Тема 4.2

«Использование СУБД MySQL и САПР phpMyAdmin»

Вступление

Мы с вами только что рассмотрели основы работы с реляционными БД на стадии проектирования и использования специальных средств создания модели БД.

Сейчас мы перейдём к конкретике: созданию БД с помощью PhpMyAdmin.

Справедливости ради следует отметить, что в реальности при разработке веб-ориентированных приложений на PHP отдельные мощные средства проектирования БД наподобие «Sparx EA» используют редко, т.к. базы данных, чаще всего, представляют собой небольшое количество таблиц с относительно небольшим количеством полей и довольно простой логикой связей.

В случае, когда таблиц много и связи сложные, приложение, как правило, само приобретает такую степень сложности и предъявляет такие требования к разработке и эксплуатации, что оптимальным решением будет переход на использование языка Java.

Связи таблиц в БД: реальность

В предыдущей теме мы рассмотрели создание БД с тремя таблицами и двумя связями типа «один ко многим».

Связи являются мощным механизмом реляционных БД, позволяющим им поддерживать т.н. «целостность данных», т.е. возможность отслеживать операции над данными и налагать на эти операции ряд ограничений.

Например, СУБД может не позволить удалить запись в родительской таблице, если ей соответствует некоторое количество записей в дочерней таблице. Или, наоборот, при удалении записи из родительской таблицы автоматически удалить все относящиеся к ней записи в дочерней таблице.

Это всё красиво в теории реляционных БД и практике «крупных» СУБД (Oracle, MS-SQL и т.п.)

В MySQL всё иначе. Долгое время MySQL был **предельно** упрощённой СУБД, поддерживающей самый минимум функций. Это, с одной стороны, – плохо, но, с другой стороны, эта СУБД была и остаётся простой в изучении и использовании и достаточно быстрой.

Связи таблиц в БД: реальность

Несмотря на то, что последние версии MySQL уже обладают огромным набором функций, некоторые ограничения до сих пор сохранились, но, что важнее, сохранилась традиция разработки БД для MySQL: такие БД, как правило, представляют собой набор несвязанных таблиц.

Более того, в очень многих задачах связи между таблицами на самом деле не нужны (например, совершенно незачем связывать таблицу «администраторы» и таблицу «рубрики новостей» (если, конечно, мы не хотим реализовать хитрый механизм разграничения прав доступа).

Но даже, если мы очень хотим организовать связи между таблицами, мы можем сделать это только в том случае, если для физического хранения данных нашей БД MySQL использует механизм InnoDB.

В использовавшемся многие годы по умолчанию механизме MyISAM связи между таблицами не поддерживались. И, совершенно очевидно, что если вам придётся устанавливать своё приложение на сервер, где по какой-то причине поддержка InnoDB отключена, работать ваша база данных или не будет вообще, или будет не так, как ожидалось.

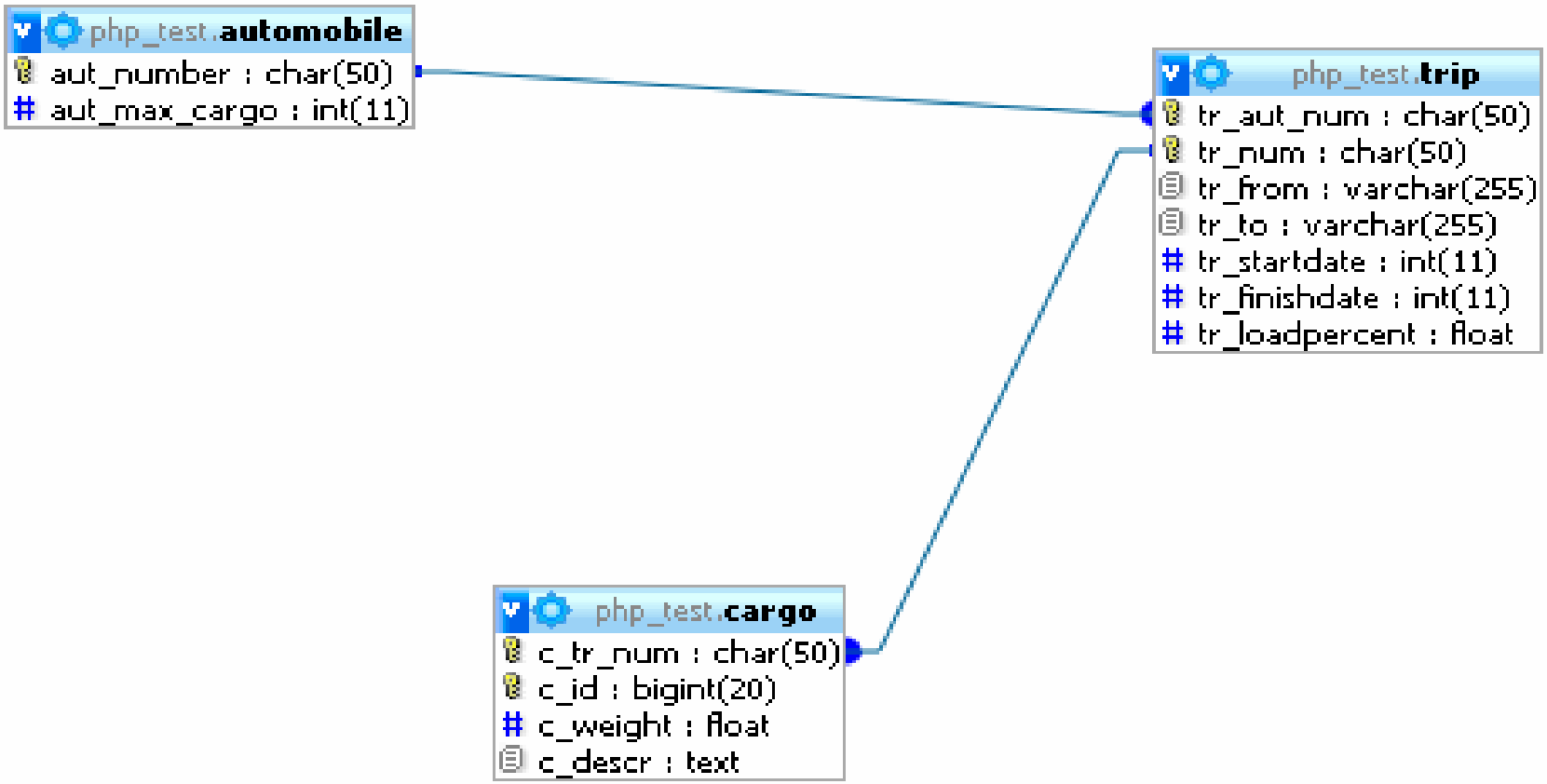
Связи таблиц в БД: реальность

Тем не менее, следует помнить, что грамотное проектирование БД с организацией всех «вспомогательных элементов» (связей, триггеров, хранимых процедур, представлений) является как хорошим тоном программирования, так и прекрасным способом повысить надёжность и безопасность вашего ПО.

Тем более, что как последние версии MySQL, так и последние версии PhpMyAdmin предоставляют множество возможностей «сделать всё правильно».

На следующем рисунке представлена модель рассмотренной в предыдущей схеме БД, открытая для просмотра и редактирования в PhpMyAdmin.

Использование PhpMyAdmin



Имена

Обратите внимание, что в представленной модели есть несколько важных моментов:

- все названия полей и таблиц приведены на английском языке в целях обеспечения максимальной совместимости с самыми разнообразными версиями СУБД;
- имена полей сформированы с префиксами, представляющими собой первые несколько символов имён таблиц, что позволят в будущем избежать неоднозначности трактовки имён полей в сложных SQL-запросах, а также повышает наглядность схемы.

Настройка PhpMyAdmin

Чтобы получить такую «красивость», как показано на предыдущем рисунке, нужно выполнить следующие действия (после того, как вы установите PhpMyAdmin).

1. Выполнить (от имени root) скрипты create_tables.sql и upgrade_tables_mysql_4_1_2+.sql (находятся в каталоге /scripts)
2. Выполнить следующие два запроса (от имени root):

```
GRANT USAGE ON mysql.* TO 'pma'@'localhost' IDENTIFIED BY 'pmapass'; GRANT SELECT ( Host, User, Select_priv, Insert_priv, Update_priv, Delete_priv, Create_priv, Drop_priv, Reload_priv, Shutdown_priv, Process_priv, File_priv, Grant_priv, References_priv, Index_priv, Alter_priv, Show_db_priv, Super_priv, Create_tmp_table_priv, Lock_tables_priv, Execute_priv, Repl_slave_priv, Repl_client_priv ) ON mysql.user TO 'pma'@'localhost'; GRANT SELECT ON mysql.db TO 'pma'@'localhost'; GRANT SELECT ON mysql.host TO 'pma'@'localhost'; GRANT SELECT (Host, Db, User, Table_name, Table_priv, Column_priv) ON mysql.tables_priv TO 'pma'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON phpmyadmin.* TO 'pma'@'localhost';
```

Настройка PhpMyAdmin

3. Скопировать файл config.sample.inc.php под именем config.inc.php (находится в корневом каталоге PhpMyAdmin)

4. Раскомментировать в нём все опции в секции /* User for advanced features */ и /* Advanced phpMyAdmin features */.

5. Всё. Теперь можно выбирать базу данных, заходить на вкладку «Дизайнер» и наслаждаться.

SQL

Язык SQL (Structured Query Language, язык структурированных запросов) – язык, на котором происходит всё взаимодействие с реляционными базами данных. Ему посвящено огромное количество литературы и, к сожалению, его детальное рассмотрение выходит за рамки нашего курса. Однако, мы будем рассматривать некоторые его элементы по ходу решения прикладных задач.

Сейчас же мы рассмотрим только набор инструкций, выполнение которых приведёт к созданию в MySQL базы данных, представленной ранее в виде схем на рисунках.

SQL

```
CREATE TABLE IF NOT EXISTS `automobile` (
  `aut_number` char(50) COLLATE cp1251_general_cs NOT NULL COMMENT 'Номер госрегистрации автомобиля',
  `aut_max_cargo` int(11) NOT NULL COMMENT 'Грузоподъёмность автомобиля',
  PRIMARY KEY (`aut_number`)
) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251_general_cs COMMENT='Информация об автомобиле';


CREATE TABLE IF NOT EXISTS `cargo` (
  `c_tr_num` char(50) COLLATE cp1251_general_cs NOT NULL COMMENT 'На каком рейсе',
  `c_id` bigint(20) NOT NULL COMMENT 'Идентификатор груза',
  `c_weight` float NOT NULL COMMENT 'Вес',
  `c_descr` text COLLATE cp1251_general_cs NOT NULL COMMENT 'Описание',
  PRIMARY KEY (`c_tr_num`),
  UNIQUE KEY `c_id` (`c_id`)
) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251_general_cs COMMENT='Груз';
```

SQL

```
CREATE TABLE IF NOT EXISTS `trip` (
  `tr_aut_num` char(50) COLLATE cp1251_general_cs NOT NULL COMMENT 'На каком автомобиле',
  `tr_num` char(50) COLLATE cp1251_general_cs NOT NULL COMMENT 'Номер рейса',
  `tr_from` varchar(255) COLLATE cp1251_general_cs NOT NULL COMMENT 'Откуда',
  `tr_to` varchar(255) COLLATE cp1251_general_cs NOT NULL COMMENT 'Куда',
  `tr_startdate` int(11) NOT NULL COMMENT 'Отбытие',
  `tr_finishdate` int(11) NOT NULL COMMENT 'Прибытие',
  `tr_loadpercent` float NOT NULL COMMENT 'Процент загруженности',
  PRIMARY KEY (`tr_aut_num`),
  UNIQUE KEY `tr_num` (`tr_num`)
) ENGINE=InnoDB DEFAULT CHARSET=cp1251 COLLATE=cp1251_general_cs COMMENT='Рейс';

ALTER TABLE `cargo`
  ADD CONSTRAINT `cargo_ibfk_1` FOREIGN KEY (`c_tr_num`) REFERENCES `trip` (`tr_num`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `trip`
  ADD CONSTRAINT `trip_ibfk_1` FOREIGN KEY (`tr_aut_num`) REFERENCES `automobile` (`aut_number`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Создание таблиц в PhpMyAdmin

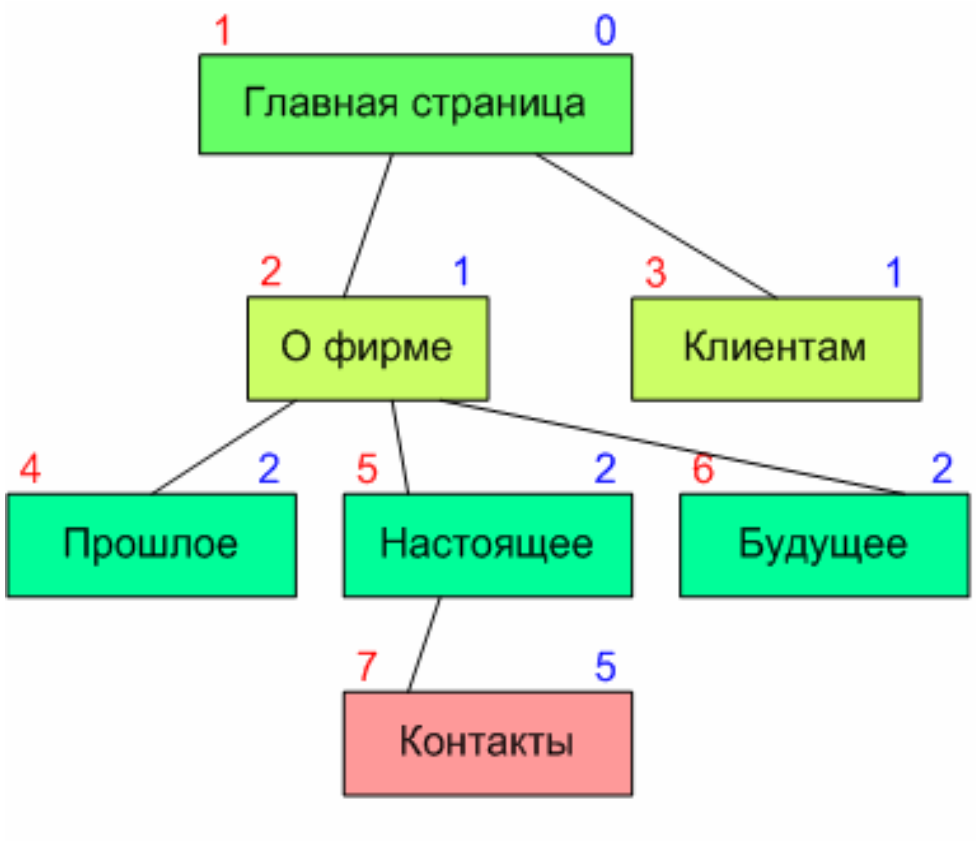
Для проектирования БД можно использовать непосредственно сам PhpMyAdmin – даже без рассмотренных выше дополнительных настроек.

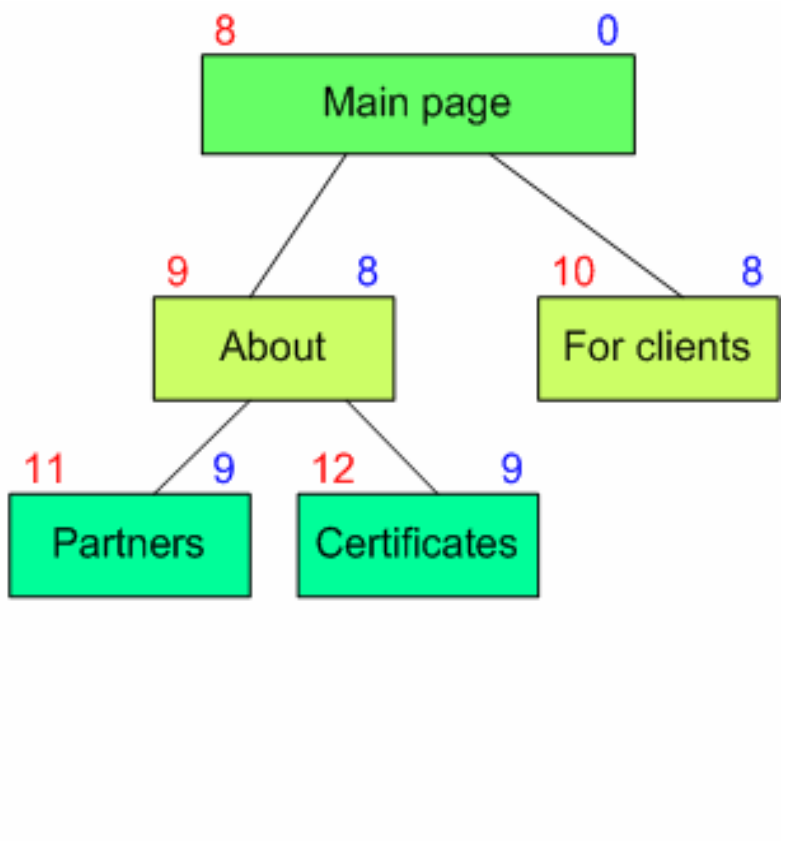
Сейчас мы кратко рассмотрим по шагам, как создать таблицу, в которой можно сохранить древовидную структуру сайта.

Но прежде – о том, как сохранить структуру сайта в одной таблице БД. Для полноты картины допустим, что наш сайт – многоязычный, причём языков может со временем добавиться.

Итак…

Создание таблиц в PhpMyAdmin





Создание таблиц в PhpMyAdmin

Красными числами в левом верхнем углу блоков помечены уникальные идентификаторы страниц (удобно сделать в таблице числовой первичный ключ с автоинкрементированием значения).

Синими числами в правом верхнем углу блоков помечены ссылки на «родительскую запись», т.е. таким образом мы указываем для любой страницы уникальный идентификатор её «страницы-родителя».

Обратите внимание, что уникальные идентификаторы всех страниц не совпадают и продолжают нумероваться и в англоязычной версии. Также заметьте, что у главных страниц нет родительских страниц – ссылка на родителя равна 0.

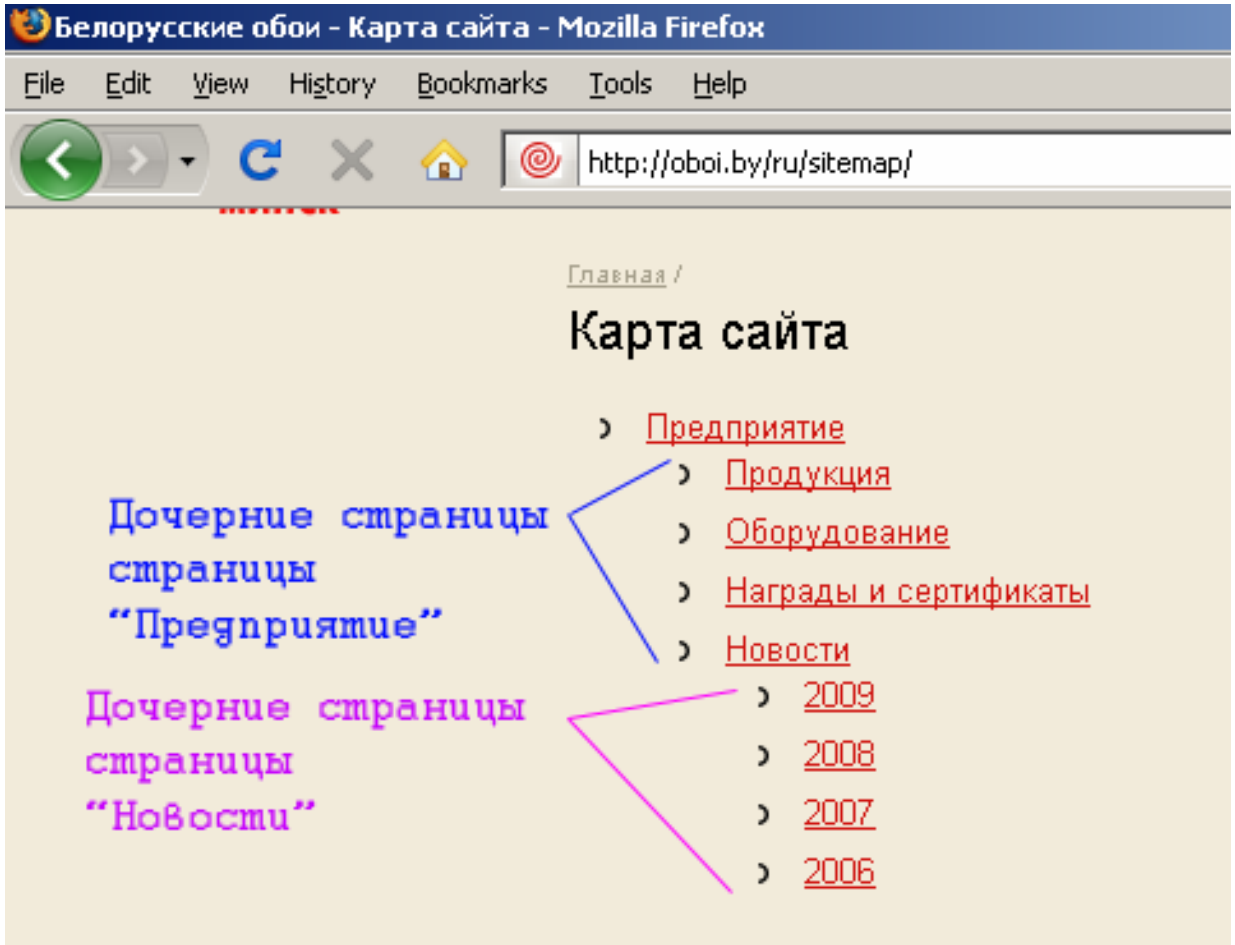
Если представить это дерево (или, если говорить совсем научно – лес деревьев, т.к. у них нет общего корня) в виде таблицы, получим такое…

Создание таблиц в PhpMyAdmin

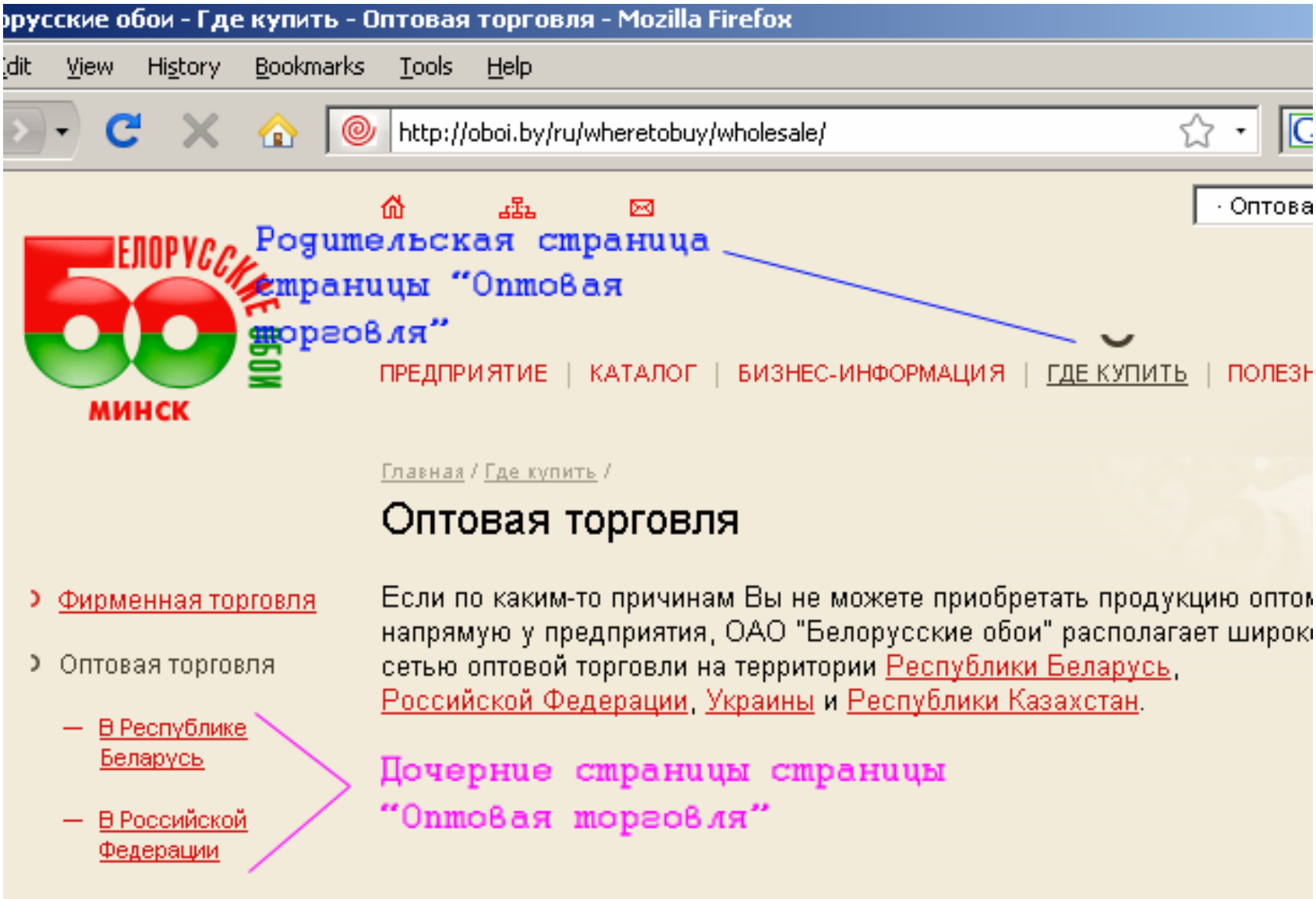
Уникальный идентификатор	Ссылка на родителя	Название
1	0	Главная страница
2	1	О фирме
3	1	Клиентам
4	2	Прошлое
5	2	Настоящее
6	2	Будущее
7	5	Контакты
8	0	Main page
9	8	About
10	8	For clients
11	9	Partners
12	9	Certificates

Создание таблиц в PhpMyAdmin

Для лучшего понимания, что такое дочерние и родительские страницы – рассмотрим два рисунка. Первый – карта сайта, второй – просто страница сайта.



Создание таблиц в PhpMyAdmin



Создание таблиц в PhpMyAdmin

Для каждой записи в рассмотренной таблице мы создадим поля:

page_id – уникальный идентификатор страницы, первичный ключ;

page_parent – ссылка на родительскую страницу;

page_order – порядок страницы в меню одного уровня;

page_lang – указание на языковую версию страницы;

page_title – значение тега TITLE;

page_menuname – название страницы в меню сайта;

page_pagename – заглавие страницы над основным текстовым блоком;

page_keyw – значение keywords;

page_desc – значение description;

page_loc – часть URL, отвечающая за формирование ссылки на страницу;

page_fakeurl – специфический URL, на который надо перейти, если пользователь зашёл на эту страницу;

page_expandchild – признак того, что при заходе на эту страницу нужно сразу же показать какую-то её подстраницу;

page_text – текстовое наполнение страницы;

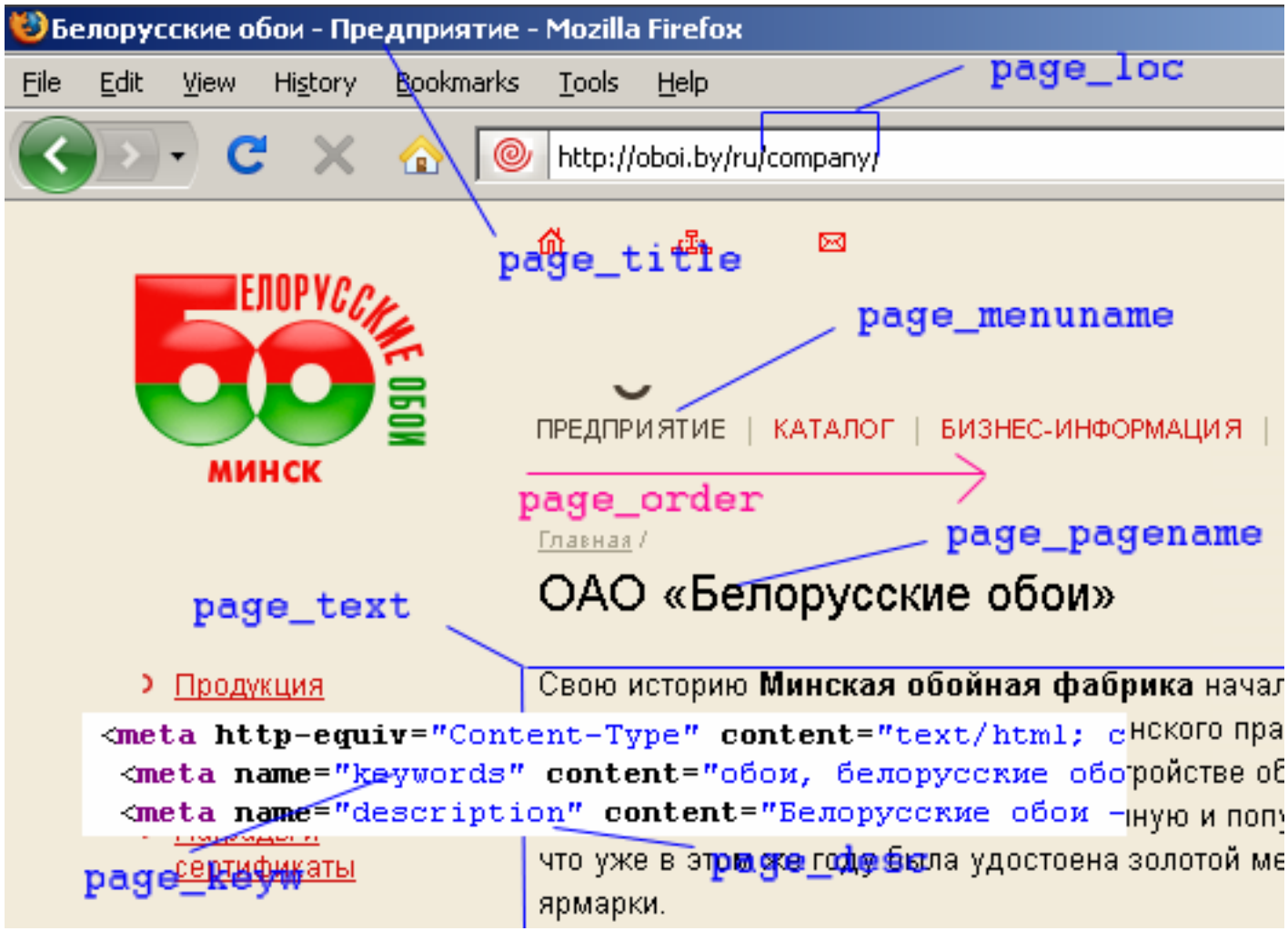
page_template – шаблон страницы (имя файла);

page_code – «обработчик страницы» (имя файла с кодом на PHP, который будет генерировать эту страницу).

При желании список таких полей можно расширить до любого разумного количества. Обратите внимание, что банеры, новости и т.п. элементы мы здесь не указываем – они будут в других таблицах, если понадобятся.

Создание таблиц в PhpMyAdmin

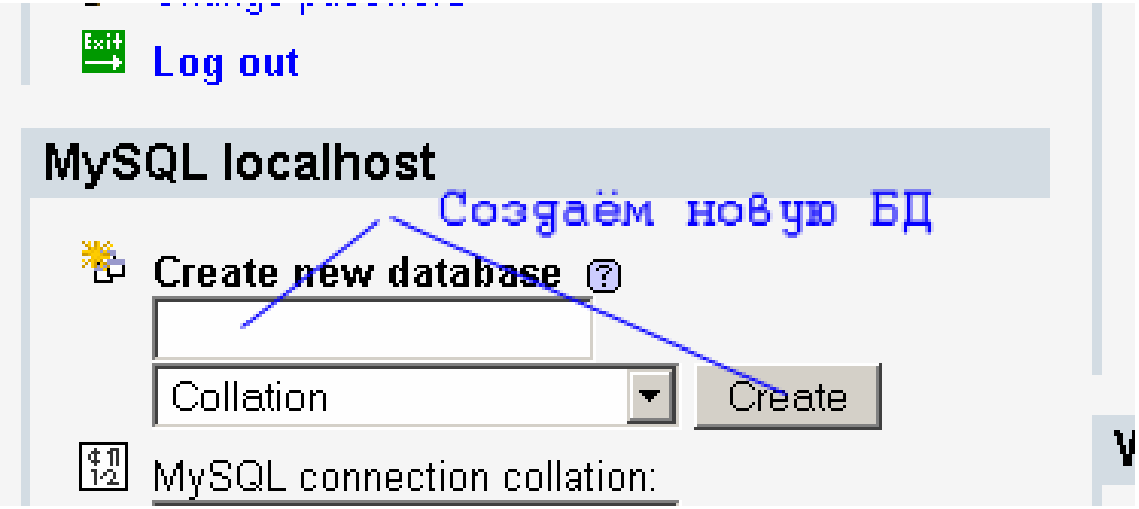
Нижеприведённый рисунок иллюстрирует, какая информация хранится в каком поле:



Создание таблиц в PhpMyAdmin

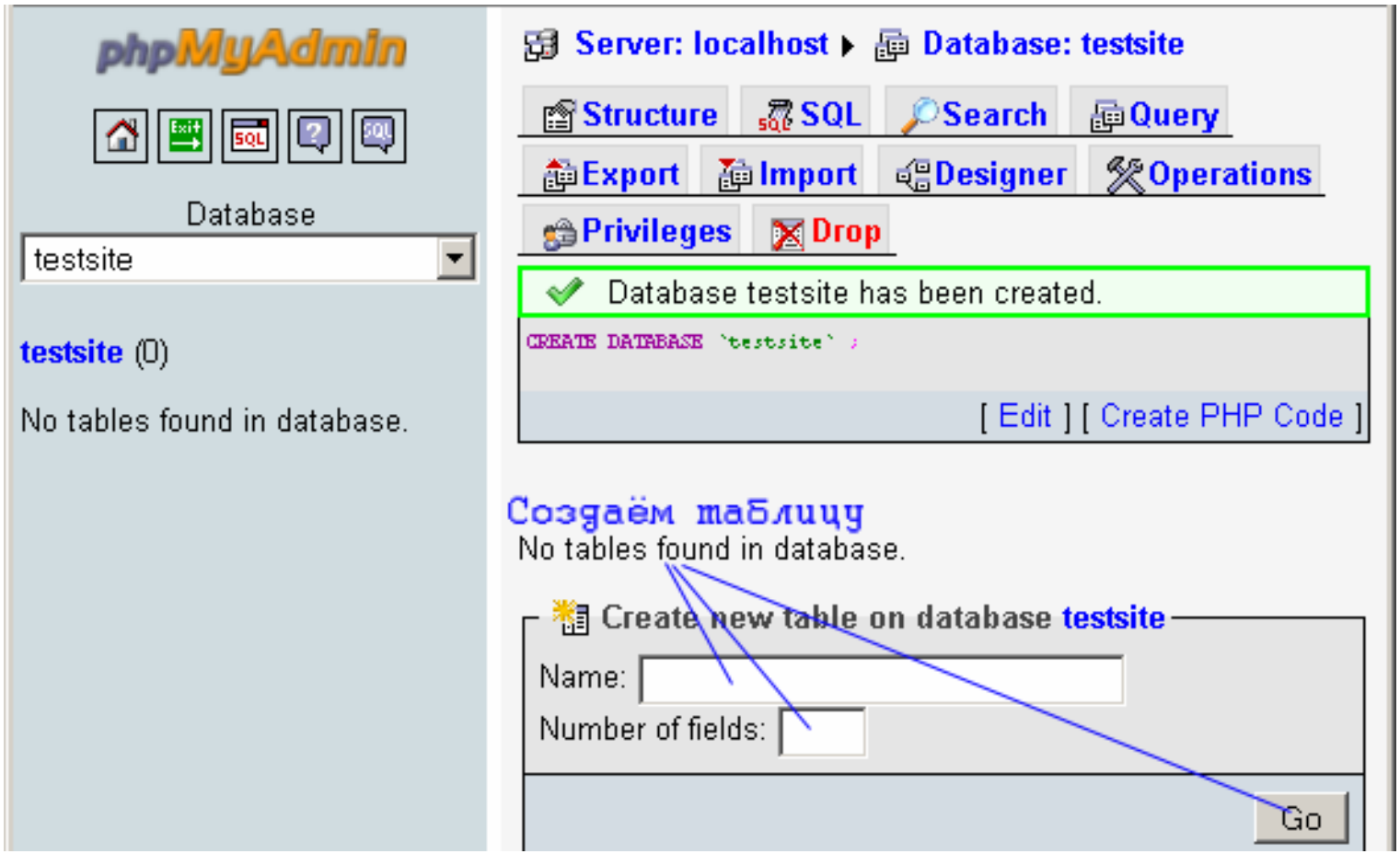
Теперь рассмотрим по шагам процесс создания таблицы в PhpMyAdmin. Процесс его установки описан в прилагаемом к материалам первого занятия видеоролике.

Итак, после того, как мы прошли авторизацию, мы можем создать базу данных:



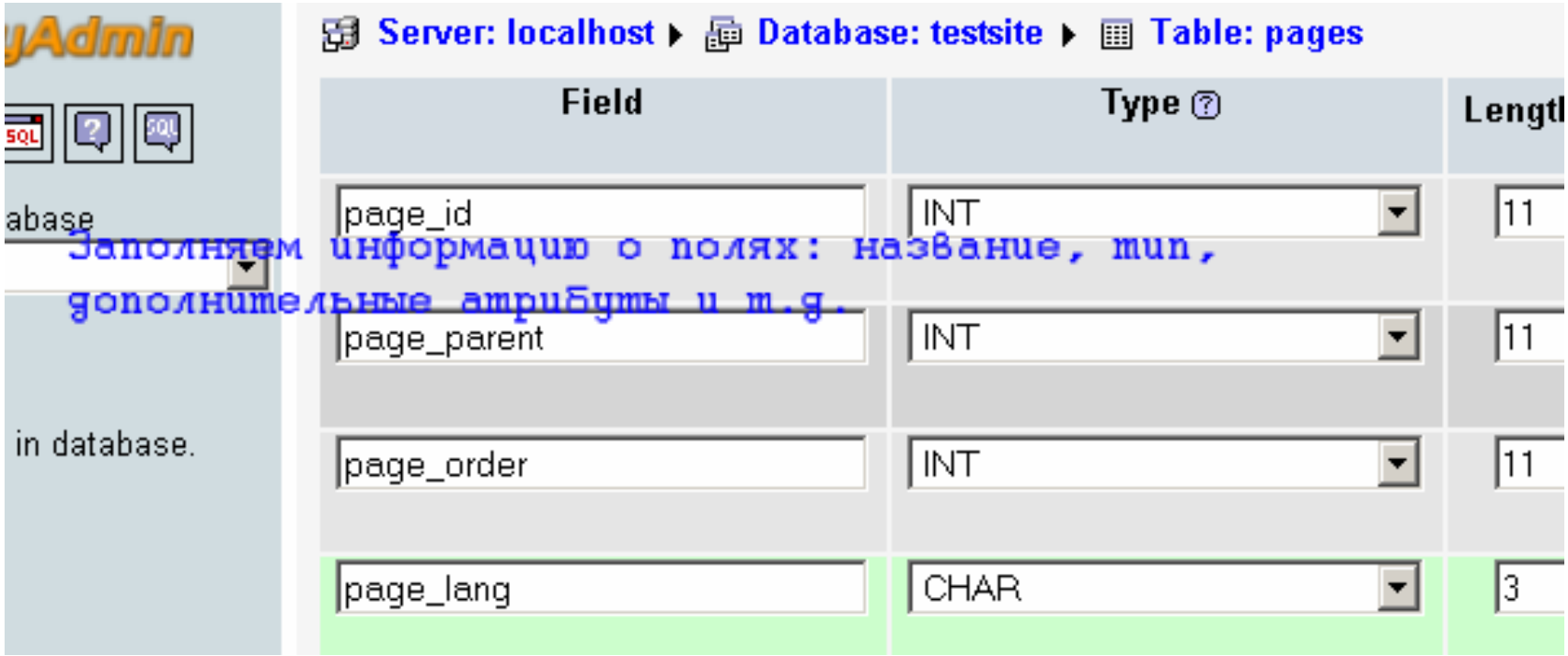
Создание таблиц в PhpMyAdmin

В новой БД создаём таблицу, указав количество полей (впоследствии поля можно добавлять и удалять, так что количество можно указывать приблизительное):



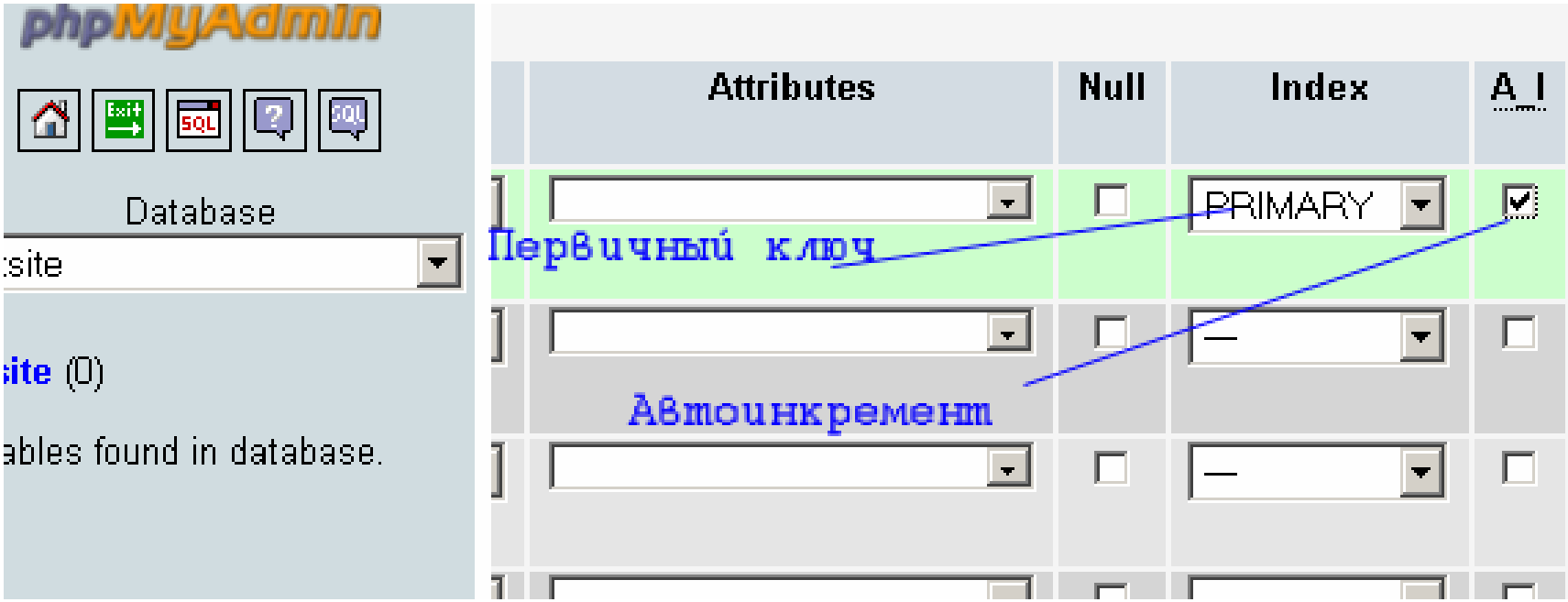
Создание таблиц в PhpMyAdmin

Заполняем информацию о таблице…



Создание таблиц в PhpMyAdmin

Полю page_id указываем, что оно – первичный ключ и автоинкрементируется:



Создание таблиц в PhpMyAdmin

После того, как вся информация заполнена, сохраняем таблицу. Таким же образом создаём остальные таблицы.

Обратите внимание, что после каждого вашего действия, приводящего к изменению в базе данных, PhpMyAdmin показывает вам вверху страницы SQL-запрос, с помощью которого он внёс это изменение. Т.о. PhpMyAdmin – ещё и удобный инструмент изучения SQL.

Реализация связей типа «1-М»

Связи «один ко многим» часто используются в задачах следующего типа:

- организовать рубрики новостей, где в каждой рубрике может быть много подрубрик;
- организовать многоуровневый каталог, где все уровни отличаются (иначе мы бы хранили всё в одной таблице, как только что рассмотрели структуру): виды товаров, производители товаров, марки товаров;
- список сотрудников, у которых может быть несколько телефонных номеров.

Решение таких задач сводится к созданию двух таблиц: родительской – в ней хранится, например, информация о рубрике новостей или о сотруднике, и дочерней – в ней хранятся, соответственно, новости или телефоны, а также присутствует информация о том, к какой рубрике или к какому сотруднику они относятся.

Реализация связей типа «1-М»

Представим эту идею на рисунке. Заметьте, что для повышения надёжности во второй таблице мы сделали первичный ключ, состоящий из двух полей (выделено курсивом), т.к. не бывает так, чтобы у одного человека было два одинаковых номера телефона.

В случае с рубриками новостей и самими новостями мы можем также сделать комбинированный первичный ключ из идентификатора новости и ссылки на её родительскую таблицу.

Сотрудники

Номер личного дела	ФИО
1	Иванов Иван Иванович
2	Петров Пётр Петрович
3	Сидоров Сидор Сидорович

Телефоны

Номер личного дела	*Номер телефона*
1	111-22-33
1	111-22-44
1	111-22-55
2	222-00-00
2	333-00-00

Реализация связей типа «1-М»

Рассмотрим чуть более сложный пример: уровней уже несколько.

Виды_организаций

ID_вида	Название_вида
1	Государственные
2	Частные
3	Криминальные

Подвиды_организаций

ID_подвида	*ID_вида*	Название_подвида
1	2	Фирмы
2	2	Магазины
3	2	Казино

Организации

ID_организации	*ID_подвида*	*ID_вида*	Информация
1	1	2	Данные A
2	3	2	Данные B
3	3	2	Данные C

Реализация связей типа «1-М»

Ситуация, которую мы видим на таком рисунке, называется «идентифицирующие связи», т.е. такие связи при которых в дочерней таблице не может быть записи, которой нет соответствующей записи в родительской таблице. Это удобно для организации целостности данных как раз в таких ситуациях, которая рассмотрена на рисунке.

Если же нам нужна возможность добавлять в дочерние таблицы записи без привязки к родительским таблицам, нам нужно использовать т.н. «неидентифицирующие связи».

Рассмотрим пример…

Реализация связей типа «1-М»

Группы_банеров

ID_группы	Название_группы
1	Автомобили
2	Недвижимость
3	Техника

Подгруппы_банеров

ID_подгруппы	ID_группы	Название_подгруппы
1	3	Компьютеры
2	0	Напитки
3	2	Офисы

Банеры

ID_банера	ID_подгруппы	ID_группы	Название
1	3	2	Офис на пр. Незав…
2	0	3	Газонокосилки…
3	2	0	Водка! Много!
4	0	0	Котята. В хорош…

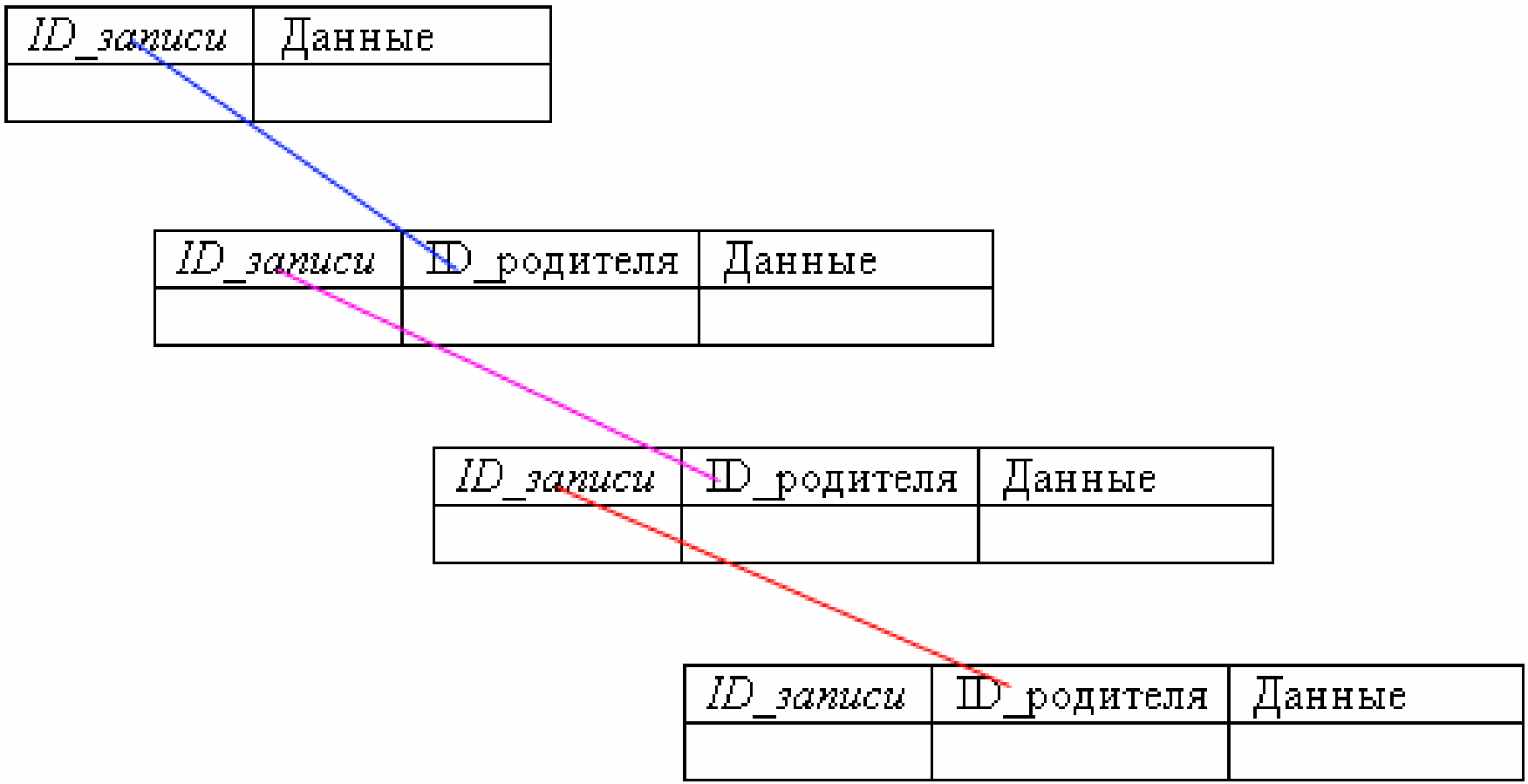
Реализация связей типа «1-М»

Такой подход также даёт нам возможность делать «ссылки через уровень» (в теории БД такого термина нет, но мы его используем, т.к. он отражает суть). Так, на только что рассмотренном примере банер «Газонокосилки» не относится ни к одной из подгрупп, но относится к группе «Техника».

Теперь о реальности. Чаще всего в разработке БД для веб-ориентированных приложений можно увидеть ситуацию, представленную на следующем рисунке. Она не является корректной с точки зрения теории реляционных БД, но настолько проста в реализации, что в некоторых случаях её можно применять. Проблемы в такой ситуации только две:

- возможно нарушение целостности данных;
- очень неудобно подсчитывать, изменять и удалять элементы нижних уровней, относящиеся к какому-то элементу верхнего уровня.

Реализация связей типа «1-М»



Реализация связей типа «М-М»

Связи типа «многие-ко-многим» тоже активно применяются. Допустим, у нас есть некие информационные блоки, каждый из которых может присутствовать на нескольких страницах сайта, в то время как на каждой странице сайта может быть несколько таких блоков. Для полноты картины добавим, что у каждого блока могут быть некоторые параметры отображения его на данной конкретной странице.

Для организации связей типа «многие-ко-многим» используется промежуточная таблица, т.е., фактически, связь «многие-ко-многим» преобразуется в две связи «один-ко-многим».

Т.о. установление связи «многие-ко-многим» сводится к созданию записи в промежуточной таблице, содержащей уникальные идентификаторы записей из обеих связываемых таблиц, а также любую дополнительную информацию.

Рассмотрим на рисунке.

Реализация связей типа «М-М»

Страницы_сайта

Поля_с_данными_страницы				*ID_страницы*
…	…	…	…	1
…	…	…	…	2
…	…	…	…	3
…	…	…	…	4

Страницы_блоки_M2M

ID_страницы	Параметры	*ID_блока*
1	…	2
1	…	4
3	…	2
4	…	2

Информационные_блоки

ID_блока	Данные
1	…
2	…
3	…
4	…

Реализация связей типа «М-М»

В общем случае дополнительные поля в промежуточной таблице могут отсутствовать, тогда там останутся только два поля: идентификатор записи первой связываемой таблицы и идентификатор записи второй связываемой таблицы.

Использование связей «многие-ко-многим» также эффективно, если мы строим, например, каталог товаров, где каждый производитель может производить разные виды товаров, а каждый вид товаров может быть произведён разными производителями.

Заключение

Т.о., как мы видим, использование баз данных позволяет быстро и легко решить широкий спектр прикладных задач. Выбор конкретной СУБД зависит от того, что за приложение мы пишем, где и как оно будет эксплуатироваться, какие требования предъявляются к его производительности, надёжности, безопасности и т.д.

Стандартными решениями являются: MySQL или PostgreSQL для проектов начального и среднего уровня; MS-SQL, DB2, Informix – для проектов «уровнем повыше», и Oracle – для крупных корпоративных проектов.