**База данных «Компьютерная фирма»**

Схема БД состоит из четырех таблиц (рис.1.1):

1. Product(maker, model, type)
2. PC(code, model, speed, ram, hd, cd, price)
3. Laptop(code, model, speed, ram, hd, screen, price)
4. Printer(code, model, color, type, price)

Таблица **Product** представляет производителя (maker), номер модели (model) и тип (PC — ПК, Laptop — портативный компьютер или Printer — принтер). Предполагается, что в этой таблице номера моделей уникальны для всех производителей и типов продуктов.

В таблице **PC** для каждого номера модели, обозначающего ПК, указаны скорость процессора — speed (МГерц), общий объем оперативной памяти - ram (Мбайт), размер диска — hd (в Гбайт), скорость считывающего устройства - cd (например, '4х') и цена — price.

Таблица **Laptop** аналогична таблице РС за исключением того, что вместо скорости CD-привода содержит размер экрана — screen (в дюймах).

В таблице **Printer** для каждой модели принтера указывается, является ли он цветным — color ('y', если цветной), тип принтера — type (лазерный — Laser, струйный — Jet или матричный — Matrix) и цена — price.

Схема данных

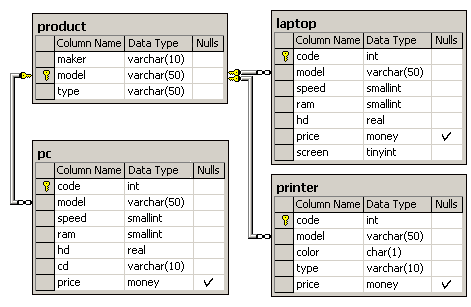


Рис. 1.1. Схема базы данных «Компьютерная фирма»

Дополнительную информацию можно извлечь из представленной на рис. 1.1 логической схемы данных. Таблицы по типам продукции (ПК, портативные компьютеры и принтеры) содержат внешний ключ (model) к таблице Product. Связь «один-ко-многим» означает, что в каждой из этих таблиц может отсутствовать модель, имеющаяся в таблице Product. С другой стороны, модель с одним и тем же номером может встречаться в такой таблице несколько раз, причем даже с полностью идентичными техническими характеристиками, так как первичным ключом здесь является столбец code. Последнее требует пояснения, так как разные люди вкладывают в понятие модели разный смысл. В рамках данной схемы считается, что модель — это единство производителя и технологии. Например, одинаковые модели могут комплектоваться технологически идентичными накопителями, но разной емкости, скажем, 60 и 80 Гбайт. В частности, это означает, что допустимо присутствие в таблице PC двух ПК с одинаковыми номерами модели, но по разной цене.

На языке предметной области данная схема может означать, что в таблице Product содержится информация обо всех известных поставщиках рассматриваемой продукции и моделях, которые они поставляют, а в остальных таблицах находятся имеющиеся в наличии (или продаже) модели. Поэтому вполне возможна ситуация, когда имеется поставщик (maker) с моделями, ни одной из которых нет в наличии.

**База данных «Фирма вторсырья»**

Фирма занимается приемом вторсырья и имеет несколько пунктов приема. Каждый пункт получает деньги для их выдачи сдатчикам в обмен на сырье. Фактически, на схеме представлены две базы данных. В каждой задаче по этой схеме используется только одна пара таблиц (либо с суффиксом «\_o», либо без него).

В таблицах Income\_o и Outcome\_o первичным ключом является пара атрибутов {point, date} — номер пункта приема и дата. Этот ключ должен моделировать ситуацию, когда сведения о получении денег на приемном пункте и их выдаче сдатчикам записываются в базу данных не чаще одного раза в день.

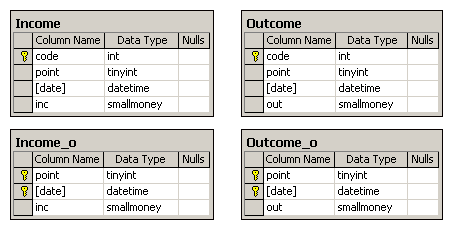


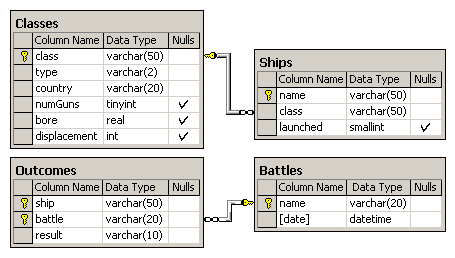
Рис. 2.1. Схема базы данных «Фирма вторсырья»

Таблица Income\_o (point, date, inc) содержит информацию о поступлении денежных сумм (inc) на пункт приема (point). Аналогичная таблица — Outcome\_o (point, date, out) — служит для контроля расхода денежных средств (out).

Вторая пара таблиц — Income (code, point, date, inc) и Outcome (code, point, date, out) — моделирует ситуацию, когда приход и расход денег может фиксироваться несколько раз в день. Следует отметить, что если записывать в последние таблицы только дату без времени (что и имеет место), то никакая естественная комбинация атрибутов не может служить первичным ключом, поскольку суммы денег также могут совпадать. Поэтому нужно либо учитывать время, либо добавить искусственный ключ. Мы использовали второй вариант, добавив целочисленный столбец code только для того, чтобы обеспечить уникальность записей в таблице.

**База данных «Корабли»**

Рассматривается база данных кораблей, участвовавших в морских сражениях второй мировой войны. Имеются следующие отношения:



Корабли в «классах» построены по одному и тому же проекту. Классу присваивается либо имя первого корабля, построенного по данному проекту, либо названию класса дается имя проекта, которое в этом случае не совпадает с именем ни одного из кораблей. Корабль, давший название классу, называется головным.

Атрибутами отношения Classes являются имя класса (class), тип (значение bb используется для обозначения боевого или линейного корабля, а bc для боевого крейсера), страну (country), которой принадлежат корабли данного класса, число главных орудий (numGuns), калибр орудий (bore — диаметр ствола орудия в дюймах) и водоизмещение в тоннах (displacement).

В отношение Ships записывается информация о кораблях: название корабля (name), имя его класса (class) и год спуска на воду (launched).

В отношение Battles включены название (name) и дата битвы (date), в которой участвовал корабль.

Отношение Outcomes используется для хранения информации о результатах участия кораблей в битвах, а именно, имя корабля (ship), название сражения (battle) и чем завершилось сражение для данного корабля (потоплен — sunk, поврежден — damaged или невредим — ok).

Отметим несколько моментов, на которые следует обратить внимание при анализе схемы на рис. 3.1. Таблица Outcomes имеет составной первичный ключ {ship, battle}. Это ограничение не позволит ввести в базу данных дважды один и тот же корабль, принимавший участие в одном и том же сражении. Однако допустимо неоднократное присутствие одного и того же корабля в данной таблице, что означает участие корабля в нескольких битвах. Класс корабля определяется из таблицы Ships, которая имеет внешний ключ (class) к таблице Classes.

Особенностью данной схемы, которая усложняет логику запросов и служит причиной ошибок при решении задач, является то, что таблицы Outcomes и Ships никак не связаны, то есть в таблице результатов сражений могут находиться корабли, отсутствующие в таблице Ships. На основании этого, казалось бы, можно сделать вывод о том, что для таких кораблей их класс неизвестен, а, следовательно, неизвестны и все технические характеристики. Это не совсем так. Как следует из описания предметной области, имя головного корабля совпадает с именем класса, родоначальником которого он является. Поэтому если имя корабля из таблицы Outcomes совпадает с именем класса в таблице Classes, то однозначно можно сказать, что это головной корабль, и, следовательно, все его характеристики нам известны.

**База данных «Аэрофлот»**

Схема БД состоит из четырех таблиц:

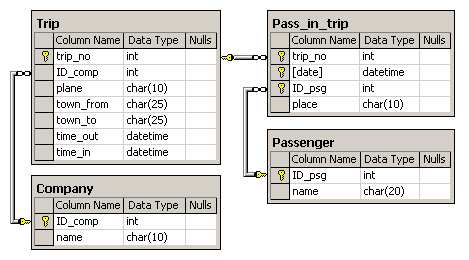


Таблица Company содержит идентификатор и название компании, осуществляющей перевозку пассажиров. Таблица Trip содержит информацию о рейсах: номер рейса, идентификатор компании, тип самолета, город отправления, город прибытия, время отправления и время прибытия. Таблица Passenger содержит идентификатор и имя пассажира. Таблица Pass\_in\_trip содержит информацию о полетах: номер рейса, дата вылета (день), идентификатор пассажира и место, на котором он сидел во время полета. При этом следует иметь в виду, что

- рейсы выполняются ежедневно, а длительность полета любого рейса менее суток;

- время и дата учитывается относительно одного часового пояса;

- время отправления и прибытия указывается с точностью до минуты;

- среди пассажиров могут быть однофамильцы (одинаковые значения поля name, например, Bruce Willis);

- номер места в салоне – это число с буквой; число определяет номер ряда, буква (a – d) – место в ряду слева направо в алфавитном порядке;

- связи и ограничения показаны на схеме данных.

**База данных «Окраска»**

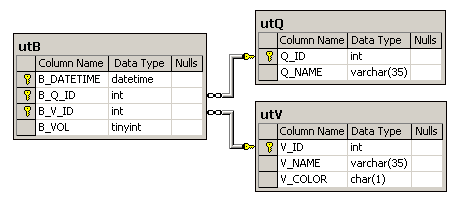


Таблица utQ содержит идентификатор и название квадрата, цвет которого первоначально черный.

Таблица utV содержит идентификатор, название и цвет баллончика с краской.

Таблица utB содержит информацию об окраске квадрата баллончиком: идентификатор квадрата, идентификатор баллончика, количество краски и время окраски.

При этом следует иметь в виду, что:

* баллончик с краской может быть одного из трех цветов - красный V\_COLOR='R', зеленый V\_COLOR='G', синий V\_COLOR='B' (латинские буквы).
* объем баллончика равен 255 и первоначально он полный;
* цвет квадрата определяется по правилу RGB, т.е. R=0,G=0,B=0 - черный, R=255, G=255, B=255 - белый;
* запись в таблице закрасок utB уменьшает количество краски в баллончике на величину B\_VOL и соответственно увеличивает количество краски в квадрате на эту же величину;
* значение 0 < B\_VOL <= 255;
* количество краски одного цвета в квадрате не превышает 255, а количество краски в баллончике не может быть меньше нуля;
* время окраски B\_DATETIME дано с точностью до секунды, т.е. не содержит миллисекунд.

Идентификаторы черных квадратов отсутствуют в таблице utB. Это следует из того, что B\_VOL не допускает NULL-значений и строго больше нуля. Таким образом, каждая запись в таблице utB описывает факт окраски одного квадрата одним баллоном, черный же квадрат (R=0,G=0,B=0) не подвергался окраске вовсе.

Ограничения схемы допускают одновременную окраску одного квадрата несколькими баллонами, как и одновременную окраску одним баллоном нескольких квадратов. Однако запрещена одновременная окраска одного квадрата одним и тем же баллоном.