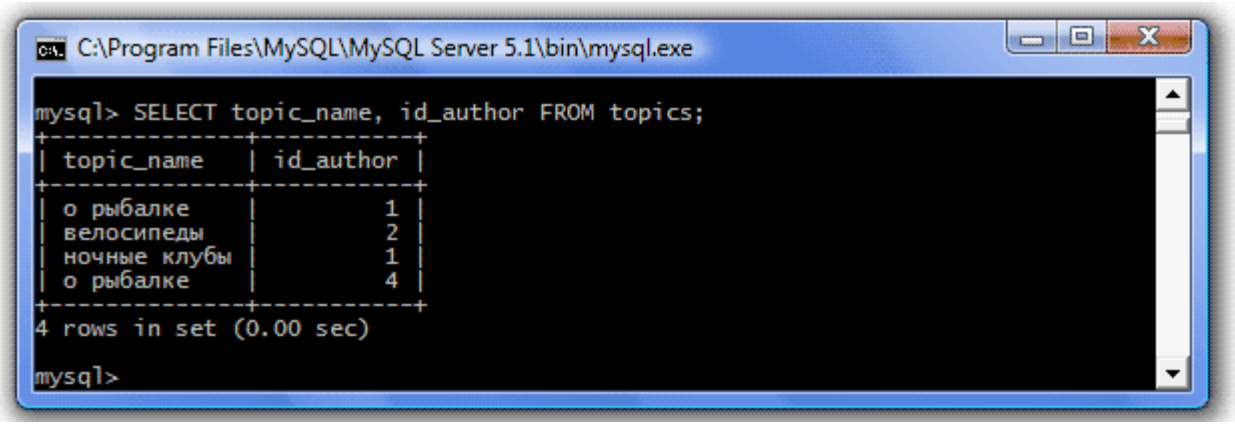
```
SELECT name FROM users WHERE id_user =
(SELECT id_author FROM topics WHERE topic_name='велосипеды');
```

Т.е. мы можем использовать любые операторы, используемые с ключевым словом WHERE (их мы изучали в прошлом уроке).

Лабораторная работа №6

Тема: Объединение таблиц (внутреннее объединение)

Предположим, мы хотим узнать, какие темы, и какими авторами были созданы. Для этого проще всего обратиться к таблице Темы (topics):



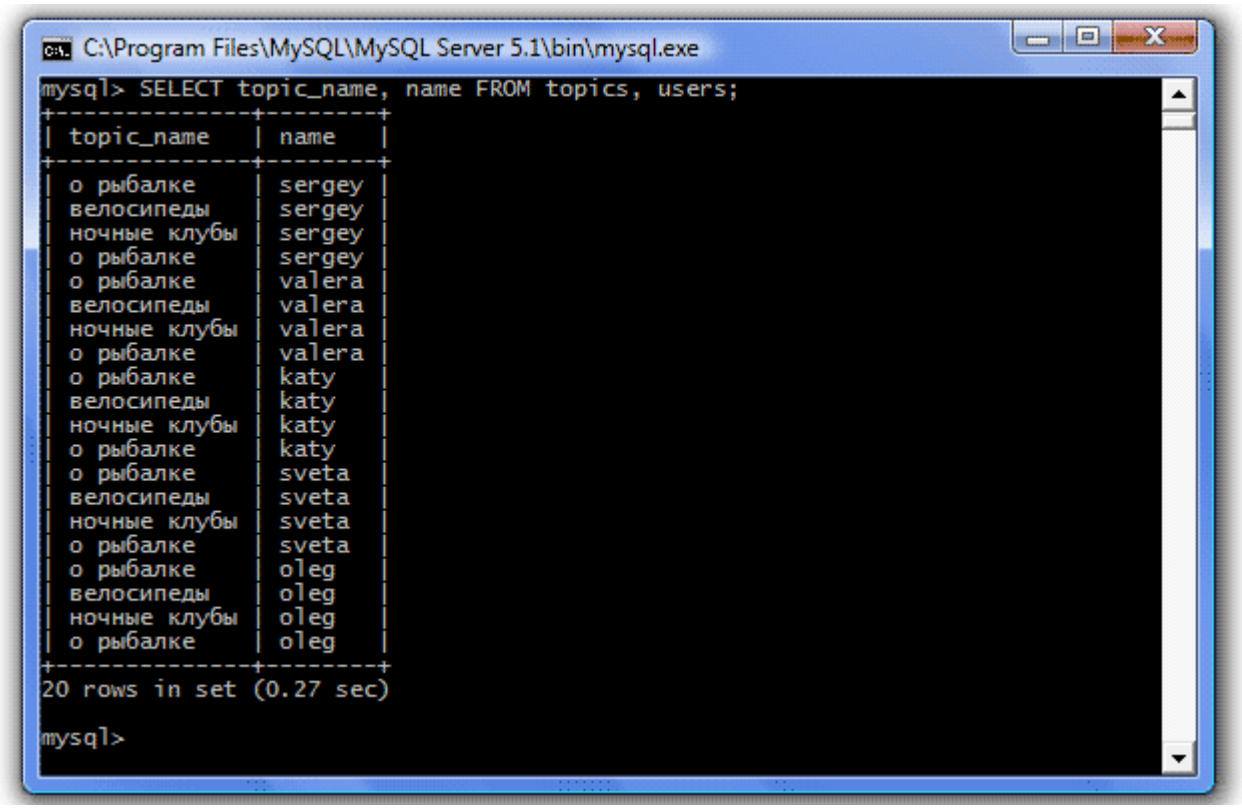
Но, что если нам необходимо, чтобы в ответе на запрос были не идентификаторы

авторов, а их имена? Вложенные запросы нам не помогут, т.к. в конечном итоге они выдают данные из одной таблицы. А нам надо получить данные из двух таблиц (Темы и Пользователи) и объединить их в одну. Запросы, которые позволяют это сделать, в SQL называются *Объединениями*.

Синтаксис самого простого объединения следующий:

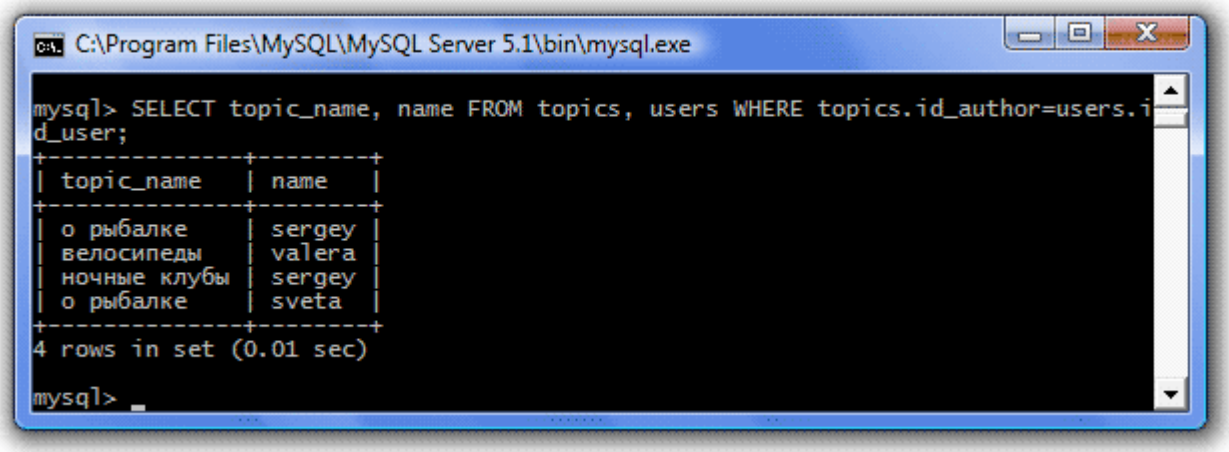
```
SELECT имена_столбцов_таблицы_1, имена_столбцов_таблицы_2 FROM
имя_таблицы_1, имя_таблицы_2;
```

Давайте создадим простое объединение:

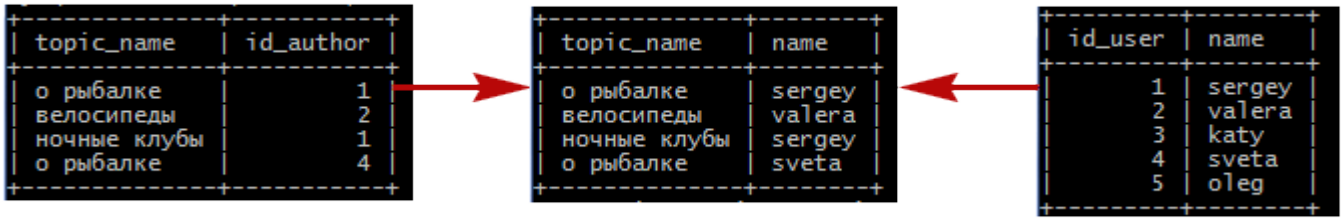


Получилось не совсем то, что мы ожидали. Такое объединение научно называется декартовым произведением, когда каждой строке первой таблицы ставится в соответствие каждая строка второй таблицы. Возможно, бывают случаи, когда такое объединение полезно, но это явно не наш случай.

Чтобы результирующая таблица выглядела так, как мы хотели, необходимо указать условие объединения. Мы связываем наши таблицы по идентификатору автора, это и будет нашим условием. Т.е. мы укажем в запросе, что необходимо выводить только те строки, в которых значения поля id_author таблицы topics совпадают со значениями поля id_user таблицы users:



На схеме будет понятнее:



Т.е. мы в запросе сделали следующее условие: если в обеих таблицах есть одинаковые идентификаторы, то строки с этим идентификатором необходимо объединить в одну результирующую строку.

Обратите внимание на две вещи:

- Если в одной из объединяемых таблиц есть строка с идентификатором, которого нет в другой объединяемой таблице, то в результирующей таблице строки с таким идентификатором не будет. В нашем примере есть пользователь Oleg (id=5), но он не создавал тем, поэтому в результате запроса его нет.

- При указании условия название столбца пишется после названия таблицы, в которой этот столбец находится (через точку). Это сделано во избежание путаницы, ведь столбцы в разных таблицах могут иметь одинаковые названия, и MySQL может не понять, о каких конкретно столбцах идет речь.

Вообще, корректный синтаксис объединения с условием выглядит так:

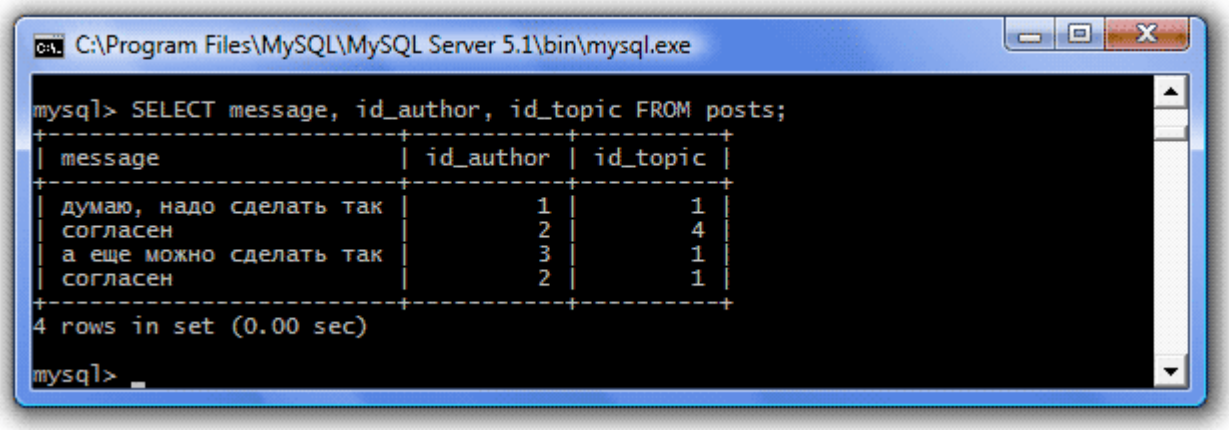
```
SELECT имя_таблицы_1.имя_столбца1_таблицы_1,
    имя_таблицы_1.имя_столбца2_таблицы_1,
    имя_таблицы_2.имя_столбца1_таблицы_2,
    имя_таблицы_2.имя_столбца2_таблицы_2
FROM
    имя_таблицы_1, имя_таблицы_2
WHERE
    имя_таблицы_1.имя_столбца_по_которому_объединяем =
    имя_таблицы_2.имя_столбца_по_которому_объединяем;
```

Если имя столбца уникально, то название таблицы можно опустить (как мы делали в примере), но делать это не рекомендуется.

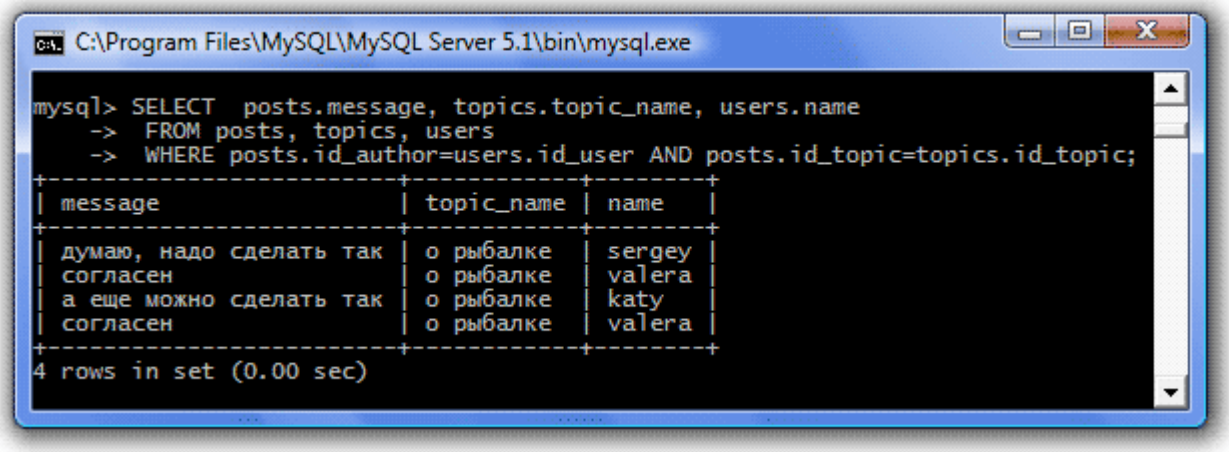
Как вы понимаете, объединения дают возможность выбирать любую информацию из любых таблиц, причем объединяемых таблиц может быть и три, и четыре, да и

условие для объединения может быть не одно.

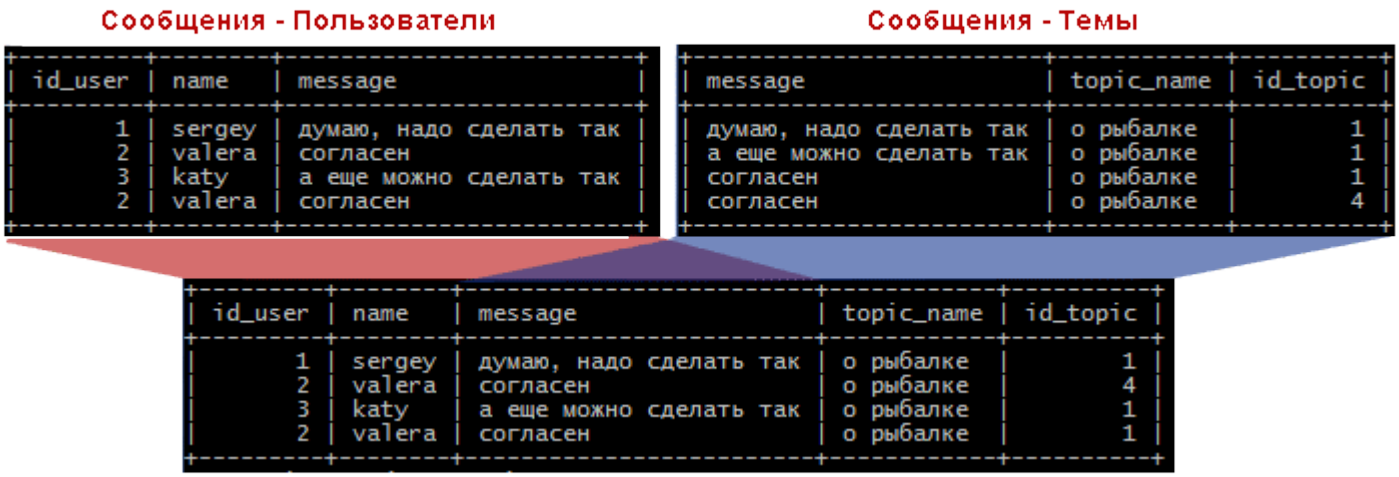
Для примера давайте создадим запрос, который покажет нам все сообщения, к каким темам они относятся и авторов этих сообщений. Конечно, вся эта информация хранится в таблице Сообщения (posts):



Но чтобы вместо идентификаторов отображались имена и названия, нам придется сделать объединение трех таблиц:



Т.е. мы объединили таблицы Сообщения и Пользователи условием posts.id_author=users.id_user, а таблицы Сообщения и Темы - условием posts.id_topic=topics.id_topic



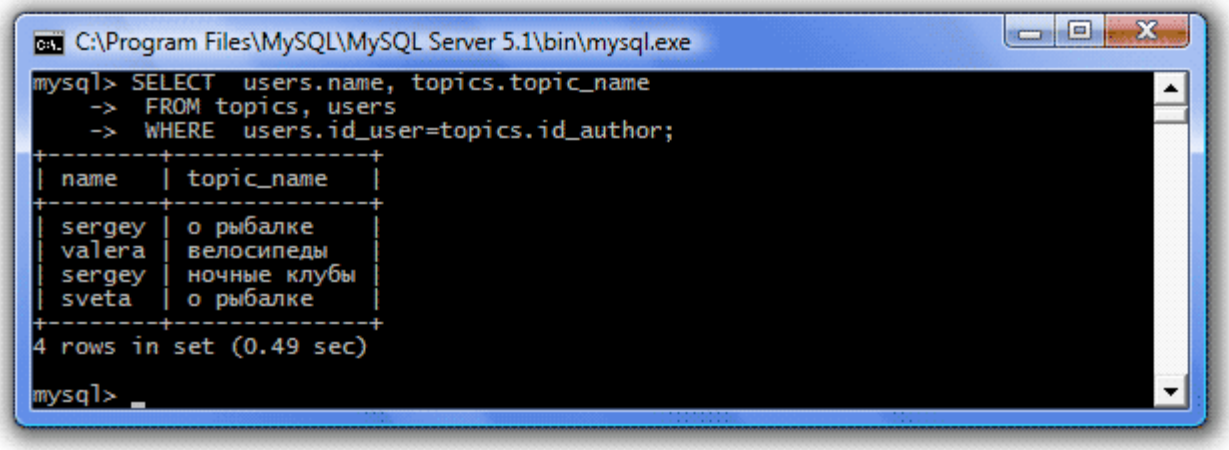
Объединения, которые мы сегодня рассматривали, называются *Внутренними объединениями*. Такие объединения связывают строки одной таблицы со строками другой таблицы (а может еще и третьей таблицы). Но бывают ситуации, когда необходимо, чтобы в результат были включены строки, не имеющие связанных. Например, когда мы создавали запрос, какие темы и какими авторами были созданы, пользователь Oleg в результирующую таблицу не попал, т.к. тем не создавал, а потому и связанной строки в объединяемой таблице не имел.

Поэтому, если нам потребуется составить несколько иной запрос - вывести всех пользователей и темы, которые они создавали, если таковые имеются - то нам придется воспользоваться *Внешним объединением*, позволяющим выводить все строки одной таблицы и имеющиеся связанные с ними строки из другой таблицы. О таких объединениях мы и будем говорить в следующем уроке.

Лабораторная работа №7

Тема: Объединение таблиц (внешнее объединение)

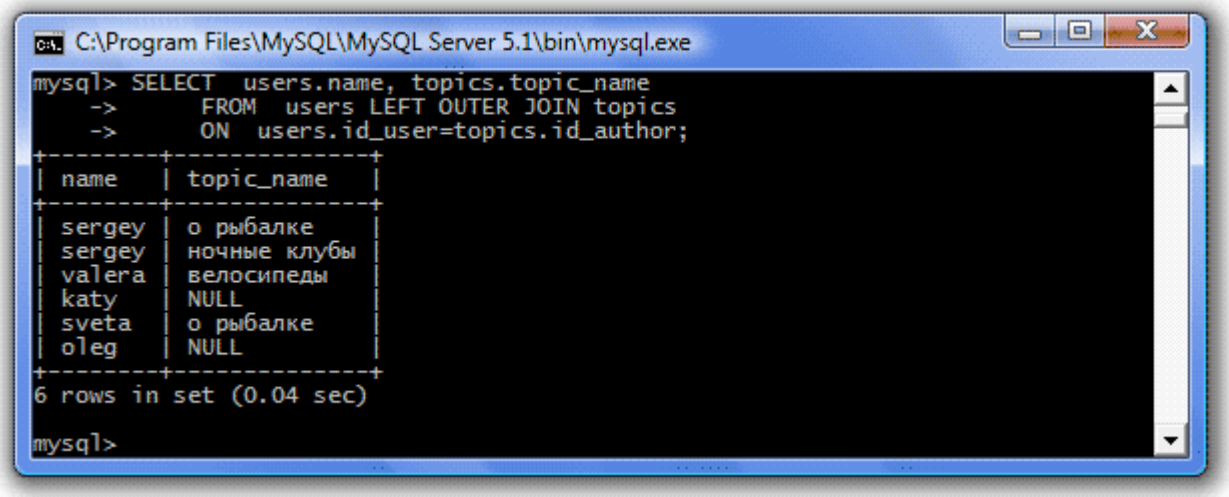
Итак, в продолжение прошлого урока, нам надо вывести всех пользователей и темы, которые они создавали, если таковые имеются. Если мы воспользуемся внутренним объединением, рассмотренным на прошлом уроке, то получим в итоге следующее:



То есть в результирующей таблице есть только те пользователи, которые создавали темы. А нам надо, чтобы выводились все имена. Для этого мы немного изменим запрос:

```
SELECT  users.name, topics.topic_name
FROM  users LEFT OUTER JOIN topics
ON  users.id_user=topics.id_author;
```

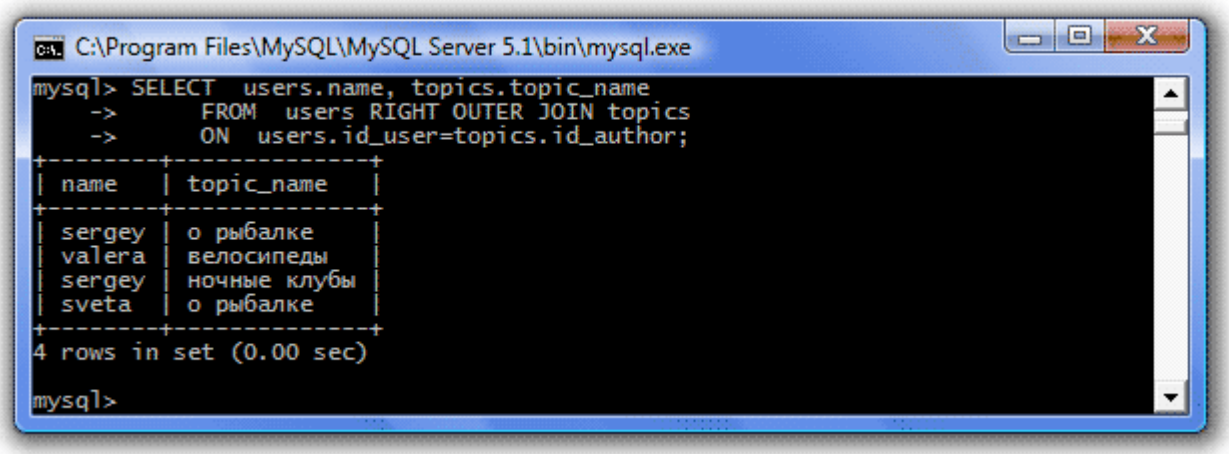
И получим желаемый результат - все пользователи и темы, ими созданные. Если пользователь не создавал тему, но в соответствующем столбце стоит значение NULL.



Итак, мы добавили в наш запрос ключевое слово - *LEFT OUTER JOIN*, указав тем самым, что из таблицы слева надо взять все строки, и поменяли ключевое

слово *WHERE* на *ON*. Кроме ключевого слова *LEFT OUTER JOIN* может быть использовано ключевое слово *RIGHT OUTER JOIN*. Тогда будут выбираться все строки из правой таблицы и имеющиеся связанные с ними из левой таблицы. И наконец, возможно полное внешнее объединение, которое извлечет все строки из обеих таблиц и свяжет между собой те, которые могут быть связаны. Ключевое слово для полного внешнего объединения - *FULL OUTER JOIN*.

Давайте поменяем в нашем запросе левостороннее объединение на правостороннее:



Как видите, теперь у нас есть все темы (все строки из правой таблицы), а вот пользователи только те, которые темы создавали (т.е. из левой таблицы выбираются только те строки, которые связаны с правой таблицей).

К сожалению полное объединение СУБД MySQL не поддерживает.

Подведем итог этого короткого урока. Синтаксис для внешнего объединения следующий:

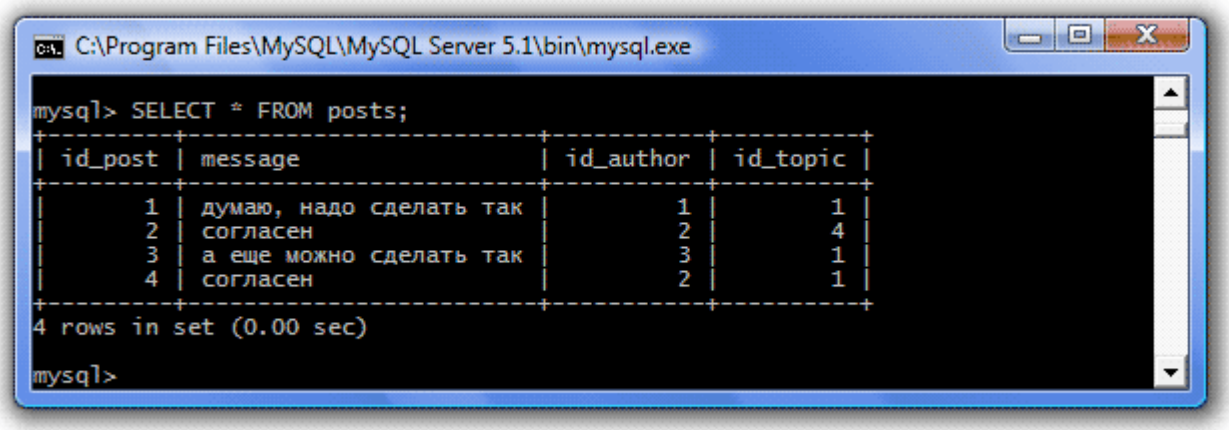
```
SELECT имя_таблицы_1.имя_столбца, имя_таблицы_2.имя_столбца
FROM имя_таблицы_1 ТИП ОБЪЕДИНЕНИЯ имя_таблицы_2
ON условие_объединения;
```

где ТИП ОБЪЕДИНЕНИЯ - либо LEFT OUTER JOIN, либо RIGHT OUTER JOIN

Лабораторная работа №8

Тема: Группировка записей и функция COUNT()

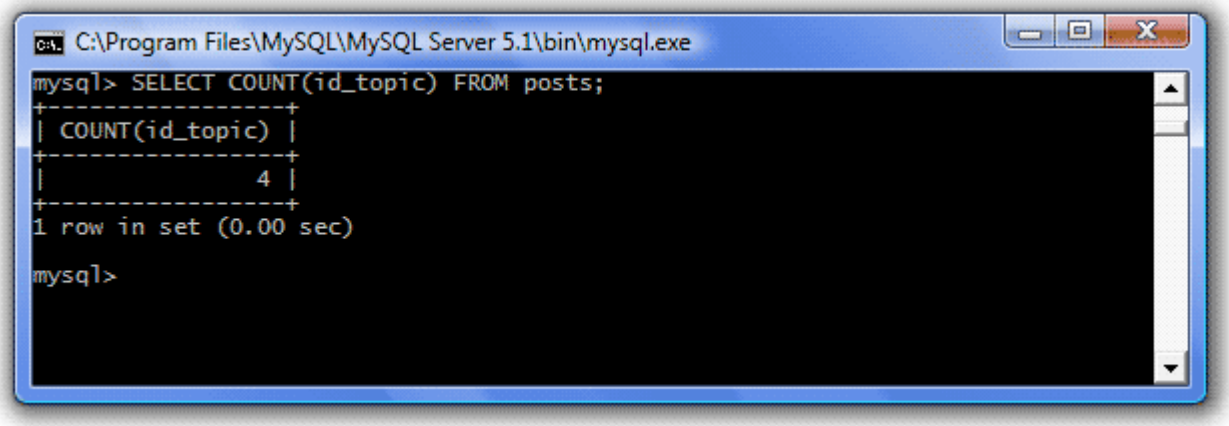
Давайте вспомним, какие сообщения и в каких темах у нас имеются. Для этого можно воспользоваться привычным запросом:



А что, если нам надо лишь узнать сколько сообщений на форуме имеется. Для этого можно воспользоваться встроенной функцией *COUNT()*. Эта функция подсчитывает число строк. Причем, если в качестве аргумента этой функции выступает *, то подсчитываются все строки таблицы. А если в качестве аргумента указывается имя столбца, то подсчитываются только те строки, которые имеют значение в указанном столбце.

В нашем примере оба аргумента дадут одинаковый результат, т.к. все столбцы таблицы имеют тип NOT NULL. Давайте напишем запрос, используя в качестве аргумента столбец id_topic:

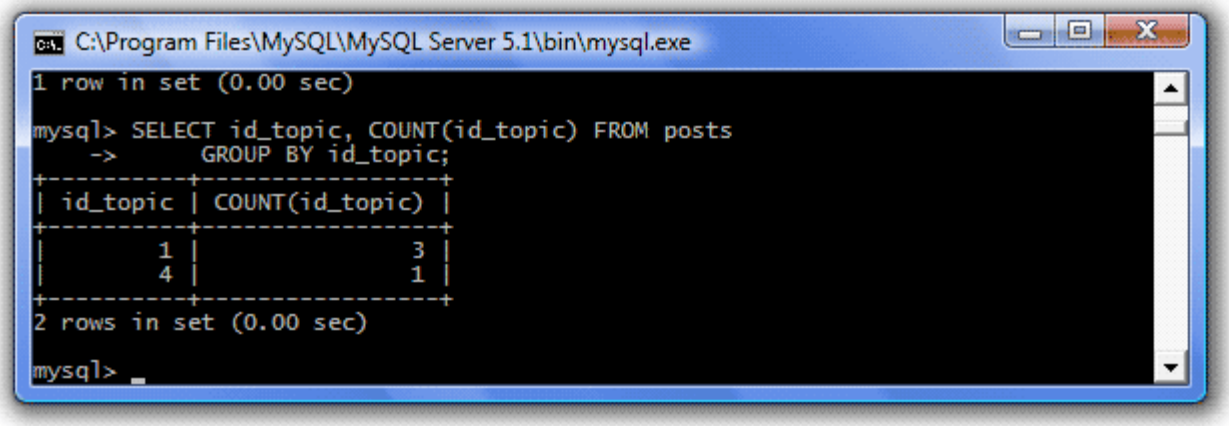
```
SELECT COUNT(id_topic) FROM posts;
```



Итак, в наших темах имеется 4 сообщения. Но что, если мы хотим узнать сколько сообщений имеется в каждой теме. Для этого нам понадобится сгруппировать наши сообщения по темам и вычислить для каждой группы количество сообщений. Для группировки в SQL используется оператор *GROUP BY*. Наш запрос теперь будет выглядеть так:

```
SELECT id_topic, COUNT(id_topic) FROM posts
GROUP BY id_topic;
```

Оператор *GROUP BY* указывает СУБД сгруппировать данные по столбцу id_topic (т.е. каждая тема - отдельная группа) и для каждой группы подсчитать количество строк:

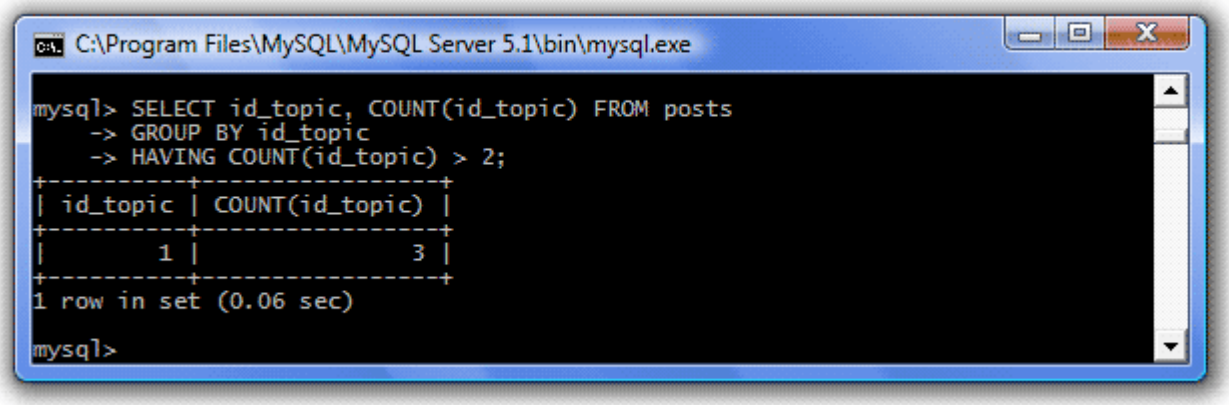


Ну вот, в теме с id=1 у нас 3 сообщения, а с id=4 - одно. Кстати, если бы в поле id_topic были возможны отсутствия значений, то такие строки были бы объединены в отдельную группу со значением NULL.

Предположим, что нас интересуют только те группы, в которых больше двух сообщений. В обычном запросе мы указали бы условие с помощью оператора *WHERE*, но этот оператор умеет работать только со строками, а для групп те же функции выполняет оператор *HAVING*:

```
SELECT id_topic, COUNT(id_topic) FROM posts
GROUP BY id_topic
HAVING COUNT(id_topic) > 2;
```

В результате имеем:



В уроке 4 мы рассматривали, какие условия можно задавать оператором *WHERE*, те же условия можно задавать и оператором *HAVING*, только надо запомнить, что *WHERE* фильтрует строки, а *HAVING* - группы.

Итак, сегодня мы узнали, как создавать группы и как подсчитать количество строк в

таблице и в группах. Вообще вместе с оператором *GROUP BY* можно использовать и другие встроенные функции, но их мы будем изучать позже.

Лабораторная работа №9

Тема: Редактирование, обновление и удаление данных

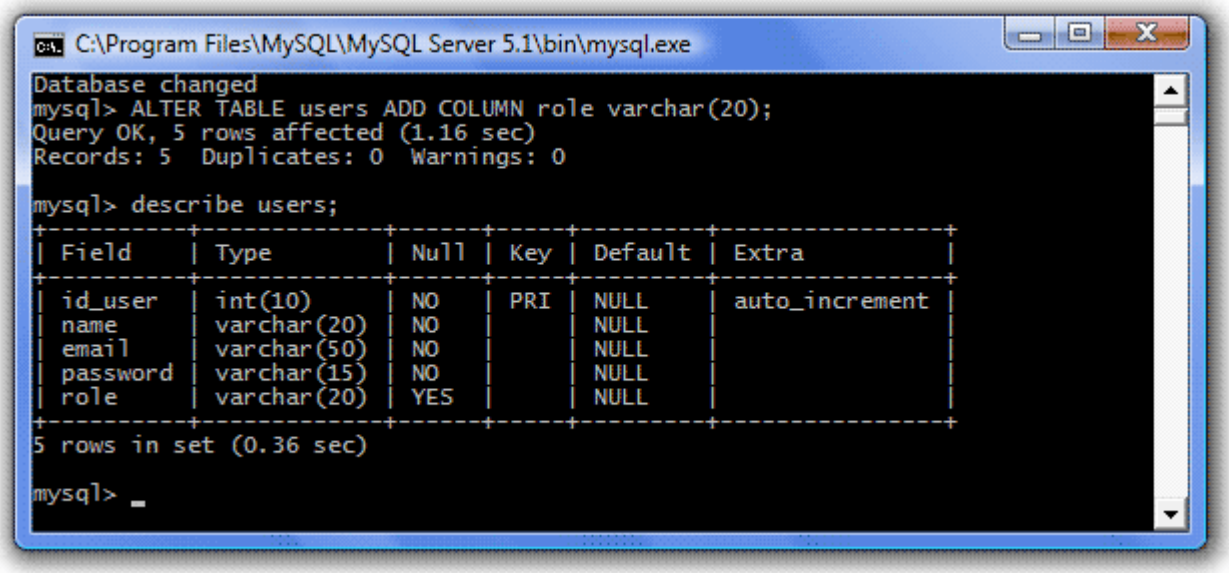
Предположим, мы решили, что нашему форуму нужны модераторы. Для этого в таблицу users надо добавить столбец с ролью пользователя. Для добавления столбцов в таблицу используется оператор *ALTER TABLE - ADD COLUMN*. Его синтаксис следующий:

```
ALTER TABLE имя_таблицы ADD COLUMN имя_столбца тип;
```

Давайте добавим столбец role в таблицу users:

```
ALTER TABLE users ADD COLUMN role varchar(20);
```

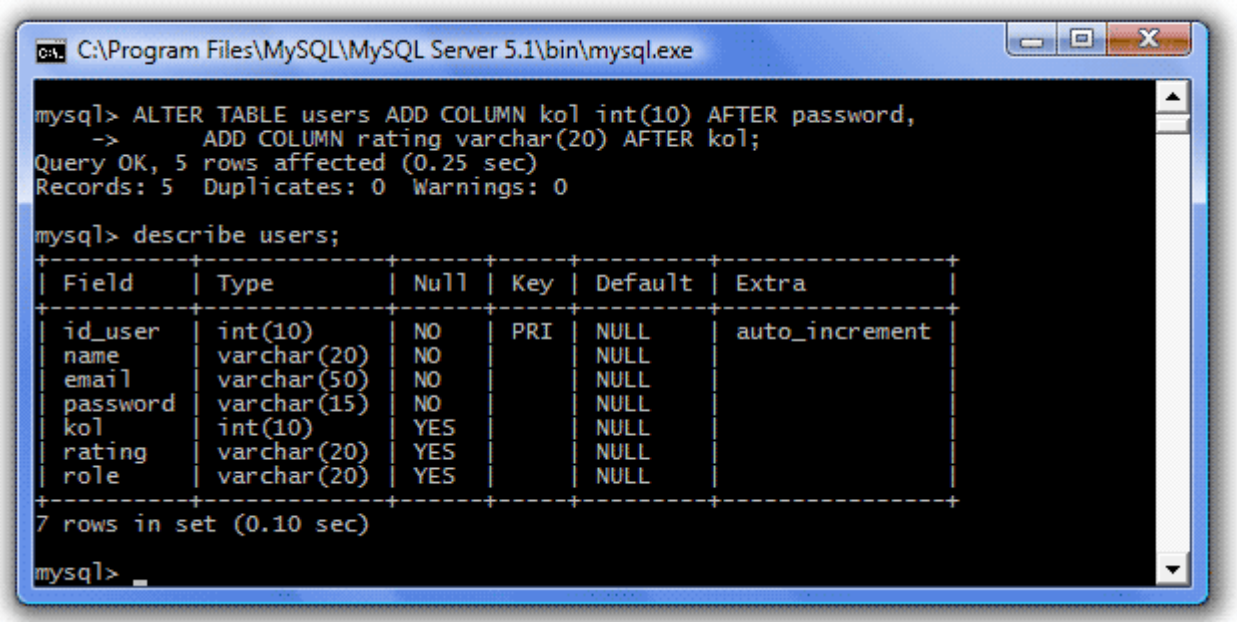
Столбец появился в конце таблицы:



Для того, чтобы указать местоположение столбца используются ключевые слова: *FIRST* - новый столбец будет первым, и *AFTER* - указывает после какого столбца поместить новый.

Давайте добавим еще два столбца: один - kol - количество оставленных сообщений, а другой - rating - рейтинг пользователя. Оба столбца вставим после поля password:

```
ALTER TABLE users ADD COLUMN kol int(10) AFTER password,
ADD COLUMN rating varchar(20) AFTER kol;
```

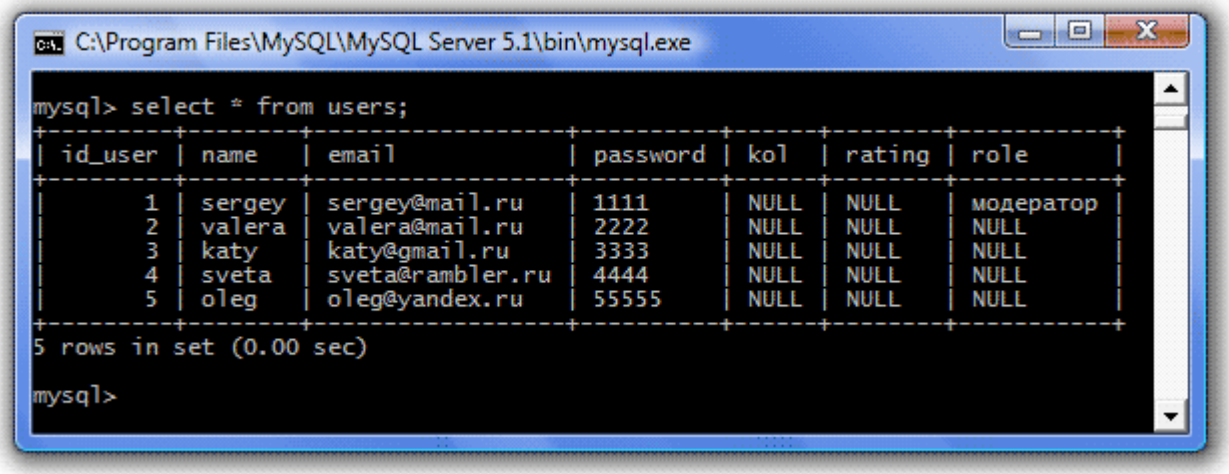


Теперь надо назначить роль модератора какому-нибудь пользователю, пусть это будет sergey с id=1. Для обновления уже существующих данных служит оператор *UPDATE*. Его синтаксис следующий:

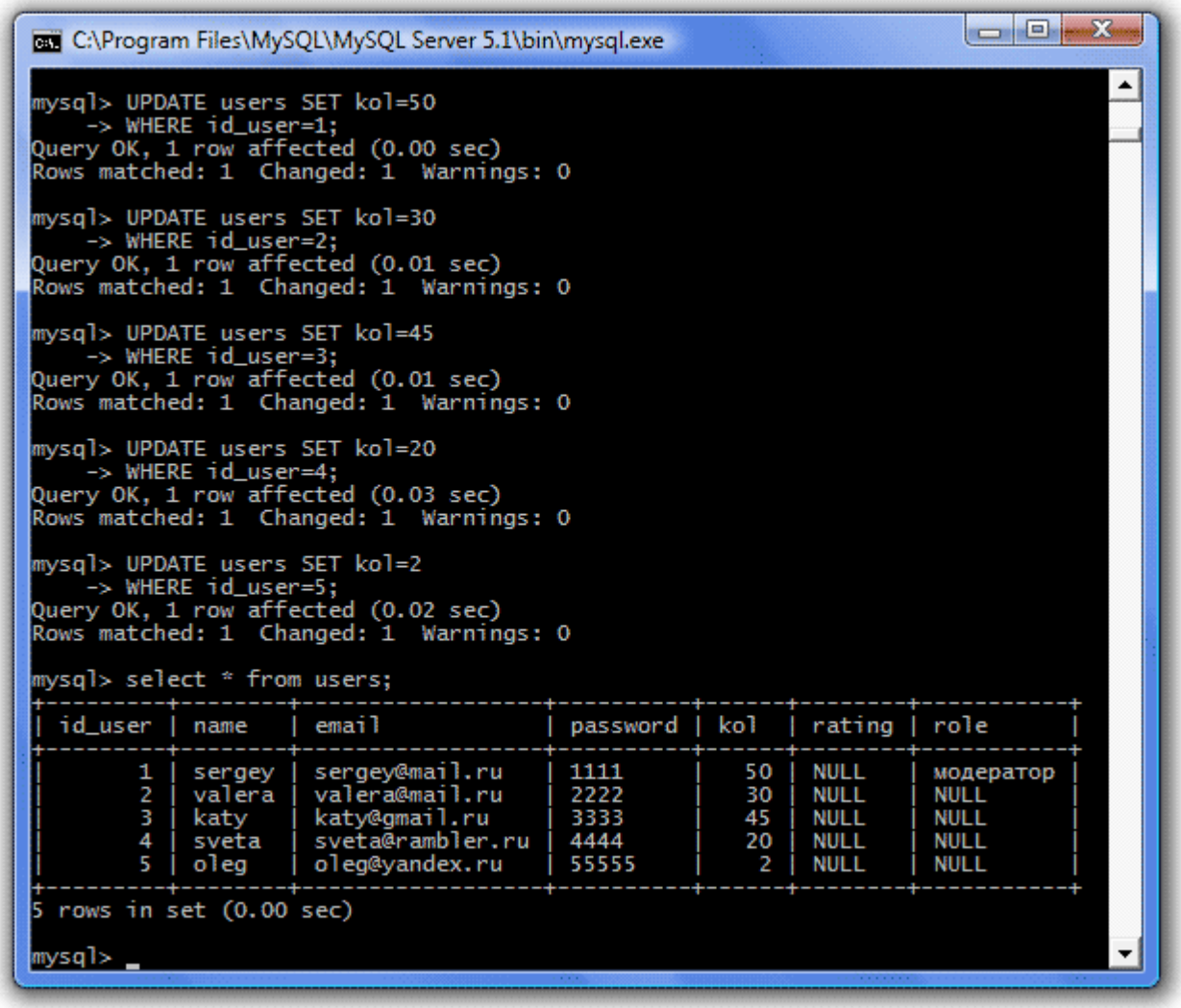
```
UPDATE имя_таблицы SET имя_столбца=значение_столбца
WHERE условие;
```

Давайте сделаем Сергея модератором:

```
UPDATE users SET role='модератор'
WHERE id_user=1;
```

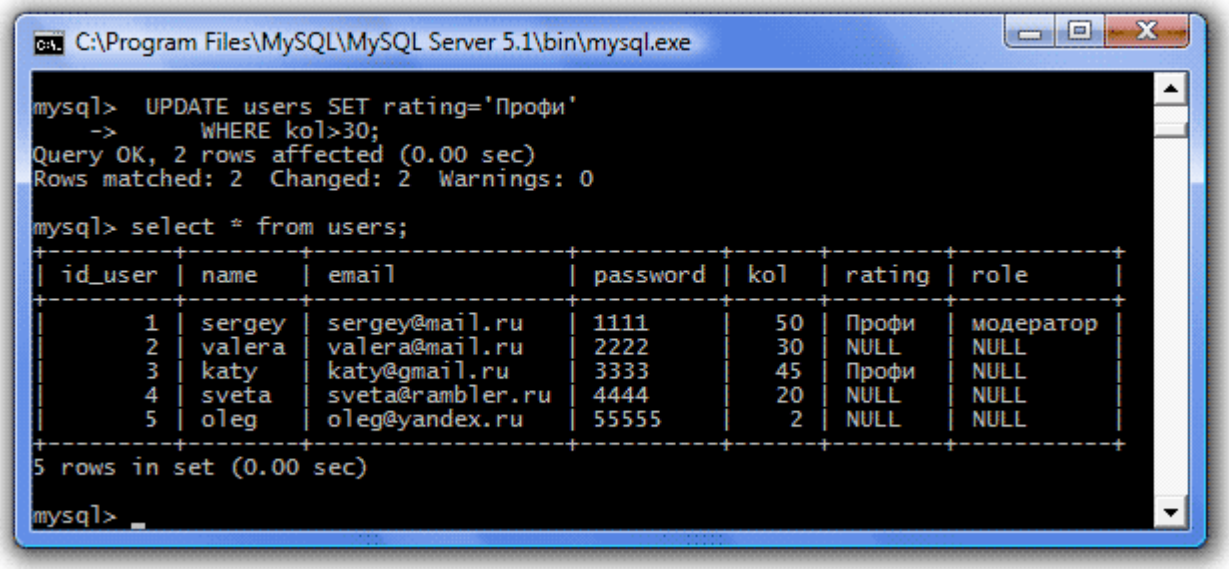


Изменять данные можно и сразу в нескольких строках и во всей таблице. Например, мы решили давать рейтинг в зависимости от количества оставленных пользователем сообщений. Давайте в нашу таблицу сначала внесем значения столбца kol так, как мы уже умеем:



А теперь давайте зададим рейтинг Профи тем, у кого количество сообщений больше 30:

```
UPDATE users SET rating='Профи'
WHERE kol>30;
```



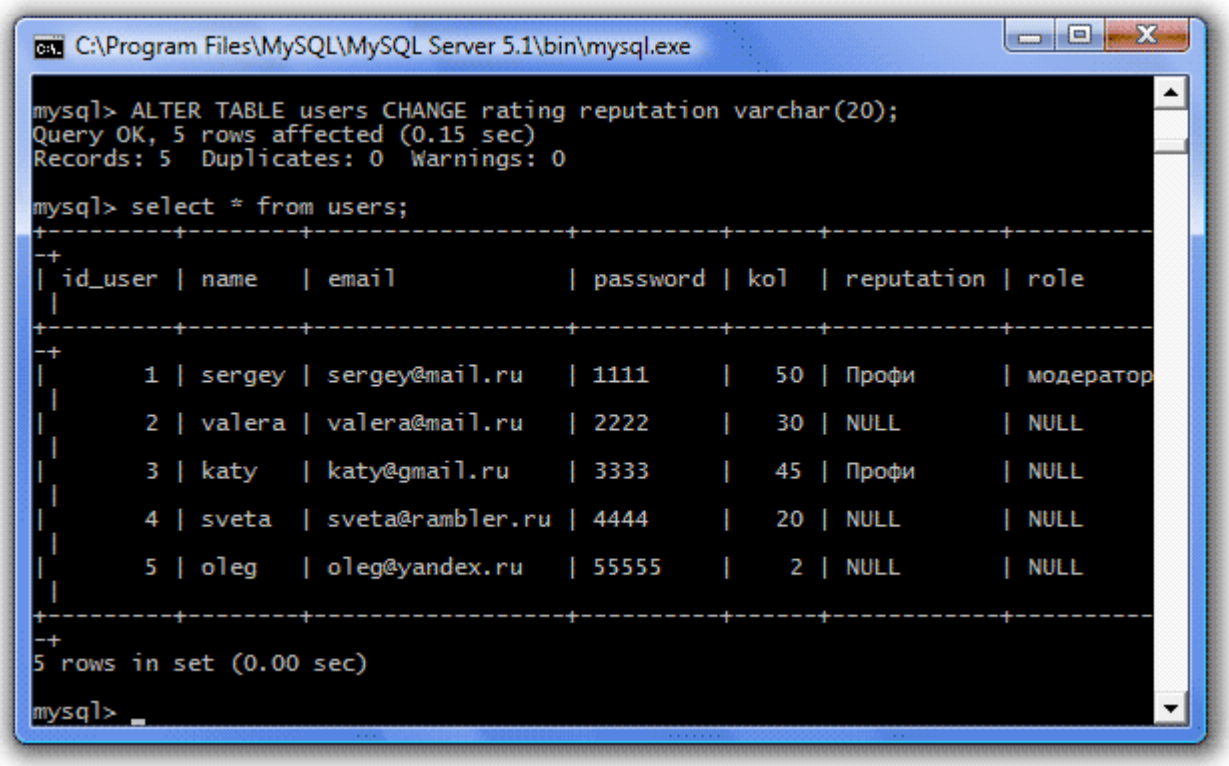
Данные изменились в двух строках, согласно заданному условию. Понятно, что если в запросе опустить условие, то данные будут обновлены во всех строках таблицы.

Предположим, что нам не нравится название Рейтинг у нашего столбца, и мы хотим переименовать столбец в Репутация - reputation. Для изменения имени существующего столбца используется оператор *CHANGE*. Его синтаксис следующий:

```
ALTER TABLE имя_таблицы CHANGE старое_имя_столбца новое_имя_столбца тип;
```

Давайте поменяем rating на reputation:

```
ALTER TABLE users CHANGE rating reputation varchar(20);
```



Обратите внимание, что тип столбца надо указывать даже, если он не меняется. Кстати, если нам понадобится изменить только тип столбца, то мы будем использовать оператор *MODIFY*. Его синтаксис следующий:

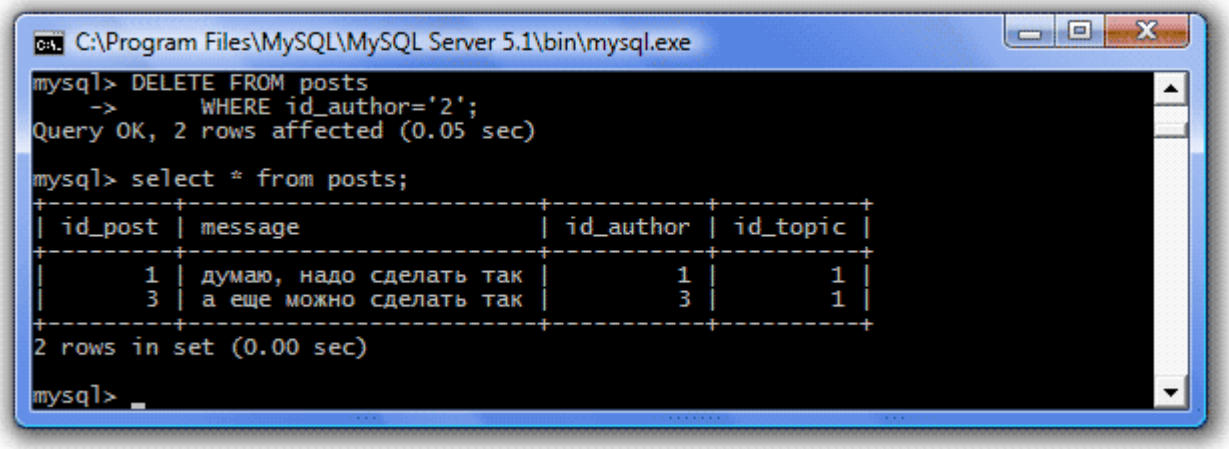
```
ALTER TABLE имя_таблицы MODIFY имя_столбца  новый_тип;
```

Последнее, что мы сегодня рассмотрим - оператор *DELETE*, который позволяет удалять строки из таблицы. Его синтаксис следующий:

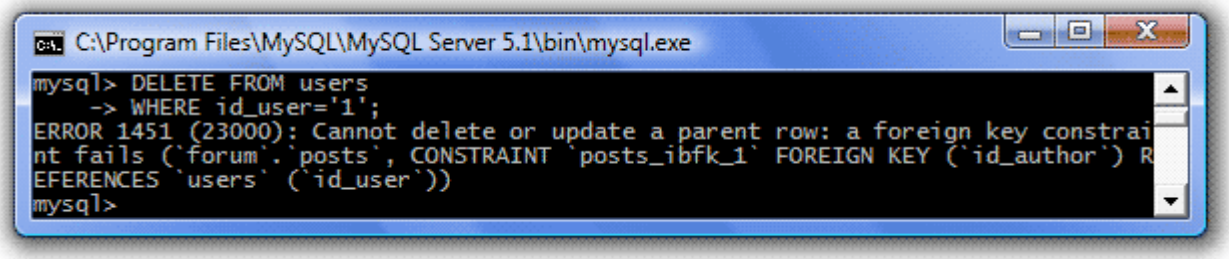
```
DELETE FROM имя_таблицы
WHERE условие;
```

Давайте из таблицы сообщений удалим те записи, которые оставлял пользователь valera (id=2):

```
DELETE FROM posts
WHERE id_author='2';
```



Понятно, если опустить условие, то из таблицы будут удалены все данные. Следует помнить, что данные СУБД даст удалить только в том случае, если они не являются внешними ключами для данных из других таблиц (поддержка целостности БД). Например, если мы захотим удалить из таблицы users пользователя, который оставлял сообщения, то нам это не удастся.



Сначала надо удалить его сообщения, а уж потом и его самого.

Давайте подведем промежуточный итог. Мы умеем создавать таблицы и связывать их между собой, обновлять, редактировать и удалять данные и извлекать данные различным образом. В принципе - это можно назвать базовыми знаниями SQL. Далее мы будем изучать встроенные функции и расширенные возможности MySQL.

Лабораторная работа №10

Тема:Встроенные функции

Функции - это операции, позволяющие манипулировать данными. В MySQL можно выделить несколько групп встроенных функций:

- Строковые функции. Используются для управления текстовыми строками, например, для обрезания или заполнения значений.

- Числовые функции. Используются для выполнения математических операций над числовыми данными. К числовым функциям относятся функции возвращающие абсолютные значения, синусы и косинусы углов, квадратный корень числа и т.д. Используются они только для алгебраических, тригонометрических и геометрических вычислений. В общем, используются редко, поэтому рассматривать их мы не будем. Но вы должны знать, что они существуют, и в случае необходимости обратиться к документации MySQL.

- Итоговые функции. Используются для получения итоговых данных по таблицам, например, когда надо просуммировать какие-либо данные без их выборки.

- Функции даты и времени. Используются для управления значениями даты и времени, например, для возвращения разницы между датами.

- Системные функции. Возвращают служебную информацию СУБД.

Для того, чтобы рассмотреть основные встроенные функции нам понадобится создать новую базу данных, чтобы в ней были числовые значения и значения даты. В уроке 5 основ баз данных мы сделали реляционную модель базы данных интернет-магазина. Пришло время реализовать ее в MySQL, заодно закрепим пройденное.

Итак, смотрим на последнюю схему урока 5 по БД и создаем БД - shop.

```
create database shop;
```

Выбираем ее для работы:

```
use shop;
```

И создаем в ней 8 таблиц, как в схеме: Покупатели (customers), Поставщики (vendors), Покупки (sale), Поставки (incoming), Журнал покупок (magazine_sales), Журнал поставок (magazine_incoming), Товары (products), Цены (prices). Один нюанс, наш магазин будет торговать книгами, поэтому в таблицу Товары мы добавим еще один столбец - Автор (author), в принципе это необязательно, но так как-то привычнее.

```
create table customers (
  id_customer int NOT NULL AUTO_INCREMENT,
  name char(50) NOT NULL,
  email char(50) NOT NULL,
  PRIMARY KEY (id_customer)
);

create table vendors (
  id_vendor int NOT NULL AUTO_INCREMENT,
  name char(50) NOT NULL,
```

```
  city char(30) NOT NULL,
  address char(100) NOT NULL,
  PRIMARY KEY (id_vendor)
);

create table sale (
  id_sale int NOT NULL AUTO_INCREMENT,
  id_customer int NOT NULL,
  date_sale date NOT NULL,
  PRIMARY KEY (id_sale),
  FOREIGN KEY (id_customer) REFERENCES customers (id_customer)
);

create table incoming (
  id_incoming int NOT NULL AUTO_INCREMENT,
  id_vendor int NOT NULL,
  date_incoming date NOT NULL,
  PRIMARY KEY (id_incoming),
  FOREIGN KEY (id_vendor) REFERENCES vendors (id_vendor)
);

create table products (
  id_product int NOT NULL AUTO_INCREMENT,
  name char(100) NOT NULL,
  author  char(50) NOT NULL,
  PRIMARY KEY (id_product)
);

create table prices (
  id_product int NOT NULL,
  date_price_changes date NOT NULL,
```

```
  price double NOT NULL,
  PRIMARY KEY (id_product, date_price_changes),
  FOREIGN KEY (id_product) REFERENCES products (id_product)
);

create table magazine_sales (
  id_sale int NOT NULL,
  id_product int NOT NULL,
  quantity int NOT NULL,
  PRIMARY KEY (id_sale, id_product),
  FOREIGN KEY (id_sale) REFERENCES sale (id_sale),
  FOREIGN KEY (id_product) REFERENCES products (id_product)
);

create table magazine_incoming (
  id_incoming int NOT NULL,
  id_product int NOT NULL,
  quantity int NOT NULL,
  PRIMARY KEY (id_incoming, id_product),
  FOREIGN KEY (id_incoming) REFERENCES incoming (id_incoming),
  FOREIGN KEY (id_product) REFERENCES products (id_product)
);
```

Обратите внимание, что в таблицах Журнал покупок, Журнал поставок и Цены первичные ключи - составные, т.е. их уникальные значения состоят из пар значений (в таблице не может быть двух строк с одинаковыми парами значений). Названия столбцов этих пар значений и указываются через запятую после ключевого слова PRIMARY KEY. Остальное вы уже знаете.

В настоящем интернет-магазине данные в эти таблицы будут заноситься посредством сценариев на каком-либо языке (типа php), нам же пока придется внести их вручную.

Можете внести любые данные, только помните, что значения в одноименных столбцах связанных таблиц должны совпадать. Либо скопируйте нижеприведенные данные:

```
INSERT INTO vendors  (name, city, address) VALUES
 ('Вильямс', 'Москва', 'ул.Лесная, д.43'),
 ('Дом печати', 'Минск', 'пр.Ф.Скорины, д.18'),
 ('БХВ-Петербург', 'Санкт-Петербург', 'ул.Есенина, д.5');

INSERT INTO customers  (name, email) VALUES
 ('Иванов Сергей', 'sergo@mail.ru'),
 ('Ленская Катя', 'lenskay@yandex.ru'),
 ('Демидов Олег', 'demidov@gmail.ru'),
 ('Афанасьев Виктор', 'victor@mail.ru'),
 ('Пажская Вера', 'verap@rambler.ru');

INSERT INTO products  (name, author) VALUES
 ('Стихи о любви', 'Андрей Вознесенский'),
 ('Собрание сочинений, том 2', 'Андрей Вознесенский'),
 ('Собрание сочинений, том 3', 'Андрей Вознесенский'),
 ('Русская поэзия', 'Николай Заболоцкий'),
 ('Машенька', 'Владимир Набоков'),
 ('Доктор Живаго', 'Борис Пастернак'),
 ('Наши', 'Сергей Довлатов'),
 ('Приглашение на казнь', 'Владимир Набоков'),
 ('Лолита', 'Владимир Набоков'),
 ('Темные аллеи', 'Иван Бунин'),
 ('Дар', 'Владимир Набоков'),
 ('Сын вождя', 'Юлия Вознесенская'),
 ('Эмигранты', 'Алексей Толстой'),
 ('Горе от ума', 'Александр Грибоедов'),
 ('Анна Каренина', 'Лев Толстой'),
```

```
('Повести и рассказы', 'Николай Лесков'),
('Антоновские яблоки', 'Иван Бунин'),
('Мертвые души', 'Николай Гоголь'),
('Три сестры', 'Антон Чехов'),
('Беглянка', 'Владимир Даль'),
('Идиот', 'Федор Достоевский'),
('Братья Карамазовы', 'Федор Достоевский'),
('Ревизор', 'Николай Гоголь'),
('Гранатовый браслет', 'Александр Куприн');

INSERT INTO incoming  (id_vendor, date_incoming) VALUES
('1', '2011-04-10'),
('2', '2011-04-11'),
('3', '2011-04-12');

INSERT INTO magazine_incoming  (id_incoming, id_product, quantity) VALUES
('1', '1', '10'),
('1', '2', '5'),
('1', '3', '7'),
('1', '4', '10'),
('1', '5', '10'),
('1', '6', '8'),
('1', '18', '8'),
('1', '19', '8'),
('1', '20', '8'),
('2', '7', '10'),
('2', '8', '10'),
('2', '9', '6'),
('2', '10', '10'),
('2', '11', '10'),
('2', '21', '10'),
```

```
('2', '22', '10'),
('2', '23', '10'),
('2', '24', '10'),
('3', '12', '10'),
('3', '13', '10'),
('3', '14', '10'),
('3', '15', '10'),
('3', '16', '10'),
('3', '17', '10');

INSERT INTO prices  (id_product, date_price_changes, price) VALUES
('1', '2011-04-10', '100'),
('2', '2011-04-10', '130'),
('3', '2011-04-10', '90'),
('4', '2011-04-10', '100'),
('5', '2011-04-10', '110'),
('6', '2011-04-10', '85'),
('7', '2011-04-11', '95'),
('8', '2011-04-11', '100'),
('9', '2011-04-11', '79'),
('10', '2011-04-11', '49'),
('11', '2011-04-11', '105'),
('12', '2011-04-12', '85'),
('13', '2011-04-12', '135'),
('14', '2011-04-12', '100'),
('15', '2011-04-12', '90'),
('16', '2011-04-12', '75'),
('17', '2011-04-12', '90'),
('18', '2011-04-10', '150'),
('19', '2011-04-10', '140'),
('20', '2011-04-10', '85'),
```

```
('21', '2011-04-11', '105'),
('22', '2011-04-11', '70'),
('23', '2011-04-11', '65'),
('24', '2011-04-11', '130');

INSERT INTO sale  (id_customer, date_sale) VALUES
('2', '2011-04-11'),
('3', '2011-04-11'),
('5', '2011-04-11');

INSERT INTO magazine_sales  (id_sale, id_product, quantity) VALUES
('1', '1', '1'),
('1', '5', '1'),
('1', '7', '1'),
('2', '2', '1'),
('3', '1', '1'),
('3', '7', '1');
```

Итак, в нашем магазине 24 наименования товара, привезенные в трех поставках от трех поставщиков, и совершенно три продажи. Все готово, можем приступать к изучению встроенных функций MySQL, чем и займемся в следующем уроке.

Лабораторная работа №11

Тема: Итоговые функции, вычисляемые столбцы и представления

Итоговые функции еще называют статистическими, агрегатными или суммирующими. Эти функции обрабатывают набор строк для подсчета и возвращения одного значения. Таких функций всего пять:

- AVG() Функция возвращает среднее значение столбца.

- COUNT() Функция возвращает число строк в столбце.

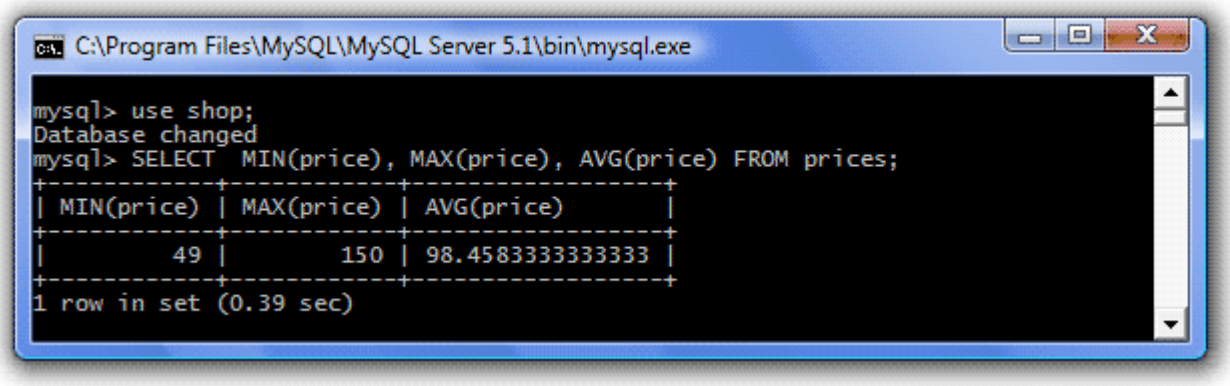
- MAX() Функция возвращает самое большое значение в столбце.

- MIN() Функция возвращает самое маленькое значение в столбце.

- SUM() Функция возвращает сумму значений столбца.

С одной из них - COUNT() - мы уже познакомились в уроке 8. Сейчас познакомимся с остальными. Предположим, мы захотели узнать минимальную, максимальную и среднюю цену на книги в нашем магазине. Тогда из таблицы Цены (prices) надо взять минимальное, максимальное и среднее значения по столбцу price. Запрос простой:

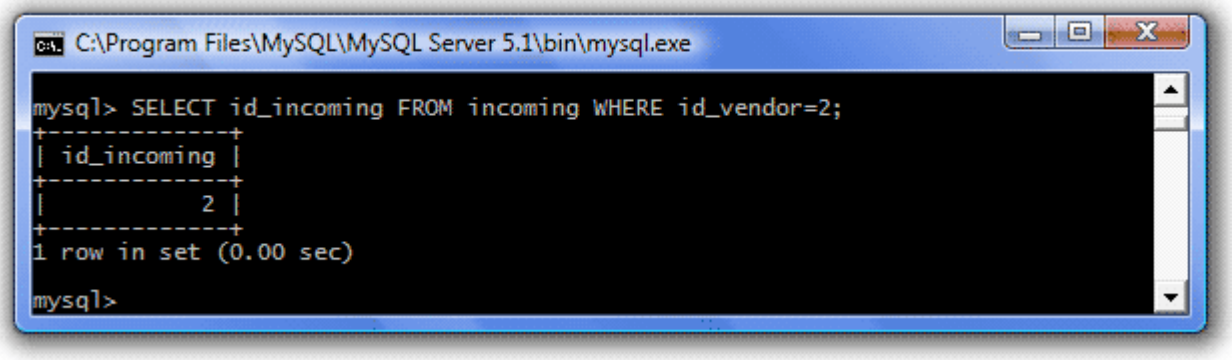
```
SELECT  MIN(price), MAX(price), AVG(price) FROM prices;
```



Теперь, мы хотим узнать, на какую сумму нам привез товар поставщик "Дом печати" (id=2). Составить такой запрос не так просто. Давайте поразмышляем, как его составить:

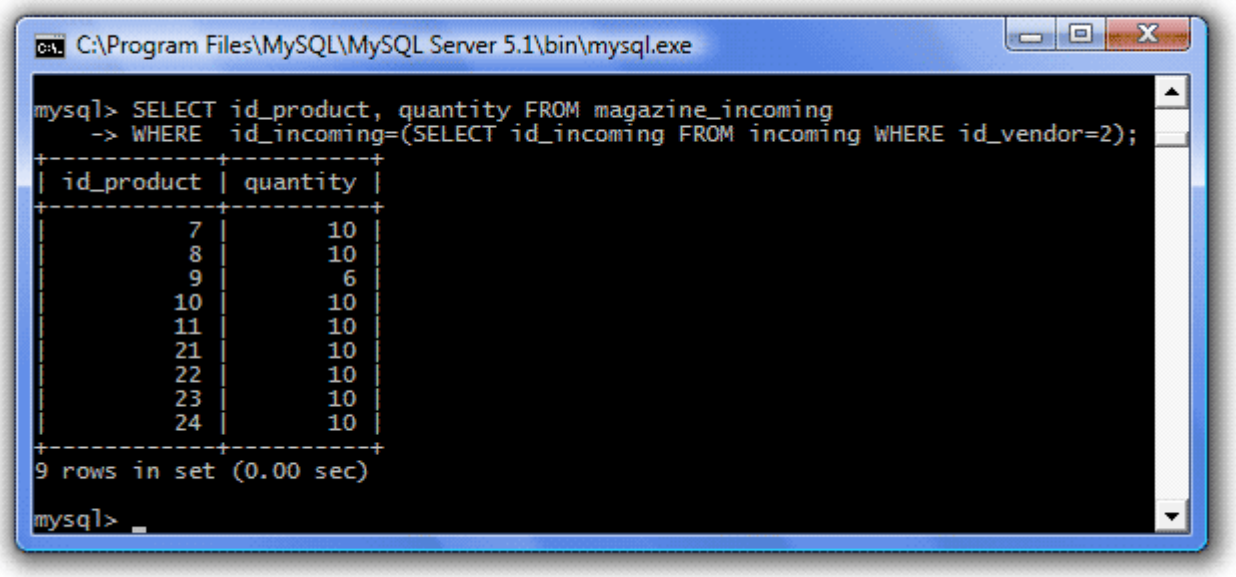
1. Сначала надо из таблицы Поставки (incoming) выбрать идентификаторы (id_incoming) тех поставок, которые осуществлялись поставщиком "Дом печати" (id=2):

```
SELECT id_incoming FROM incoming
WHERE id_vendor=2;
```



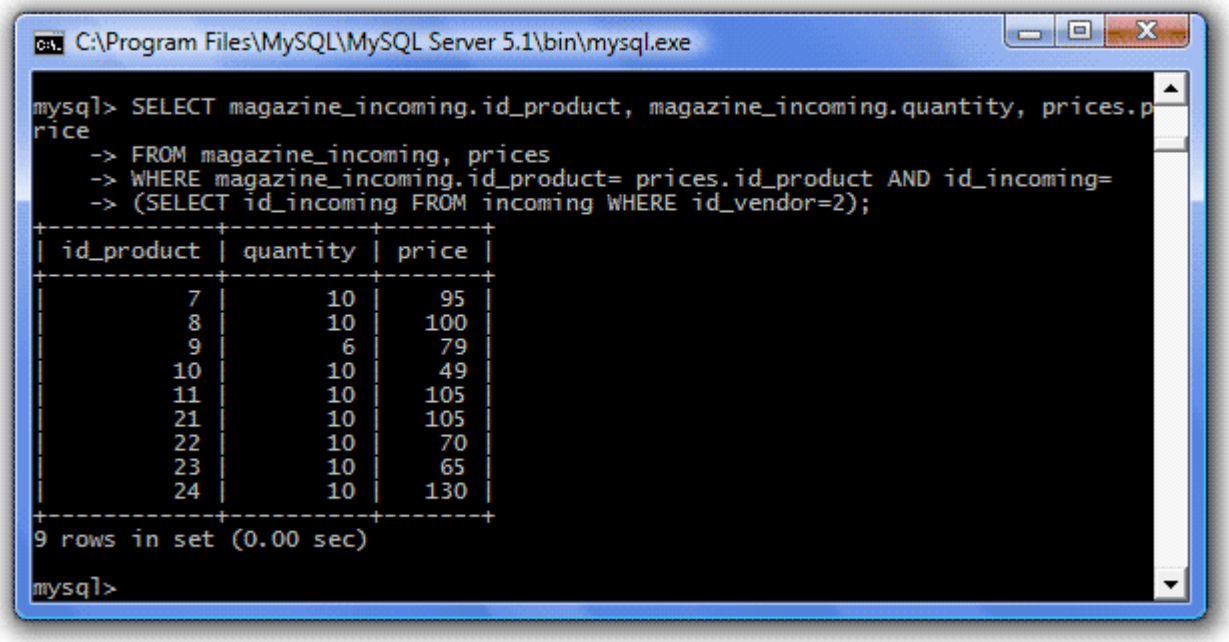
2. Теперь из таблицы Журнал поставок (magazine_incoming) надо выбрать товары (id_product) и их количества (quantity), которые осуществлялись в найденных в пункте 1 поставках. То есть запрос из пункта 1 становится вложенным:

```
SELECT id_product, quantity FROM magazine_incoming
WHERE  id_incoming=(SELECT id_incoming FROM incoming WHERE id_vendor=2);
```



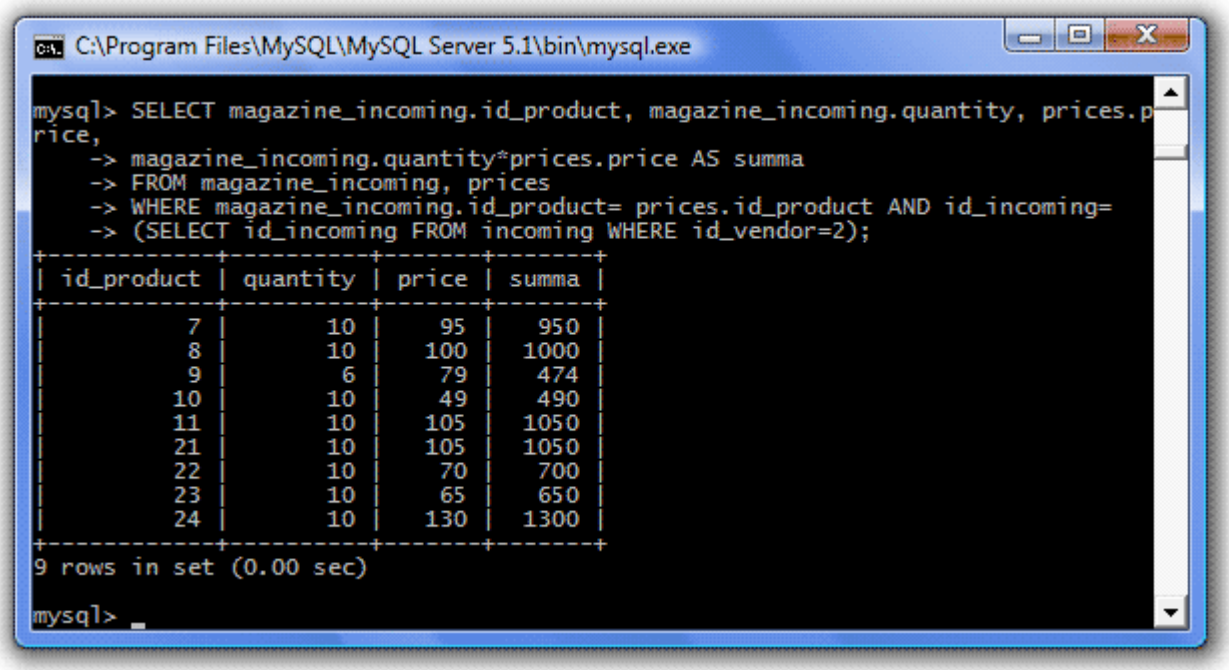
3. Теперь нам надо добавить в результирующую таблицу цены на найденные товары, которые хранятся в таблице Цены (prices). То есть нам понадобится объединение таблиц Журнал поставок (magazine_incoming) и Цены (prices) по столбцу id_product:

```
SELECT magazine_incoming.id_product, magazine_incoming.quantity, prices.price
FROM magazine_incoming, prices
WHERE magazine_incoming.id_product= prices.id_product AND id_incoming=
  (SELECT id_incoming FROM incoming WHERE id_vendor=2);
```



4. В получившейся таблице явно не хватает столбца Сумма, то есть вычисляемого столбца. Возможность создания таких столбцов предусмотрена в MySQL. Для этого надо лишь указать в запросе имя вычисляемого столбца и что он должен вычислять. В нашем примере такой столбец будет называться summa, а вычислять он будет произведение столбцов quantity и price. Название нового столбца отделяется словом AS:

```
SELECT magazine_incoming.id_product, magazine_incoming.quantity, prices.price,
magazine_incoming.quantity*prices.price AS summa
FROM magazine_incoming, prices
WHERE magazine_incoming.id_product= prices.id_product AND id_incoming=
(SELECT id_incoming FROM incoming WHERE id_vendor=2);
```



5. Отлично, нам осталось лишь просуммировать столбец summa и наконец-то узнаем, на какую сумму нам привез товар поставщик "Дом печати". Синтаксис для использования функции SUM() следущий:

```
SELECT  SUM(имя_столбца) FROM имя_таблицы;
```

Имя столбца нам известно - summa, а вот имени таблицы у нас нет, так как она является результатом запроса. Что же делать? Для таких случаев в MySQL существуют Представления. Представление - это запрос на выборку, которому присваивается уникальное имя и который можно сохранять в базе данных, для последующего использования.

Синтаксис создания представления следующий:

```
CREATE VIEW имя_представления AS запрос;
```

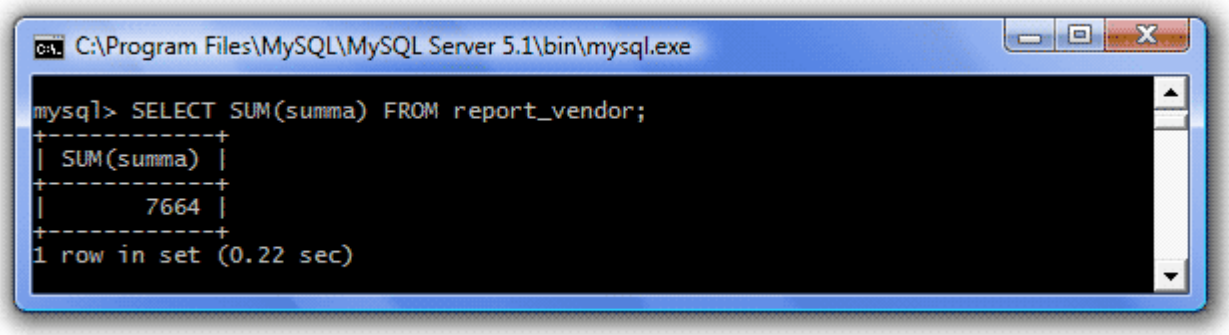
Давайте сохраним наш запрос, как представление с именем report_vendor:

```
CREATE VIEW report_vendor AS
SELECT magazine_incoming.id_product, magazine_incoming.quantity, prices.price,
```

```
magazine_incoming.quantity*prices.price AS summa
FROM magazine_incoming, prices
WHERE magazine_incoming.id_product= prices.id_product AND id_incoming=
(SELECT id_incoming FROM incoming WHERE id_vendor=2);
```

6. Вот теперь можно использовать итоговую функцию SUM():

```
SELECT SUM(summa) FROM report_vendor;
```



Вот мы и достигли результата, правда для этого нам пришлось использовать вложенные запросы, объединения, вычисляемые столбцы и представления. Да, иногда для получения результата приходится подумать, без этого никуда. Зато мы коснулись двух очень важных тем - вычисляемые столбцы и представления. Давайте поговорим о них поподробнее.

Вычисляемые поля (столбцы)

На примере мы рассмотрели сегодня математическое вычисляемое поле. Здесь хотелось бы добавить, что использовать можно не только операцию умножения (*), но и вычитание (-), и сложение (+), и деление (/). Синтаксис следующий:

```
SELECT имя_столбца_1, имя_столбца_2, имя_столбца_1*имя_столбца_2 AS
имя_вычисляемого_столбца
```

```
FROM имя_таблицы;
```

Второй нюанс - ключевое слово AS, мы его использовали для задания имени вычисляемого столбца. На самом деле с помощью этого ключевого слова задаются псевдонимы для любых столбцов. Зачем это нужно? Для сокращения и читаемости кода. Например, наше представление могло бы выглядеть так:

```
CREATE VIEW report_vendor AS
SELECT A.id_product, A.quantity, B.price, A.quantity*B.price AS summa
FROM magazine_incoming AS A, prices AS B
WHERE A.id_product= B.id_product AND id_incoming=
(SELECT id_incoming FROM incoming WHERE id_vendor=2);
```

Согласитесь, что так гораздо короче и понятнее.

Представления

Синтаксис создания представлений мы уже рассматривали. После создания представлений, их можно использовать так же, как таблицы. То есть выполнять запросы к ним, фильтровать и сортировать данные, объединять одни представления с другими. С одной стороны это очень удобный способ хранения частоприменяемых сложных запросов (как в нашем примере).

Но следует помнить, что представления - это не таблицы, то есть они не хранят данные, а лишь извлекают их из других таблиц. Отсюда, во-первых, при изменении данных в таблицах, результаты представления так же будут меняться. А во-вторых, при запросе к представлению происходит поиск необходимых данных, то есть производительность СУБД снижается. Поэтому злоупотреблять ими не стоит.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
КЫРГЫЗСКОЙ РЕСПУБЛИКИ

КЫРГЫЗ РЕСПУБЛИКАСЫНЫН
БИЛИМ БЕРҮҮ ЖАНА ИЛИМ МИНИСТРЛИГИ
ОШ МАМЛЕКЕТТИК УНИВЕРСИТЕТИ
МАТЕМАТИКА ЖАНА ИНФОРМАЦИЯЛЫК ТЕХНОЛОГИЯЛАР
ФАКУЛЬТЕТИ
АССТ КАФЕДРАСЫ

«Бекитилди»
АССТ кафедрасынын
2020-жылдын 27-августунда
өткөрүлгөн
№1-протоколунда
Каф. башч., доц.: Молдояров У.

БААЛОО КАРАЖАТТАРЫНЫН ФОНДУ

Дисциплина: Берилгендер базасы

Багыты: « ССИБИ »

Окутуунун формасы: Күндүзгү

Окуу жылы: 2020-2021

Түзгөн: улук окутуучу _____ Г.И. Сейитказыева

Ош – 2020

БААЛОО КАРАЖАТТАРЫНЫН ФОНДУНУН ПАСПОРТУ

“Берилгендер базасы” дисциплинасы боюнча учурдагы жана аралык текшерүүнү, экзаменди уюштуруу үчүн баалоо каражаттарынын фонду НББПнын негизинде түзүлгөн жумушчу программага ылайыкталып түзүлдү.

“Берилгендер базасы” дисциплинасы боюнча баалоо каражаттарынын фонду билим берүүнүн талаптарына ылайык студенттердин ээ болгон билимдерин, билгичтиктерин, көндүмдөрүн, компотенциялардын калыптануу деңгээлин текшерүүгө арналган.

№№ п/п	Дисциплинанын текшерилүүчү бөлүмдөрү	Компетенциялардын коду	Баалоо каражатынын аталышы
1	СУБД жөнүндө түшүнүк.MySql программалоо чөйрөсү	ОК-3, ИК-1	Суроо, тест
2	MySql тилинин операторлору	ПК-10, ПК-12	Суроо, тест
3	My sql тилинде функиялардын колдонулуштары	ПК-5, ПК-12	Суроо, тест
4	Сакталаган процедуралар жана функциялар.	ПК-5,ПК10	Суроо, тест

“Берилгендер базасы” дисциплинасы боюнча баалоо каражаттарынын фонду өз ичине төмөнкүлөрдү камтыйт:

1.Учурдагы текшерүүнү уюштуруу үчүн баалоо каражаттары:

-Теориялык алган билимдерин жана атайын терминдерди туура колдоно билгичтиктерин баалоо үчүн лабораториялык тапшырмалардын топтому

- Стандарттык билимдердин деңгээлин текшерүү үчүн тесттер.

2. Аралык текшерүүнү уюштуруу үчүн баалоо каражаттары:

- Экзаменди жүргүзүү үчүн жыйынтыктоочу текшерүүчү практикалык тапшырмалар, тесттер.

“Берилгендер базасы” дисциплинасын окуп үйрөнүүдө калыптануучу компотенциялар.

ОК-3: Заманбап билим берүүчүлүк жана маалыматтык технологияларын пайдаланып өз алдынча жаңы билимдерге ээ болууга жөндөмдүү.

ПК-5:Медициналык маалыматтар системаларынын жана берилгендер базасынын абалын жана өнүгүүсүн анализдөөгө жөндөмдүү

Билет	**Колдоно алат**	**Ээ болот**
MySql каражатында берилгендер базасын түзүүнү билет	Маалыматтарды берилгендер базасын түзүүдө колдоно алат	Берилгендер базасын түзүү ыкмаларына ээ болот

ИК-1: Максатты коюу, ага жетүү жолдорун, маалыматты анализдөөгө, кабылдоого жөндөмдүү.

Билет	**Колдоно алат**	**Ээ болот**
Таблицалар алардын арсындагы байланыштарды түзүүнү билет	Өзү кызыкан чөйрөнүн маалыматтарын кайра иштетүүгө колдоно алат	Практикалык маселелерди чечүү көндүмдөрүнө ээ болот

ПК-12:Медициналык маалыматтар системасына жетүүнү башкарууну билет

Билет	**Колдоно алат**	**Ээ болот**

Алган билимин өндүрүштүк маселелерде колдонулушун билет	Техникалык маселелерди туура **колдоно алат.**	Проектирлөө методдорун **ээ болот**

Тесттирлөөнү баалоо критерийлери

Тестке берилчү баллдар	**Баалоо критерийи**
Бир туура жоопко 1 балл	*Жооптун туура тандалган варианты*

УЧУРДАГЫ ТЕКШЕРҮҮ ҮЧҮН БААЛОО КАРЖАТТАРЫ

«Берилгендер базасы» предмети боюнча тесттик суроолор

1. СУБД деген эмнени тушундурот?
    1. берилгендер базасынын обьектерин түзүүчү программалык камсыздоо
    2. берилгендер базасын түзүү, өзгөртүү аныктоо жана ага жетүүнү башкарууга мүмкүндүк берүүчү программалык камсыздоо
    3. берилгендерди жаңылоо, өчүрүү, берилгендер базасына жүргүзүлгөн запросторду аткаруучу программалык камсыздоо
2. Кайсыл команданын жырдамында берилгендер базасынын обьектеринин структурасын түзүп, өзгөртүп, өчүрүүгө болот?
    a. DDL
    b. Alter table
    c. Creat table
3. UPDATE операторунун кызматы?
    a. талаанын жолчосуна атайын көрсөтулбөгөн учурда туруучу маанини берүүгө болот
    b. таблицага кийирилген маалымааттарды көрүү үчүн колдонулат
    c. таблицвнын жазууларындагы маанилерди өзгөртүү үчүн колдонулат
4. Берилгендер базасында түзүлгөн талаанын индекси?

a. AUTO_INCREMENT
b. NOT NULL
c. PRIMARY KEY

5. ASC параметри менен DESC параметринин айырмасы кандай?
   a. ASC кичине мааниден чоңго карай өсүү тартибинде DESC чоңдон кичине маанига карай кемүү тартибинде сортировкалайт
   b. ASC чоңдон кичине маанини карай кемүү тартибинде DESC кичине мааниден чоңго карай өсүү тартибинде сортировкалайт
   c. билбейм
6. Таблицадагы талаанын атын жана тибин өзгөртүү үчүн кайсыл операторду колдонобуз?
   a. UPDATE
   b. SELECT*from
   c. Change
7. SHOW операторунун жыйынтыгы эмнени берет?
   a. берилгендер базасын жана таблицаларды өчүрөт
   b. берилгендер базасынын атын өзгөртөт
   c. берилгендер базасын же таблицалардын тизмесин көргөзөт
8. Байланыштардын типтерин көрсөт?
   a. Бирден-бирге
   b. Бирден көпкө, көптөн бирге
   c. Көптөн-көпкө

9. Кайсы конструкциянын жардамында системада, канча берилгендер базасы бар

   экендигин текшерүүгө болот?
   a. Count databases
   b. Check databases
   c. Show databases

10. MySQLде төмөнкүлөрдөн кайсы тип жок?

a. Char
b. Enum
c. String

11. MySQLде кайсы оператор өчүрүү үчүн колодонулат?
    a. Remove
    b. Droup
    c. Collapse

12. MySQLдин башка СУБДлардан артыкчылыгы?
    a. Тармака туташуу мүмкүнүчүлүгү
    b. Интерфейси
    c. Операторлору

13. MySQLде сортировкалоо оператолору?
    a. Select, describe
    b. Show, alter
    c. Asc desc

14. MySQLде таблицага талаа кошуу үчүн кайсы оператор колдонулат?
    a. Add
    b. Decimal
    c. Select

15. Select оператору кайсы учурда колдонулат?
    a. Өчүрүү үчүн
    b. Таблицага киргизилген маанилерди көрүү үчүн
    c. Берилгендер базасын активдештирүү үчүн

16. SQL тилинин жардамында Pasport таблицасындагы FirstName талаасынын мааниси “Eliza” болгон жолчону кантип өчүрүүгө болот?
    a. DELETE Row FirstName=’Eliza’ FROM Pasport
    b. DELETE FirstName=’Eliza’ FROM Pasport
    c. DELETE FROM Pasport WHERE FirstName=’Eliza’

17. Pasport таблицасындагы FirstName талаасына карап кемитилген тартипте сорттолгон жазууларды кантип алып чыга алабыз?

a. SELECT*FROM Pasport SORT ‘FirstName’ DESC
b. SELECT*FROM Pasport ORDER BY FirstName DESC
c. SELECT*FROM Pasport ORDER BY ‘FirstName’ DESC

18. SQL аббревиатурасы эмнени түшүндүрөт?

a. Strong Question Language
b. Structured Question Language
c. Structured Query Language

19. SQL тилинин жардамында Pasport таблицасынын Adres талаасына ‘Kyrgyzstan,Osh’ маанисин кантип кийиребиз?

a. INSERT INTO Pasport(‘Kyrgyzstan,Osh’) INTO Adres
b. INSERT INTO Pasport(Adres) Values(‘Kyrgyzstan,Osh’)
c. INSERT (‘Kyrgyzstan,Osh’) INTO Pasport(Adres)

20. Work таблицасындагы Major талаасынын ‘Programming’ маанисин ‘Translater’ маанисине кантип өзгөртүүгө болот?

a. MODIFY Work SET Major=’Programming’ TO ‘Translater’
b. MODIFY Work SET Major=’Translater’ WHERE=‘Programming’
c. UPDATE Work SET Major=’Translater’ WHERE=‘Programming’

21. Биз түзгөн Info_person берилгендер базасын кантип өчүрөбүз?

a. DROP Info_person
b. DROP DATABASES Info_person
c. DELETE Info_person

22. Pasport таблицасындагы Age талаасынын 25 ден 50 жашка чейинки маанилерин чыгаруу буйругу кандай жазылат?

a. SELECT * FROM Pasport WHERE Age>25 AND Age<50
b. SELECT * FROM Pasport WHERE BETWEEN 25 AND 50
c. SELECT FROM Pasport WHERE BETWEEN 25 AND 50

23. MySQLде кайсы циклдик операторлор колдонулат?

a. While, repeat, loop;
b. While, repeat, for
c. While, repeat, if

24. Учурдагы датаны сааты кайсы команданын жардамыда коро алабыз?

a. CURDATE(), CURTIME(),NOW();
b. NOWDATE(),NOWTIME;
c. CURENT_DATE(), CURENT_TIME();

25. AVG() функциясынын кызматы кандай?

a. Таблицадагы жазуулардын орточо маанисин чыгарып берет;
b. Талаанын жазууларынын орточо маансин чыгарып берет;
c. Эки вариант тен туура;

26. Мамычанын тибин озгортуу учун кайсы операторду колдонобуз жана анын синтаксиси кандай жазылат?

a. **Modify** операторунун жардамында, **ALTER TABLE** *таблицанын аты* **MODIFY** *мамычанын аты жаны тип*;
b. **Modify** операторунун жардамында, **ALTER TABLE** *таблицанын аты* **MODIFY** *жолчонун аты жаны тип*;
c. **Modify** операторунун жардамында, **ALTER TABLE** *берилгендер базасынын аты* **MODIFY** *мамычанын аты жаны тип*;

27. Group By операторунун кызматы ……

a. SQL тилинде жазууларды групировкалоо б.а таблицанын ичиндеги мамылчалардагы жазууларды групалаштыруу ;
b. SQL тилинде таблицаларды групировкалоо б.а таблицанын ичиндеги жазууларды групалаштыруу ;

28. Сырткы таблицаларды бири-бири менен байланыштыруунун синтаксиси кандай жазылат?

a. **SELECT** *1-таб_аты.мамычанын_аты, 2-таб_аты.мамычанын аты* **FROM** 1-*таб_аты байланыштыруунун тиби 2-таб_аты* **ON** *байланышуунун шарты*;
b. **SELECT** *1-таб_аты.мамычанын_аты, 2-таб_аты.мамычанын аты* **FROM** 1-*таб_аты байланыштыруунун тиби 2-таб_аты* **WHERE** *байланышуунун шарты*;
c. **SELECT** *1-таб_аты.мамычанын_аты, 2-таб_аты.мамычанын аты* **FROM** 1-*таб_аты байланыштыруунун тиби 2-таб_аты* ;

29.SQL аббревиатурасы эмнени түшүндүрөт?

a. Strong Question Language
b. Structured Question Language
c. Structured Query Language

30.Биз түзгөн Info_person берилгендер базасын кантип өчүрөбүз?

a. DROP Info_person
b. DROP DATABASES Info_person
c. DELETE Info_person

31. UPDATE операторунун кызматы?

а) талаанын жолчосуна атайын көрсөтулбөгөн учурда туруучу маанини берүүгө болот

б) таблицага кийирилген маалымааттарды көрүү үчүн колдонулат

в) таблицвнын жазууларындагы маанилерди өзгөртүү үчүн колдонулат

32. ASC параметри менен DESC параметринин айырмасы кандай?

а) ASC кичине мааниден чоңго карай өсүү тартибинде DESC чоңдон кичине мааниге карай кемүү тартибинде сортировкалайт

б) ASC чоңдон кичине маанини карай кемүү тартибинде DESC кичине мааниден чоңго карай өсүү тартибинде сортировкалайт

в) билбейм

33. Таблицадагы талаанын атын жана тибин өзгөртүү үчүн кайсыл операторду колдонобуз?

а) UPDATE

б) SELECT*from

в) Change

34. SHOW операторунун жыйынтыгы эмнени берет?

а) берилгендер базасын жана таблицаларды өчүрөт

б) берилгендер базасынын атын өзгөртөт

в) берилгендер базасын же таблицалардын тизмесин көргөзөт

35. Байланыштардын типтерин көрсөт?

а) Бирден-бирге, Бирден көпкө, көптөн бирге, Көптөн-көпкө

б) Бирден-бирге, Бирден көпкө,

в) билбейм

36. AUTO_INCREMENT атрибутунун функциясы кандай?

a) Ачкычтык талаа түзүү мүмкүндүгүн берет

b) Уникалдуу катар номерди автоматтык түрдө коёт

c) Таблица түзөт

37. DESCRIBE операторунун кызматы-

a) Таблицаны конструктор абалында (структурасын) чагылтат

b) Таблицага жаңы талаа кошуу мүмкүнчүлүгүн берет

c) Таблицадагы маалыматтарды толугу менен чагылтат

38. mysql программасы үчүн команда киргизилип жатканда регистр мааниге ээби?

a. Ооба

b. Жок

c. Билбейм

39. FOREIGN KEY –

a) Ички ключ

b) Сырткы ключ

c) Ачкычтык талаа

40. Таблицадагы талаанын атын өзгөртүүчү команданы тандагыла;

a) ALTER TABLE table1 Change tuulgan_jyly tuulgan datasy varchar (10);

b) INSERT INTO table1 Change tuulgan_jyly tuulgan datasy varchar (10);

c) ALTER TABLE table1 drop tuulgan_jyly tuulgan datasy varchar (10);

41. MySQLде таблицага талаа кошуу үчүн кайсы оператор колдонулат?

a) Add

b) Decimal

c) Select

42. Select оператору кайсы учурда колдонулат?

a) Өчүрүү үчүн

b) Таблицага киргизилген маанилерди көрүү үчүн

c) Берилгендер базасын активдештирүү үчүн

43. Берилгендер базасын түзүүнүн синтаксиси

а)Create database Базанын_аты;

б) use database базанын_аты;

в)create базанын_аты database;

44. Учурда активдүү берилгендер базасын билүү үчүн.....

а) use database;

б) SELECT database ();

в)show databases;

45. Таблицаны кайсы оператордун жардамында өчүрөбүз?

а) SELECT table;

б)Drop databases;

в) Drop table;

46. DESCRIBE- операторун.....

а) База же таблицанын тизмесин үчүн колдонобуз .

б)таблицанын структурасын көрүү үчүн колдонобуз.

в) таблицанын структурасын өзгөртүү үчүн колдонобуз

47. NOT NULL- .....

а) Талаа бүтүн сандарды кабыл алат

б) MySQLдин атайын атрубуту.

в)Бул тип берилгенде талаанын ар бр жолчосу толтурулууга тийиш, жолчолор бош маанини кабыл албайт.

48. Таблицаны өзгөртүү үчүн кайсы операторду колдонобуз?4

а) ALTER TABLE.

б)ADD tabble

в)DESCRIBE

Студенттердин өз алдынча иштери

Студенттердин өз алдынча иштери пландоодо жана тапшырмаларды даярдоодо Блумдун 6 деңгээлдүү таксономиясы (билүү, түшүнүү, колдонуу, анализ, синтез, баалоо) колдонулат.

9.3.1. Билүү, түшүнүү жана колдонуу үчүн берилген тапшырмалар

32. Берилгендердин иерархиялык моделинин өзгөчөлүктөрү
33. Берилгендердин тармактык моделинин өзгөчөлүктөрү
34. Таблица түзүү.
35. Таблицанын ичин толтуруу, толтуруунун жолдору
36. MySQLдин структурасы
37. Клиент-сервердик архитектура
38. Структура дистрибутивдердин структурасы
39. Официалдуу документация

9.3.2. Анализдөө жана синтездөө үчүн берилген тапшырмалар

40. MySQLди орнотуу
41. MySQLдин иш жөндөмдүүлүгүн текшерүү
42. MySQLди Linux ОСна орнотуу
43. Конфигурациялык файл
44. Берилгендердин каталогун жылдыруу

45. Учрдагы версиясын жаңылоо
46. Обзор утилит MySQL утилитасынын обзору
47. *mysql утилитасы*
48. Командалык жолчо
49. Кодировканы ылайыктоо
50. *SELECT* и *LOAD DATA операторлорунун колдонулушу*
51. BACKUP TABLE *и* RESTORE TABLE

9.3.3. Баалоо үчүн берилген тапшырмалар

52. Таблицаларды түзүү жана маалыматтар менен толтуруу жолдору
53. Таблицалардын ортосундагы байланыштарды түзүү
54. Бирден –бирге байланышы
55. Бирден – көпкө байланышы
56. Көптөн –көпкө байланышы
57. SELECT-операторун колдонулушу
58. Функциялардын колдонулушу
59. IF… THEN… ELSE…операторлору
60. Математикалык функциялар
61. Дата жана время функциялары
62. Жолчолук функциялар

Студенттер өз алдынча тапшырмаларды төмөндөгү методдордун бири менен коргойт (кафедрадагы кезекчилик мезгилинде, сабактан кийин, ишемби күнү, модулдук жумада):

- Презентация;
- Реферат;
- Оозэки баяндоо;

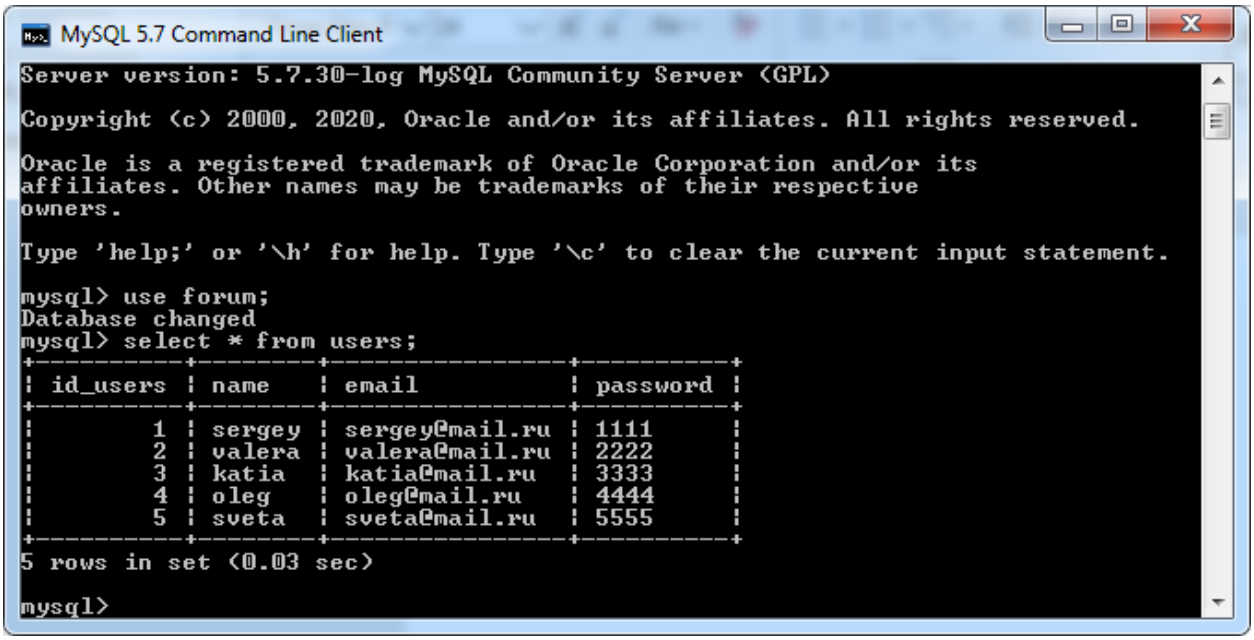
Практикалык курс үчүн тапшырмалар

1. ITAS_kafedrasy аталышындагы берилгендер базасын түзгүлө

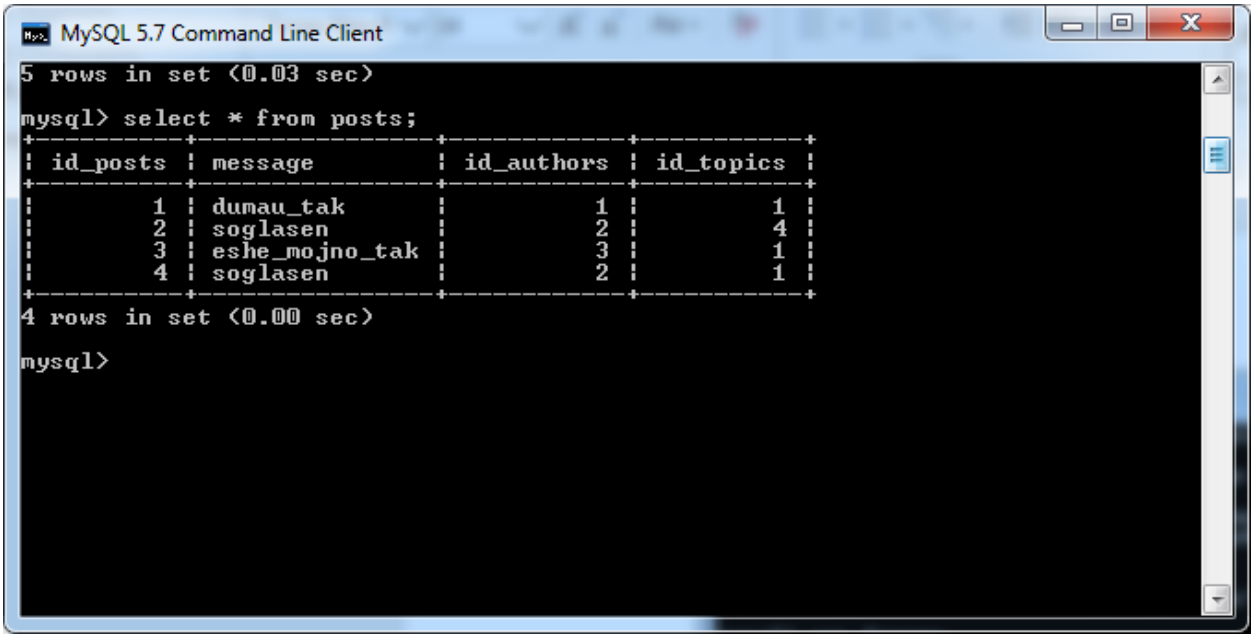
2. Бул берилгендер базасына Okutuuchular аталышындагы таблица түзгүлө. Таблица төмөнкү талаалардан турсун.

   Kod Familia Imia Staj doljnost

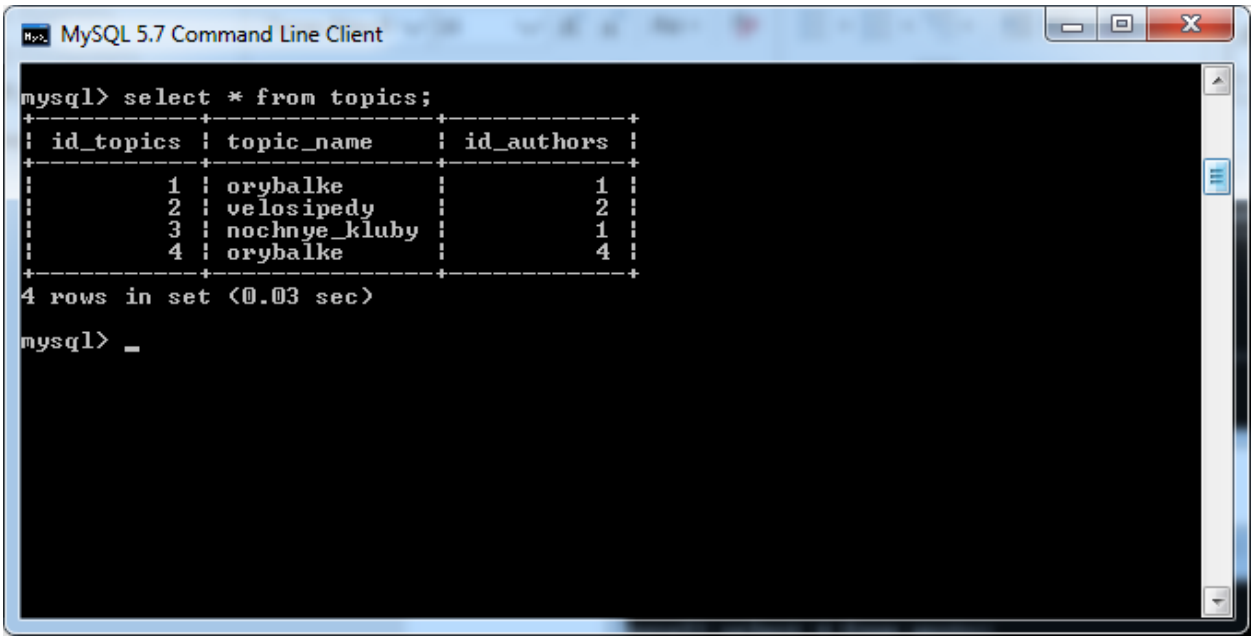
3. Okutuuchular таблицасына data_rojdenia талаасын кошкула.
4. Okutuuchular таблицасынан doljnost талаасын өчүрүп салгыла
5. Staj талаасын zarplata талаасына өзгөрткүлө
6. **Users** таблицасынын **name** талаасын өсүү тартибинде сортировкалоонун кодун жазгыла.
7. **Users** таблицасынын **password** талаасын кемүү тартибинде сортировкалагыла



8. **Posts** таблицасында **id_authors** талаасын группировкалоонун коду кандайча жазылат?
9. **Users** жана **posts** таблицаларынын **name, message** талааларын бириктирүүнүн кодун жазгыла.



10. **Topics** таблицасынын каалаган эки талаасынан турган виртуалдык таблица түзгүлө.
11. **Union** операторуна бир мисал келтиргиле



12. **Кардиология** аталышындагы берилгендер базасын түз жана ага 5 талаадан турган **Врачтар** таблицасын түз. Таблицанын ичин толтуруп, **Туулган жылы** талаасы боюнча кемүү тартибинде сортировкала.

13. **Урология** аталышындагы берилгендер баазасын түз. Ага 6 тааладан турган **Бейтаптар** таблицасын түз. Таблицаны толтуруп каалаган 3 талаасынан виртуалдык таблица түз.
14. **Терапия** аталышындагы берилгендер базасын түз жана ага каалаган 2 таблица түз. **UNION** операторунун жардамында 2 **SELECT** операторун бириктир.
15. **Неврология** аталышындагы берилгендер базасын түз жана ага каалаган эки таблица түзүп алардын талааларынан ички бириктирүүнү жүргүз.
16. **Аптека** аталышындагы берилгендер базасын түз ага **Даарылар** аталышындагы таблица түз. Баасы талаасын 5 ке көбөйтүп жыйынтыкты чыгар.

Бланкалык же компьютердик тестирлөө.

10. *Жыйынтыктоочу экзамендеги тапшырмалар (үлгү)*

53. Берилгендер базасын түзүү
54. Берилгендер базасын тандоо
55. Таблица түзүү
56. Берилгендердин сандык тиби
57. Берилгендердин жолчолук тиби
58. Берилгендердин календардык тиби
59. NULL берилгендер тиби
60. Таблицанын структурасын көрүү
61. Мамычалардын параметлери
62. Берилгендер базасынын, таблицалардын талаалардын мүмкүн болгон аттары
63. WHERE операторлору, мисалдар
64. ORDER BY, GROUP BY операторлору, мисалдар
65. DEFAULT, ALTER TABLE операторлорунун колдонулуштары
66. VIEW, UNION операторлору
67. Таблицанын параметрлери
68. Убактылуу таблицалар

69. Таблицанын көчүрмөсүн түзүү
70. Таблицанын оперативдик эске жайгаштыруу
71. Бир нече таблицаларды бир таблицага бириктирүү
72. Таблицаларды өчүрүү
73. Таблицаны өзгөртүү
74. Талааларды кошу
75. Талааны өчүрүү
76. Бар талаалардын атын өзгөртүү
77. Таблицанын атын өзгөртүү
78. Таблицанын параметрлерин өзгөртүү
79. Таблицаны калыбына келтирүү
80. REPAIR TABLE оператору
81. Таблицанын текшерүүчү суммасы
82. Индекстер
83. Ички жана сырткы ключтөр
84. AUTO_INCREMENT атрибутунун жардамында уникалдуу индексти түзүү
85. Кадимки жана уникалдуу индекстер
86. Таблицага индекстерди кошуу жана өчүрүү
87. Индекстерди калыбына келтирүү
88. *INSERT опреаторнунун колдонулушу*
89. *TIMESTAMP берилгендер тиби*
90. Сандык маанилерди толтуруу
91. Жолчолук маанилерди толтуруу
92. Календардык маанилерди толтуруу
93. *UNIXSTAMP форматында маанилерди толтуруу*
94. *AUTO_INCREMENT механизми*
95. *Эсептөөчү маанилерди толтуруу*
96. *INSERT... SELECT операторунунун колдонулушу*
97. *Берилгендерди өчүрүү*
98. *DELETE оператору*
99. *TRUNCATE оператору*

100. *Бир нече таблицалардан өчүрүү*
101. *Бир нече таблицалардан каскаддык өчүрүү*
102. *Жазууларды жаёылоо*
103. *CASCADE оператору*
104. *UPDATE көп таблицалуу оператору*

Лабораториялык иштерди баалоону жүргүзүү

Теориялык билимдерин бышыктоо үчүн лабораториялык тапшырмалар берилет. Лабораториялык жумуштарды аткаруу лабораториялык сааттар учурунда жана үйгө берилет.

Лабораториялык жумуштарды баалоо критерийлери

Баалар	Баалоо критерийлери
5 баллов	Тапшырма толук аткарылып, теориялык билими менен бышыкталса.
4 балла	Тапшырма толук аткарылып, негизделбесе, теориялык билими толук болбой калса
3 балла	Тапшырма толук аткарылбай, жарымдан көбү аткарылса, бышыкталбаса.
2 балла	Тапшырма аткарылбаса

Баллдарды топтоонун картасы – сабактардын бардык түрлөрүндөгү текшерүү боюнча канча балл (максималдуу) ала тургандыгы жөнүндө студенттерге жеткирилүүчү маалымат.

Студенттер баллдарды модулдарда төмөнкүдөй топтошот:

1-модулда эки учурдагы текшерүү (УТ1, УТ2) жана бир аралыктагы текшерүү (АТ1) уюштурулат. Ар бир текшерүү үчүн 30 баллдык баалоо системасы колдонулат. Баллдар тапшырмалар менен кошо тааныштырылат.

УТ1 текшерүүсү 4-жумада, УТ2 текшерүүсү 8-жумада уюштурулат, ал эми аралыктагы текшерүү дагы 8-жумада уюштурулат.

УТ1 деп 4-жумага чейин өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $УТ1 = \frac{Лек + Лаб + СӨАИ}{З}$.

УТ2 деп сабак башталгандан баштап 4-жумадан 8-жумага чейин өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $УТ2 = \frac{Лек + Лаб + СӨАИ}{З}$.

Ведомостко жана журналга УТ1, УТ2 лердин жыйынтыктары коюлат.

8-жумада 1-модулдун материалдары боюнча 1-аралыктагы текшерүү уюштурулат. Мында 1-модулда өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз: $АТ1 = \frac{Лек + Лаб + СӨАИ}{З}$.

1-модулда баалоо учурдагы текшерүүлөрдүн жана 1-аралыктагы текшерүүнүн арифметикалык орточосу менен аныкталат: $М1 = \frac{УТ1 + УТ2 + АТ1}{З}$.

2-модулдагы баалоо 1-модулдагы баалоо сыяктуу эле аткарылат.

Жыйынтыктоочу текшерүүдө семестрде ичинде өтүлгөн лекциялык материалдарды өздөштүргөндүгү, аткарылган лабораториялык жана өз алдынча иштер боюнча баалоонун арифметикалык орточосун алабыз:

$$ЖТ = \frac{Лек + Лаб + СӨАИ}{З}$$

Экзамендеги баалоо модулдардын жана жыйынтыктоочу текшерүүнүн арифметикалык орточосу менен сыйлык (***С***) баллдардын суммасы менен аныкталат:

$$Экз = \frac{М1 + М2 + ЖТ}{З} + С.$$

Баллдар тапшырмаларды берүүдө кошо көрсөтүлөт. ***С*** – сыйлык баллдар «Билимди баалоо системасы» жөнүндөгү жободо көрсөтүлгөн.

Балл коюу саясаты

Учурдагы, аралыктагы жана жыйынтыктоочу текшерүүлөр «Билимди баалоо» жөнүндөгү жобо менен аныкталат.

Студенттин билим деңгээли 100 баллдык системада төмөнкү эрежеге ылайык коюлат:

Рейтинг (балл)	**Тамгалык система боюнча баа**	**GPA боюнча баалоонун цифралык эквиваленти**	**Традициялык системе боюнча баа**
87 – 100	A	4,0	Эң жакшы
80 – 86	B	3,33	Жакшы
74 – 79	C	3,0	
68 – 73	Д	2,33	Канааттандыраарлык
61 – 67	E	2,0	
31 -60	FX	0	Канааттандыраарлык эмес
0 - 30	F	0	

Экзаменде бааны коюуда объективдүүлүк жана акыйкаттуулук принциптеринин негизинде студенттин билиминин сапаты бардык тараптан анализделип, модулдук-рейтингдик системанын жобосуна ылайык коюлат.