Язык SQL

Сленг для СУБД

Когда мы приезжаем в чужую страну, перед нами встает проблема языкового барьера. Впрочем, на элементарном английском языке можно объясниться практически в любой стране мира. Аналогично обстоят дела и с современными СУБД, но язык общения с ними гораздо проще — SQL, хотя и здесь имеют место различные диалекты.

Для хранения и упорядочения информации сегодня повсеместно служат базы данных. Современные БД не только могут принять на хранение данные, изменить их и выдать определенному пользователю, но также и провести их сложную обработку, например подсчитать баланс или провести переоценку и еще многое другое.

Что с ними делать?

Операций, которые можно произвести с СУБД, существует всего четыре — добавление записи, удаление, изменение и выборка данных по условию. Эти операции составляют часть языка SQL, которая называется «язык манипуляций с данными» или DML (Data Manipulation Language). Но прежде нужно определить, как данные будут расположены, в каких «емкостях». Для этого существуют операторы второй части, называемой языком определения данных — DDL (Data Defifnition Language). DDL позволяет создать в базе данных таблицы, индексы для быстрого поиска, взаимосвязи между таблицами, различные формы представлений и другие объекты. В нем, как в DML, мало основных операций, их всего три — создать, изменить и удалить.

MySQL

Многие online-сервисы сегодня реализованы с использованием простого сервера MySQL. Вычислительная мощность у него небольшая, и он не поддерживает средств сохранения целостности данных, но тем не менее он вполне подходит для размещения в Интернете небольшого количества данных и динамического построения web-страниц. »

Операторы DDL

Оператор CREATE служит для создания всего, что может располагаться в базе данных, и даже для самой базы данных. Его оформление можно описать следующим образом: CREATE <объект> <описание>. Для создания новой базы данных оператор будет выглядеть так: CREATE DATABASE web.

Далее в web нужно создать таблицы в зависимости от того, какая информация будет храниться в ней. Предположим, это будет информация о посетителях сайта. Для этого мы предложим посетителю зарегистрироваться и ввести следующую информацию о себе: имя, фамилию, адрес, e-mail и телефон. Если мы планируем реализовать подписку на рассылку новостей, то предложим пользователю выбрать интересующие его темы из списка. При желании можно дополнить базу данных и другими возможностями.

Сущности бытия

Итак, мы сформировали две сущности (Entity): сущность 1 — посетитель, назовем ее Visitor; и сущность 2 — список рассылок, на которые он подписался, назовем ее Subscription. Для списка рассылки достаточно хранить два атрибута — название и периодичность выхода.

Теперь откроем доступную нам CASE-систему для проектирования базы данных. К примеру, можно воспользоваться Power Designer 7.5, www.sybase.ru (рис. 1). Нарисуем эти сущности и определим между ними связь «Многие-ко-многим», так как у нашего сайта будет много посетителей, и добавим к сущностям их уникальные идентификаторы — idVisitor и idSubscr соответственно.

Часть связи со стороны посетителя, определяющую его подписку, назовем Visitor_Subscriptions, а часть со стороны подписки, определяющую того, кто на нее подписался, — Subscribed_By_Visitor.

Для реляционных моделей данных, с которыми работают SQL-серверы, отношения «многие-ко-многим» должны быть преобразованы в «один-ко-многим». Для этого перестроим модель в конкретные таблицы базы данных, которые Power Designer сам создаст при преобразовании концептуальной модели в физическую. Чтобы он правильно все сделал, укажем ему тип базы данных — MySQL (рис. 2).

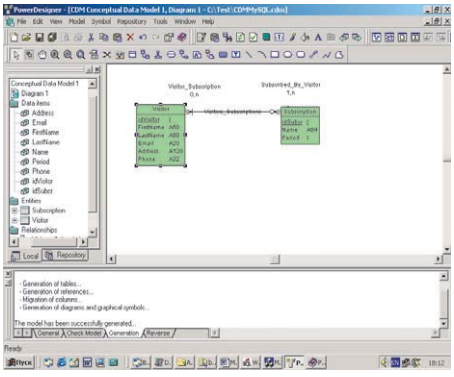
Дизайнер построит физическую модель, в которой будет три таблицы. Последняя сводит связь «многие-ко-многим» к формуле «один-ко-многим». Она будет содержать всего два поля для идентификации конкретного посетителя с конкретной рассылкой (рис. 3).

Атрибуты таблиц

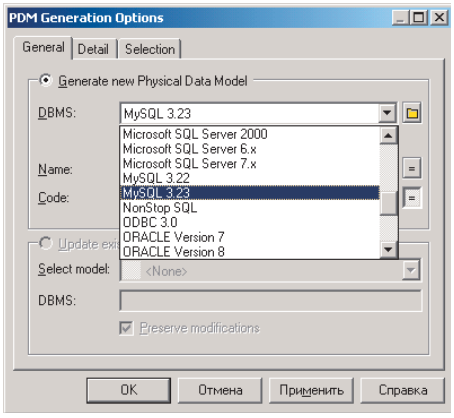
Оператор CREATE TABLE создает таблицу в базе данных. В нем следует перечислить создаваемые колонки, их типы и размеры. Атрибут primary key указывает, которая из колонок будет выступать в качестве первичного ключа — поля, которое однозначно идентифицирует запись. Также в качестве атрибута указывается, может ли значение в колонке быть неопределенным (NULL); оно означает, что полю не присвоено значение. NULL не равно NULL, любое сравнение двух величин, одна из которых (или обе) равна NULL, даст результат «ложь». Для проверки, является ли значение NULL, в языке SQL есть специальные операции IS NULL и IS NOT NULL.

Создаем таблицы

Первая таблица, которую нам предстоит создать, — таблица с данными о посетителях сайта (рис. 4). Выполним этот оператор и все последующие:



▲ Рис. 1. Sybase Power Designer 7.5 — система визуального проектирования моделей данных



▲ Рис.2 Выбор СУБД в Power Designer

У истоков SQL

Начало организации данных

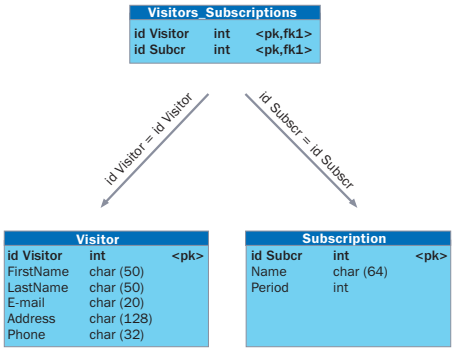
Когда-то, еще относительно недавно, в 70-х годах XX века, базы данных представляли собой файлы — хранилища данных в определенных уникальных форматах. Чтобы получить к ним доступ, необходимо было пользоваться соответствующими программными библиотеками и писать некоторый программный код на каком-либо языке программирования, иным языком, надо было быть программистом узкой специализации.

Как раз в то время фирма IBM проводила исследовательские работы в обозначенной области. Их задачей было создать средство, позволяющее описывать практически любые действия с данными, которые может выполнить компьютер, таким образом, чтобы все это можно было выразить в удобной и понятной для пользователя форме. В результате этих исследований, в компании разработали Структурный Язык Запросов — SQL (Structured Query Language). На основе английского языка он позволяет описывать действия, которые требуют выполнить от сервера базы данных.

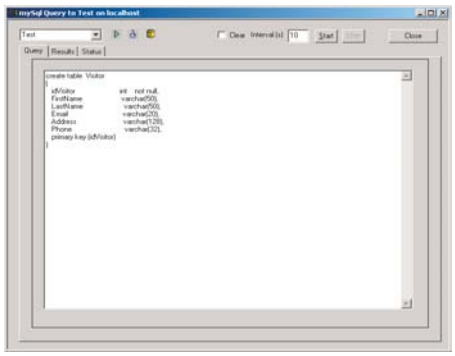
Разработки IBM заинтересовали ряд других производителей. К примеру, фирма Relation Software Inc в 1979 году выпустила свою новую СУБД, работающую с запросами на SQL. Эта СУБД и сама фирма Relation Software, переименованная позднее в ORACLE Inc, приобрели мировую известность. Компания и по сей день успешно работает в различных областях информационных технологий.

Все современные СУБД умеют работать с языком SQL, а некоторые из них даже содержат собственные процедурные расширения этого языка. В 1986 году ANSI и ISO разработали стандарт языка, который потом периодически обновлялся и дополнялся (последний раз в 1992 году).

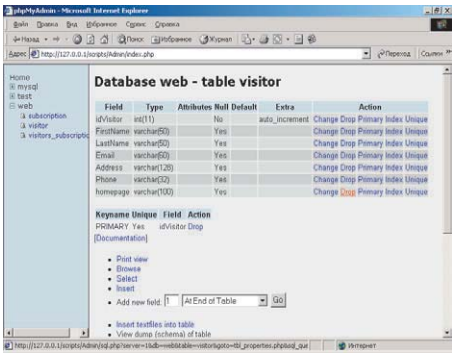
Мы рассматриваем основы языка SQL на примере очень популярной СУБД MySQL.



▲ Рис. 3. Модель данных списка тематической рассылки



▲ Рис. 4. Программа Manager, входящая в комплект MySQL для Windows



▲ Рис. 5. Программа администрирования базы данных MySQL

```
create unique index Visitor_PK on visitor
(
 idVisitor
);
create table visitors_subscriptions
(
 idVisitor int not null,
 idSubcr int not null,
 primary key (idVisitor, idSubcr)
);
create unique index Visitors_Subscripti-
ons_PK on visitors_subscriptions
(
 idVisitor,
 idSubcr
);
create index Subscribed_By_Visitor_FK on
visitors_subscriptions
(
 idVisitor
);
create index Visitor_Subscription_FK on
visitors_subscriptions
(
 idSubcr
);
create table Subscription
(
 idSubcr int not null auto_increment,
 Name varchar(64),
 Period int,
 primary key (idSubcr)
);
create unique index Subscription_PK on
subscription
(
 idSubcr
);
```

Для создания индексов оператор CREATE имеет следующий формат: CREATE INDEX <имя индекса> ON <имя таблицы> (<список колонок, таблицы, которые в него входят>).

Если в системе установлена поддержка языка PHP, можно воспользоваться программой-администратором phpMySQLAdmin (http://phpwizard.net/projects/phpMyAdmin). Она позволяет создавать новые базы данных MySQL, имеет очень удобный интерфейс и обширные возможности (рис 5).

Изменение таблиц

Чтобы изменить структуру таблицы путем добавления, изменения или удаления колонок, нужно воспользоваться оператором ALTER. Например: ALTER TABLE Visitor ADD homepage VARCHAR (100) — добавит колонку homepage в таблицу visitor.

Для изменения колонки применяется функция MODIFY. Атрибут auto_increment означает, что при вводе значение колонке можно не присваивать, сервер будет считать его сам, каждый раз увеличивая на единицу последний присвоенный номер: ALTER TABLE visitor MODIFY idVisitor int not null auto_increment.

Удаляется колонка оператором ALTER с функцией DROP: ALTER TABLE visitor DROP homepage. Наконец, чтобы удалить объект из базы, существует оператор DROP, например: DROP TABLE visitor.

Типы данных

Список возможных типов колонок всегда можно найти в документации и многочисленных описаниях по базам данных, в частности, по MySQL. Типы наиболее часто встречающиеся на практике это:

INT — целое число, VARCHAR — строка. MySQL поддерживает строки длиной до 255 символов. Если нужно хранить больше символов, для этого есть тип TEXT. FLOAT и DOUBLE для действительных значений, BLOB, LONGBLOB — для хранения изображений или мультимедийной информации, Date, Time, DateTime и Timestamp — для хранения даты и времени в различных видах.

Размещение данных на хранение

Теперь, когда таблицы созданы, можно в них что-либо положить. Для этого нужно перейти к рассмотрению второй части операторов — DML. Для добавления записи в таблицу служит оператор INSERT, изменения — UPDATE (REPLACE), удаления — DELETE, чтения — SELECT.

Оператор INSERT имеет следующий синтаксис: INSERT INTO <список колонок> VALUES <список соответствующих значений>. Для примера заполним таблицу подписок:

```
INSERT INTO subscription
      VALUES ('Автомобили', '3');
INSERT INTO subscription
      VALUES ('Компьютеры (железо)', '5');
INSERT INTO subscription
      VALUES ('Программы (Software)', '5');
INSERT INTO subscription
      VALUES ('Политика', '14');
INSERT INTO subscription
```

Select

```
SELECT name, Period FROM subscription
WHERE Period <= 5 ORDER BY Period
```

name	Period
Автомобили	3
Новости культуры	3
Компьютеры (железо)	5
Программы (Software)	5
Игры	5
Работа и вакансии	5

▲ Табл. 1. Выборка по условию

Join

```
SELECT V.FirstName,V.LastName, .Email,S.Name,S.Period FROM visitor V
LEFT JOIN visitors_subscriptions VS ON VS.idVisitor=V.idVisitor
LEFT JOIN subscription S ON S.idSubcr=VS.idSubcr
```

FirstName	LastName	Email	Name	Period
Иван	Мухин	muchin@mail.ru	Автомобили	3
Иван	Мухин	muchin@mail.ru	Компьютеры (железо)	5
Иван	Мухин	muchin@mail.ru	Программы (Software)	5
Иван	Мухин	muchin@mail.ru	Экономика	14
Иван	Мухин	muchin@mail.ru	Игры	5
Николай	Петров	petrov@rambler.ru	Политика	14
Николай	Петров	petrov@rambler.ru	Работа и вакансии	5
Николай	Петров	petrov@rambler.ru	Новости культуры	3

▲ Табл. 2. Соединение таблиц

Outer

```
SELECT S.Name,S.Period, V.FirstName,V.LastName, V.Email FROM
(subscription S LEFT OUTER JOIN visitors_subscriptions VS ON S.idSubcr=
VS.idSubcr ) LEFT OUTER JOIN visitor V ON VS.idVisitor=V.idVisitor
```

Name	Period	FirstName	LastName	Email
Автомобили	3	Иван	Мухин	muchin@mail.ru
Компьютеры (железо)	5	Иван	Мухин	muchin@mail.ru
Программы (Software)	5	Иван	Мухин	muchin@mail.ru
Политика	14	Николай	Петров	petrov@rambler.ru
Экономика	14	Иван	Мухин	muchin@mail.ru
Игры	5	Иван	Мухин	muchin@mail.ru
Работа и вакансии	5	Николай	Петров	petrov@rambler.ru
Новости культуры	3			

▲ Табл. 3. Вывод с учетом пустых записей

```
    VALUES ('Экономика', '14');
INSERT INTO subscription
    VALUES ('Игры', '5');
INSERT INTO subscription
        VALUES ('Работа и вакансии', '5');
INSERT INTO subscription
        VALUES ('Новости культуры', '3');
```

Таблицу можно также заполнить путем выборки данных из других таблиц. В этом случае используется следующая форма записи: INSERT INTO <таблица> SELECT <колонки> FROM <таблица>.

Практическое размещение

После того как пользователь ввел свои данные в форму на web-странице, их нужно передать серверу для записи в таблицу Visitor. Для этого сформируем запрос вида:

```
INSERT INTO visitor VALUES ('Иван', 'Мухин',
'muchin@mail.ru', 'Севастопольский просп.
58-11', '123-45-67', '');
```

Все строковые значения должны быть заключены в апострофы, даже если они пустые.

После регистрации пользователь, возможно, захочет подписаться на некоторые из новостей. Для этого нужно вставить записи с его номером и номерами выбранных им групп в переходную таблицу Visitors_subscriptions:

```
INSERT INTO visitors_subscriptions
        VALUES ( '1', '1')
INSERT INTO visitors_subscriptions
        VALUES ( '1', '5');
```

Предполагается, что посетитель зарегистрировался под номером 1 и выбрал новости с номерами 1 и 5.

Запрос на получение данных

Для запроса данных из базы в SQL существует оператор SELECT (выбрать). Синтаксис этого оператора прост: SELECT <что выбираем> FROM <откуда> WHERE <условие отбора> GROUP BY <как сгруппировать > ORDER BY <как отсортировать >.

В простейшем случае, если нам нужно получить все записи из таблицы без каких-либо условий, оператор SELECT выглядит так: SELECT * FROM visitor.

Звездочка означает, что требуется взять значения всех колонок, которые имеются в таблице. Если нужны не все, а только некоторые из них, или выражения, составленные из значений колонок, они перечисляются через запятую (табл. 1).

Соединение нескольких таблиц

Для соединения таблиц используем оператор LEFT JOIN (соединение слева). Все зависит от направления — что с чем соединяется. Если посмотреть, на что подписался каждый из посетителей, то записям из Visitor мы сопоставим записи из Visitors_subscriptions с таким же idVisitor, а им, в свою очередь, записи из Subscription, где idSubcr совпадают. С JOIN связано слово ON, которое задает условие соединения: по каким общим полям таблицы склеиваются. Так как мы используем имя таблицы в нескольких местах, удобно назначить ей псевдоним (alias). Псевдоним указывается сразу после имени через пробел. Если имени таблицы назначен псевдоним, то все ссылки на эту таблицу должны быть только через него (табл. 2).

Если же, наоборот, нам нужно посмотреть по группам новостей, кто подписался на них, порядок таблиц меняется на обратный. Если на новость никто не подписан, но она все равно должна быть выведена, добавляется слово OUTER (открытый), означающее, что учитывать все записи, даже пустые. Как видим, на новости культуры не подписался никто (табл. 3).

Изменение и удаление данных

Предположим, у г-на Мухина изменился e-mail, и он хочет получать подписку на свой новый адрес. Нужно изменить его в базе данных. Для этого выполним следующий запрос: UPDATE Visitor SET Email='ivanmukhin@pisem.net' WHERE idVisitor = 1. Для удаления значения из таблицы служит оператор DELETE (DELETE FROM Visitor WHERE idVisitor=1).

Использование SQL

Конечно, в реальных программах операторы SQL не вводятся вручную, однако умение правильно составить запрос может пригодиться любому пользователю для использования недокументированных возможностей ПО.

SQL-запросы формируются программой в виде символьной строки и затем передаются через соответствующий программный интерфейс серверу базы данных. От последнего получается результат выполнения, который передается программе в виде массивов, коллекций и других структур данных. Производители инструментальных средств поставляют собственные драйверы, и это позволяет программисту формировать и выполнять запросы к различным СУБД без переделки программного кода. Хотя при этом все равно приходится учитывать особенности диалектов языка SQL для различных серверов. Но это отдельная тема, и о них мы поговорим в следующий раз.

■ ■ ■ Сергей Бабичев