Базы данных
SQL, часть 2

В.Н.Лукин

29.10.2020

Естественное соединение

Определение. *Пусть* $r(R)$ *и* $s(S)$ *— отношения,* $RS=R \cup S$. *Естественным соединением r и s называется максимальное отношение*

$$r \parallel s = q(RS) = \{t(RS) \mid \exists\, t_r \in r,\ t_s \in s:\ t_r = t(R),\ t_s = t(S)\}$$

Пусть $R \cap S = X \neq \varnothing$. Тогда результат соединения состоит из всех кортежей: $t(X) = t_r(X) = t_s(X)$.

Если $R \cap S = \varnothing$, соединение $r \parallel s$ называется *декартовым произведением* и обозначается $r \times s$.

В нём каждый кортеж $t_r \in r$ соединен с каждым кортежем $t_s \in s$.

Соединение в SQL (1)

В SQL легко реализовать эквисоединение.

С помощью переименования и эквисоединения можно реализовать и естественное соединение.

Но в SQL можно поступить проще: атрибут можно именовать с указанием имени его таблицы, что позволяет обеспечить уникальность имен даже в случае непустого пересечения схем:

<*имя таблицы*>.<*имя атрибута*>.

Соединение в SQL (2)

Рассмотрим простейший вариант эквисоединения, отвечающем операции реляционной алгебры:

$$r[A_1 = B_1, A_2 = B_2, \ldots, A_m = B_m]s,$$

где r и s — соединяемые отношения, а $A_1, A_2, \ldots, A_m$ и $B_1, B_2, \ldots, B_m$ — атрибуты, по равенству которых производится соединение.

В SQL соответствующий оператор выглядит так:

Select <*список атрибутов*> ***From*** *r*, *s*
Where $A_1 = B_1$ ***and*** $A_2 = B_2$ ***and***... ***and*** $A_m = B_m$;

Если вместо равенства поставить любой допустимый знак отношения, получится θ-соединение.

Соединение в SQL (3)

Этот вид соединения основан на ссылочной целостности. Если для значения одного из атрибутов нет равного ему значения второго, запись в выборку не включается.

Если один из атрибутов – родительский ключ, а другой внешний, необходимое условие включения записи в выборку – существование значения родительского ключа, равного любому внешнему.

Это и есть условие ссылочной целостности. В качестве родительского ключа обычно выбирают первичный.

Соединение в SQL
пример (1)

Подобрать продавцов и покупателей, живущих в одном городе.

Select *Имя_пок*, *Имя_прод*, *покупатель.Город*
From *покупатель*, *продавец*
Where *покупатель.Город = продавец.Город*;

В списке атрибутов *Имя_пок* и *Имя_прод* однозначно определяют таблицы, из которых выбираются.

Атрибут *Город* есть в обеих таблицах, поэтому следует уточнить, атрибут какой таблицы выбирается. В данном случае это безразлично, так как их значения равны.

Соединение в SQL
пример (2)

Теперь *Ном_пок* в таблице *покупатель* — первичный (родительский) ключ, а в таблице *заказ* — внешний.

Select *Имя_пок, Ном_зак*
From *покупатель, заказ*
Where *заказ.Ном_пок = покупатель.Ном_пок*;

На один внешний ключ приходится ровно один родительский, значит, количество записей в выборке равно количеству записей в таблице *заказ*.

Если значений внешнего ключа нет в родительских, записей станет меньше (нарушение ссылочной целостности).

Если для одного значения внешнего ключа нашлось более одного значения родительского, записей станет больше (нарушение условия однозначности первичного ключа).

Многократное соединение (1)

Соединение может применяться не только к двум таблицам: их может быть и больше, и меньше.

Во втором случае соединяемые таблицы должны иметь разные имена, иначе одноименные атрибуты различить невозможно.

Для выхода из положения таблице присваивают локальное, временное имя, которое используется наряду с основным. Это имя называется *переменной области определения*, или *переменной корреляции*, или *псевдонимом*.

Многократное соединение (2)

Таблица может иметь несколько псевдонимов.

Существует два формата определения псевдонима:

Select *<список атрибутов>* ***From*** *r пн*1*, s пн*2 …

Select *<список атрибутов>* ***From*** *r* ***As*** *пн*1, *s* ***As*** *пн*2 …

Здесь *пн*1 и *пн*2 — псевдонимы таблиц *r* и *s* соответственно.

Многократное соединение пример (1)

Найти заказы покупателей из городов, не совпадающих с городами продавцов. В этом случае соединяются три таблицы.

Select *Ном_зак, Имя_пок* ***From*** *заказ, покупатель, продавец*
Where *заказ.Ном_пок = покупатель.Ном_пок*
and *заказ.Ном_прод = продавец.Ном_прод*
and *покупатель.Город <> продавец.Город;*

Многократное соединение пример (2)

Найти покупателей, имеющих попарно равную значимость. В этом случае одна таблица соединяется с собой же.

Select *a.Имя_пок* ***As*** *Пок*1, *b.Имя_пок* ***As*** *Пок*2,
a.Значимость
From *покупатель* ***As*** *a, покупатель* ***As*** *b*
Where *a.Значимость = b.Значимость*;

Результат получается правильным, но неудобным: возникает много повторов из-за перестановок значений атрибутов.

Многократное соединение пример (3)

Чтобы избавиться от повторов, существует следующий нехитрый прием:

Select *a.Имя_пок, b.Имя_пок, a.Значимость*

From *покупатель* ***As*** *a, покупатель* ***As*** *b*

Where *a.Значимость = b.Значимость*

and *a.Имя_пок < b.Имя_пок;*

Соединение таблицы с собой пример (1)

Рассмотрим базу данных для генеалогического дерева, где в одной таблице хранятся данные о человеке и его родителях. Отношение с рекурсивной схемой:

человек(*Ид, ФИО, Отец, Мать*).

Здесь *Ид, Отец, Мать* — атрибуты над одним доменом: идентификатором человека. Ключ — атрибут *Ид*. *Отец* и *Мать* — внешние ключи, которые ссылаются на ту же таблицу. Необходимо выдать список людей с их родителями.

Соединение таблицы с собой пример (2)

Если есть две таблицы: списки людей и отцов, задача легко решается соединением этих таблиц по ключам (*человек.Отец*, *отец.Ид*). Аналогично с таблицей матерей.

Если таблица одна, используем ее в запросе трижды, давая ей разные псевдонимы:

Select *ребенок.ФИО, отец.ФИО, мать.ФИО*
From *человек* ***As*** *ребенок, человек* ***As*** *отец,*
человек ***As*** *мать*
Where *ребенок.Отец = отец.Ид*
and *ребенок.Мать = мать.Ид*;

Вложенные запросы (1)

В SQL множество значений может задаваться запросом, который заключается в круглые скобки.

Полученное множество содержит атомарные элементы: запрос должен выбирать один атрибут.

В результате запроса множество может быть пустым, содержать один элемент или более одного.

Если множество содержит один элемент, для него определены обычные операции сравнения.

Если элементов больше, определена лишь операция принадлежности элемента множеству.

Вложенные запросы (2)

В результате запроса может образоваться несколько одинаковых значений и по умолчанию все они включаются в выборку.

То есть на самом деле речь идет не о множестве, а о мультимножестве. Использование варианта ***Distinct*** приводит к исключению повторов.

Запрос, формирующий множество, называют вложенным запросом. Вложенный запрос сам может иметь вложенный запрос.

Вложенные запросы пример (1)

Найти все заказы, которые выполняет продавец Сидоров.

Select *** ***From*** *заказ*

Where *Ном_прод =* (***Select*** *Ном_прод* ***From*** *продавец*

Where *Имя_прод = ’Сидоров’*);

Запрос корректен, если есть ровно один продавец Сидоров. Если такого нет — результат не определен, если их более одного — результат ошибочен.

.

Вложенные запросы пример (2)

Найти все заказы, сумма которых больше, чем средняя за 4 октября.

Select * ***From*** *заказ*
Where *Сумма* > (***Select*** *AVG*(*Сумма*) ***From*** *заказ*
Where *Дата* = ’04.10’);

.

Вложенные запросы пример (3)

Найти все заказы для продавцов из Москвы.

Select *** ***From*** *заказ*
Where *Ном_прод* ***in*** (***Select*** *Ном_прод* ***From*** *продавец*
Where *Город = ’Москва’*);

Такой же результат можно получить, используя операцию соединения:

Select *a.** ***From*** *заказ* ***As*** *a, продавец* ***As*** *b*
Where *a.Ном_прод = b.Ном_прод*
and *Город = ’Москва’*;

Вложенные запросы
пример (4)

Найти комиссионные всех продавцов, обслуживающих покупателей из Москвы.

Select *Комиссия* ***From*** *продавец*
Where *Ном_прод* ***in***
(***Select*** *Ном_прод* ***From*** *заказ*
Where *Ном_пок* ***in***
(***Select*** *Ном_пок* ***From*** *покупатель*
Where *Город = ’Москва’*));

Вложенные запросы пример (5)

Определить количество покупателей, имеющих значимость, превышающую среднюю для покупателей из Тулы.

Select *Значимость, COUNT*(***Distinct*** *Ном_пок*)
From *покупатель*
Having *Значимость* >
(***Select*** *AVG*(*Значимость*) ***From*** *покупатель*
Where *Город = ’Тула’*)
Group By *Значимость*;

Надо учитывать, что время на выполнение запроса разными способами может существенно разниться.

Связанные запросы

Для реализации вложенного запроса может потребоваться информация из таблиц внешнего запроса. Запросы такого типа называются *связанными*.

При вычислении вложенного запроса внутренний подзапрос вычисляется автономно до выполнения внешнего запроса.

Выполнение внутреннего подзапроса зависит от состояния таблиц, объявленных во внешнем запросе. Он должен выполняться для каждой строки внешней таблицы, от которой зависит.

Связанные запросы правила выполнения

1. Выбрать очередную строку внешней таблицы, от которой зависит внутренний подзапрос, она называется *строкой-кандидатом*.
2. Сохранить ее под псевдонимом, указанном во внешней фразе ***From***.
3. Выполнить подзапрос. Используемые значения строки-кандидата называются *внешними ссылками*.
4. Оценить результат внешнего запроса на основании результата подзапроса.
5. Повторить процедуру для следующих строк.

Связанные запросы
пример (1)

Выбрать всех покупателей, которые сделали заказы 3 октября.

Select * ***From*** *покупатель* ***As*** *a*
Where ’03.10’ ***in*** (***Select*** *Дата* ***From*** *заказ* ***As*** *b*
Where *a.Ном_пок = b.Ном_пок*);

Связанный запрос можно заменить соединением. Нужно внимательно отнестись к выбору варианта, от этого существенно зависит производительность.

Select Distinct *a.** ***From*** *покупатель* ***As*** *a*, *заказ* ***As*** *b*
Where *a.Ном_пок = b.Ном_пок* ***and*** *Дата =* ’03.10’;

Связанные запросы
пример (2)

Выбрать имена и номера продавцов, обслуживающих более одного покупателя.

Select *Ном_прод, Имя_прод* ***From*** *продавец* ***As*** *a*
Where 1 < (***Select*** *COUNT*(***Distinct*** *Ном_пок*) ***From*** *заказ*
Where *Ном_прод = a.Ном_прод*);

Связанный запрос во внутреннем подзапросе может содержать ту же таблицу, что и во внешнем. Тогда одна и та же таблица в разных подзапросах играет разные роли. Одновременно с выборкой строки-кандидата фиксируется и состояние таблицы, чтобы восстановить его после выполнения.

Связанные запросы
пример (3)

Для каждого покупателя выбрать все заказы на сумму, большую средней суммы его заказов.

Select * From *заказ* ***As*** *a*
Where *Сумма* > (***Select*** *AVG*(*Сумма*) ***From*** *заказ* ***As*** *b*
Where *a.Ном_пок = b.Ном_пок*);

Так же, как и в случае вложенного запроса, связанный подзапрос может использоваться во фразе ***Having***, если условие накладывается на группу записей.

Предикаты, определенные на подзапросах

В состав логических выражений SQL могут входить предикаты, определенные на подзапросах:

- признак того, что подзапрос не пуст (***Exists***),
- признак того, что все элементы удовлетворяют некоторому условию (***All***),
- признак того, что существует хотя бы один элемент, удовлетворяющий некоторому условию (***Any***, ***Some***).

Предикаты
Exists

Функция ***Exists*** истинна, если ее аргумент (подзапрос) содержит хотя бы один элемент, в противном случае она ложна.

Легко видеть, что в подзапросе этой функции использование агрегатных функций бессмысленно.

Рассмотрим применение функции ***Exists*** на примерах.

Предикаты
Exists, пример (1)

Выбрать всех покупателей из Тулы, если хотя бы один тульский покупатель сделал заказ.

Select *** ***From*** *покупатель*
Where *Город = ’Тула’* ***and Exists***
(***Select*** *** ***From*** *заказ* ***Where*** *Ном_пок* ***in***
(***Select*** *Ном_пок* ***From*** *покупатель*
Where *Город = ’Тула’*));

Здесь в конструкция ***Select*** *… используется в подзапросе. Кроме ***Exists*** в подзапросе вариант «*» не употребляется.

Предикаты
Exists, пример (2)

Выбрать номера всех продавцов, у которых более одного покупателя.

Select Distinct *Ном_прод* ***From*** *заказ* ***As*** *a*
Where Exists(***Select*** * ***From*** *заказ* ***As*** *b*
Where *a.Ном_прод = b.Ном_прод*
and *a.Ном_пок <> b.Ном_пок*);

В данном случае ***Exists*** используется в сочетании со связанным подзапросом.

Предикаты
Exists, пример (3)

Дополним предыдущий пример соединением: выберем имена всех продавцов, у которых более одного покупателя.

```
Select Distinct a.Ном_прод, Имя_прод
    From заказ As a, продавец As c
    Where Exists(Select * From заказ As b
        Where a.Ном_прод = b.Ном_прод
            and a.Ном_пок <> b.Ном_пок)
        and a.Ном_прод = c.Ном_прод;
```

Предикаты
All

Предикат ***All*** имеет вид

<переменная> θ ***All*** (*<подзапрос>*),

где θ — операция сравнения.

Он истинен, если каждое значение подзапроса удовлетворяет условию.

Для операции равенства применяется редко: все элементы должны быть равны между собой.

Неравенство обозначает, что левая часть не равна ни одному из элементов выборки.

В этом случае он легко реализуется операцией ***in***.

Предикаты
All, пример

Более интересны операции «больше», «меньше», …

Выбрать всех покупателей, значимость которых выше значимости любого покупателя из Тулы:

Select *** ***From*** *покупатель*
Where *Значимость* > ***All***(***Select*** *Значимость* ***From*** *покупатель*
Where *Город = 'Тула'*);

Этот же запрос с ***Exists***:

Select * From *покупатель* ***As*** *a*
Where not Exists(***Select * From*** *покупатель* ***As*** *b*
Where *a.Значимость <= b.Значимость*
and *b.Город = 'Тула'*);

Предикаты
Any (синоним — *Some*)

Предикат *Any* имеет вид

<*переменная*> θ ***Any*** (<*подзапрос*>),

где θ — операция сравнения.

Он истинен, если хотя бы одно значение подзапроса удовлетворяет условию.

Этот предикат чаще используется для операции равенства, чем для «больше», «меньше» и т. п.

Предикаты
Any, пример

Выбрать всех покупателей, живущих в городе, где есть продавец :

***Select** * **From** покупатель*
***Where** Город = **Any**(**Select** Город **From** продавец*);

Этот же запрос с ***in***:

***Select** * **From** покупатель*
***Where** Город **in** (**Select** Город **From** продавец*);

Этот же запрос с ***Exists***:

***Select** * **From** покупатель **As** a*
***Where not Exists**(**Select** * **From** продавец **As** b*
***Where** a.Город = b.Город*);

Предикаты

Запрос с ***Any*** и с ***All*** может быть выражен через ***Exists***, но часто требуется более глубокая вложенность и встречаются связанные подзапросы.

Уточним поведение предикатов ***Any***, ***All*** и ***Exists*** в особых случаях.

Для пустой выборки ***All*** принимает значение *истина*, ***Any*** и ***Exists*** — *ложь*.

При сравнении с ***null*** предикаты ***Any*** и ***All*** принимают неопределенное значение, ***Exists*** никогда неопределенное значение не принимает.

Объединение (1)

Операция объединения — ***Union***, объединяются два независимых подзапроса.

Она объединяет два множества, значит, элементы исходных множеств должны быть однотипными. В каждом подзапросе должно быть одинаковое количество столбцов, они должны быть сравнимы.

Согласно стандарту ANSI сравнимость сводится к тому, чтобы тип и размер столбцов совпадали.

Другое ограничение: если ***null***-значения запрещены в одном подзапросе, они должны быть запрещены и в другом.

Объединение (2)

Нельзя использовать ***Union*** в подзапросах, а в объединяемых выборках — агрегатные функции. В конкретных реализациях могут быть и другие ограничения.

При объединении из результата автоматически исключаются тождественные строки в отличие от команды ***Select***. Чтобы их оставить, следует использовать вариант ***Union All***.

Для получения требуемого порядка строк в выборке используют, как обычно, фразу ***Order By***, она ставится единственный раз, причем в конце.

Объединение
пример

Выбрать номера покупателей, значимость которых выше 200 или которые сделали заказ на сумму более 3000.

Select *Ном_пок* ***From*** *Покупатель*

Where *Значимость* > 200

Union

Select *Ном_пок* ***From*** *Заказ*

Where *Сумма* > 3000 ***Order By*** *Ном_пок*;

.

Объединение (3)

Объединение — операция над двумя операндами, поэтому для объединения более чем двух выборок используют скобочные конструкции.

(***Select*** *<выборка* 1> ***Union*** [***All***] ***Select*** *<выборка* 2>)

Union [***All***]

(***Select*** *<выборка* 3> ***Union*** [***All***] ***Select*** *<выборка* 4>) …

Очевидно, что различный порядок выполнения действий при использовании ***Union*** и ***Union All*** может привести к разным результатам.

Добавление данных (1)

Пусть схема содержит *n* атрибутов.

Добавление данных в таблицу — команда ***Insert***:

Insert Into <*таблица*> ***Values*** (<*знач*$_1$>, <*знач*$_2$>, …<*знач*$_n$>);

Согласно стандарту *ANSI*, пустое значение (***null***) не может быть записано в таблицу непосредственно.

Однако оно возникает, если воспользоваться формой записи из двух списков: атрибуты и их значения. В этом случае могут быть указаны не все атрибуты:

Insert Into <*таблица*> (<*атр*$_1$>, <*атр*$_2$>, …<*атр*$_k$>)
Values (<*знач*$_1$>, <*знач*$_2$>, …<*знач*$_k$>);

Добавление данных (2)

Вместо указания значений можно использовать запрос. Существует два варианта: с заполнением всех атрибутов без их перечисления и заполнение части атрибутов с их указанием в списке.

Insert Into <*таблица*>
Select … From … Where …;

Insert Into <*таблица*> (<amp_1>, <amp_2>, …<amp_k>)
Select … From … Where …;

Еще одна возможность: в качестве значения <$знач_i$> может быть задано любое допустимое выражение.

Добавление данных (3)

Очевидные замечания:

1) таблица уже должна существовать;
2) полученные значения по смыслу должны соответствовать схеме таблицы или списку атрибутов;
3) типы (домены) выбираемых значений должны соответствовать типам (доменам) атрибутов таблицы;
4) добавленные записи не должны нарушать уникальность ключа.

Добавление данных пример

Добавить заказ в таблицу заказов.

Insert Into *заказ* ***Values*** (337, 5500, 28, 17, ’07.10’);

Добавить итоги в таблицу итогов, если она заранее создана.

Insert Into *итоги* (*Дата, Итог*)

Select *Дата, SUM*(*Сумма*) ***From*** *заказ*

Group By *Дата*;

Удаление строк данных

Исключение строк из таблицы производится командой ***Delete***.

Существует два варианта команды:

- для очистки таблицы целиком ;
- для удаления найденных строк.

Delete From <*таблица*>;

Delete From <*таблица*> ***Where*** <*условие*>;

Изменение значений атрибутов

Изменение значений производится командой ***Update***, которая имеет две модификации: меняются значения во всей таблице, и в пределах выборки:

Update <*таблица*>
    Set <$атр_1$> = <$знач_1$>, …<$атр_k$> = <$знач_k$>;
Update <*таблица*>
    Set <$атр_1$> = <$знач_1$>, …<$атр_k$> = <$знач_k$>
    Where <*условие*>;

В команде изменения значение может быть задано выражение. В условии могут использоваться подзапросы.

Изменение значений атрибутов пример

Увеличить значимость всех покупателей из Тулы на 10:

Update *покупатель*
Set *Значимость = Значимость* + 10
Where *Город = 'Тула'*

Увеличить комиссионные всем продавцам, имеющим более трех покупателей, на 0,1:

Update *продавец*
Set *Комиссия = Комиссия* + 0.1
Where 3 < (***Select*** *COUNT*(***Distinct*** *Ном_пок*)
From *заказ*
Where *заказ.Ном_прод = продавец.Ном_прод*);

Создание таблицы (1)

Создание таблицы производится командой:

Create Table *<таблица>*

(*<атр$_1$> <тип$_1$>* [*<(размер$_1$)>*] [*<ограничения$_1$>*]*,*

<атр$_2$> <тип$_2$> [*<(размер$_2$)>*] [*<ограничения$_2$>*]*,*

...

<атр$_n$> <тип$_n$> [*<(размер$_n$)>*] [*<ограничения$_n$>*]

[*,* ***Primary Key*** (*<первичный ключ>*)]*;*

[*,* ***Foreign Key*** (*<внешний ключ>*)

References *<таблица$_2$> <родительский ключ>*])*;*

Создание таблицы (2)

Здесь:

- *<таблица>* — имя создаваемой таблицы,
- *<атр>* — имя атрибута,
- *<тип>* — тип атрибута,
- <(*размер*)> — размер атрибута,
- *<ограничения>* — перечень ограничений целостности,
- *<первичный ключ>* — перечень атрибутов, входящих в первичный ключ,
- *<внешний ключ>* — перечень атрибутов, входящих во внешний ключ,
- *<таблица2>* — таблица, на которую ссылается данная,
- *<родительский ключ>* — атрибуты в *таблице*2, на которые ссылается внешний ключ.

Создание таблицы (3)

Ограничения на атрибут накладываются для контроля целостности. Они могут быть разными:

- запрет на пустоту (***not null***),
- указание на уникальность значения (***unique***),
- атрибут — первичный ключ (***primary key***)
- атрибут — внешний ключ (***references*** *таблица2>* (<*родительский ключ*>)),
- необходимость удовлетворять условиям (***check***(<*предикат*>)),
- значение по умолчанию (***default*** = <значение>).

Создание таблицы (4)

Первичный ключ описывается отдельно, если в его состав входит более одного атрибута. Список его атрибутов записывается через запятую.

То же относится и к внешнему ключу, состоящему из нескольких атрибутов. Здесь нужно обеспечить сравнимость соответствующих атрибутов из внешнего и родительского ключей.

Необходимо обеспечить ссылочную целостность: на каждое значение внешнего ключа должно быть соответствующее значение родительского.

Создание таблицы
пример (1)

Создадим таблицы для нашего глобального примера.

Create Table *продавец*

(*Ном_прод*	***Numeric***(2)	***primary key,***
Имя_прод	***Char***(40)	***not null,***
Город	***Char***(20)	***default****='Москва'*,
Комиссия	***Numeric***(4,2)	***check***(*Комиссия* < 1));

Create Table *покупатель*

(*Ном_пок*	***Numeric***(2)	***primary key,***
Имя_пок	***Char***(40)	***not null,***
Город	***Char***(20)	***default*** *= 'Москва'*,
Значимость	***Numeric***(3));	

Создание таблицы пример (2)

```
Create Table заказ
    (Ном_зак          Numeric(3)     primary key,
     Сумма            Numeric(7,1)   not null,
     Ном_пок          Numeric(2)     references
                                     покупатель(Ном_пок),
     Ном_прод Numeric(2)             references
                                     продавец(Ном_прод),
    Дата              Date           not null);
```

Изменение структуры таблицы (1)

Выполняется командой ***Alter Table***.

Применение её к рабочей таблице рискованно, лучше создать новую таблицу и загрузить в нее данные из старой.

Команда ***Alter Table*** добавляет и удаляет атрибуты, изменяет их описание, изменяет описание таблицы.

Alter Table *<таблица>* ***Add*** *<атр> <тип>* [*<(размер)>*] [*<ограничения>*]*;*

Alter Table *<таблица>* ***Alter*** *<атр>*
Set *<значение по умолчанию>;*

Alter Table *<таблица>* ***Drop*** *<атр> <стиль удаления>;*

Изменение структуры таблицы (2)

Описание добавления атрибута аналогично описанию при создании таблицы.

Изменение атрибута затрагивает лишь его значение по умолчанию. Есть два варианта: ***default***=<значение> и ***Drop default***. В первом случае оно изменяется, во втором — убирается (по умолчанию станет ***null***).

Изменение не влияет на уже существующие значения.

Удаление атрибута выполняется в одном из стилей:

- строгом (***Restrict***) – команда отвергается, если атрибут используется в ограничениях целостности других атрибутов;
- каскадном (***Cascade***) – удаляются все ограничения целостности с этим атрибутом.

В любом варианте команда отвергается, если удаляется последний атрибут таблицы.

Удаление таблицы

Удаление таблицы может быть выполнено тогда, когда она пустая, то есть из нее предварительно были удалены все данные.

Команда удаления:

Drop Table <*таблица*> <*стиль удаления*>;

Удаление таблицы выполняется в одном из стилей:

- строгом (***Restrict***) – команда отвергается, если имя таблицы используется в ограничениях целостности;
- каскадном (***Cascade***) – удаляются все ограничения целостности с этим именем.

Индексы

Чтобы ускорить поиск в таблицах, SQL предоставляет возможность использовать индексы.

Создание индекса производится следующей командой:

Create Index *<имя индекса>*

On *<таблица>* (*<amp$_1$>*, *<amp$_2$>*, …*<amp$_k$>*);

Если создается уникальный индекс, вместо ***Create Index*** используется вариант ***Create Unique Index***.

Удаление индекса производится командой

Drop Index *<имя индекса>*;

Варианты соединения в SQL (1)

Для корректности операции соединения необходима ссылочная целостность.

Предположим, в торговый зал магазина попал товар, код которого по небрежности или в спешке не внесли в номенклатуру. Товар был успешно продан, сведения о продаже попали в базу, но при выдаче финансового отчёта за день выяснилась недостача: сведения о продаже этого товара не вошли в отчёт.

Причина понятная: записи о продаже товара не соединились с соответствующей записью номенклатуры, потому что её не было.

Варианты соединения в SQL (2)

Неполнота естественного и θ-соединения была замечена довольно давно.

Ещё в 1979 г. Э. Кодд предложил ввести новый вид соединения – внешнее соединение, чтобы избежать потери информации при попытке соединения несоединимых кортежей.

Для этого несоединимый кортеж дополнялся атрибутами другого отношения с пустыми значениями.

Понятно, что может быть три варианта операции: левое, правое и полное внешние соединения.

Варианты соединения в SQL (3)

В первом случае в соединение входят все кортежи первого отношения. Если для них есть соединимые кортежи второго отношения, результат формируется обычным образом, если нет – вместо значений атрибутов второго кортежа в результат записываются пустые значения.

Правое внешнее соединение устроено так же, но роли первого и второго отношения меняются.

Полное внешнее соединение не допускает потери ни справа, ни слева.

14 видов соединения в SQL (1)

Эти операции можно выразить известными нам средствами SQL, но их представления выглядят громоздко.

Поэтому в стандарте предусмотрены специальные средства, составляющие подъязык соединений.

В нём рассматривается 14 видов соединения:

1) внутреннее по условию;
2) внутреннее по совпадению значений указанных одноимённых столбцов;
3) естественное внутреннее;

14 видов соединения в SQL (2)

4) левое внешнее по условию;
5) правое внешнее по условию;
6) полное внешнее по условию;
7) левое внешнее по совпадению значений указанных одноимённых столбцов;
8) правое внешнее по совпадению значений указанных одноимённых столбцов;
9) полное внешнее по совпадению значений указанных одноимённых столбцов;
10) естественное левое;
11) естественное правое;
12) естественное полное;
13) соединение объединением;
14) прямое

14 видов соединения в SQL примеры

Покажем суть каждого типа соединений на примерах.
Даны списки преподавателей, дисциплин и таблица, где указано, кто какие дисциплины может вести.

Преп

идП	*ФИО*
1	Иванов
4	Петров
5	Сидоров
2	Егоров

Дисц

идД	*ДСЦ*
1	БД
2	ЧМ
3	ФАН
4	КА
5	ТВ

ПД

идП	*идД*
1	1
4	1
5	2
2	3
3	4

14 видов соединения в SQL (1)

1. Внутреннее по условию

Преп ***inner join*** *ПД* ***on*** *Преп.идП = ПД.идП*

идП	*ФИО*	*ПД.идП*	*идД*
1	Иванов	1	1
4	Петров	4	1
5	Сидоров	5	2
5	Сидоров	5	3

14 видов соединения в SQL (2)

2. Внутреннее по совпадению значений указанных одноимённых столбцов (здесь совпадает с 1)

Преп ***inner join*** *ПД* ***using*** (*идП*)

3. Естественное внутреннее

Преп ***natural inner join*** *ПД*

Преп ***inner join*** *ПД* ***using*** (*идП*)

14 видов соединения в SQL (2а)

4. Левое внешнее по условию (для каждой дисциплины – кто ведёт)

Дисц ***left outer join*** *ПД* ***on*** *Дисц.идД = ПД.идД*

идД	*ДСЦ*	*ПД.идП*	*идД*
1	БД	1	1
1	БД	4	1
2	ЧМ	5	2
3	ФАН	5	3
4	КА	3	4
5	ТВ		5

14 видов соединения в SQL (3)

5. Правое внешнее по условию (для преподавателя – кто ведёт дисциплины (3 уволился))

Преп ***right outer join*** *ПД* ***on*** *Преп.идП = ПД.идП*

идП	*ФИО*	*ПД.идП*	*идД*
1	Иванов	1	1
4	Петров	4	1
5	Сидоров	5	2
5	Сидоров	5	3
		3	4

14 видов соединения в SQL (3а)

6. Полное внешнее по условию (кто ведёт дисциплины и кто свободен (2))

Преп ***full outer join*** *ПД* ***on*** *Преп.идП = ПД.идП*

идП	*ФИО*	*ПД.идП*	*идД*
1	Иванов	1	1
4	Петров	4	1
5	Сидоров	5	2
5	Сидоров	5	3
2	Егоров		
		3	4

14 видов соединения в SQL (3б)

7. Левое внешнее по совпадению значений указанных одноимённых столбцов (для каждого преподавателя – кто ведёт)

Преп ***left outer join*** *ПД* ***using***(*идП*)

идП	*ФИО*	*ПД.идП*	*идД*
1	Иванов	1	1
4	Петров	4	1
5	Сидоров	5	2
5	Сидоров	5	3
2	Егоров		

14 видов соединения в SQL (4)

8. Правое внешнее по совпадению значений указанных одноимённых столбцов (кто ведёт дисциплины и кто свободен (2)) (здесь совпадает с 5)

Преп ***right outer join*** *ПД* ***using***(*идП*)

9. Полное внешнее по совпадению значений указанных одноимённых столбцов (здесь совп. с 6)

Преп ***full outer join*** *ПД* ***using***(*идП*)

10. Естественное левое

Преп ***natural left outer join*** *ПД*

Преп ***left outer join*** *ПД* ***using***(*идП*)

14 видов соединения в SQL (5)

11. Естественное правое

Дисц ***natural right outer join*** *ПД*

Дисц ***right outer join*** *ПД* ***using***(*идД*)

12. Естественное полное

Дисц ***natural full outer join*** *ПД*

Дисц ***full outer join*** *ПД* ***using***(*идД*)

13. Соединение объединением

Преп ***union join*** *ПД*

14. Прямое

Преп ***cross join*** *Дисц*

Конец второй части

На этом краткое знакомство с языком SQL и предоставляемыми им возможностями заканчивается.

Подробно его можно изучить по литературе, приведенной в библиографии.

Для знакомства с конкретной реализацией следует обратиться к специальной литературе.

Литература

1. *Грабер М*. Введение в SQL.
2. *Дейт К.* Введение в системы баз данных.
3. *Дейт К*. Руководство по реляционной СУБД DB2.
4. *Конноли Т. и др*. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.
5. *Кренке Д.* Теория и практика построения баз данных.
6. *Кузнецов С. Д.* Основы баз данных: учебное пособие.