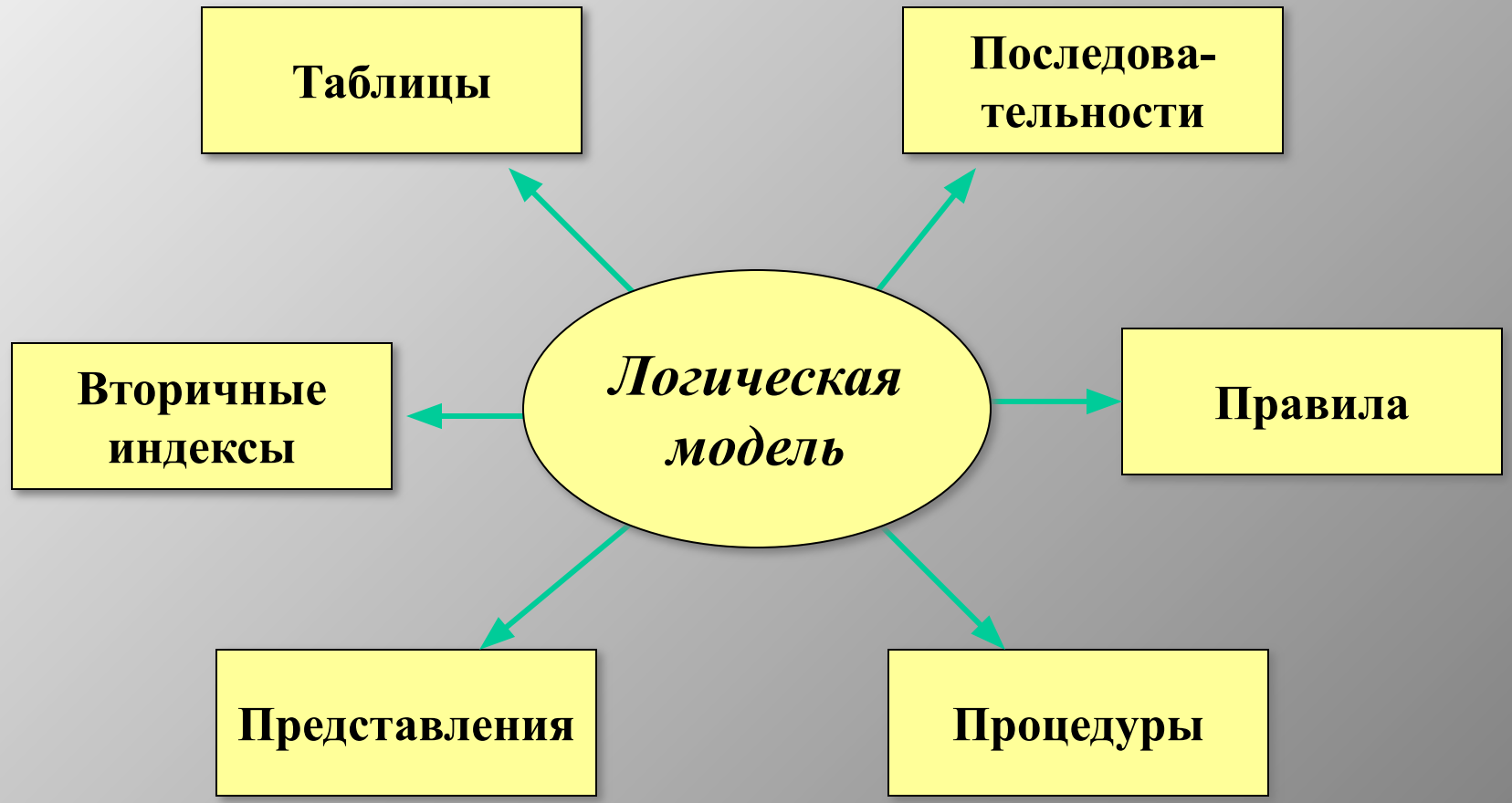
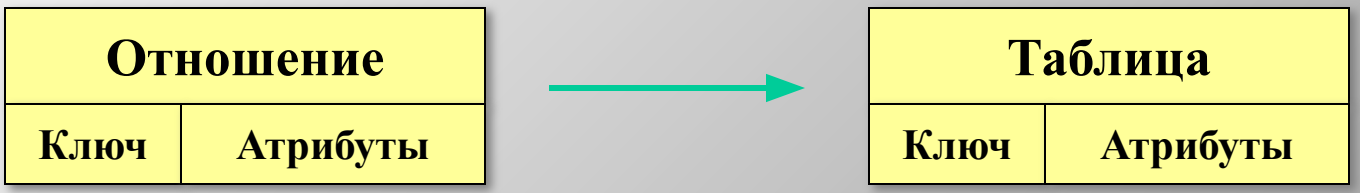
Тема 3.
Логические модели данных

Элементы логической модели

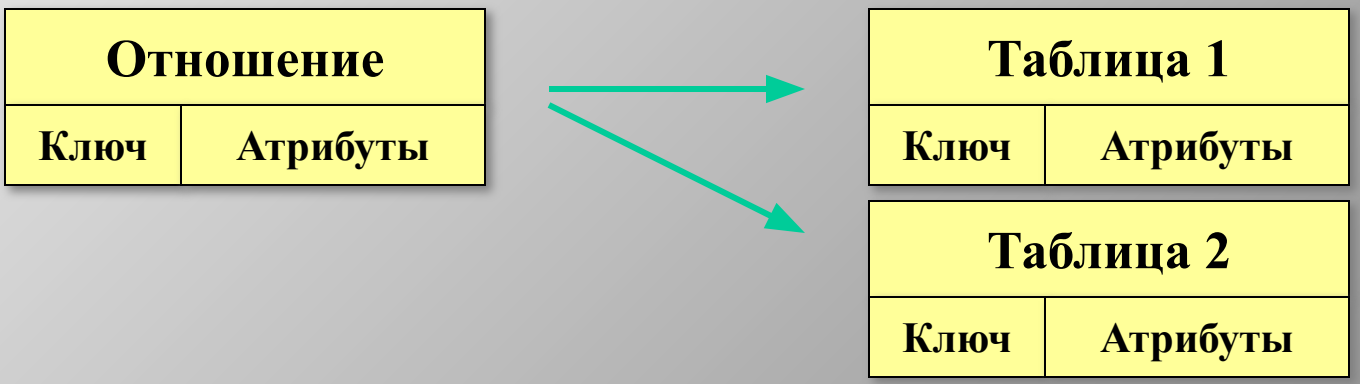


Таблицы базы данных

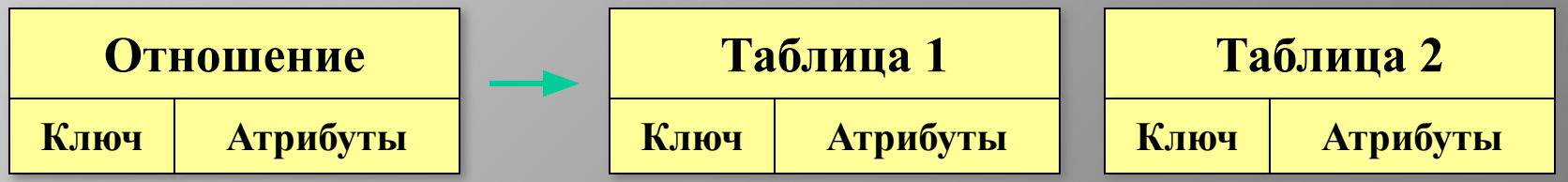
1. Отображение "отношение - таблица"



2. Вертикальная сегментация



3. Горизонтальная сегментация



Вертикальная сегментация отношений

Заказ			
Номер	**Дата**	**Клиент**	**Статус**
1	12.01	A	Выполнен
2	14.01	B	Выполнен
...			
312	18.05	C	Выполнен
313	20.05	A	В работе
314	20.05	B	В работе
315	21.05	C	Принят

Архив заказов			
Номер	**Дата**	**Клиент**	**Статус**
1	12.01	A	Выполнен
2	14.01	B	Выполнен
...			
312	18.05	C	Выполнен

Текущие заказы			
Номер	**Дата**	**Клиент**	**Статус**
313	20.05	A	В работе
314	20.05	B	В работе
315	21.05	C	Принят

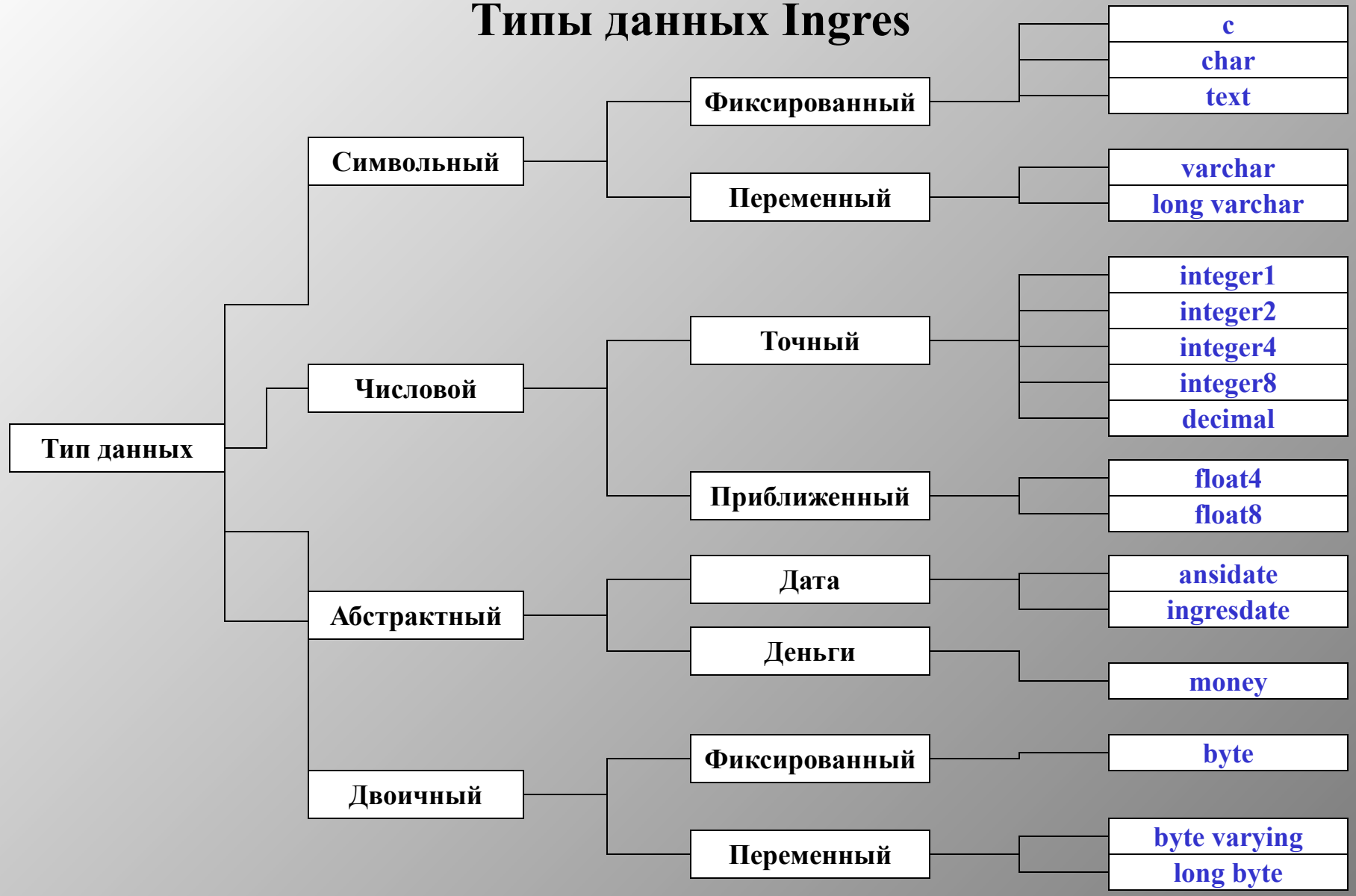
Горизонтальная сегментация отношений

Изделие		
Номер	**Наименование**	**Описание**
1	Стойка 3 полки	Для сборки ст...
...		
2000	Подставка	Подставка име...

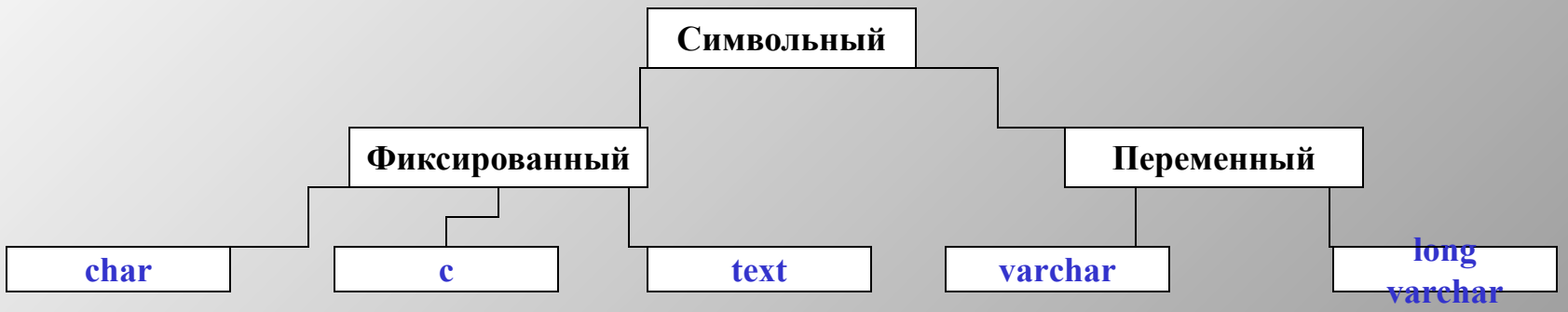
Данные изделия	
Номер	**Наименование**
1	Стойка 3 полки
...	
2000	Подставка

Описание изделия	
Номер	**Описание**
1	Для сборки стойки нужно...
...	
2000	Подставка имеет нижнюю...

Типы данных Ingres



Символьные типы данных



Тип **char** может содержать как отображаемые, так и неотображаемые символы. В таблицах без сжатия данных строка дополняется пробелами. Если сжатие задано, то концевые пробелы исключаются. В операциях сравнения пробелы принимаются во внимание, а в случае, когда строки имеют разную длину, наименьшая строка дополняется пробелами. Строки 'ABC' и 'ABC ' эквивалентны для операции сравнения.

Строка символов типа **c** может содержать только отображаемые символы, при вводе неотображаемые символы заменяются пробелами. В операциях сравнения пробелы не принимаются во внимание, то есть строки '955 76 49' и '9557649' эквивалентны.

Тип **text** может содержать все символы ASCII, за исключением нулевого, который при вводе преобразуется в пробел. В операциях сравнения пробелы принимаются во внимание.

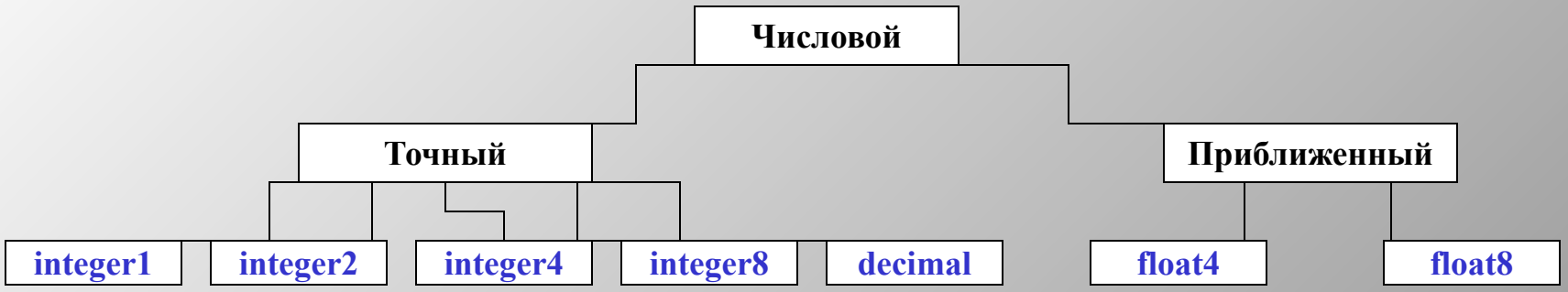
В типе данных **varchar** первые два байта используются для сохранения длины строки. Строка типа **varchar** может содержать любые символы, включая неотображаемые, а также нулевой символ ('\0'). В операциях сравнения пробелы принимаются во внимание.

Тип данных **long varchar** имеет такие же характеристики, как и varchar, но может содержать строку размером до 2 Гбайт.

Функции для работы со строками

Функция	Описание
charextract(c1, n)	Возвращает *n*-й байт строки **c1**. Если *n* превышает размер строки, то функция возвращает пробел
concat(c1, c2)	Соединение строк. Размер возвращаемой строки равен размеру строк **c1** и **c2**
left(c1, len)	Возвращает первые **len** символов строки **c1**. Если результат – строка с фиксированной длиной, то она дополняется до размера строки **c1** пробелами
length(c1)	Возвращает длину строки **c1**, если строка имеет тип данных с фиксированной длиной, и количество символов в строке **c1**, если строка имеет тип данных с переменной длиной
locate(c1, c2)	Возвращает номер байта, с которого начинается первое появление подстроки **c2** в строке **c1**, с учетом пробелов в конце подстроки **c2**. Если подстрока не найдена, то функция возвращает **size(c1)+1**
lowercase(c1)	Преобразует строку **c1** к нижнему регистру
pad(c1)	Дополняет строку **c1** пробелами
right(c1, len)	Возвращает последние **len** символов строки **c1**. Если результат – строка с фиксированной длиной, то она дополняется до размера строки **c1** пробелами
size(c1)	Возвращает заданную длину строки **c1**
trim(c1)	Удаляет хвостовые пробелы из строки **c1**
uppercase(c1)	Преобразует строку **c1** к верхнему регистру

Числовые типы данных



К целочисленным типам данных относятся **integer1**, **integer2**, **integer4**, **integer8**, размер которых равен соответственно 1, 2, 4 и 8 байт. Десятичный тип данных **decimal** определяется в терминах точности (precision) и масштаба (scale). Под точностью понимается общее количество цифр в числе, а под масштабом – количество цифр после десятичной точки. Например для числа 12345.67890 точность равна десяти, а масштаб – пяти. Допустимая точность для decimal от 1 до 31 цифры.

К типам данных с плавающей точкой относятся **float4** размером 4 байт и **float8** размером 8 байт. Число с плавающей точкой представляется как мантисса и экспонента. Например, для числа 123E4 мантисса равна 123, а экспонента равна 4. Точность для типа
данных float4 составляет 7 цифр, а для типа данных float8 – 15 цифр.

Математические функции

Функция	Описание
abs(*n*)	Абсолютная величина аргумента *n*
acos(*n*)	Арккосинус *n*
asin(*n*)	Арксинус *n*
atan(*n*)	Арктангенс *n*
ceil(*n*)	Усечение *n* до целого в верхнюю сторону
cos(*n*)	Косинус *n*
exp(*n*)	Экспонента *n*
log(*n*)	Натуральный логарифм *n*
mod(*n*, *b*)	Остаток от целого деления *n* на *b* (*n* и *b* должны быть целыми числами)
pi	Число π
power(*x*, *y*)	Возведение *x* в степень *y*
round(*n*, *i*)	Округление *n* до *i*-го разряда
sign(*n*)	Знак. Возвращает –1, если $n < 0$; 0, если $n = 0$; 1, если $n > 0$
sin(*n*)	Синус *n*
sqrt(*n*)	Квадратный корень из *n*
tan(*n*)	Тангенс *n*
trunc(*n*, *i*)	Усечение *n* до *i*-го разряда в нижнюю сторону

Денежный тип данных

money

Тип данных **money** имеет точность два знака после десятичной точки и допускает значения от $-999999999999.99 до $+999999999999.99, где $ является знаком денежной единицы, по умолчанию – доллар. Знак денежной единицы и точность в диапазоне от 0 до 2 можно задавать с помощью переменных среды II_MONEY_FORMAT и II_MONEY_PREC соответственно. начения в этом формате округляются до двух знаков после деся-тичной точки. Под тип данных money отводятся 8 байт.

Временной тип данных

ingresdate

Тип данных **ingresdate** позволяет представлять как абсолютную дату
и время, так и временной диапазон. Значение типа данных ingresdate описывается как строка символов в апострофах, например '15-nov-93 10:30'. Поддерживается достаточно много форматов даты, и их виды задаются через переменную среды II_DATE_FORMAT. Временной диапазон также заключается в апострофы и может быть задан в годах, месяцах, днях, часах, минутах, секундах или в их комбинации, например '5 years 8 months', '23 hrs 38 mins 53 secs'. Допустимый диапазон значений для years от –800 до +800, для months от –9611 до 9611, для days от –292 559 до +292 559. Тип данных ingresdate занимает 12 байт.

Функции для работы с датами

Функция	Описание
date_trunc(unit, date)	Возвращает усеченную дату, способ усечения задается параметром **unit**, который может принимать следующие значения: **second**, **minute**, **hour**, **day**, **week**, **month**, **quarter**, **year**
date_part(unit, date)	Возвращает часть даты, заданную параметром **unit**
date_gmt(date)	Преобразует абсолютную дату к дате по Гринвичу
interval(unit, interval)	Преобразует интервал времени к формату с плавающей точкой в единицах, заданных параметром **unit**

Сводная таблица типов данных

Тип	Описание
c(n)	Строка символов фиксированной длины размером *n* символов, *n* не должно превышать размер страницы
char(n)	Строка символов фиксированной длины размером *n* символов, *n* не должно превышать размер страницы
varchar(n)	Строка символов переменной длины, *n* – максимальное число символов в строке, *n* не должно превышать размер страницы
long varchar	Текстовые данные размером до 2 Гбайт
text(n)	Строка символов фиксированной длины размером *n* символов, *n* не должно превышать размер страницы
integer1	Целое однобайтное число в диапазоне от –128 до 127
integer2	Целое 2-байтное число в диапазоне от –32 768 до 32 767
integer4	Целое 4-байтное число в диапазоне от –2 147 483 648 до 2 147 483 647
Integer8	Целое 8-байтное число в диапазоне от $-2^{63}-1$ до 2^{63}
decimal	Десятичное число с фиксированной точкой, максимальное число разрядов – 31
float4	4-байтное число с плавающей точкой в диапазоне от –1,0E+38 до +1,0E+38 (с точностью 7 цифр)
float8	8-байтное число с плавающей точкой в диапазоне от –1,0E+38 до +1,0E+38 (с точностью 16 цифр)
ingresdate	12-байтная дата, может содержать дату и время или интервалы времени в диапазоне от 01.01.1532 до 31.12.2382

Сводная таблица типов данных (окончание)

Тип	Описание
money	8-байтная денежная единица в диапазоне от –999 999 999 999,99 до 999 999 999 999,99
byte(n)	Двоичные данные фиксированной длины размером *n* байт, *n* не должно превышать размер страницы
byte varying(n)	Двоичные данные переменной длины размером *n* байт, *n* не должно превышать размер страницы
long byte	Двоичные данные размером до 2 Гбайт

В Ingres Open Source для типов данных **c**, **char**, **varchar**, **text**, **byte** и **byte varying** устанавливается лимит в 2000 байт для стандартной страницы 2 Кбайт. Этот лимит может быть увеличен путем увеличения размера страницы со стандартного в 2 Кбайт до 4, 8, 16, 32 или 64 Кбайт. Однако в любом случае лимит не может превышать 32 000 байт.

В СУБД Ingres Open Source имеются специальные типы данных – ключи, которые позволяют автоматически генерировать уникальные значения атрибутов в пределах одной таблицы или для базы данных в целом. Для приложений, требующих управления санкционированным доступом к данным, могут применяться метки защиты – специальный тип данных для контроля доступа.

Неопределенные значения атрибутов

Номер	Обозначение	Наименование	Цвет	Цена
1	КР.121.00.01	Стойка 4 полки 1200 × 800	Синий	3640.00
2	КР.121.00.02	Стойка 4 полки 1200 × 800	Зеленый	3680.00
3	КР.116.00.01	Подставка малая	Красный	null
4	КР.117.00.01	Подставка средняя	Красный	845.00

Неопределенное значение **null** не следует путать с нулевым значением, пустой строкой или строкой, заполненной пробелами. Любая арифметическая операция с неопределенным значением дает в результате также неопределенное значение, например 5 + null = null. Любая логическая операция сравнения, в которой участвует неопределенное значение, дает в результате ложное значение (false).

Для хранения признака неопределенного значения атрибута в каждой записи таблицы используется дополнительное поле в 1 байт. Размер записи таблицы равен, таким образом, сумме размеров внутреннего представления атрибутов плюс один байт на каждый атрибут, который может иметь неопределенное значение.

Структуры хранения данных

HEAP	HASH	ISAM	BTREE

Структура **heap**: размещение и доступ к данным последовательны; у таблицы не может быть ключа; записи всегда добавляются в конец таблицы. Эта структура хранения является самой быстрой для добавления данных и самой медленной для чтения, поскольку при любом запросе просматриваются все страницы таблицы.

Структура **hash**: таблица должна иметь ключ, состоящий из одного или нескольких атрибутов; доступ к записям осуществляется по значению ключа с использованием специального алгоритма. Структура хранения hash работает быстро при доступе по точному значению ключа. Вместе с тем, структура hash неэффективна при поиске по диапазону значений и по части составного ключа.

Структура **isam**: данные хранятся с использованием статического индекса; для структуры хранения isam должен быть задан ключ из одного или нескольких атрибутов, записи сортируются по значению ключа; при добавлении в таблицу или изменении данных индекс не изменяется. Эта структура хранения эффективна при доступе по точному значению ключа, по диапазону значений ключа и по левой части составного ключа. Структура isam не эффективна при быстром росте таблицы.

Структура **btree**: доступ к данным – с использованием динамического индекса. В данной структуре индекс растет при росте таблицы, что исключает проблемы, которые характерны для структуры isam. Структура btree эффективна при доступе по точному значению, по диапазону значений и по левой части составного ключа; не эффективна для статических таблиц.

Вторичные индексы

Часто возникают ситуации, когда в условиях запросов фигурируют неключевые атрибуты – тогда для повышения эффективности доступа применяются вторичные индексы. Вторичный индекс представляет собой инвертированную таблицу, ключом которой является неключевой атрибут (или несколько атрибутов) основной таблицы. Записи индекса связаны с записями основной таблицы с помощью специальных указателей.

Представления

Представление (**view**) является “виртуальной таблицей”, содержимое которой определяется результатом лежащего в его основе запроса на чтение. При создании представление получает имя и его определение сохраняется в базе данных. Представления используются по нескольким причинам. С их помощью можно ограничить доступ к данным, разрешая пользователям видеть только некоторые из строк и столбцов таблицы. С помощью представления можно упростить проектирование запросов, заменяя запрос к нескольким таблицам запросом к представлению. Если доступ к данным или ввод данных осуществляется с помощью представления, СУБД может автоматически проверять, выполняются ли заданные условия целостности. Недостатками представлений являются замедление выполнения запросов (СУБД приходится преобразовывать запрос к представлению в запрос к исходным таблицам) и ограничения на их обновления.

Процедуры

Процедура базы данных представляет собой набор операторов SQL, хранимых и исполняемых как единый блок. Преимущество использования процедур заключается в сокращении обмена запросами между приложением и СУБД, а также в увеличении быстродействия приложения за счет того, что процедуры хранятся в подготовленном к выполнению виде. Использование процедур сокращает время разработки приложений, поскольку приложения могут использовать общие процедуры базы данных. Использование процедур также позволяет повысить уровень защиты данных путем предоставления пользователям доступа в процедуре к операциям над таблицами, без предоставления полного доступа к этим таблицам.

Правила

В Ingres Open Source с некоторым событием, происходящим при изменении базы данных, можно связать сопутствующее действие, которое СУБД будет автоматически выполнять при каждом возникновении события. Такое действие должно задаваться как процедура базы данных. Механизм вызова процедур при появлении события называется правилом базы данных (**rule**). Изменения базы данных могут быть результатом добавления, изменения или удаления данных.

К преимуществам использования правил можно отнести сокращение размера приложений за счет того, что правила хранятся в базе данных и вызываются при каждом добавлении или обновлении. Однако у правил имеется ряд недостатков. Правила приводят к усложнению базы данных, так как логика правил становится частью базы данных. Также исчезает возможность управлять из приложения всеми происходящими в базе данных процессами, поскольку программные запросы могут вызывать выполнение различных скрытых действий.

Последовательности

В реляционных СУБД в силу применяемой модели данных отсутствует понятие последовательного номера записи в таблице. Это приводит к сложностям при решении таких задач, как нумерация записей таблицы или результатов запроса, генерация последовательных ключей и им подобных. Для того чтобы облегчить выполнение подобных действий, в СУБД Ingres Open Source применяются генераторы последовательностей (**sequence**).

Последовательности задаются в логической модели данных. Каждая последовательность имеет уникальное имя и набор параметров, задающий стартовое значение, шаг инкрементации, конечное значение, цикличность и т. п.