МИНОБРНАУКИ РОССИИ

Передовые инженерные школы

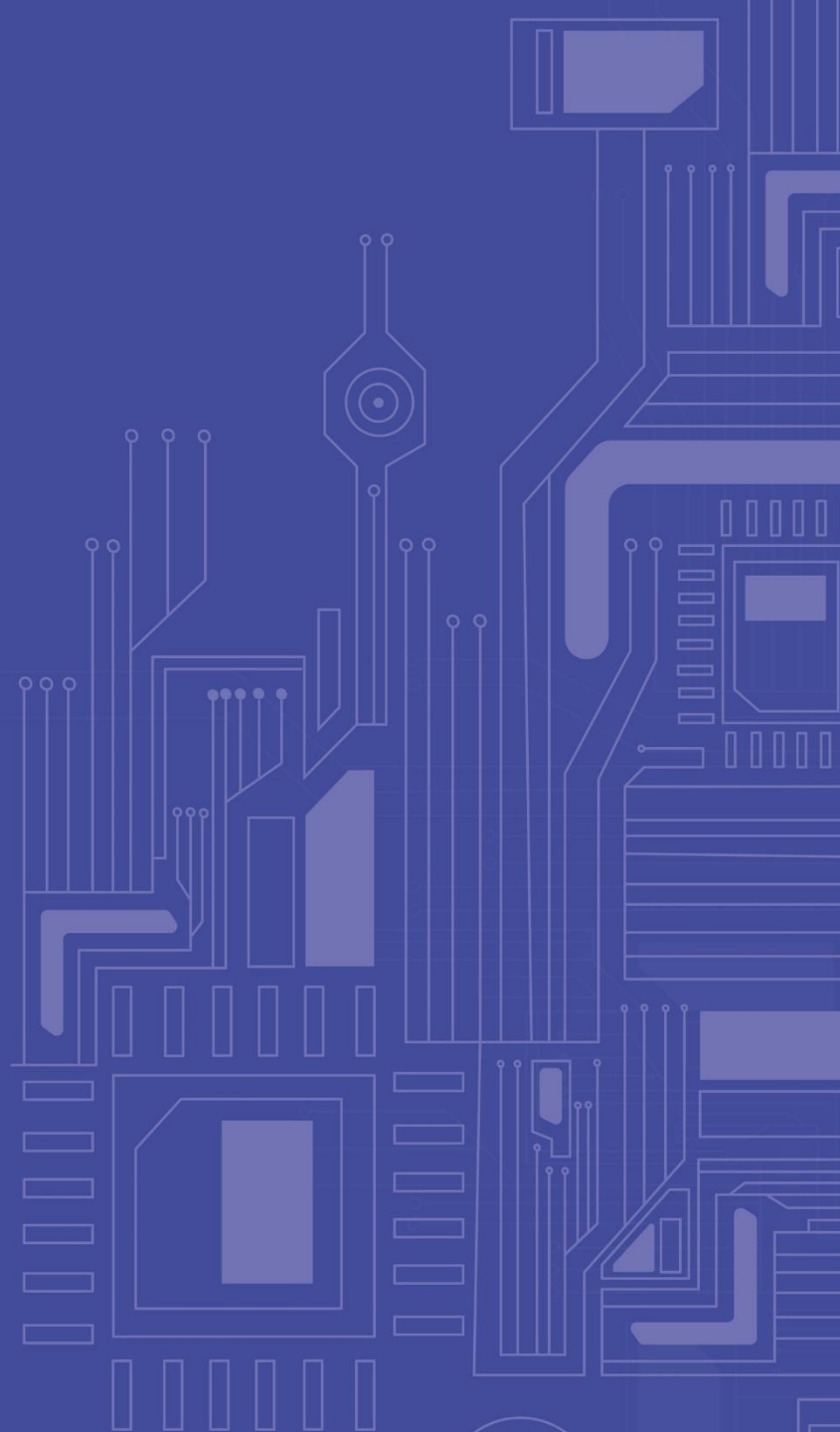
СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Лекция 9

ПЛАН ЛЕКЦИИ

- Типы данных SQL
- Домены
- Таблицы

ТИПЫ ДАННЫХ SQL



КАТЕГОРИИ ТИПОВ ДАННЫХ

Данные, хранящиеся в столбцах таблиц SQL-ориентированной базы данных, являются типизированными, т. е. представляют собой значения одного из типов данных, предопределенных в языке SQL или определяемых пользователями путем применения соответствующих средств языка. Для этого при определении таблицы каждому ее столбцу назначается некоторый тип данных (или домен), и в дальнейшем СУБД должна следить, чтобы в каждом столбце каждой строки каждой таблицы присутствовали только допустимые значения.

КАТЕГОРИИ ТИПОВ ДАННЫХ SQL

- точные числовые типы (exact numerics);
- приближенные числовые типы (approximate numerics);
- типы символьных строк (character strings);
- типы битовых строк (bit strings);
- типы даты и времени (datetimes);
- типы временных интервалов (intervals);
- булевский тип (Booleans);
- типы коллекций (collection types);
- анонимные строчные типы (anonymous row types);
- типы, определяемые пользователем (user-defined types);
- ссылочные типы (reference types)

В столбцах таблиц, определенных на любых типах данных, наряду со значениями этих типов, допускается сохранение неопределенного значения, которое обозначается ключевым словом NULL.

- Результатом выражений вида *x a_op* **NULL**, **NULL** *a_op x*, **NULL** *a_op* **NULL** является **NULL** для всех арифметических операций.
- Значением выражений *x comp_op* **NULL**, **NULL** *comp_op x*, **NULL** *comp_op* NULL для всех операций сравнения является третье логическое значение ***unknown***.

ТОЧНЫЕ ЧИСЛОВЫЕ ТИПЫ (1/2)

К категории *точных числовых типов* в SQL относятся те типы, значения которых точно представляют числа. Типы данных этой категории распадаются на две части: *истинно целые типы* (INTEGER и SMALLINT) и *типы, допускающие наличие дробной части* (NUMERIC и DECIMAL).

Истинно целые типы

- Тип **INTEGER** служит для представления целых чисел. Точность чисел (число сохраняемых бит) определяется в реализации. При определении столбца данного типа достаточно указать просто INTEGER.
- Тип **SMALLINT** также служит для представления целых чисел. Точность определяется в реализации, но она не должна быть больше точности типа INTEGER. При определении столбца указывается просто SMALLINT.
- Литералы типов целых чисел представляются в виде строк символов, изображающих десятичные числа; в начале строки могут присутствовать символы «+» или «-» (если символ знака отсутствует, подразумевается «+»). Примеры литералов типов INTEGER и SMALLINT: 1826545, 876.

ТОЧНЫЕ ЧИСЛОВЫЕ ТИПЫ (2/2)

Точные типы, допускающие наличие дробной части

- Тип **NUMERIC**. На самом деле, это не просто тип данных, а параметризуемый тип. При определении столбца можно указать спецификацию NUMERIC (p, s), где p и s – литералы истинно целого типа, и p задает *точность значений* (число сохраняемых бит), а s – шкалу (число десятичных цифр в дробной части). Задаваемая шкала не должна быть отрицательной и не должна превышать значение точности. При определении столбца можно использовать сокращенные формы спецификации типа – NUMERIC и NUMERIC (p). Первая форма предполагает использование точности, определяемое по умолчанию в реализации, и шкалы, равной нулю, а вторая – использование заданной точности и шкалы, равной нулю. Допустимые диапазоны значений p и s определяются в реализации.
- Тип **DECIMAL**. Этот тип аналогичен типу NUMERIC. Отличие состоит в том, что если при определении столбца типа DECIMAL задается точность p, то на самом деле используется точность m, определяемая в реализации, такая, что m > p. Шкала всегда устанавливается такой, как явно или неявно (по умолчанию) задается. При указании типа столбца можно использовать спецификации DECIMAL, DECIMAL (p) и DECIMAL (p, s).
- Литералы типов точных чисел, допускающих наличие дробной части, представляются в виде строк символов, изображающих десятичные числа, в начале которых могут присутствовать символы «+» или «-» (если символ знака отсутствует, подразумевается «+»), а внутри последовательности цифр может присутствовать символ «.». Примеры литералов типов NUMERIC и DECIMAL: 125, 26.36.

ПРИБЛИЖЕННЫЕ ЧИСЛОВЫЕ ТИПЫ

- Тип **REAL**. Значения типа соответствуют числам с плавающей точкой одинарной точности. Точность определяется в реализации, но обычно совпадает с точностью одинарной плавающей арифметики, поддерживаемой на аппаратной платформе, которая используется реализацией. При определении столбца указывается просто REAL.
- Тип **DOUBLE PRECISION**. Точность значений этого типа определяется в реализации, но она должна быть больше точности типа REAL. Обычно приближенным числам SQL с двойной точностью соответствуют поддерживаемые аппаратурой числа с плавающей точкой двойной точности. При определении столбца указывается просто DOUBLE PRECISION.
- Тип **FLOAT**. Это параметризуемый тип, значение параметра p которого задает необходимую точность значений. Требуется, чтобы реально обеспечиваемая реализацией точность значений была не меньше p. Допустимый диапазон значений параметра p определяется в реализации. При определении столбца можно указать либо FLOAT (p), либо просто FLOAT. В последнем случае подразумевается точность, определяемая реализацией по умолчанию.
- Литералы приближенных числовых типов представляются в виде литерала точного числового типа, за которым могут следовать символ «E» и литерал целого числового типа. Примеры литералов приближенных числовых типов: 123, 123.12, 123E12, 123.12E12. Литеральное выражение xEy представляет значение x*(10y).

ТИПЫ СИМВОЛЬНЫХ СТРОК

В SQL определены три параметризуемых *типа символьных строк*: CHARACTER (или CHAR), CHARACTER VARYING (или CHAR VARYING, или VARCHAR) и CHARACTER LARGE OBJECT (или CLOB).

- Тип **CHARACTER**. Значениями типа являются символьные строки. Конкретный набор допустимых символов определяется в реализации, но, как правило, включает набор символов ASCII. При определении столбца допускается использование спецификаций CHARACTER (x) и просто CHARACTER. Последний вариант эквивалентен заданию CHARACTER (1). После определения столбца типа CHARACTER (x) СУБД будет резервировать место для хранения x символов этого столбца во всех строках соответствующей таблицы. Если, например, определен столбец типа CHARACTER (8), и в некоторой строке таблицы в него заносится символьная строка длиной пять символов, то реально будут храниться восемь символов, последние три из которых будут пробелами.
- Тип **CHARACTER VARYING**. При определении столбца допускается использование спецификаций CHARACTER VARYING (x) и просто CHARACTER VARYING. Последний вариант эквивалентен заданию CHARACTER VARYING (1). Если в некоторой таблице определяется столбец типа CHARACTER VARYING (x), то в каждой строке этой таблицы значения данного столбца будут занимать ровно столько места, сколько требуется для сохранения соответствующей символьной строки (но ни одна такая строка не может состоять более чем из x символов).
- Тип **CHARACTER LARGE OBJECT**. Этот тип данных предназначен для определения столбцов, хранящих большие и разные по размеру группы символов. При определении столбца задается спецификация CLOB (z), где z задает максимальный размер соответствующей группы символов. Максимально возможное значение параметра z определяется в реализации, но, очевидно, что оно должно быть существенно больше максимально возможного значения параметра x, присутствующего в типах CHAR и CHAR VARYING.

ОПЕРАЦИИ НАД ТИПАМИ СИМВОЛЬНЫХ СТРОК

Определен ряд операций, которые можно выполнять над символьными строками. Перечислим некоторые из них.

- Операция конкатенации (обозначается в виде «||») возвращает символьную строку, произведенную путем соединения строк-операндов в том порядке, в каком они заданы.
- Функция выделения подстроки (**SUBSTRING**) принимает три аргумента – строку, номер начальной позиции и длину – и возвращает строку, выделенную из строки-аргумента в соответствии со значениями двух последних параметров.
- Функция **UPPER** возвращает строку, в которой все строчные буквы строки-аргумента заменяются прописными. Функция **LOWER**, наоборот, заменяет в заданной строке все прописные буквы строчными.
- Функция определения длины (**CHARACTER_LENGTH, OCTET_LENGTH, BIT_LENGTH**) возвращает длину заданной символьной строки в символах, октетах или битах (в зависимости от вида вычисляющей функции) в виде целого числа.
- Функция определения позиции (**POSITION**) определяет первую позицию в строке S, с которой в нее входит заданная строка S1 (если не входит, то возвращается значение нуль).

Литералы типов символьных строк представляются в виде последовательностей символов, заключенных в одинарные или двойные кавычки. В первом случае среди набора символов литерала допускается наличие символов двойной кавычки, а во втором – символов одинарной кавычки. Примеры литералов символьных строк: ’ABCDEF’, ’Ab"Ctd’, "Fbcdef", "ab’cdtF".

ТИПЫ БИТОВЫХ СТРОК

В SQL определены три параметризуемых *типа битовых строк*: BIT, BIT VARYING и BINARY LARGE OBJECT (или BLOB).

- Тип **BIT**. Значениями типа являются битовые строки. При определении столбца допускается использование спецификаций BIT (x) и просто BIT. Последний вариант эквивалентен заданию BIT (1). После определения столбца типа BIT (x) СУБД будет резервировать место для хранения x бит этого столбца во всех строках соответствующей таблицы.
- Тип **BIT VARYING**. При определении столбца допускается использование только спецификации без умолчания вида BIT VARYING (x), где значение x определяет максимальную длину битовой строки, которую можно хранить в данном столбце.
- Тип **BINARY LARGE OBJECT**. Этот тип данных предназначен для определения столбцов, хранящих большие и разные по размеру группы байтов. При определении столбца задается спецификация **BLOB** (z), где z задает максимальный размер соответствующей группы байтов. С технической точки зрения типы CLOB и BLOB очень похожи. Их разделение требуется для того, чтобы подчеркнуть, что значения типа CLOB состоят из символов (в частности, в них может осмысленно производиться текстовый поиск), а значения типа BLOB состоят из произвольных байтов, не обязательно кодирующих символы.

ОПЕРАЦИИ НАД ТИПАМИ БИТОВЫХ СТРОК

Над битовыми строками определен ряд операций. Некоторые из них мы рассмотрим.

- Битовая конкатенация (обозначается в виде **||**), которая возвращает результирующую битовую строку, полученную путем конкатенации строк-аргументов в том порядке, в котором они заданы.
- Функция извлечения подстроки из битовой строки. Синтаксис и семантика этой функции идентичны синтаксису и семантике функции **SUBSTRING** для символьных строк, за исключением того, что первый аргумент и возвращаемое значение являются битовыми строками.
- Функция определения длины (**OCTET_LENGTH, BIT_LENGTH**) возвращает длину заданной битовой строки в октетах или битах в зависимости от выбранной функции.
- Функция определения позиции (**POSITION**) определяет первую позицию в битовой строке S, с которой в нее входит строка S1. Если строка S1 не входит в строку S, возвращается значение нуль.

Литералы типов битовых строк представляются как заключенные в одинарные кавычки последовательности символов «0» и «1», предваряемые символом «B»; или предваряемые символом «X» последовательности символов, которые изображают шестнадцатеричные цифры (за цифрой «9» следуют «A», «B», «C», «D», «E» и «F»). Примеры литералов типов битовых строк: B’0111001111000111111111’, X’78FBCD0012FFFFA’.

ТИПЫ ДАТЫ И ВРЕМЕНИ

Возможность сохранения в базе данных информации о дате и времени очень важна с практической точки зрения. Достаточно вспомнить взбудоражившую весь мир «проблему 2000 года», одним из основных источников которой было некорректное хранение дат в базах данных. В стандарте SQL поддержке средств работы с датой и временем уделяется большое внимание. В частности, поддерживаются специальные *«темпоральные» типы данных* DATE, TIME, TIMESTAMP, TIME WITH TIME ZONE и TIMESTAMP WITH TIME ZONE.

Тип даты

- Тип DATE. Значения этого типа состоят из компонентов-значений года, месяца и дня некоторой даты. Значение года состоит из четырех десятичных цифр и соответствует летоисчислению от Рождества Христова до 9999 г. Значение месяца состоит из двух десятичных цифр и варьируется от 01 до 12. Значение номера дня месяца состоит из двух десятичных цифр и варьируется от 01 до 31, хотя значение месяца даты может накладывать ограничения на возможность использования значений дня месяца 29, 30 и 31. В стандарте SQL не накладываются какие-либо ограничения на внутренний способ представления дат, используемый в реализации. При определении столбца типа DATE указывается просто DATE.

Литералы типа DATE представляются в виде строки «’yyyy-mm-dd’», где символы y, m и d должны изображать десятичные числа. Например, литерал DATE ’1949-04-08’ представляет дату 8 апреля 1949 г.

ТИПЫ ВРЕМЕНИ

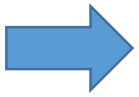
- Тип **TIME**. Значения этого параметризованного типа состоят из компонентов-значений часа, минуты и секунды некоторого времени суток. Значение часа состоит ровно из двух десятичных цифр и варьируется от 00 до 23. Значение минуты состоит из двух десятичных цифр и варьируется от 00 до 59. Основное значение секунды также состоит из двух цифр, но может включать дополнительные цифры, представляющие доли секунды. Так что в целом значение секунды варьируется от 00 до 61.999... В значении времени присутствуют две лишние секунды, поскольку Всемирная служба времени иногда добавляет две секунды к последней минуте года для синхронизации мирового времени с реальным. Решение о поддержке этих «високосных» секунд принимается на уровне реализации. Число цифр в доле секунды также определяется в реализации. В стандарте требуется только то, чтобы это число было не меньше шести. При определении столбца типа TIME может указываться TIME (p) (значение p задает точность долей секунды) или просто TIME (в этом случае доли секунды не учитываются).

Литералы типа TIME представляются в виде строки TIME ’hh:mm-ss:f...f’, где символы h, m, s и f должны изображать десятичные числа. Например, литерал TIME ’16:33-20:333’ представляет время суток 16 часов 33 минуты 20 и 333 тысячных секунды.

ТИПЫ ВРЕМЕННОЙ МЕТКИ

Типы временной метки

- Тип **TIMESTAMP**. Значения этого параметризованного типа состоят из компонентов — значений года, месяца и дня некоторой даты, а также компонентов — значений часа, минуты и секунды некоторого времени суток. Число десятичных цифр в значениях-компонентах и ограничения этих значений такие же, как у значений типов DATE и TIME. При определении столбца типа TIMESTAMP может указываться TIMESTAMP (p) (значение p задает точность долей секунды) или просто TIMESTAMP (в этом случае, в отличие от типа данных TIME, по умолчанию принимается, что в доли секунды используются шесть десятичных цифр).



Литералы типа TIMESTAMP представляются в виде строки TIMESTAMP 'yyyy-mm-dd hh:mm-ss:f...f', где символы y, m, d, h, m, s и f должны изображать десятичные числа.

Типы времени и временной метки с временной зоной

- Тип **TIME WITH TIME ZONE**. Этот тип данных похож на тип TIME с тем лишь отличием, что значения типа TIME WITH TIME ZONE включают дополнительный компонент — значение, характеризующее смещение соответствующего времени относительно гринвичского времени (теперь его называют UTC – universal time coordinated). Деталей представления этого дополнительного компонента мы касаться не будем.
- Тип **TIMESTAMP WITH TIME ZONE**. Этот тип данных отличается от типа TIMESTAMP тем, что значения типа TIMESTAMP WITH TIME ZONE включают дополнительный компонент-значение, характеризующее смещение соответствующего времени относительно гринвичского.

ТИПЫ ВРЕМЕННЫХ ИНТЕРВАЛОВ (1/2)

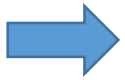
Вообще говоря, ***временным интервалом*** называется разность между двумя значениями даты или времени. В SQL определены две категории *типов временных интервалов*: «год-месяц» и «день-время суток». Временные интервалы языка SQL не привязываются к начальному и/или конечному значению даты/времени, а описывают только протяженность во времени. В общем случае при определении столбца типа временного интервала указывается INTERVAL start (p) [TO end (q)], где в качестве «start» и «end» могут задаваться YEAR, MONTH, DAY, HOUR, MINUTE и SECOND.
Параметр p задает требуемую точность лидирующего поля интервала (число десятичных цифр). Параметр q может задаваться только в том случае, когда в качестве end используется SECOND, и указывает точность долей секунды. Если говорить более точно, возможны следующие вариации типов временных интервалов.

- **Типы категории «день-время суток».** При определении столбца можно использовать комбинации, представленные справа. Если значение параметра p не указывается явно, по умолчанию принимается его значение «2». Значением параметра q по умолчанию является «6».

```
INTERVAL DAY (p),
INTERVAL DAY,
INTERVAL DAY (p) TO HOUR,
INTERVAL DAY TO HOUR,
INTERVAL DAY (p) TO MINUTE,
INTERVAL DAY TO MINUTE,
INTERVAL DAY (p) TO SECOND (q),
INTERVAL DAY TO SECOND (q),
INTERVAL DAY (p) TO SECOND,
INTERVAL DAY TO SECOND,
INTERVAL HOUR (p),
INTERVAL HOUR, INTERVAL HOUR (p) TO MINUTE,
INTERVAL HOUR TO MINUTE,
INTERVAL HOUR (p) TO SECOND (q),
INTERVAL HOUR TO SECOND (q),
INTERVAL HOUR TO SECOND,
INTERVAL MINUTE (p),
INTERVAL MINUTE,
INTERVAL MINUTE (p) TO SECOND (q),
INTERVAL MINUTE TO SECOND (q),
INTERVAL MINUTE (p) TO SECOND,
INTERVAL MINUTE TO SECOND,
INTERVAL SECOND (p, q),
INTERVAL SECOND (p),
INTERVAL SECOND.
```

ТИПЫ ВРЕМЕННЫХ ИНТЕРВАЛОВ (2/2)

- Типы категории «год-месяц». Можно определить столбцы следующих типов: INTERVAL YEAR, INTERVAL YEAR (p) (значения этих типов – временные интервалы в годах), INTERVAL MONTH, INTERVAL MONTH (p) (значения этих типов – временные интервалы в месяцах), INTERVAL YEAR TO MONTH, INTERVAL YEAR (p) TO MONTH (значения этих типов – временные интервалы в годах и месяцах). Если значение параметра p не указывается явно, по умолчанию принимается его значение «2».



Пример литерала одной из разновидностей типа INTERVAL: INTERVAL ’10:20’ MINUTE TO SECOND – временной интервал в 10 минут и 20 секунд.

Над значениями темпоральных типов могут выполняться арифметические операции, смысл которых определяется следующей таблицей:

Тип первого операнда	Операция	Тип второго операнда	Тип результата
Datetime	-	Datetime	Interval
Datetime	+ или -	Interval	Datetime
Interval	+	Datetime	Datetime
Interval	+ или -	Interval	Interval
Interval	* или /	Numeric	Interval
Numeric	*	Interval	Interval

БУЛЕВСКИЙ ТИП ДАННЫХ

При определении столбца булевского типа указывается просто спецификация **BOOLEAN**. *Булевский тип* состоит из трех значений: true, false и unknown. (соответствующие литералы обозначаются TRUE, FALSE и UNKNOWN). Поддерживается возможность построения булевских выражений, которые вычисляются в трехзначной логике.

Таблицы истинности основных логических операций

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

ТИПЫ КОЛЛЕКЦИЙ (1/2)

Начиная с SQL:1999, в языке поддерживается возможность использования типов данных, значения которых являются коллекциями значений некоторых других типов. Обычно под термином *коллекция* понимается одно из следующих образований: массив, список, множество и мультимножество. В варианте SQL:1999, принятом в 1999 г., были специфицированы только типы массивов. В стандарте SQL:2003 появилась спецификация типа мультимножества.

Типы массивов

Любой возможный *тип массива* получается путем применения конструктора типов **ARRAY.** При определении столбца, значения которого должны принадлежать некоторому типу массива, используется конструкция ***dt*** **ARRAY** [**mc**], где ***dt*** специфицирует некоторый допустимый в SQL тип данных, а **mc** является литералом некоторого точного числового типа с нулевой длиной шкалы и определяет максимальное число элементов в значении типа массива.

Элементам каждого значения типа массива соответствуют их порядковые номера, называемые индексами. Значение индекса всегда должно принадлежать отрезку [1, **mc**]. Значениями типа массива **dt ARRAY [mc]** являются все массивы, состоящие из элементов типа dt, максимальное значение индекса которых **cs** не превосходит значения mc. При сохранении в базе данных значения типа массива занимает столько памяти, сколько требуется для сохранения **cs** элементов.

Основными операциями над массивами являются выборка значения элемента массива по его индексу, изменение некоторого элемента массива или массива целиком и конкатенация (сцепление) двух массивов. Кроме того, для любого значения типа массива можно узнать значение его **cs**.

ТИПЫ КОЛЛЕКЦИЙ (2/2)

Типы мультимножеств

При определении столбца таблицы *типа мультимножеств* используется конструкция **dt MULTISET**, где **dt** задает тип данных элементов конструируемого типа мультимножеств. Значениями типа мультимножеств являются мультимножества, т. е. неупорядоченные коллекции элементов одного и того же типа, среди которых допускаются дубликаты. Например, значениями типа INTEGER MULTISET являются мультимножества, элементами которых — целые числа. Примером такого значения может быть мультимножество {12, 34, 12, 45, -64}.

В отличие от массива, мультимножество является неограниченной коллекцией; при конструировании типа мультимножеств не указывается предельная кардинальность значений этого типа. Однако это не означает, что возможность вставки элементов в мультимножество действительно не ограничена; стандарт всего лишь не требует явного объявления границы. Ситуация аналогична той, которая возникает при работе с таблицами, для которых в SQL не объявляется максимально допустимое число строк.

ОПЕРАЦИИ НАД ТИПАМИ КОЛЛЕКЦИЙ

Для типов мультимножеств поддерживаются операции преобразования типа значения-мультимножества к типу массивов или другому типу мультимножеств с совместимым типом элементов (операция **CAST**), для удаления дубликатов из мультимножества (функция **SET**), для определения числа элементов в заданном мультимножестве (функция **CARDINALITY**), для выборки элемента мультимножества, содержащего в точности один элемент (функция **ELEMENT**). Кроме того, для мультимножеств обеспечиваются операции объединения (**MULTISET UNION**), пересечения (**MULTISET INTERSECT**) и определения разности (**MULTISET EXCEPT**). Каждая из операций может выполняться в режиме с сохранением дубликатов (режим **ALL**) или с устранением дубликатов (режим **DISTINCT**).

Расширенные в SQL:2003 возможности работы с типами коллекций являются принципиально важными. Даже при наличии определяемых пользователями типов данных и типов массивов SQL:1999 не предоставлял полных возможностей для преодоления исторически присущего реляционной модели данных вообще и SQL в частности ограничения «плоских таблиц». После появления конструктора типов мультимножеств и устранения ограничений на тип данных элементов коллекции это историческое ограничение полностью ликвидировано. Мультимножество, типом элементов которого является анонимный строчный тип (см. ниже), представляет собой полный аналог таблицы. Тем самым, в базе данных допускается произвольная вложенность таблиц. Возможности выбора структуры базы данных безгранично расширяются.

АНОНИМНЫЕ СТРОЧНЫЕ ТИПЫ

Анонимный строчный тип данных - это конструктор типов ROW, позволяющий производить безымянные типы строк (кортежей). Любой возможный строчный тип получается путем использования конструктора ROW. При определении столбца, значения которого должны принадлежать некоторому строчному типу, используется конструкция ROW (fld_1, fld_2, …, fld_n), где каждый элемент fld_i, определяющий поле строчного типа, задается в виде тройки **fldname, fldtype, fldoptions**. Подэлемент fldname задает имя соответствующего поля строчного типа. Подэлемент **fldtype** специфицирует тип данных этого поля. В качестве типа данных поля строчного типа можно использовать любой допустимый в SQL тип данных, включая типы коллекций, определяемые пользователями типы и другие строчные типы. Необязательный подэлемент **fldoptions** может задаваться для указания применяемого по умолчанию порядка сортировки, если соответствующий подэлемент fldtype указывает на тип символьных строк, а также должен задаваться, если fldtype указывает на ссылочный тип. *Степенью строчного типа* называется число его полей.

ПОЛЬЗОВАТЕЛЬСКИЕ ТИПЫ ДАННЫХ

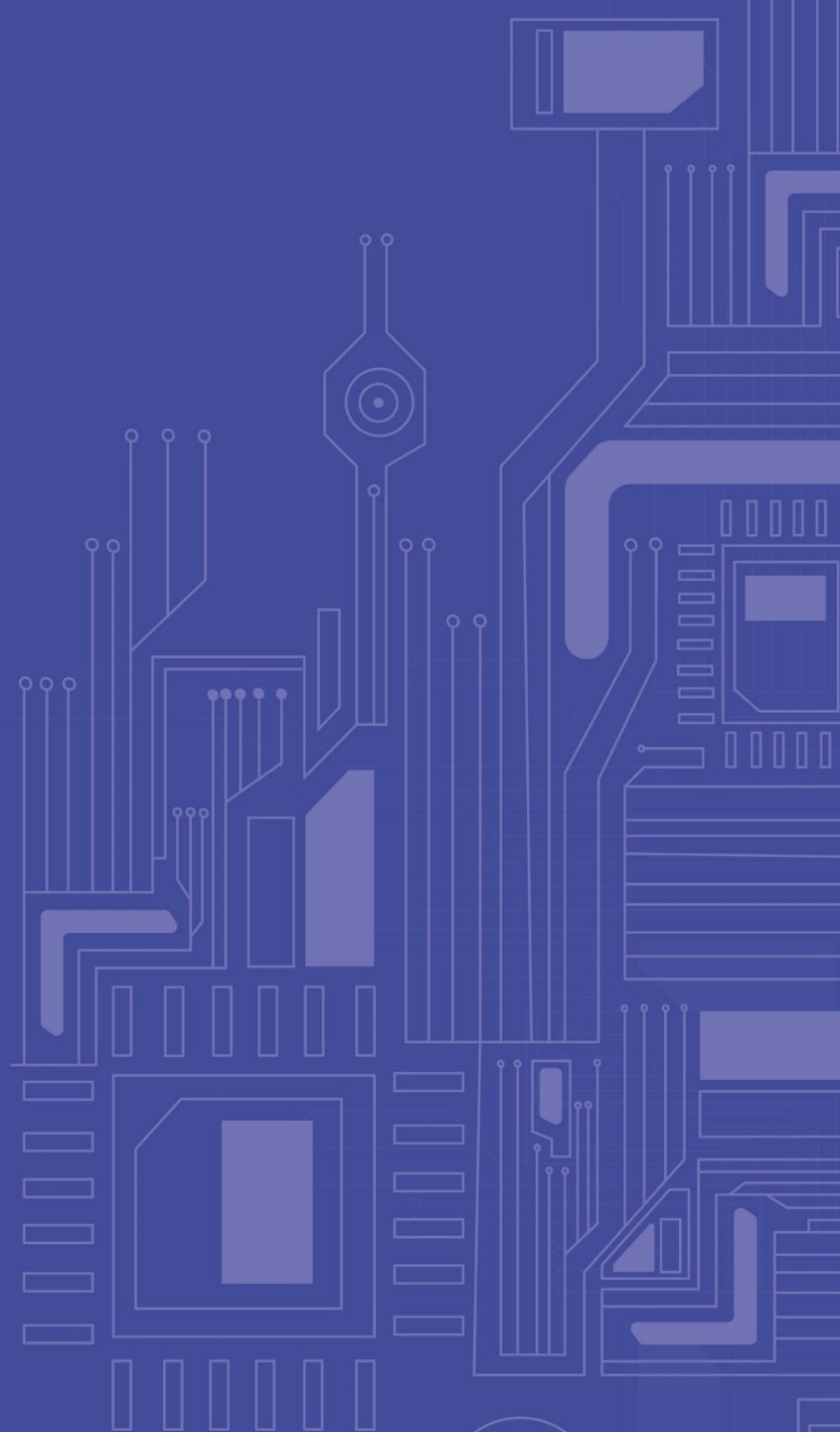
Структурные типы (Structured Types). Соответствующие возможности SQL:1999 позволяют определять долговременно хранимые, именованные типы данных, включающие один или более атрибутов любого из допустимых в SQL типа данных, в том числе другие структурные типы, типы коллекций, строчные типы и т. д. Стандарт SQL не накладывает ограничений на сложность получаемой в результате структуры данных, однако не запрещает устанавливать такие ограничения в реализации. Дополнительные механизмы определяемых пользователями методов, функций и процедур позволяют определить поведенческие аспекты структурного типа.

Индивидуальные типы (Distinct Types). Можно определить долговременно хранимый, именованный тип данных, опираясь на единственный предопределенный тип. Например, можно определить индивидуальный тип данных PRICE, опираясь на тип DECIMAL (5, 2). Тогда значения типа PRICE представляются точно так же, как значения типа DECIMAL (5, 2). Однако в SQL:1999 индивидуальный тип не наследует от своего опорного типа набор операций над значениями. Например, чтобы сложить два значения типа PRICE требуется явно сообщить системе, что с этими значениями нужно обращаться как со значениями типа DECIMAL (5, 2). Другая возможность состоит в явном определении методов, функций и процедур, связанных с данным индивидуальным типом.

ССЫЛОЧНЫЕ ТИПЫ ДАННЫХ

Обеспечивается механизм конструирования типов (***ссылочных типов***), которые могут использоваться в качестве типов столбцов некоторого вида таблиц (типизированных таблиц). Фактически значениями ссылочного типа являются строки соответствующей типизированной таблицы. Более точно, каждой строке типизированной таблицы приписывается уникальное значение (нечто вроде первичного ключа, назначаемого системой или приложением), которое может использоваться в методах, определенных для табличного типа, для уникальной идентификации строк соответствующей таблицы. Эти уникальные значения называются *ссылочными значениями*, а их тип – ссылочным типом. Ссылочный тип может содержать только те значения, которые действительно ссылаются на экземпляры указанного типа (т. е. на строки соответствующей типизированной таблицы).

ДОМЕНЫ И ПРЕОБРАЗОВАНИЯ ТИПОВ



ПОНЯТИЕ ДОМЕНА В SQL (УСТАРЕВАЕТ)

Домен является долговременно хранимым, именованным объектом схемы базы данных. Домены можно создавать (определять), изменять (изменять определения) и ликвидировать (отменять определение). Имена доменов можно использовать при определении столбцов таблиц. Можно считать, что в SQL *определение домена* представляет собой вынесенное за пределы определения индивидуальной таблицы «родовое» определение столбца, которое можно использовать для определения различных реальных столбцов реальных базовых таблиц.

Для определения домена в SQL используется оператор CREATE DOMAIN. Общий синтаксис этого оператора следующий:

```
domain_definition ::= CREATE DOMAIN domain_name [AS] data_type
    [ default_definition ]
    [ domain_constraint_definition_list ]
```

Раздел default_definition имеет вид:
DEFAULT { literal | niladic_function | NULL }

Ниладические (служебные) функции
- USER
- CURRENT_USER
- SESSION_USER
- SYSTEM_USER
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP

Элемент списка **domain_constraint_definition_list** имеет вид

```
[CONSTRAINT constraint_name]
    CHECK (conditional_expression)
```

Наиболее естественным видом ограничения домена является следующий:

```
CHECK (VALUE IN (list_of_valid_values))
```

ИЗМЕНЕНИЕ И ОТМЕНА ОПРЕДЕЛЕНИЯ ДОМЕНА

ИЗМЕНЕНИЕ ОПРЕДЕЛЕНИЯ ДОМЕНА

Общий синтаксис изменения

```
domain_alternation ::=
    ALTER DOMAIN domain_name domain_alternation_action
domain_alternation_action ::=
      domain_default_alternation_action
    | domain_constraint_alternation_action
```

Изменение значения по умолчанию

```
domain_default_alternation_action ::=
      SET default_definition
    | DROP DEFAULT
```

Изменение ограничений

```
domain_constraint_alternation_action ::=
      ADD domain_constraint_definition
    | DROP CONSTRAINT constraint_name
```

ОТМЕНА ОПРЕДЕЛЕНИЯ ДОМЕНА

```
DROP DOMAIN domain_name {RESTRICT | CASCADES}
```

Если в операторе указано RESTRICT, и если соответствующий домен использован в определении некоторого столбца, в определении некоторого представления или в определении ограничения целостности, то оператор DROP DOMAIN отвергается. В противном случае определение домена ликвидируется.

Если в операторе DROP DOMAIN указано CASCADES, то оператор выполняется всегда. При этом уничтожаются все представления и ограничения целостности, в определении которых использовалось имя данного домена. Столбцы, определенные на этом домене, автоматически переопределяются следующим образом:

- считается, что каждый такой столбец теперь относится к определяющему типу уничтожаемого домена;
- если у столбца не было определено собственное значение по умолчанию, то считается, что теперь у него имеется такое значение по умолчанию, совпадающее со значением по умолчанию уничтожаемого домена;
- каждый столбец наследует все ограничения уничтожаемого домена.

ПРИМЕРЫ ОПРЕДЕЛЕНИЙ И ИЗМЕНЕНИЙ ДОМЕНОВ

```
CREATE DOMAIN EMP_NO AS INTEGER
    CHECK (VALUE BETWEEN 1 AND 10000);
```

```
CREATE DOMAIN SALARY AS NUMERIC (10, 2)
    DEFAULT 10000.00
    CHECK (VALUE BETWEEN 10000.00 AND 20000000.00)
    CONSTRAINT SAL_NOT_NULL CHECK (VALUE IS NOT NULL);
```

Немного поупражняемся с доменом SALARY. Для изменения значения заработной платы по умолчанию с 10000 на 11000 руб. нужно выполнить оператор

```
ALTER DOMAIN SALARY SET DEFAULT 11000.00;
```

Для отмены значения по умолчанию в домене SALARY следует воспользоваться оператором

```
ALTER DOMAIN SALARY DROP DEFAULT;
```

Если к определению домена SALARY требуется добавить ограничение (например, запретить значение зарплаты, равное 15000 руб.), необходимо выполнить оператор

```
ALTER DOMAIN SALARY ADD CHECK (VALUE <> 15000.00);
```

Наконец, если требуется отменить (именованное!) ограничение целостности, препятствующее наличию неопределенных значений в столбцах, которые определены на домене SALARY, то нужно выполнить оператор

```
ALTER DOMAIN SALARY DROP CONSTRAINT SAL_NOT_NULL;
```

ПРЕОБРАЗОВАНИЯ ТИПОВ

В языке SQL обеспечивается возможность использования в различных операциях не только значений тех типов, для которых предопределена операция, но и значений типов, неявным или явным образом приводимых к требуемому типу.

В SQL поддерживается совместимость некоторых типов данных за счет *неявного преобразования* значений одного типа к значениям другого типа данных. Тип данных A приводим к типу данных B в том и только в том случае, когда в любом месте, где ожидается значение типа B, может быть использовано значение типа A.

ОСНОВНЫЕ ПРАВИЛА ПРИВОДИМОСТИ ТИПОВ

- **Типы символьных строк.** Тип CHARACTER (x) приводим к любому типу CHARACTER (y), если y $\geq$ x. Типы VARCHAR (x) и CHARACTER (x) приводимы к любому типу VARCHAR (y), если y $\geq$ x. Типы CHARACTER (x) и VARCHAR (x) приводимы к любому типу CLOB.
- **Типы битовых строк.** Тип BIT (x) приводим к любому типу BIT (y), если y $\geq$ x. Типы BIT VARYING (x) и BIT (x) приводимы к любому типу BIT VARYING (y), если y $\geq$ x.
- **Типы BLOB.** Тип BLOB (x) приводим к любому типу BLOB (y), если y $\geq$ x.
- **Типы точных чисел**. Тип EN (p_1, s_1) приводим к любому типу EN (p_2, s_2), у которого $s_2 \geq s_1$ и p_2 определяется в реализации. Тип EN (p, s) приводим к любому типу приблизительных чисел AN (p_1), где p_1 определяется в реализации.
- **Типы приблизительных чисел.** Тип AN (p_1) приводим к любому типу AN (p_2), если $p_2 \geq p_1$.

ЯВНЫЕ ПРЕОБРАЗОВАНИЯ И ОПЕРАТОР CAST

```
CAST ({scalar-expression | NULL } AS
      {data_type | domain_name})
```

Оператор преобразует значение заданного скалярного выражения к указанному типу или к базовому типу указанного домена. Результатом применения оператора CAST к неопределенному значению является неопределенное значение.

Примем следующие обозначения типов данных:

EN – точные числовые типы (Exact Numeric)
AN – приблизительные числовые типы (Approximate Numeric)
C – типы символьных строк (Character)
FC – типы символьных строк постоянной длины (Fixed-length Character)
VC – типы символьных строк переменной длины (Variable-length Character)
B – типы битовых строк (Bit String)
FB – типы битовых строк постоянной длины (Fixed-length Bit String)
VB – типы битовых строк переменной длины (Variable-length Bit String)
D – тип Date
T – типы Time
TS – типы Timestamp
YM – типы Interval Year-Month
DT – типы Interval Day-Time

ЯВНЫЕ ПРЕОБРАЗОВАНИЯ

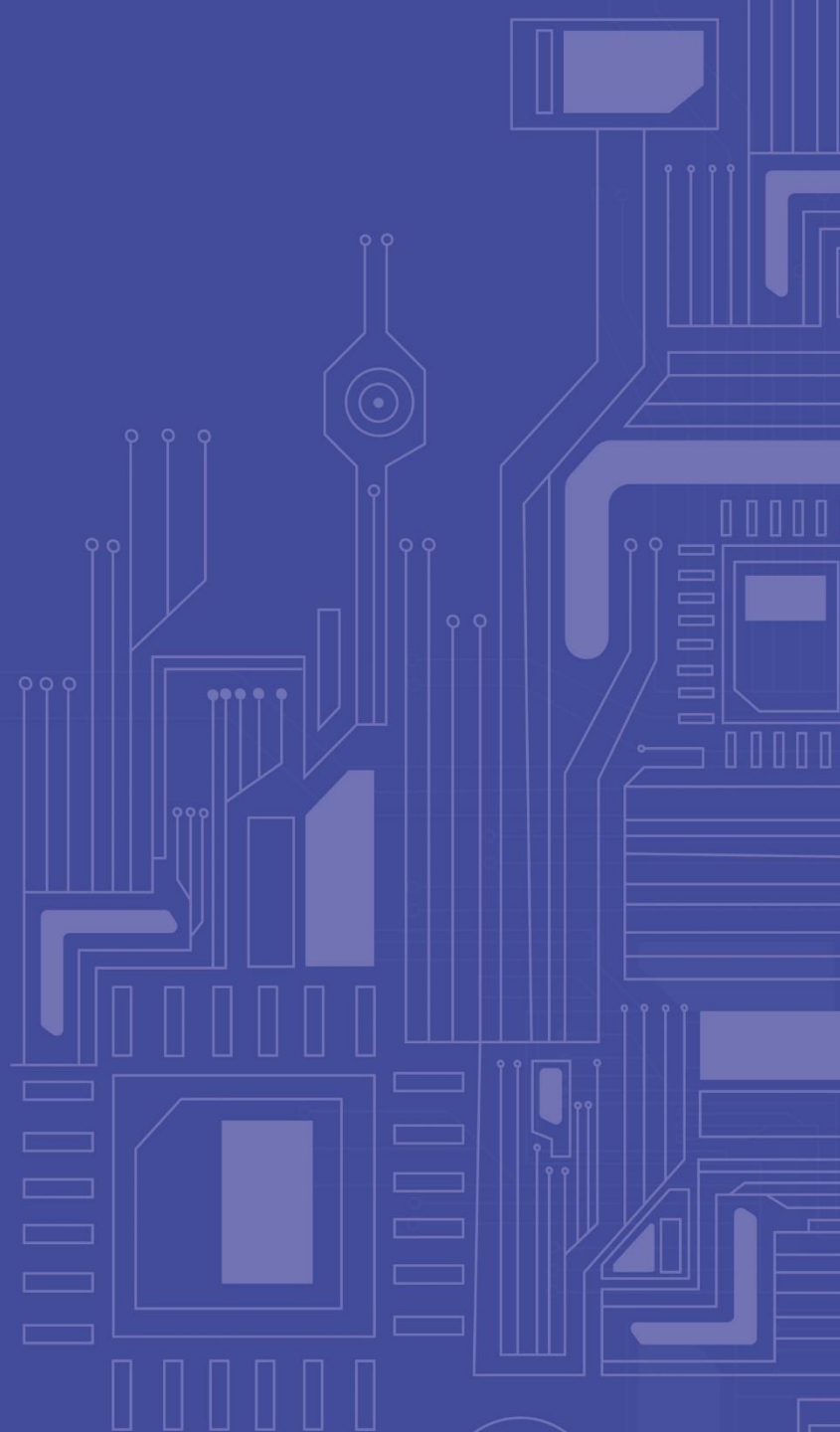
Пусть TD – это тип данных, к которому производится преобразование, а SD – тип данных операнда. Тогда допустимы следующие комбинации :

SD	TD										
		EN	AN	VC	FC	VB	FB	D	T	TS	YM	DT
	EN	Да	Да	Да	Да	Нет	Нет	Нет	Нет	Нет	?	?
	AN	Да	Да	Да	Да	Нет	Нет	Нет	Нет	Нет	Нет	Нет
	C	Да	Да	?	?	Да	Да	Да	Да	Да	Да	Да
	B	Нет	Нет	Да	Да	Да	Да	Нет	Нет	Нет	Нет	Нет
	D	Нет	Нет	Да	Да	Нет	Нет	Да	Нет	Да	Нет	Нет
	T	Нет	Нет	Да	Да	Нет	Нет	Нет	Да	Да	Нет	Нет
	TS	Нет	Нет	Да	Да	Нет	Нет	Да	Да	Да	Нет	Нет
	YM	?	Нет	Да	Да	Нет	Нет	Нет	Нет	Нет	Да	Нет
	DT	?	Нет	Да	Да	Нет	Нет	Нет	Нет	Нет	Нет	Да

По поводу ячеек таблицы, содержащих знак вопроса, необходимо сделать несколько оговорок:

1. если TD – интервал и SD – тип точных чисел, то TD должен содержать единственное поле даты-времени;
2. если TD – тип точных чисел и SD – интервал, то SD должен содержать единственное поле даты-времени;
3. если SD – тип символьных строк и TD – тип символьных строк постоянной или переменной длины, то набор символов SD и TD должен быть одним и тем же.

ТАБЛИЦЫ



БАЗОВАЯ ТАБЛИЦА В SQL МОДЕЛИ

Как и в реляционной модели данных, в модели SQL поддерживается единственная родовая структура данных, называемая в данном случае *базовой таблицей*.

- Коренное отличие базовой таблицы от истинного отношения состоит в том, что тело таблицы не обязательно является множеством. Среди строк тела таблицы могут встречаться дубликаты, и в общем случае тело базовой таблицы SQL представляет собой мультимножество строк.

- Порождаемые таблицы SQL, которые формируются при выполнении запросов к SQL-ориентированной базе данных, еще более отдаляют SQL от реляционной модели. В таких таблицах может отсутствовать и правильно сформированный заголовок (могут иметься одноименные столбцы).

В операторе SQL **CREATE TABLE** специфицируются не только столбцы таблицы, но и ограничения целостности, которым должны удовлетворять данные, хранящиеся в базовой таблице. Эти ограничения являются частным случаем ограничений базы данных целиком, для определения которых, а также изменения и ликвидации определений имеются специальные операторы.

Базовые (реально хранимые в базе данных) таблицы создаются (определяются) с использованием оператора **CREATE TABLE**. Для изменения определения базовой таблицы применяется оператор **ALTER TABLE**. Уничтожить хранимую таблицу (отменить ее определение) можно с помощью оператора **DROP TABLE**.

ОПРЕДЕЛЕНИЕ БАЗОВОЙ ТАБЛИЦЫ

Базовые (реально хранимые в базе данных) таблицы создаются (определяются) с использованием оператора CREATE TABLE. Для изменения определения базовой таблицы применяется оператор ALTER TABLE. Уничтожить хранимую таблицу (отменить ее определение) можно с помощью оператора DROP TABLE. Создание базовой таблицы, кроме создания соответствующих описателей, порождает новую область внешней памяти, в которой будут храниться данные, поставляемые пользователями.

```
base_table_definition ::= CREATE TABLE base_table_name (base_table_element_commalist)

base_table_element ::= column_definition | base_table_constraint_definition
```

Здесь **base_table_name** задает имя новой (изначально пустой) базовой таблицы. Каждый элемент определения базовой таблицы является либо определением столбца, либо определением табличного ограничения целостности.

Определение столбца

```
column_definition ::= column_name
    { data_type | domain_name }
    [ default_definition ]
    [ column_constraint_definition_list ]
```

Значения столбца по умолчанию

```
DEFAULT { literal | niladic_function | NULL }
```

- если в определении столбца явно присутствует раздел DEFAULT, то значением столбца по умолчанию является значение, указанное в этом разделе;
- иначе, если столбец определяется на домене и в определении этого домена явно присутствует раздел DEFAULT, то значением столбца по умолчанию является значение, указанное в этом разделе;
- иначе значением по умолчанию столбца является NULL.

ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ СТОЛБЦА

```
column_constraint_definition ::=
    [ CONSTRAINT constraint_name ]
    NOT NULL
    | { PRIMARY KEY | UNIQUE }
    | references_definition
    | CHECK ( conditional_expression )
```

Ограничение NOT NULL означает, что в определяемом столбце никогда не должны содержаться неопределенные значения. Если определяемый столбец имеет имя C, то это ограничение эквивалентно следующему табличному ограничению: CHECK (C IS NOT NULL).

В определение столбца может входить одно из ограничений уникальности: ограничение первичного ключа (PRIMARY KEY) или ограничение возможного ключа (UNIQUE). Ограничение PRIMARY KEY, в дополнение к требованию NOT NULL значений самого столбца, влечет за собой ограничение NOT NULL для определяемого столбца. Эти ограничения столбца эквивалентны следующим табличным ограничениям: PRIMARY KEY (C) и UNIQUE (C).

```
references_definition ::=
    REFERENCES base_table_name [ (column_commalist) ]
        [ MATCH { SIMPLE | FULL | PARTIAL } ]
        [ ON DELETE referential_action ]
        [ ON UPDATE referential_action ]
```

Ограничение **references_definition** означает объявление определяемого столбца внешним ключом таблицы. Ограничение эквивалентно следующему табличному ограничению: FOREIGN KEY (C) references_definition.

Проверочное ограничение CHECK (conditional_expression) приводит к тому, что в данном столбце могут находиться только те значения, для которых вычисление conditional_expression не приводит к результату false. В условном выражении проверочного ограничения столбца разрешается использовать имя только определяемого столбца.

ТАБЛИЧНЫЕ ОГРАНИЧЕНИЯ (1/2)

```
base_table_constraint_definition ::=
    [ CONSTRAINT constraint_name ]
    { PRIMARY KEY | UNIQUE } ( column_commalist )
    | FOREIGN KEY ( column_commalist )
      references_definition
    | CHECK ( conditional_expression )
```

Имеется три разновидности табличных ограничений: ограничение первичного или возможного ключа (PRIMARY KEY или UNIQUE), ограничение внешнего ключа (FOREIGN KEY) и проверочное ограничение (CHECK). Любому ограничению может явным образом назначаться имя, если перед определением ограничения поместить конструкцию CONSTRAINT constraint_name.

Ограничение первичного или возможного ключа

Табличное ограничение первичного или возможного ключа { PRIMARY_KEY | UNIQUE } (column_commalist) означает требование уникальности составных значений указанной группы столбцов. Ограничение PRIMARY KEY, в дополнение к этому, влечет ограничение NOT NULL для всех столбцов, упоминаемых в определении ограничения. В определении таблицы допускается произвольное число определений возможного ключа (для разных комбинаций столбцов), но не более одного определения первичного ключа.

В языке SQL действительно допускается определение таблиц, у которых отсутствуют возможные ключи. Эта особенность языка, среди прочего, очевидным образом противоречит базовым требованиям реляционной модели данных.

ТАБЛИЧНЫЕ ОГРАНИЧЕНИЯ (2/2)

Проверочное табличное ограничение

Определение табличного ограничения вида CHECK (conditional_expression) приводит к тому, что указанное условное выражение будет вычисляться при каждой попытке обновления соответствующей таблицы. Таблица находится в соответствии с данным проверочным табличным ограничением, если для всех строк таблицы результатом вычисления соответствующего условного выражения не является *false*.

Табличное ограничение внешнего ключа

Табличное ограничение FOREIGN KEY (column_commalist) references_definition означает объявление внешним ключом группы столбцов, имена которых перечислены в списке column_commalist.

```
references_definition ::=
    REFERENCES base_table_name [ (column_commalist) ]
        [ MATCH { SIMPLE | FULL | PARTIAL } ]
        [ ON DELETE referential_action ]
        [ ON UPDATE referential_action ]
```

Если определение ссылок включает список столбцов (column_commalist), то этот список должен совпадать со списком имен столбцов, использованных в некотором определении первичного или возможного ключа (PRIMARY_KEY или UNIQUE) в определении таблицы base_table_name (T). Если в определении ссылок список столбцов явно не задан, то считается, что он совпадает со списком столбцов, использованных в определении первичного ключа (PRIMARY_KEY) таблицы T.

ТАБЛИЧНОЕ ОГРАНИЧЕНИЕ ВНЕШНЕГО КЛЮЧА (1/2)

Разновидности способов сопоставления значений внешнего и возможного ключей

Пусть определяемая таблица имеет имя S, а базовая таблица base_table_name имеет имя T.

СМЫСЛ НЕОБЯЗАТЕЛЬНОГО РАЗДЕЛА ОПРЕДЕЛЕНИЯ ВНЕШНЕГО КЛЮЧА MATCH { SIMPLE | FULL | PARTIAL }.

Ограничение внешнего ключа удовлетворяется в том и только в том случае, когда для каждой строки таблицы S выполняется одно из следующих условий:

Раздел отсутствует или имеет вид MATCH SIMPLE

- какой-либо столбец, входящий в состав внешнего ключа, содержит NULL;
- таблица T содержит в точности одну строку, такую, что значение внешнего ключа в данной строке таблицы S совпадает со значением соответствующего возможного ключа в этой строке таблицы T.

Раздел MATCH присутствует и имеет вид MATCH PARTIAL

- каждый столбец, входящий в состав внешнего ключа, содержит NULL;
- таблица T содержит по крайней мере одну такую строку, что для каждого столбца данной строки таблицы S, значение которого отлично от NULL, его значение совпадает со значением соответствующего столбца возможного ключа в этой строке таблицы T.

ТАБЛИЧНОЕ ОГРАНИЧЕНИЕ ВНЕШНЕГО КЛЮЧА (2/2)

Раздел MATCH имеет вид MATCH FULL

- каждый столбец, входящий в состав внешнего ключа, содержит NULL;
- ни один столбец, входящий в состав внешнего ключа, не содержит NULL, и таблица T содержит в точности одну строку, такую, что значение внешнего ключа в данной строке таблицы S совпадает со значением соответствующего возможного ключа в этой строке таблицы T.

Только при наличии спецификации MATCH FULL ссылочное ограничение соответствует требованиям реляционной модели. Тем не менее, в определении ограничения внешнего ключа базовых таблиц в SQL по умолчанию предполагается наличие спецификации MATCH SIMPLE.

Поддержка ссылочной целостности и ссылочные действия

В связи с определением ограничения внешнего ключа осталось рассмотреть еще два необязательных раздела – ON DELETE referential_action и ON UPDATE referential_action.

```
referential_action ::=
   { NO ACTION | RESTRICT | CASCADE
    | SET DEFAULT | SET NULL }
```

Чтобы объяснить, в каких случаях и каким образом выполняются эти действия, требуется сначала определить понятие ссылающейся строки (referencing row).

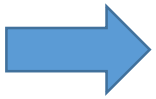
REFERENCING ROW

В определении ограничения внешнего ключа **отсутствует раздел MATCH** или **присутствуют** спецификации **MATCH SIMPLE либо MATCH FULL**

Тогда для данной строки **t** таблицы **T** строкой таблицы **S**, ссылающейся на строку **t**, называется каждая строка таблицы **S**, значение внешнего ключа которой совпадает со значением соответствующего возможного ключа строки **t**.

В определении ограничения внешнего ключа **присутствует** спецификация **MATCH PARTIAL**

Тогда для данной строки **t** таблицы **T** строкой таблицы **S**, ссылающейся на строку **t**, называется каждая строка таблицы **S**, отличные от NULL значения столбцов внешнего ключа которой совпадают со значениями соответствующих столбцов соответствующего возможного ключа строки **t**.



В случае MATCH PARTIAL строка таблицы S называется ссылающейся исключительно на строку t таблицы T, если эта строка таблицы S является ссылающейся на строку t и не является ссылающейся на какую-либо другую строку таблицы T.

ССЫЛОЧНЫЕ ДЕЙСТВИЯ (ON DELETE)

Пусть определение ограничения внешнего ключа содержит раздел **ON DELETE** referential_action. Предположим, что предпринимается попытка удалить строку t из таблицы T. Тогда:

- если в качестве требуемого ссылочного действия указано `NO ACTION` или `RESTRICT`, то операция удаления отвергается, если ее выполнение вызвало бы нарушение ограничения внешнего ключа;
- если в качестве требуемого ссылочного действия указано `CASCADE`, то строка `t` удаляется, и если в определении ограничения внешнего ключа отсутствует раздел `MATCH` или присутствуют спецификации `MATCH SIMPLE` или `MATCH FULL`, то удаляются все строки, ссылающиеся на `t`. Если же в определении ограничения внешнего ключа присутствует спецификация `MATCH PARTIAL`, то удаляются только те строки, которые ссылаются исключительно на строку `t`;
- если в качестве требуемого ссылочного действия указано `SET DEFAULT`, то строка `t` удаляется, и во всех столбцах, которые входят в состав внешнего ключа, всех строк, ссылающихся на строку `t`, проставляется заданное при их определении значение по умолчанию. Если в определении внешнего ключа содержится спецификация `MATCH PARTIAL`, то подобному воздействию подвергаются только те строки таблицы `S`, которые ссылаются исключительно на строку `t`;
- если в качестве требуемого ссылочного действия указано `SET NULL`, то строка `t` удаляется, и во всех столбцах, которые входят в состав внешнего ключа, всех строк, ссылающихся на строку `t`, проставляется `NULL`. Если в определении внешнего ключа содержится спецификация `MATCH PARTIAL`, то подобному воздействию подвергаются только те строки таблицы `S`, которые ссылаются исключительно на строку `t`.

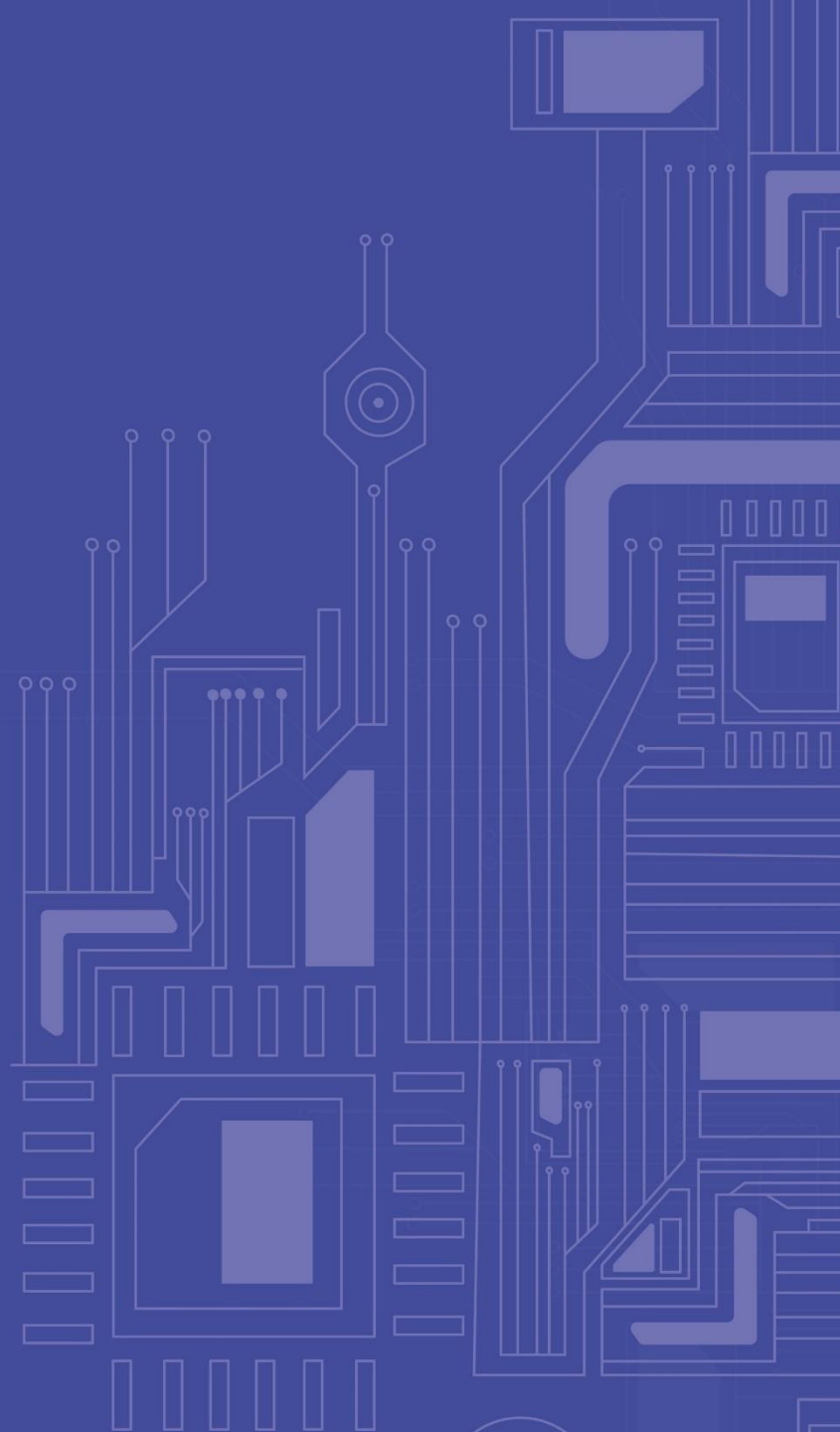
ССЫЛОЧНЫЕ ДЕЙСТВИЯ (ON UPDATE)

Пусть определение ограничения внешнего ключа содержит раздел **ON UPDATE** referential_action. Предположим, что предпринимается попытка обновить столбцы соответствующего возможного ключа в строке t из таблицы T. Тогда:

- если в качестве требуемого ссылочного действия указано NO ACTION или RESTRICT, то операция обновления отвергается, если ее выполнение вызвало бы нарушение ограничения внешнего ключа;
- если в качестве требуемого ссылочного действия указано CASCADE, то строка t обновляется, и если в определении ограничения внешнего ключа отсутствует раздел MATCH или присутствуют спецификации MATCH SIMPLE или MATCH FULL, то соответствующим образом обновляются все строки, ссылающиеся на t (в них должным образом изменяются значения столбцов, входящих в состав внешнего ключа). Если же в определении ограничения внешнего ключа присутствует спецификация MATCH PARTIAL, то обновляются только те строки, которые ссылаются исключительно на строку t;
- если в качестве требуемого ссылочного действия указано SET DEFAULT, то строка t обновляется, и во всех столбцах, которые входят в состав внешнего ключа и соответствуют изменяемым столбцам таблицы T, всех строк, ссылающихся на строку t, проставляется заданное при их определении значение по умолчанию. Если в определении внешнего ключа содержится спецификация MATCH PARTIAL, то подобному воздействию подвергаются только те строки таблицы S, которые ссылаются исключительно на строку t, причем в них изменяются значения только тех столбцов, которые не содержали NULL;
- если в качестве требуемого ссылочного действия указано SET NULL, то строка t обновляется, и во всех столбцах, которые входят в состав внешнего ключа и соответствуют изменяемым столбцам таблицы T, всех строк, ссылающихся на строку t, проставляется NULL. Если в определении внешнего ключа содержится спецификация MATCH PARTIAL, то подобному воздействию подвергаются только те строки таблицы S, которые ссылаются исключительно на строку t.


ВСЁ!

СЕМИНАР



ПРИМЕРЫ ОПРЕДЕЛЕНИЯ БАЗОВЫХ ТАБЛИЦ (1/6)

EMP:

EMP_NO : EMP_NO
EMP_NAME : VARCHAR
EMP_BDATE : DATE
EMP_SAL : SALARY
DEPT_NO : DEPT_NO
RO_NO : PRO_NO

```
(1)     CREATE TABLE EMP (
(2)     EMP_NO EMP_NO PRIMARY KEY,
(3)     EMP_NAME VARCHAR(20) DEFAULT 'Incognito' NOT NULL,
(4)     EMP_BDATE DATE DEFAULT NULL CHECK (
        VALUE >= DATE '1917-10-24'),
(5)     EMP_SAL SALARY,
(6)     DEPT_NO DEPT_NO DEFAULT NULL REFERENCES
        DEPT ON DELETE SET NULL,
(7)     PRO_NO PRO_NO DEFAULT NULL,
(8)     FOREIGN KEY PRO_NO REFERENCES PRO (PRO_NO)
        ON DELETE SET NULL,
(9)     CONSTRAINT PRO_EMP_NO CHECK
      ((SELECT COUNT (*) FROM EMP E
          WHERE E.PRO_NO = PRO_NO) <= 50));
```

Столбцы EMP_NO, EMP_SAL, DEPT_NO, PRO_NO, DEPT_TOTAL_SAL, DEPT_MNG и PRO_MNG определяются на ранее определенных доменах (определения доменов EMP_NO и SALARY приведены в лекции). Первичными ключами отношений EMP, DEPT и проектов PRO являются столбцы EMP_NO, DEPT_NO и PRO_NO соответственно. В таблице EMP столбцы DEPT_NO и PRO_NO являются внешними ключами, указывающими на отдел, в котором работает служащий, и на выполняемый им проект соответственно. В таблице DEPT внешним ключом является столбец DEPT_NO, указывающий на служащего, являющегося руководителем соответствующего отдела, а в таблице PRO внешним ключом является столбец PRO_MNG, указывающий на служащего, являющегося менеджером соответствующего проекта.

ПРИМЕРЫ ОПРЕДЕЛЕНИЯ БАЗОВЫХ ТАБЛИЦ (2/6)

```
(1)     CREATE TABLE EMP (
(2)     EMP_NO EMP_NO PRIMARY KEY,
(3)     EMP_NAME VARCHAR(20) DEFAULT 'Incognito' NOT NULL,
(4)     EMP_BDATE DATE DEFAULT NULL CHECK (
        VALUE >= DATE '1917-10-24'),
(5)     EMP_SAL SALARY,
(6)     DEPT_NO DEPT_NO DEFAULT NULL REFERENCES
        DEPT ON DELETE SET NULL,
(7)     PRO_NO PRO_NO DEFAULT NULL,
(8)     FOREIGN KEY PRO_NO REFERENCES PRO (PRO_NO)
        ON DELETE SET NULL,
(9)     CONSTRAINT PRO_EMP_NO CHECK
      ((SELECT COUNT (*) FROM EMP E
         WHERE E.PRO_NO = PRO_NO) <= 50));
```

В части (1) указывается, что создается таблица с именем EMP. В части (2) определяется столбец EMP_NO на домене EMP_NO. У этого столбца не определено значение по умолчанию, и он объявлен первичным ключом таблицы (это ограничение целостности добавляется через AND к ограничениям, унаследованным столбцом от определения домена). Помимо прочего, это означает неявное указание запрета для данного столбца неопределенных значений.

В части (3) определен столбец EMP_NAME на базовом типе данных символьных строк переменной длины с максимальной длиной 20. Для столбца указано значение по умолчанию – строка 'Incognito', и в качестве ограничения целостности запрещены неопределенные значения. В части (4) определяется столбец EMP_BDATE (дата рождения служащего). Он имеет тип данных DATE, значением по умолчанию является NULL (даты рождения некоторых служащих неизвестны). Кроме того, ограничение столбца запрещает принимать на работу лиц, о которых известно, что они родились до Октябрьского переворота. В части (5) определен столбец EMP_SAL на домене SALARY. Значение по умолчанию и ограничения целостности наследуются из определения домена.

ПРИМЕРЫ ОПРЕДЕЛЕНИЯ БАЗОВЫХ ТАБЛИЦ (3/6)

```
(1)     CREATE TABLE EMP (
(2)     EMP_NO EMP_NO PRIMARY KEY,
(3)     EMP_NAME VARCHAR(20) DEFAULT 'Incognito' NOT NULL,
(4)     EMP_BDATE DATE DEFAULT NULL CHECK (
        VALUE >= DATE '1917-10-24'),
(5)     EMP_SAL SALARY,
(6)     DEPT_NO DEPT_NO DEFAULT NULL REFERENCES
        DEPT ON DELETE SET NULL,
(7)     PRO_NO PRO_NO DEFAULT NULL,
(8)     FOREIGN KEY PRO_NO REFERENCES PRO (PRO_NO)
        ON DELETE SET NULL,
(9)     CONSTRAINT PRO_EMP_NO CHECK
      ((SELECT COUNT (*) FROM EMP E
          WHERE E.PRO_NO = PRO_NO) <= 50));
```

В части (6) столбец DEPT_NO определяется на одноименном домене (для наших целей его определение несущественно), но явно объявляется, что значением по умолчанию этого столбца будет NULL (некоторые служащие не приписаны ни к какому отделу). Кроме того, добавляется ограничение внешнего ключа: столбец DEPT_NO ссылается на первичный ключ таблицы DEPT. Определено ссылочное действие: при удалении строки из таблицы DEPT во всех строках таблицы EMP, ссылавшихся на эту строку, столбцу DEPT_NO должно быть присвоено неопределенное значение.

В части (7) определяется столбец PRO_NO. Его определение аналогично определению столбца DEPT_NO, но ограничение внешнего ключа вынесено в часть (8), где оно определяется в полной форме как табличное ограничение. Наконец, в части (9) определяется табличное проверочное ограничение с именем PRO_EMP_NO, которое требует, чтобы ни в одном проекте не участвовало больше 50 служащих.

ПРИМЕРЫ ОПРЕДЕЛЕНИЯ БАЗОВЫХ ТАБЛИЦ (4/6)

DEPT:

DEPT_NO : DEPT_NO
DEPT_NAME : VARCHAR
DEPT_EMP_NO : INTEGER
DEPT_TOTAL_SAL : SALARY
DEPT_MNG : EMP_NO

```
(1)     CREATE TABLE DEPT (
(2)     DEPT_NO DEPT_NO PRIMARY KEY,
(3)     DEPT_EMP_NO INTEGER NO NULL CHECK (
        VALUE BETWEEN 1 AND 100),
(4)     DEPT_NAME VARCHAR(200) DEFAULT 'Nameless' NOT NULL,
(5)     DEPT_TOTAL_SAL SALARY DEFAULT 1000000.00
   NO NULL CHECK (VALUE > = 100000.00),
(6)     DEPT_MNG EMP_NO DEFAULT NULL
      REFERENCES EMP ON DELETE SET NULL
      CHECK (IF (VALUE IS NOT NULL) THEN
         ((SELECT COUNT(*) FROM DEPT
            WHERE DEPT.DEPT_MNG = VALUE) = 1),
(7)     CHECK (DEPT_EMP_NO =
      (SELECT COUNT(*) FROM EMP
         WHERE DEPT_NO = EMP.DEPT_NO)),
(8)     CHECK (DEPT_TOTAL_SAL >=
      (SELECT SUM(EMP_SAL) FROM EMP
         WHERE DEPT_NO = EMP.DEPT_NO)));
```

В части (3) столбец DEPT_EMP_NO (число служащих в отделе) определен на базовом типе INTEGER без значения по умолчанию, с запретом неопределенного значения и с проверочным ограничением, устанавливающем допустимый диапазон значений числа служащих в отделе. Еще одно проверочное ограничение этого столбца – (7) – вынесено на уровень определения табличного ограничения. Это ограничение устанавливает, что в каждой строке таблицы DEPT значение столбца DEPT_EMP_NO должно равняться общему числу строк таблицы EMP, в которых значение столбца DEPT_NO равно значению одноименного столбца данной строки таблицы DEPT.

В части (5) для определения столбца DEPT_TOTAL_SAL (объем фонда заработной платы отдела) используется домен SALARY. Но при этом явно установлено значение столбца по умолчанию (отличное от значения по умолчанию домена), запрещено наличие неопределенных значений и введено дополнительное проверочное ограничение, определяющее нижний порог объема фонда заработной платы отдела. Еще одно проверочное ограничение – (8) – вынесено на уровень определения табличного ограничения. Это ограничение устанавливает, что в каждой строке таблицы DEPT значение столбца DEPT_TOTAL_SAL должно быть не меньше суммы значений столбца EMP_SAL во всех строках таблицы EMP, в которых значение столбца DEPT_NO равно значению одноименного столбца данной строки таблицы DEPT.

ПРИМЕРЫ ОПРЕДЕЛЕНИЯ БАЗОВЫХ ТАБЛИЦ (5/6)

```
(1)     CREATE TABLE DEPT (
(2)     DEPT_NO DEPT_NO PRIMARY KEY,
(3)     DEPT_EMP_NO INTEGER NO NULL CHECK (
        VALUE BETWEEN 1 AND 100),
(4)     DEPT_NAME VARCHAR(200) DEFAULT 'Nameless' NOT NULL,
(5)     DEPT_TOTAL_SAL SALARY DEFAULT 1000000.00
   NO NULL CHECK (VALUE > = 100000.00),
(6)     DEPT_MNG EMP_NO DEFAULT NULL
      REFERENCES EMP ON DELETE SET NULL
      CHECK (IF (VALUE IS NOT NULL) THEN
         ((SELECT COUNT(*) FROM DEPT
            WHERE DEPT.DEPT_MNG = VALUE) = 1),
(7)     CHECK (DEPT_EMP_NO =
      (SELECT COUNT(*) FROM EMP
         WHERE DEPT_NO = EMP.DEPT_NO)),
(8)     CHECK (DEPT_TOTAL_SAL >=
      (SELECT SUM(EMP_SAL) FROM EMP
         WHERE DEPT_NO = EMP.DEPT_NO)));
```

Определение столбца DEPT_MNG – часть (6). Этот столбец объявляется внешним ключом таблицы DEPT. Но мы хотим сказать больше. У отдела могут временно отсутствовать руководители, поэтому в столбце допускаются неопределенные значения. Но если у отдела имеется руководитель, то он должен являться руководителем только этого отдела. На первый взгляд можно было бы воспользоваться ограничением столбца UNIQUE. Но такое ограничение допускало бы наличие неопределенного столбца DEPT_MNG только в одной строке таблицы DEPT, а мы хотим допустить отсутствие руководителя у нескольких отделов. Поэтому потребовалось ввести более громоздкое проверочное ограничение столбца.

Ограничение (9) в первом определении и ограничения (7) и (8) во втором определении внешне похожи, но сильно отличаются по своей сути. Ограничения (7) и (8) связаны с агрегатной семантикой столбцов DEPT_EMP_NO и DEPT_TOTAL_SAL таблицы DEPT. Отмена ограничений изменила бы смысл этих столбцов. Ограничение (9) является текущим административным ограничением. Если руководство предприятия примет решение разрешить использовать в одном проекте более 50 служащих, ограничение можно отменить без изменения смысла столбцов таблицы EMP. Имея это в виду, мы ввели явное имя ограничения (9), чтобы при необходимости имелась простая возможность отменить это ограничение с помощью оператора ALTER TABLE.

ПРИМЕРЫ ОПРЕДЕЛЕНИЯ БАЗОВЫХ ТАБЛИЦ (6/6)

PRO:

PRO_NO : PRO_NO
PRO_TITLE : VARCHAR
PRO_SDATE : DATE
PRO_DURAT : INTERVAL
PRO_MNG : EMP_NO
PRO_DESC : CLOB

```
(1)     CREATE TABLE PRO (
(2)     PRO_NO PRO_NO PRIMARY KEY,
(3)     PRO_TITLE VARCHAR(200)DEFAULT 'No title' NOT NULL,
(4)     PRO_SDATE DATE DEFAULT CURRENT_DATE NOT NULL,
(5)     PRO_DURAT INTERVAL YEAR DEFAUL INTERVAL '1'
        YEAR NOT NULL,
(6)     PRO_MNG EMP_NO UNIQUE NOT NULL
        REFERENCES EMP ON DELETE NO ACTION,
(7)     PRO_DESC CLOB(10M));
```

Столбец PRO_SDATE содержит дату начала проекта, а столбец PRO_DURAT – продолжительность проекта в годах. В этом определении имеет смысл прокомментировать часть (6). Мы считаем, что если отдел, по крайней мере временно, может существовать без руководителя, то у проекта всегда должен быть менеджер. Поэтому определение столбца PRO_MNG является гораздо более строгим, чем определение столбца DEPT_MNG в таблице DEPT. Сочетание ограничений UNIQUE и NOT NULL при отсутствии значений по умолчанию приводит к абсолютной уникальности значений столбца PRO_MNG. Другими словами, этот столбец обладает всеми характеристиками первичного ключа, хотя объявлен только как возможный ключ. Кроме того, он объявлен как внешний ключ с действием при удалении строки таблицы EMP с соответствующим значением первичного ключа NO ACTION, запрещающим такие удаления. В совокупности это гарантирует, что у любого проекта будет существовать менеджер, являющийся служащим предприятия. В части (5) столбец PRO_DESC (описание проекта) определен как большой символьный объект с максимальным размером 10 Мбайт.

РАЗНЫЕ БАЗЫ ДАННЫХ И СУБД

Как получить список всех таблиц (включая служебные) из разных БД:

```
-- H2, HSQLDB, MySQL, PostgreSQL, SQL Server
        SELECT table_schema, table_name
        FROM information_schema.tables
-- DB2
        SELECT tabschema, tabname
        FROM syscat.tables
-- Oracle
        SELECT owner, table_name
        FROM all_tables
-- SQLite
        SELECT name
        FROM sqlite_master
-- Teradata
        SELECT databasename, tablename
        FROM dbc.tables
```

ПРИМЕР СОЗДАНИЯ ТАБЛИЦ В MS SQL

Создать *таблицу* для хранения данных о товарах, поступающих в продажу в некоторой торговой фирме. Необходимо учесть такие сведения, как название и тип товара, его цена, сорт и город, где товар производится.

```
CREATE TABLE Товар
(Название       VARCHAR(50) NOT NULL,
    Цена        MONEY NOT     NULL,
    Тип         VARCHAR(50) NOT NULL,
    Сорт        VARCHAR(50),
    ГородТовара VARCHAR(50))
```

Создать таблицу для сохранения сведений о постоянных клиентах с указанием названий города и фирмы, фамилии, имени и отчества клиента, номера его телефона.

```
CREATE TABLE  Клиент
    (Фирма       VARCHAR(50) NOT NULL,
    Фамилия      VARCHAR(50) NOT NULL,
    Имя          VARCHAR(50) NOT NULL,
    Отчество     VARCHAR(50),
    ГородКлиента VARCHAR(50),
    Телефон      CHAR(10) NOT NULL)
```

ПРИМЕРЫ SELECT В MS SQL

Составить *список* сведений о всех клиентах.

```
SELECT * FROM Клиент
```

Составить список всех фирм.

```
SELECT ALL Клиент.Фирма
FROM Клиент
```

Или (что эквивалентно)

```
SELECT Клиент.Фирма
FROM Клиент
```

Составить список всех уникальных фирм.

```
SELECT DISTINCT Клиент.Фирма
FROM Клиент
```

ПРИМЕР БД С ТЕСТАМИ ПРИБОРОВ

```
ivasonik@DESKTOP-8O2FCMA:~$ sqlite3 dev-tests.s3db
```

```
ivasonik@DESKTOP-8O2FCMA:~$ nano schema.sql
```

schema.sql

```
CREATE table devices (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        sn TEXT NOT NULL UNIQUE
);

CREATE TABLE tests (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        ts INTEGER NOT NULL,
        device_id INTEGER NOT NULL REFERENCES devices(id),
        result INTEGER --0 - fail, 1 - passed
);
```

```
ivasonik@DESKTOP-8O2FCMA:~$ sqlite3 dev-tests.s3db <schema.sql
```