Функциональные зависимости

Для демонстрации основных идей данной темы, будет использоваться несколько измененная версия отношения поставок SP, содержащая атрибут City, представляющий город соответствующего поставщика. Это измененное отношение будет называться SCP.

Sno	City	Pno	Qty
1	Смоленск	1	100
1	Смоленск	2	100
2	Владимир	1	200
2	Владимир	2	200
3	Владимир	2	300
4	Смоленск	2	400
4	Смоленск	4	400
4	Смоленск	5	400

Следует четко различать:
1. значение этого отношения в определенный момент времени;
2. и набор всех возможных значений, которые данное отношение может принимать в различные моменты времени.

Примеры ФЗ, которым удовлетворяет отношение SCP в данном состоянии:
{ Sno } -> {City}
{Sno,Pno} -> {Qty}
{Sno,Pno} -> {City}
{Sno,Pno} -> { City, Qty }
{Sno,Pno} -> {Sno}
{Sno,Pno} -> { Sno, Pno, City, Qty }
{ Sno } -> {Qty}
{ Qty } -> {Sno}

Определение 1. Пусть R - это отношение, а X и Y - произвольные подмножества множества атрибутов отношения R. Тогда Y функционально зависимо от X, что в символическом виде записывается как X -> Y тогда и только тогда? Когда любое значение множества X связано в точности с одним значением множества Y.

Левая и правая стороны ФЗ будут называются детерминантом и зависимой частью соответственно.

Определение 2. Пусть R является переменной-отношением, а X и Y - произвольными подмножествами множества атрибутов переменной-отношения R. Тогда X -> Y тогда и только тогда, когда для любого допустимого значения отношения R любое значение X связано в точности с одним значением Y.

Определение 2а. Пусть R(A1, A2, ..., An) - схема отношения. Функциональная зависимость, обозначаемая X -> Y между двумя наборами атрибутов X и Y, которые являются подмножествами R определяет ограничение на возможность существования кортежа в некотором отношении r. Ограничение означает, что для любых двух кортежей t1 и t2 в r, для которых имеет место t1[X] = t2[X], также имеет место t1[Y] = t2[Y].

1. Если ограничение на схеме отношения R утверждает, что не может быть более одного кортежа со значением атрибутов X в любом отношении экземпляре отношения r, то X является потенциальным ключом R. Это означает, что X -> Y для любого подмножества атрибутов Y из R. Если X является потенциальным ключ R, то X -> R.
2. Если X -> Y в R, это не означает, что Y -> X в R.

Функциональная зависимость является семантическим свойством, т. е. свойством значения атрибутов.
Примеры безотносительных ко времени ФЗ для переменной-отношения SCP:

{Sno,Pno} -> {Sno,Pno} -> {Sno,Pno} -> {Sno,Pno} -> {Sno,Pno} -> { Sno } ->
{Qty}
{City}
{ City, Qty }
{Sno}
{ Sno, Pno, City, Qty } {City}

Следует обратить внимание на ФЗ, которые выполняются для отношения SCP, но не выполняются «всегда» для переменной-отношения SCP:

{Sno} -> {Qty} {Qty} -> {Sno}
Если X является потенциальным ключом переменной-отношения R, то все атрибуты Y переменной-отношения R должны быть обязательно ФЗ от X (это следует из определения потенциального ключа). Если переменная-отношение R удовлетворяет ФЗ X -> Y и X не является потенциальным ключом, то R будет характеризоваться некоторой избыточностью. Например, в случае отношения SCP сведения о том, что каждый данный поставщик находится в данном городе, будут повторяться много раз (это хорошо видно из таблицы).
Эта избыточность приводит к разным аномалиям обновления, получившим такое название по историческим причинам. Под этим понимаются определенные трудности, появляющиеся при выполнении операций обновления INSERT, DELETE и UPDATE. Рассмотрим избыточность, соответствующую ФЗ (Sno -> City). Ниже поясняются проблемы, которые возникнут при выполнении каждой из указанных операций обновления.

1. Операция INSERT. Нельзя поместить в переменную-отношение SCP информацию о том, что некоторый поставщик находится в определенном городе, не указав сведения хотя бы об одной детали, поставляемой этим поставщиком.
2. Операция DELETE. Если из переменной-отношения SCP удалить кортеж, который является единственным для некоторого поставщика, будет удалена не только информация о поставке поставщиком некоторой детали, но также информация о том, что этот поставщик находится в определенном городе. В действительности проблема заключается в том, что в переменной-отношении SCP содержится слишком много собранной в одном месте информации, поэтому при удалении некоторого кортежа теряется слишком много информации.
3. Операция UPDATE. Название города для каждого поставщика повторяется в переменной-отношении SCP несколько раз, и эта избыточность приводит к возникновению проблем при обновлении.

Для решения всех этих проблем, как предлагалось выше, необходимо выполнить декомпозицию переменной-отношения SCP на две следующие переменные-отношения.
S' { Sno, City }
SP { Sno, Pno, Qty }

Даже если ограничиться рассмотрением ФЗ, которые выполняются «всегда», множество ФЗ, выполняющихся для всех допустимых значений данного отношения, может быть все еще очень большим. Поэтому встает задача сокращения множества ФЗ до компактных размеров. Важность этой задачи вытекает из того, что ФЗ являются ограничениями целостности. Поэтому при каждом обновлении данных в СУБД все они должны быть проверены. Следовательно, для заданного множества ФЗ S желательно найти такое множество T, которое (в идеальной ситуации) было бы гораздо меньшего размера, чем множество S, причем каждая ФЗ множества S могла бы быть заменена ФЗ множества T. Если бы такое множество T было найдено, то в СУБД достаточно было бы использовать ФЗ из множества T, а ФЗ из множества S подразумевались бы автоматически.

Очевидным способом сокращения размера множества ФЗ было бы исключение тривиальных зависимостей, т. е. таких, которые не могут не выполняться. Примером тривиальной ФЗ для отношения SCP может быть { Sno, Pno } -> { Sno }.

Определение 3. ФЗ (X -> Y) тривиальная тогда и только тогда, когда Y включается в X.
Одни ФЗ могут подразумевать другие ФЗ. Например, зависимость
{ Sno, Pno } -> { City, Qty }

подразумевает следующие ФЗ

{ Sno, Pno } -> City { Sno, Pno } -> Qty

В качестве более сложного примера можно привести переменную-отношение R с атрибутами A, B и C, для которых выполняются ФЗ (A -> B) и (B -> C). В этом случае также выполняется ФЗ (A -> C), которая называется транзитивной ФЗ.

Определение 4. Множество всех ФЗ, которые задаются данным множеством ФЗ S, называется замыканием S и обозначается символом S+.

Из сказанного выше становится ясно, что для выполнения сформулированной задачи следует найти способ вычисления S+ на, основе S.

Первая попытка решить эту проблему принадлежит Армстронгу (Armstrong), который предложил набор правил вывода новых ФЗ на основе заданных (эти правила также называются аксиомами Армстронга).

Пусть в перечисленных ниже правилах A, B и C – произвольные подмножества множества атрибутов заданной переменной-отношения R, а символическая запись AB означает { A, B }. Тогда правила вывода определяются следующим образом.

1. Правило **рефлексивности**: (B принадлежит A) следовательно (A -> B)
2. Правило **дополнения**: (A -> B) следовательно AC -> BC
3. Правило **транзитивности**: (A -> B) и (B -> C) следовательно (A -> C)

Каждое и этих правил может быть непосредственно доказано на основе определения ФЗ. Более того, эти правила являются полными в том смысле, что для заданного множества ФЗ S минимальный набор ФЗ, которые подразумевают все зависимости из множества S, может быть выведен из S на основе этих правил. Они также являются исчерпывающими, поскольку никакие дополнительные ФЗ (т.е. ФЗ, которые не подразумеваются ФЗ множества S) с их помощью не могут быть выведены. Иначе говоря, эти правила могут быть использованы для получения замыкания S+.

Из трех описанных выше правил для упрощения задачи практического вычисления замыкания S+ можно вывести несколько дополнительных правил. (Примем, что D - это другое произвольное подмножество множества атрибутов R.)

4. Правило **самоопределения**: A -> A
5. Правило **декомпозиции**: (A -> BC) следовательно (A -> B) и (A -> C)
6. Правило **объединения**: (A -> B) и (A -> C) следовательно (A -> BC)
7. Правило **композиции**: (A -> B) и (C -> D) следовательно (AC -> BD)

Кроме того, Дарвен (Darwen) доказал следующее правило, которое назвал общей теоремой объединения:

8. (A->B) и (C->D) следовательно (A(C–B)->BD).

Упражнение. Пусть дана некоторая переменная-отношение R с атрибутами A, B, C, D, E, F и следующими ФЗ:
A -> BC
B -> E
CD -> EF
Показать, что для переменной-отношения R также выполняется ФЗ (AD -> F), которая вследствие этого принадлежит замыканию заданного множества ФЗ.

1. A -> BC (дано)
2. A -> C (следует из п. 1 согласно правилу декомпозиции)
3. AD -> CD (следует из п. 2 согласно правилу дополнения)
4. CD -> EF (дано)
5. AD -> EF (следует из п. 3 и 4 согласно правилу транзитивности)
6. AD -> F (следует из п. 5 согласно правилу декомпозиции)

Хотя эффективный алгоритм для вычисления замыкания S+ на основе множества ФЗ S так и не сформулирован, можно описать алгоритм, который определяет, будет ли данная ФЗ находиться в данном замыкании. Для этого, прежде всего, следует дать определение понятию **суперключ**.

Определение 5. Суперключ переменной-отношения R - это множество атрибутов переменной-отношения R, которое содержит в виде подмножества (но не обязательно собственного подмножества), по крайней мере, один потенциальный ключ.

Из этого определения следует, что суперключи для данной переменной-отношения R - это такие подмножества K множества атрибутов переменной-отношения R, что ФЗ (K -> A) истинна для каждого атрибута A переменной- отношения R.

Предположим, что известны некоторые ФЗ, выполняющиеся для данной переменной-отношения, и требуется определить потенциальные ключи этой переменной-отношения. По определению потенциальными ключами называются неприводимые суперключи. Таким образом, выясняя, является ли данное множество атрибутов K суперключом, можно в значительной степени продвинуться к выяснению вопроса, является ли K потенциальным ключом. Для этого нужно определить, будет ли набор атрибутов переменной-отношения R множеством всех атрибутов, функционально зависящих от K. Таким образом, для заданного множества зависимостей S, которые выполняются для переменной-отношения R, необходимо найти способ определения множества всех атрибутов переменной-отношения R, которые функционально зависимы от K, т.е. так называемое замыкание K^+ множества K для S.

Алгоритм вычисления этого замыкания имеет вид.

```
function Closure(K: SetOfAttr; S: SetOfFD): SetOfAttr;
var
        Jold, Jnew: SetOfAttr;
begin
        Jnew := K;
        repeat
                Jold := Jnew;
                foreach ((X -> Y in S) do
                        if (X принадлежит Jnew) then Jnew=Jnew + Y;
        until (Jold = Jnew);
        return Jold;
end; // of Closure
```

Определение. Суперключ в схеме отношения R(A1, A2, ... , An) - это набор атрибутов S из R со свойством, что ни для каких двух кортежей t1 и t2 в любом отношении над схемой R не будет выполняться равенство t1[S] = t2[S].

Определение. Потенциальный ключ K - это суперключ с дополнительным свойством: удаление любого атрибута из K приведет к тому, что K перестанет быть суперключом.

Определение. Атрибут в схеме отношения R называется первичным атрибутом R, если он является членом некоторого потенциального ключа R. Атрибут называется непервичным, если он не является первичным атрибутом, то есть, если он не является членом какого-либо потенциального ключа.

Алгоритм нахождения ключа K схемы отношения R для заданного множество функциональных зависимостей F

Вход: Схема отношения R и множество функциональных зависимостей F на атрибутах R.

1. Пусть K := R.

2. Для каждого атрибута из K

```
    {
        вычислить Closure({K - A), F);
        если замыкание содержит все атрибуты из R, то установите K: = K - {A}
    };
```

Заметьте, что алгоритм определяет только один ключ из множества возможных ключей на R; возвращаемый ключ зависит от порядка, в котором атрибуты удаляются из R на шаге 2.

Упражнение. Пусть дана переменная-отношение R с атрибутами A, B, C, D, E и F и следующими ФЗ:
A -> BC
E -> CF
B -> E
CD -> EF
Вычислить замыкание $\{ A, B \}^{+}$ множества атрибутов { A, B }, исходя из заданного множества ФЗ.

1. Присвоим Jold и Jnew начальное значение - множество {A,B}.
2. Выполним внутренний цикл четыре раза - по одному разу для каждой заданной ФЗ. На первой итерации (для зависимости A -> BC) будет обнаружено, что левая часть действительно является подмножеством замыкания Jnew. Таким образом, к Jnew можно добавить атрибуты B и C. Jnew теперь представляет собой множество {A,B,C}.
3. На второй итерации (для зависимости E -> CF) обнаруживается, что левая часть не является подмножеством полученного до этого момента результата, который, таким образом, остается неизменным.
4. На третьей итерации (для зависимости B -> E) к Jnew будет добавлено множество E, которое теперь будет иметь вид {A, B, C, E}.
5. На четвертой итерации (для зависимости CD -> EF) Jnew останется неизменным.
6. Далее внутренний цикл выполняется еще четыре раза. На первой итерации результат останется прежним, на
второй он будет расширен до {A, B, C, E, F}, а на третьей и четвертой - снова не изменится.
7. Наконец после еще одного четырехкратного прохождения цикла замыкание Jnew останется неизменным и весь
процесс завершится с результатом $\{A, B\}^{+}$ = {A, B, C, E, F}. // Конец упр.

Выводы.

1. Для заданного множества ФЗ S легко можно указать, будет ли заданная ФЗ (X -> Y) следовать из S, поскольку это возможно тогда и только тогда, когда множество Y является подмножеством замыкания X^{+} множества X для заданного множества S. Таким образом, представлен простой способ определения, будет ли данная ФЗ (X->Y) включена в замыкание S^{+} множества S.
2. Напомним определение понятия суперключа. Суперключ переменной-отношения R - это множество атрибутов переменной-отношения R, которое в виде подмножества (но необязательно собственного подмножества) содержит по крайней мере один потенциальный ключ. Из этого определения прямо следует, что суперключи для данной пеменной-отношения R - это такие подмножества K множества атрибутов переменной-отношения R, что ФЗ (K -> A) будет истинна для каждого атрибута A переменной-отношения R. Другими словами, множество K является суперключ тогда и только тогда, когда замыкание K^{+} для множества K в пределах заданного мноства ФЗ является множеством абсолютно всех атрибутов переменной-отношения R. (Кроме того, множество K является потенциальным ключом тогда и только тогда, когда оно является неприводимым суперключом).

Определение 6. Два множества ФЗ S1 и S2 эквивалентны тогда и только тогда, когда они являются покрытиями друг для друга, т. е. S1+ = S2+.

Упражнение.
Эквивалентны ли следующие множества ФЗ: F= {A ->C, AC ->D,E ->AD, E ->H} and G = {A ->CD, E ->AH}.
Проверим, что G покрывается множеством F и F покрывается множеством G

G покрывается множеством F:
{A}+ = {A, C, D} (относительно F), покрывается A ->CD из G
{E}+ = {E, A, D, H, C} (относительно F), покрывается E ->AH в G

F покрывается множеством G:
{A}+ = {A, C, D} (относительно G), покрывается A ->C в F

{A, C}+= {A, C, D} (относительно G), покрывается AC ->D в F
{E}+ = {E, A, H, C, D} (относительно G), покрывается E ->AD и E ->H в F

Каждое множество ФЗ эквивалентно, по крайней мере, одному неприводимому множеству.

Определение 7. Множество ФЗ является неприводимым тогда и только тогда, когда оно обладает всеми перечисленными ниже свойствами.

1. Каждая ФЗ этого множества имеет одноэлементную правую часть.
2. Ни одна ФЗ множества не может быть устранена без изменения замыкания этого множества.
3. Ни один атрибут не может быть устранен из левой части любой ФЗ данного множества без изменения замыкания множества.
   Если I является неприводимым множеством, которое эквивалентно множеству S, то проверка выполнения ФЗ из множества I автоматически обеспечит выполнение ФЗ из множества S.

Пример. Рассмотрим переменную-отношение деталей P с функциональными зависимостями, перечисленными ниже.

Pno -> Pname
Pno -> Color
Pno -> Weight
Pno -> City

Нетрудно заметить, что это множество функциональных зависимостей является неприводимым:

1. правая часть каждой зависимости содержит только один атрибут,
2. левая часть, очевидно, является неприводимой,
3. ни одна из функциональных зависимостей не может быть опущена без изменения замыкания множества (т. е. без
   утраты некоторой информации).

В противоположность этому приведенные ниже множества функциональных зависимостей не являются неприводимыми.

1. Pno -> { Pname, Color }
   Pno -> Weight
   Pno -> City

(Правая часть первой ФЗ не является одноэлементным множеством.)

2. { Pno, Pname } -> Color
   Pno -> Pname
   Pno -> Weight
   Pno -> City

(Первую ФЗ можно упростить, опустив атрибут Pname в левой части без изменения замыкания, т. е. она не является неприводимой слева.)

3. Pno -> Pno
   Pno -> Pname
   Pno -> Color
   Pno -> Weight
   Pno -> City

(Здесь первую ФЗ можно опустить без изменения замыкания.)

Можно сделать утверждение, что для любого множества ФЗ существует, по крайней мере, одно эквивалентное множество, которое является неприводимым. Это достаточно легко продемонстрировать на следующем примере. Пусть дано исходное множество ФЗ S. Тогда благодаря правилу декомпозиции можно без утраты общности предположить, что каждая ФЗ в этом множестве S имеет одноэлементную правую часть. Далее для каждой ФЗ f из этого множества S следует проверить каждый атрибут A в левой части зависимости f. Если множество S и множество зависимостей, полученное в результате устранения атрибута A в левой части зависимости f, эквивалентны, значит этот атрибут следует удалить. Затем для каждой оставшейся во множестве S зависимости f, если множества S и S \ {f} эквивалентны, следует удалить зависимость f из множества S. Получившееся в результате таких действий множество S является неприводимым и эквивалентно исходному множеству S.

Алгоритм поиска минимального покрытия F для множества функциональных зависимостей E

Вход: Множество функциональных зависимостей E.
1. Пусть F: = E.
2. Заменить каждую функциональную зависимость X -> {A1, A2, ..., An} из F на n функциональных зависимостей X -> A1, X -> A2, ..., X -> An.
3. Для каждой функциональной зависимости X -> A из F
    1. Для каждого атрибута B, который является элементом X
        1. если { {F – {X -> A} } объединить { (X – {B}) -> A} } эквивалентно F заменить X -> A на (X – {B}) -> A в F.
4. Для каждой оставшейся функциональной зависимости X -> A из F
    1. если {F - {X -> A}} эквивалентно F, удалить X -> A из F.

Упражнение. Пусть дана переменная-отношение R с атрибутами A, B, C, D и следующими функциональными зависимостями.

A -> BC
B -> C
A -> B
AB -> C
AC -> D

Найти неприводимое множество функциональных зависимостей эквивалентное данному множеству.
1. Прежде всего, следует переписать заданные ФЗ таким образом, чтобы каждая из них имела одноэлементную правую часть.

A -> B
A -> C
B -> C
A -> B
AB -> C
AC -> D

Нетрудно заметить, что зависимость A -> B записана дважды, так что одну из них можно удалить.
2. Затем в левой части зависимости AC -> D может быть опущен атрибут C, поскольку дана зависимость A -> C, из которой по правилу дополнения можно получить зависимость A -> AC. Кроме того, дана зависимость AC -> D, из которой по правилу транзитивности можно получить зависимость A -> D. Таким образом, атрибут C в левой части исходной зависимости AC -> D является избыточным.
3. Далее можно заметить, что зависимость AB -> C может быть исключена, поскольку дана зависимость A -> C, из которой по правилу дополнения можно получить зависимость AB -> CB, а затем по правилу декомпозиции - зависимость AB -> C.
4. Наконец зависимость A -> C подразумевается зависимостями A -> B и B -> C, так что она также может быть отброшена. В результате получается неприводимое множество зависимостей.

A -> B
B -> C
A -> D

Множество ФЗ I, которое неприводимо и эквивалентно другому множеству ФЗ S, называется неприводимым покрытием множества S. Таким образом, с тем же успехом в системе вместо исходного множества ФЗ S может использоваться его неприводимое покрытие I (здесь следует повторить, что для вычисления неприводимого эквивалентного покрытия I необязательно вычислять замыкание S+). Однако необходимо отметить, что для заданного множества ФЗ не всегда существует уникальное неприводимое покрытие.

Проектирование базы данных может быть выполнено с использованием двух подходов: снизу вверх (bottom-up) или сверху вниз (top-down).

1. Методология проектирования bottom-up (также называемая методологией синтеза) рассматривает основные связи между отдельными атрибутами в качестве отправной точки и использует их, чтобы построить схемы отношений схем. Этот подход не пользуется популярностью на практике, потому что она страдает от проблемы того, чтобы собрать большое число бинарных связей между атрибутами в качестве отправной точки. На пактике это сделать почти невозможно.
2. Методология проектирования top-down (также называемая методологией анализа) начинается с некоторого набора отношений, состоящих из атрибутов. Затем отношения анализируются отдельно или совместно, в результате чего происходит их декомпозиция до тех пор, пока не будут достигнуты все желаемые свойства.

   Неявными целями обеих методологий являются сохранение информации и минимизация избыточности.

Процесс нормализации схем отношений, основанный на операции декомпозии, должен обладать следующими свойства:

1. неаддитивностью JOIN или JOIN без потерь информации (NJP) , которая гарантирует, что в результате декомпозиции не появятся лишние кортежи.
2. свойством сохранения зависимостей (DPP), которое гарантирует, что каждая функциональная зависимость будет представлена в каком-либо отдельном отношении после декомпозиции.

Свойство NJP является очень критическим и должно быть достигнуто любой ценой. Свойство DPP желательно, но не всегда достижимо.