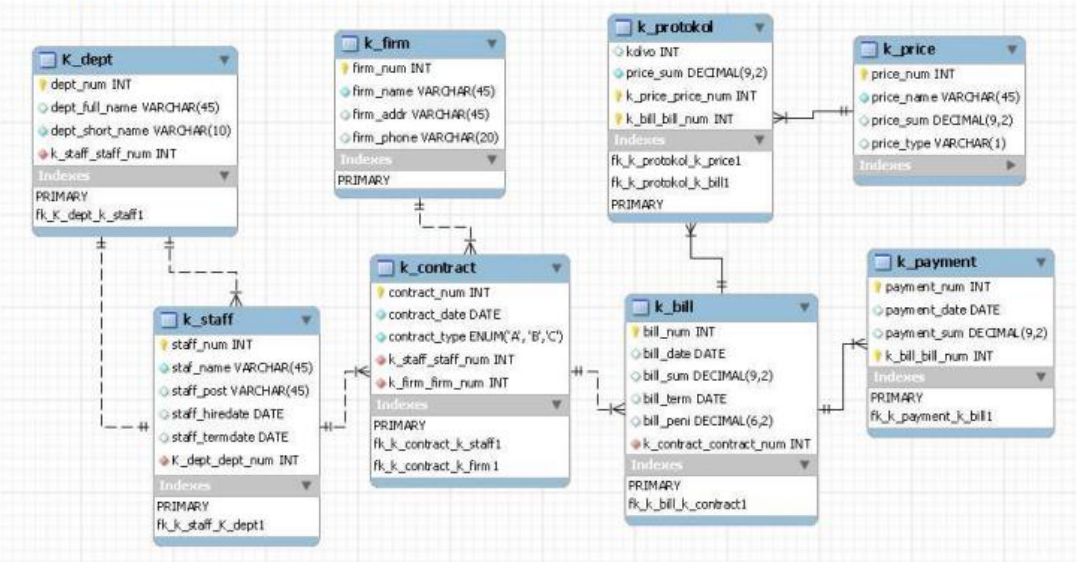
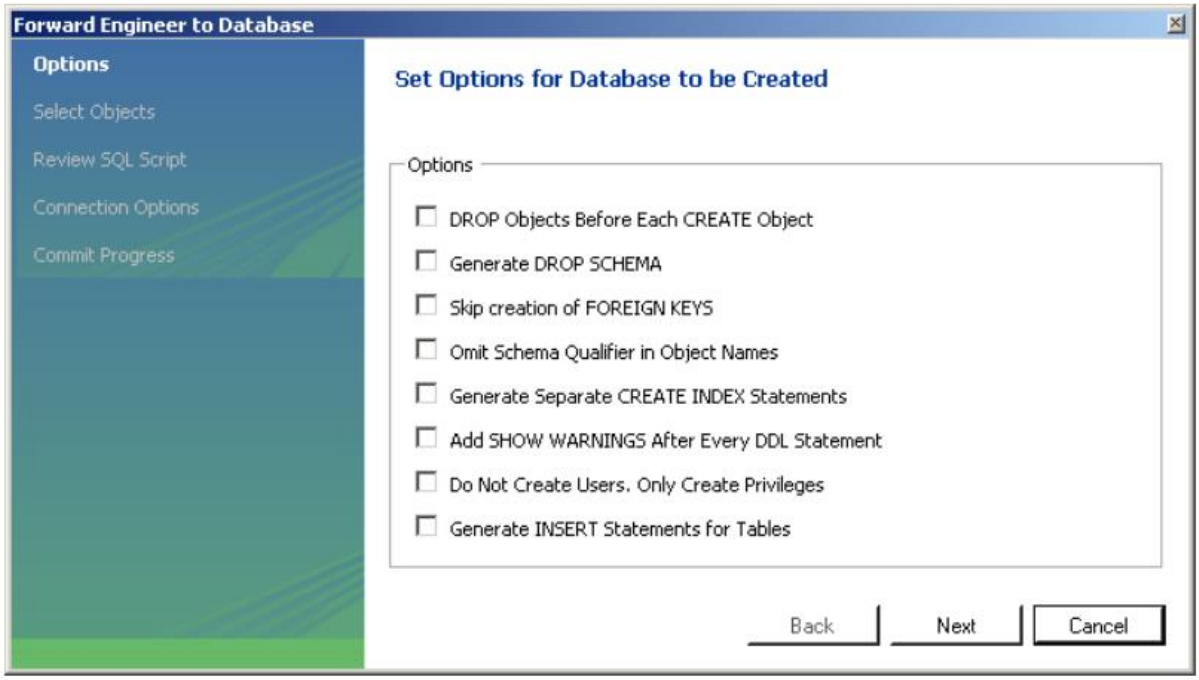
3) Создание базы данных из EER-диаграммы

На предыдущем этапе мы разработали EER-диаграмму для нашей предметной области:



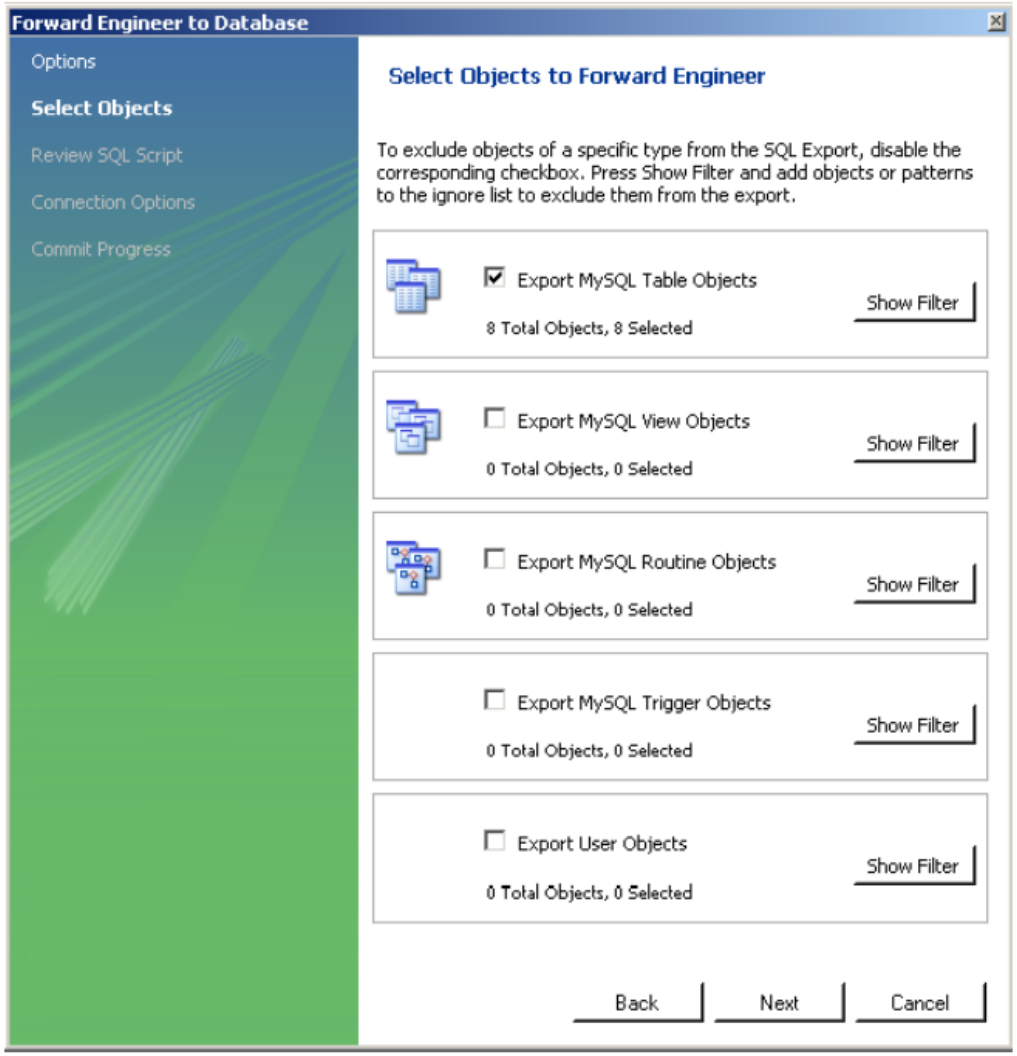
Теперь на основе этой диаграммы создадим физическую базу данных. Выберем пункт меню **Database - Forward Engeneer**. Запустится мастер построения базы данных.

На первом шаге можно задать некоторые дополнительные действия. Для начала ничего на этой странице не выбираем, нажимаем **Next.**

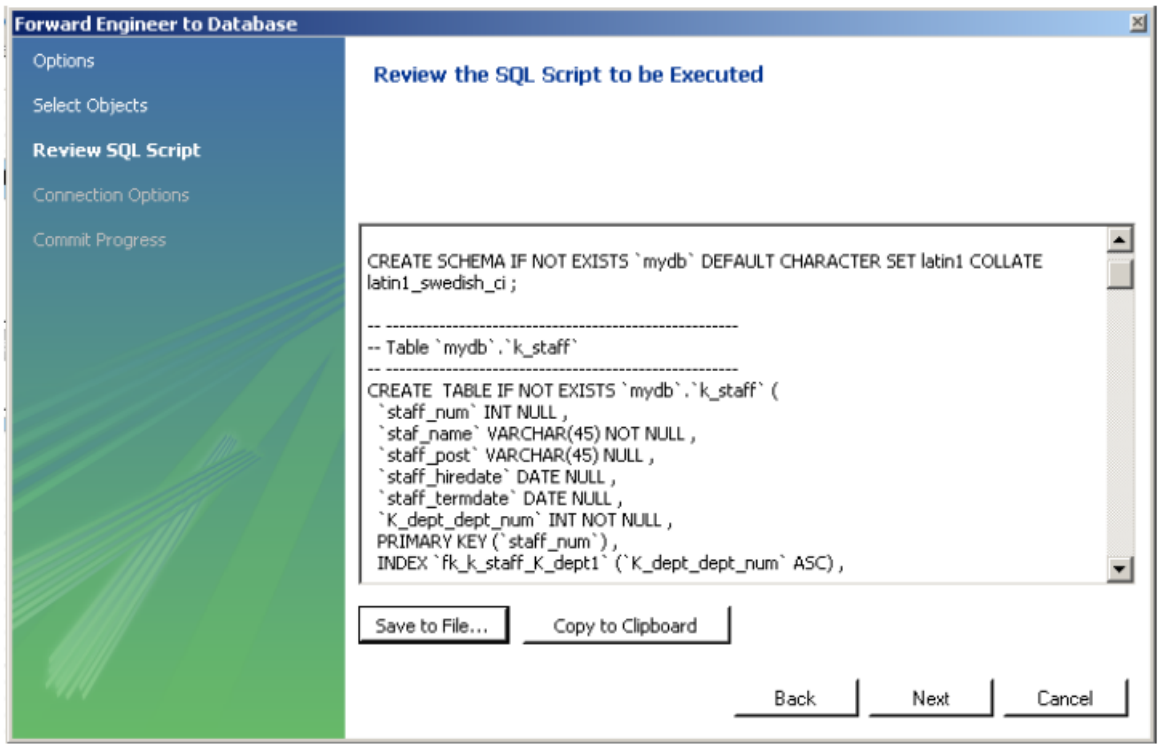


Примечание: при повторном создании базы данных нужно будет включить первые два флажка для удаления старых таблиц.

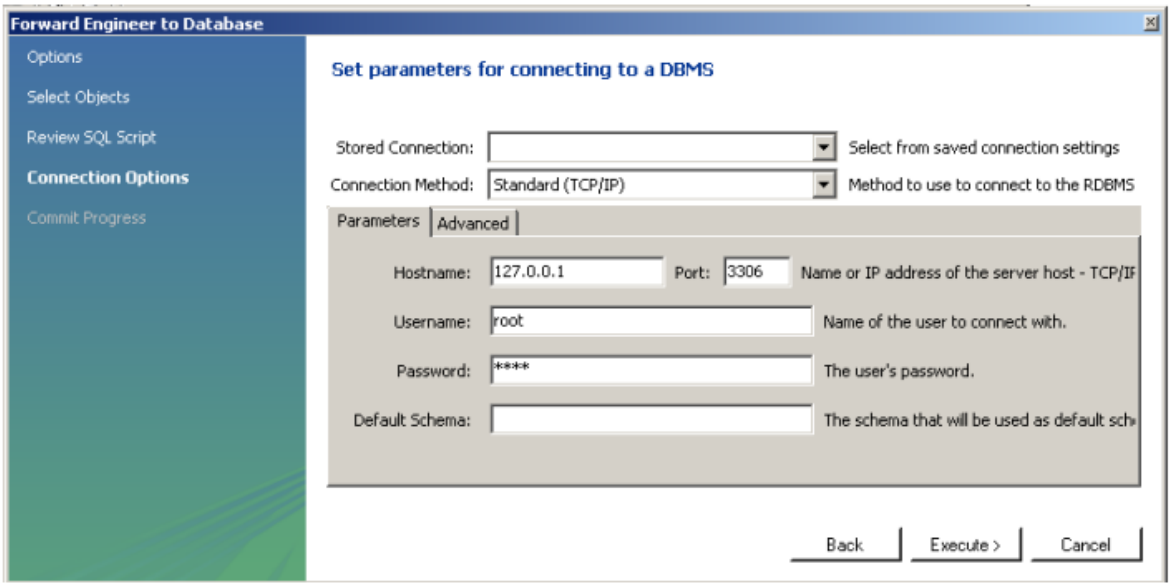
На второй странице включен флажок, указывающий, что мы создаем таблицы (всего 8 шт.). Других объектов пока у нас нет.



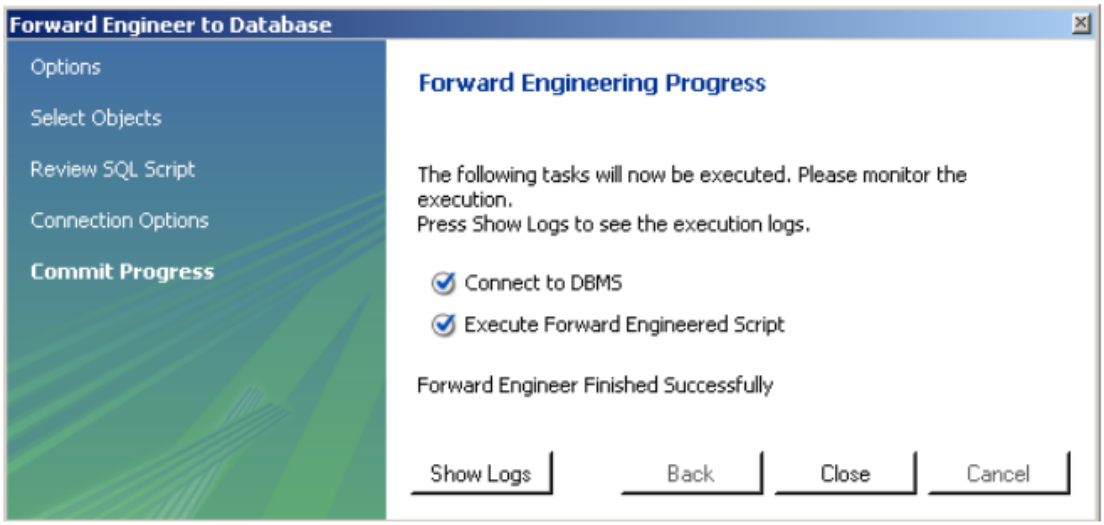
На следующей странице показывается текст сценария для создания базы данных. Его можно просмотреть прямо на месте, а также сохранить в файл (кнопка **Save to file**…).



Далее запрашивается логин и пароль для подключения к серверу:

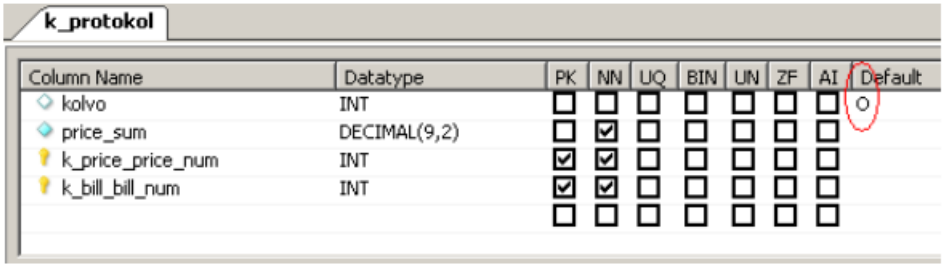


Если нет никаких ошибок, то получим окно с сообщением об успешном результате:

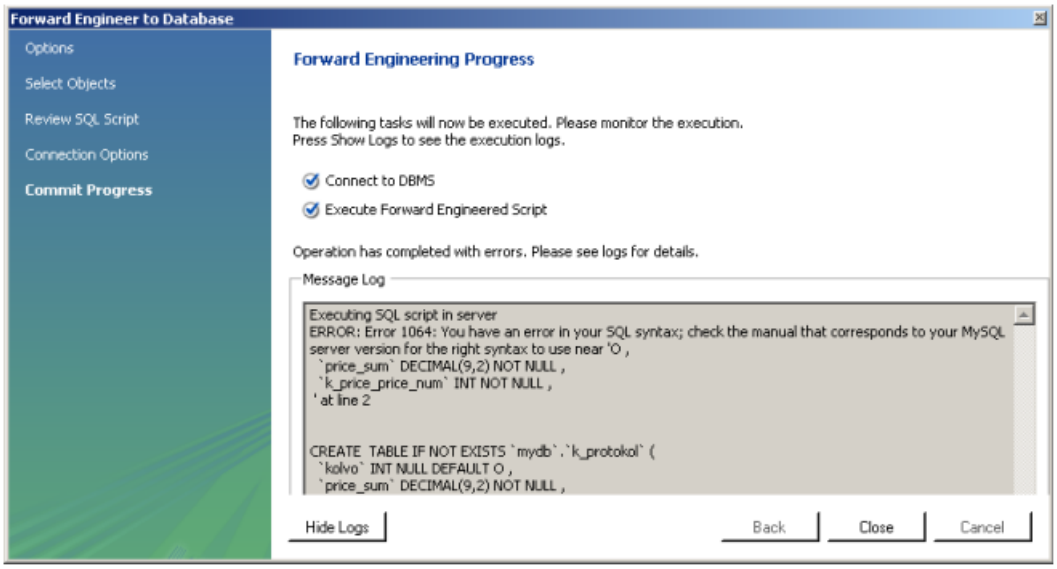


В противном случае можно нажать на кнопку **Show logs** и посмотреть протокол ошибок.

Какие могут быть ошибки? Например, в таблице ПротоколСчета мы захотели указать для количества значение по умолчанию «ноль», а вместо этого набрали букву О:



Получим ошибку:



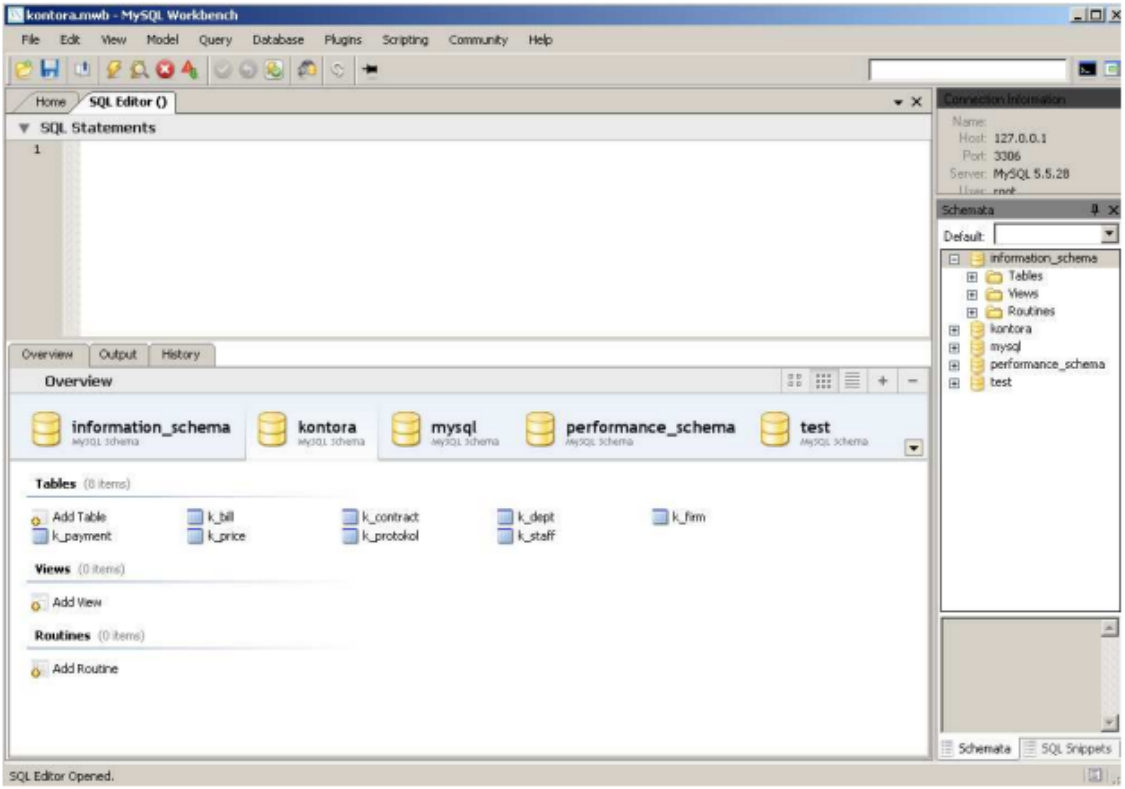
Заполнение базы данных, модификация данных

На предыдущем этапе мы создали базу данных. Теперь будем ее заполнять данными. Подключимся к серверу, в секции

щелкнем по ссылке Open Connection to Start Querying. В открывшемся окне нужно задать **username** и **password**, и нажать на кнопку «ОК».

Мы подключились к **MySQL server**.

В этом режиме работы рабочая область **MySQL Workbench** разделена на 3 окна:



- Верхнее окно **SQL Statements** предназначено для ввода и выполнения команд SQL. Внимание! В OS **Windows XP** текст, набранный в этом окне, автоматически не сохраняется. Если вы переместились из этого окна в какой-то другой режим работы, текст может быть потерян.
- Нижнее окно с несколькими вкладками показывает структуру имеющихся баз данных и позволяет ими управлять. Например,

если “дважды щелкнуть” по какой-либо таблице, то откроется дополнительное окно со структурой этой таблицы в нижней части рабочей области.

- Правое окно содержит иерархию объектов сервера.

Для заполнения базы данных в **MySQL Workbench** есть несколько возможностей. Рассмотрим три из них.

1-ый способ заполнения базы данных – используем команду INSERT

Самый универсальный и гибкий способ создания данных состоит в использовании SQL-команды INSERT. Формат у нее такой:

```
INSERT INTO ИмяТаблицы [(СписокСтолбцовТаблицы)]
VALUES (СписокЗначений);
```

В квадратных скобках указываются необязательные элементы команды. Если в этой команде пропустить СписокСтолбцовТаблицы, то имеются в виду ВСЕ столбцы, и именно в таком порядке, в каком они были определены при создании таблицы.

SQL-команды нужно набирать в окне **SQL statement**. Для выполнения команд нужно выбрать меню **Query – Execute** или кнопку на панели инструментов или нажать Ctrl+Enter.

Можно набрать несколько команд и выполнить их **все вместе**, или выделить **отдельную** команду (как для копирования) и выполнить только ее.

Текст SQL-команд, который также называют SQL-сценарием, можно (и нужно!) сохранять в файл. По умолчанию тип файла **sql**.

Заполним таблицу **Предприятия**:

```
# выберем базу данных

USE kontora;

# добавим строки

INSERT INTO k_firm (firm_name, firm_addr)
     VALUES('Альфа', 'Москва');

INSERT INTO k_firm (firm_name, firm_addr)
     VALUES('Бета', 'Казань');
```

```
INSERT INTO k_firm (firm_name, firm_addr)
     VALUES('Гамма', 'Париж');

INSERT INTO k_firm (firm_name, firm_addr)
     VALUES('Дельта', 'Лондон');

INSERT INTO k_firm (firm_name, firm_addr)
     VALUES('Омега', 'Токио');


# посмотрим результат

SELECT * FROM k_firm;
```

Обратите внимание, что мы не задавали значения для столбца *firm_num*, поскольку этот столбец имеет свойство **Auto increment**, и сервер его заполняет сам, натуральными числами.

Overview | Output | History | Result (1)

Export

	firm_num	firm_name	firm_addr	firm_phone
▶	1	Альфа	Москва	NULL
	2	Бета	Казань	NULL
	3	Гамма	Париж	NULL
	4	Дельта	Лондон	NULL
	5	Омега	Токио	NULL

Fetched: 5

Заполним **Отдел**

```
INSERT INTO k_dept (dept_short_name, dept_full_name)
     VALUES('Sales', 'Отдел продаж');

INSERT INTO k_dept (dept_short_name, dept_full_name)
     VALUES('Mart', 'Отдел маркетинга');

INSERT INTO k_dept (dept_short_name, dept_full_name)
     VALUES('Cust', 'Отдел гарантийного обслуживания');

SELECT * FROM k_dept;
```

Overview | Output | History | Result (1)

Export

dept_num	dept_full_name	dept_short_name	k_staff_staff_num
1	Отдел продаж	Sales	NULL
2	Отдел маркетинга	Mart	NULL
3	Отдел гарантийного обслуживания	Cust	NULL

Fetched: 3

Заполним таблицу **Сотрудник**. Обратите внимание, что в этой таблице можно указывать **только** такой номер отдела, который **существует** в таблице **Отдел**! (Оставить это поле пустым тоже можно.)

```
INSERT INTO k_staff
(staff_name, K_dept_dept_num, staff_hiredate, staff_post)
     VALUES('Иванов', 1, '1999-01-01', 'Менеджер');

INSERT INTO k_staff
(staff_name, K_dept_dept_num, staff_hiredate, staff_post)
     VALUES('Петров', 2, '2010-10-13','Менеджер');

INSERT INTO k_staff
(staff_name, K_dept_dept_num, staff_hiredate, staff_post)
     VALUES('Сидоров', 3, '2005-12-01','Менеджер');

INSERT INTO k_staff
(staff_name, staff_hiredate, staff_post)
     VALUES('Семенов', '1990-01-01','Директор');

INSERT INTO k_staff
(staff_name, K_dept_dept_num, staff_hiredate, staff_post)
     VALUES('Григорьев', 3, '2008-12-19','Программист');

SELECT * FROM k_staff;
```

Overview | Output | History | Result (1)

Export

staff_num	staff_name	staff_post	staff_hiredate	staff_termdate	K_dept_dept_num
1	Иванов	Менеджер	1999-01-01	NULL	1
2	Петров	Менеджер	2010-10-13	NULL	2
3	Сидоров	Менеджер	2005-12-01	NULL	3
4	Семенов	Директор	1990-01-01	NULL	NULL
5	Григорьев	Программист	2008-12-19	NULL	3

Fetched: 5

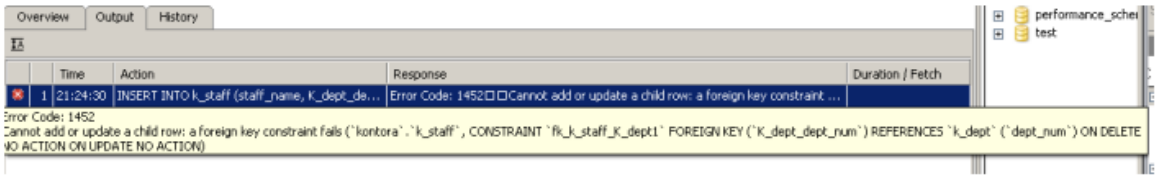
Что же будет, если указать несуществующий номер отдела?

```
INSERT INTO k_staff
(staff_name, K_dept_dept_num, staff_hiredate, staff_post)
     VALUES('Смит', 10, '2013-01-01','Консультант');
```

Будет получена следующая ошибка:

```
Error code: 1452

Cannot add or update a child row: a foreign key constraint fails
(`kontora`,`k_staff`, CONSTRAINT `fk_staff_k_dept` FOREIGN KEY
(`k_dept_dept_num`) REFERENCES `k_dept` (`dept_num`) …
```



Заполним таблицу **Договор**

```
INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('A', 1, 1,'2011-11-01');

INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('B', 1, 2,'2011-10-01');

INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('C', 1, 1,'2011-09-01');

INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('A', 2, 2,'2011-11-15');

INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('B', 2, 2,'2011-08-01');

INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('C', 3, 1,'2011-07-15');

INSERT INTO k_contract
(contract_type, k_firm_firm_num, k_staff_staff_num, contract_date)
     VALUES('A', 4, 1,'2011-11-12');


SELECT * FROM k_contract;
```

Overview | Output | History | Result (1)

Export

contract_num	contract_date	contract_type	k_staff_staff_num	k_firm_firm_num
1	2011-11-01	A	1	1
2	2011-10-01	B	2	1
3	2011-09-01	C	1	1
4	2011-11-15	A	2	2
5	2011-08-01	B	2	2
6	2011-07-15	C	1	3
7	2011-11-12	A	1	4

Fetched: 7

Заполним таблицу **Счет**

```
INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(1, '2011-11-12', '2011-12-12', 1000);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(1, '2011-12-12', '2012-01-12', 2000);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(1, '2012-01-12', '2012-02-12',2000);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(2, '2011-12-12', '2012-01-12', 6000);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(2, '2012-01-12', '2012-02-12', 2000);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(3, '2012-01-12', '2012-02-12', 2500);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(4, '2011-12-12', '2012-01-12', 1500);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(5, '2011-12-12', '2012-01-12', 1200);

INSERT INTO k_bill
(k_contract_contract_num, bill_date, bill_term, bill_sum)
     VALUES(5, '2012-01-12', '2012-02-12', 10000);
```

```
SELECT * FROM k_bill;
```

Overview | Output | History | Result (1)

Export

bill_num	bill_date	bill_sum	bill_term	bill_peni	k_contract_contract_num
1	2011-11-12	1000.00	2011-12-12	NULL	1
2	2011-12-12	2000.00	2012-01-12	NULL	1
3	2012-01-12	2000.00	2012-02-12	NULL	1
4	2011-12-12	6000.00	2012-01-12	NULL	2
5	2012-01-12	2000.00	2012-02-12	NULL	2
6	2012-01-12	2500.00	2012-02-12	NULL	3
7	2011-12-12	1500.00	2012-01-12	NULL	4
8	2011-12-12	1200.00	2012-01-12	NULL	5
9	2012-01-12	10000.00	2012-02-12	NULL	5

Fetched: 9

И остальные таблицы:

```
SELECT * FROM k_payment;
```

Overview | Output | History | Result (1)

Export

payment_num	payment_date	payment_sum	k_bill_bill_num
1	2011-12-15	1000.00	2
1	2012-01-13	1500.00	3
1	2012-01-12	1000.00	4
1	2012-01-05	100.00	7
1	2011-12-25	1000.00	8
2	2012-01-15	500.00	3
2	2012-01-12	900.00	7

```
SELECT * FROM k_price;
```

Overview | Output | History | Result (1)

Export

price_num	price_name	price_sum	price_type
1	Материализация духов	1000.00	У
2	Раздача слонов	100.00	У
3	Слоновий бивень	3000.00	Т
4	Моржовый клык	1500.00	Т
5	Копыто Пегаса	5000.00	Т

'У' означает услугу, 'Т' – товар.

```
SELECT * FROM k_protokol;
```

Overview | Output | History | Result (1)

Export

kolvo	price_sum	k_price_price_num	k_bill_bill_num
1	1000.00	1	1
2	1000.00	1	2
1	1000.00	1	5
2	1000.00	1	6
1	1000.00	1	8
20	100.00	2	3
10	100.00	2	5
5	100.00	2	6
2	100.00	2	8
2	3000.00	3	4
1	1500.00	4	7
2	5000.00	5	9

Кроме команды добавления данных INSERT, есть полезные команды изменения данных UPDATE и удаления данных DELETE.

Формат команды UPDATE:

```
UPDATE [INTO] ИмяТаблицы SET ИмяСтолбца=НовоеЗначение
[WHERE Условие];
```

Квадратные скобки означают необязательную часть команды. Если условия нет, то изменяются ВСЕ строки заданной таблицы.

Применим эту команду на практике. Если вы обратили внимание, в таблице **Отдел** остался незаполненным столбец *k_staff_staff_num,* означающий номер сотрудника – руководителя отдела.

```
UPDATE k_dept SET k_staff_staff_num=2
     WHERE dept_short_name='Mart';
UPDATE k_dept SET k_staff_staff_num=3
     WHERE dept_short_name='Cust';
UPDATE k_dept SET k_staff_staff_num=1
     WHERE dept_short_name='Sales';
```

Результат:

Overview | Output | History | Result (1)

Export

dept_num	dept_full_name	dept_short_name	k_staff_staff_num
1	Отдел продаж	Sales	1
2	Отдел маркетинга	Mart	2
3	Отдел гарантийного обслуживания	Cust	3

Формат команды DELETE:

```
DELETE [FROM] имя_таблицы [WHERE условие];
```

Квадратные скобки означают необязательную часть команды. Если условия нет, то удаляются ВСЕ строки заданной таблицы.

Пример: удаляем фирму с номером 5:

```
DELETE FROM k_firm WHERE firm_num=5;
```

Результат успешный. А что будет, если попробовать удалить фирму с номером 1? У этой фирмы есть подчиненные строки в таблице Договор.

Ошибка:

```
Error Code 1451

Cannot delete or update a parent row: a foreign key constraint
fails (`kontora`, `k_contract`, CONSTRAINT `fk_contract_k_firm`
FOREIGN KEY (`k_firm_firm_num`) REFERENCES `k_firm` (`firm_num`) …
```

Overview | Output | History

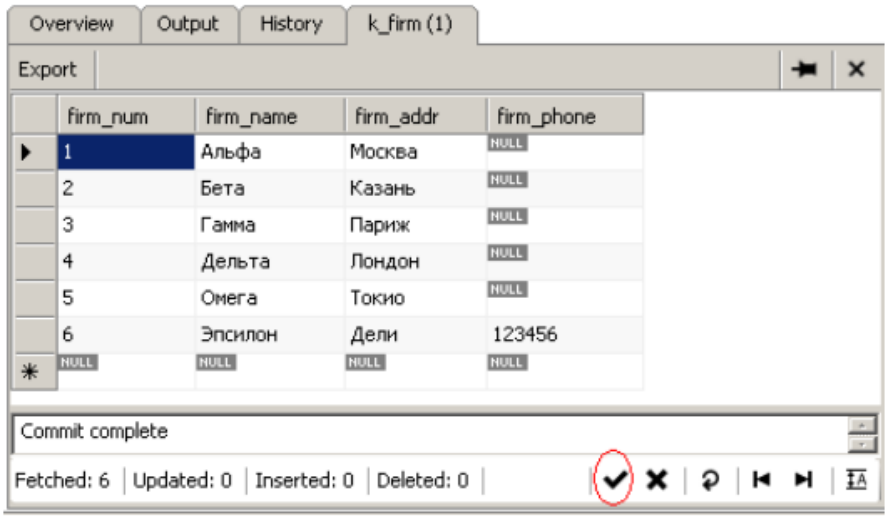
	Time	Action	Response	Duration / Fetch
1	21:53:23	DELETE FROM k_firm WHERE firm_num=1	Error Code: 1451 Cannot delete or update a parent row: a foreign ke...	

Error Code: 1451
Cannot delete or update a parent row: a foreign key constraint fails (`kontora`.`k_contract`, CONSTRAINT `fk_k_contract_k_firm1` FOREIGN KEY (`k_firm_firm_num`) REFERENCES `k_firm` (`firm_num`) ON DELETE NO ACTION ON UPDATE NO ACTION)

2-ой способ заполнения базы данных – используем визуальные средства

Чтобы заполнять базу данных с помощью визуальных средств, в окне сервера нужно “дваждыщелкнуть” по нужной таблице (или выполнить команду EDIT ИмяТаблицы). Откроется окно

редактирования, в котором можно изменять и добавлять данные. Не забывайте сохранять изменения нажатием на кнопку «галочка»!



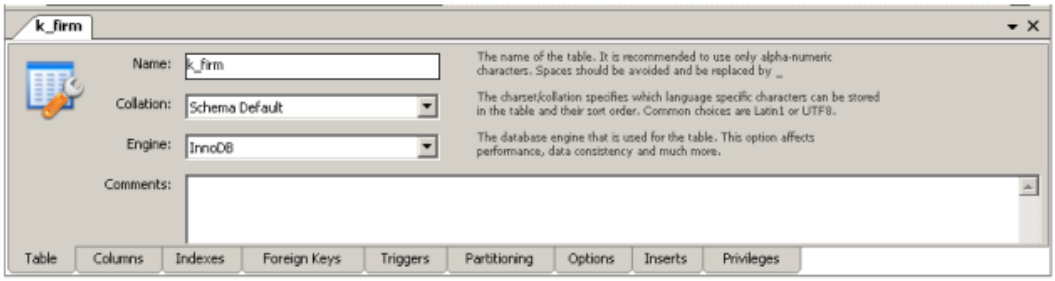
Этот способ добавления данных очень легкий – проблема возникает только при необходимости переноса данных на другой компьютер. Простых путей для копирования данных **нет**. Можно использовать выгрузку в текстовые файлы.

3-ий способ заполнения базы данных – данные хранятся в EER-модели

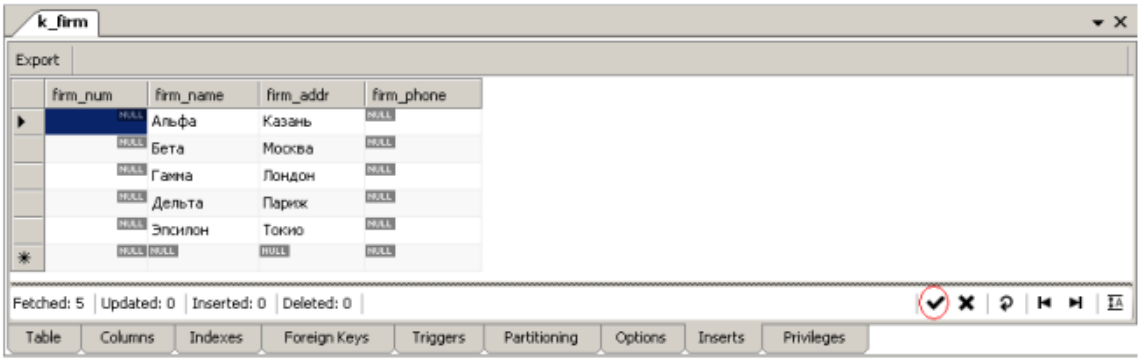
Без этого способа заполнения можно вполне обойтись, но для полноты картины расскажем о нем тоже.

Для применения этого способа придётся вернуться на шаг назад и открыть EER-диаграмму.

Как вы помните, если в диаграмме дважды щелкнуть по имени таблицы, то открывается окно ее свойств:



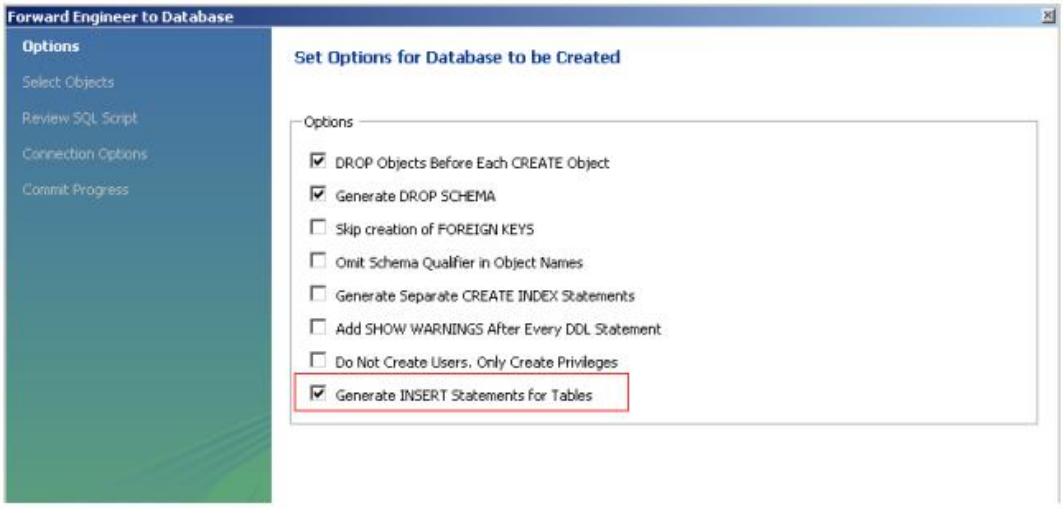
При разработке структуры таблицы мы использовали вкладку **Columns.** Теперь переключимся на вкладку **Inserts** и заполним данные в таблице**.**



НомерФирмы можем не заполнять, у него есть свойство Auto Increment. Телефон мы тоже не заполнили, он необязательный. После заполнения данных не забудьте нажать на кнопку «галочка», чтобы сохранились изменения в модели.

При использовании этого способа есть очень существенная проблема. При заполнении таблиц не проверяются никакие ограничения, ни на типы полей, ни на внешние ключи. Поэтому очень легко сделать ошибку.

Этот режим работы похож на предыдущий способ заполнения базы данных, но у него есть **очень важное отличие!** Заполненные таким образом данные хранятся **только в EER-модели**, на стороне сервера их **нет**. Для того чтобы данные появились на стороне сервера, нужно заново выполнить генерацию базы данных из EER-модели, как во втором задании. При этом обязательно нужно отметить флажок **Generate INSERT Statements for Tables:**



В этот момент проявят себя все ранее сделанные ошибки при вводе данных.

Разумеется, при этом старая база данных **удаляется**, вместе со **всеми** ранее введенными данными.