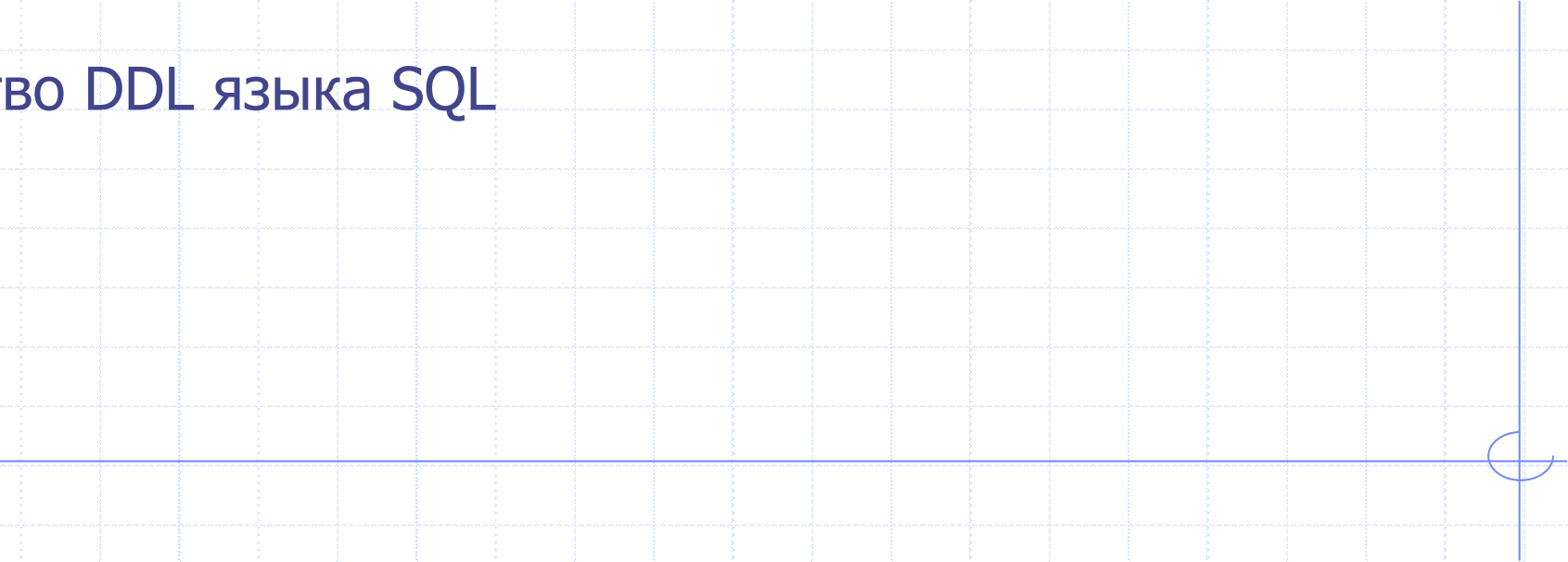


Схема базы данных

Подмножество DDL языка SQL



Метаданные

- ◆ Все объекты, определенные в базе данных называются *метаданными* и *определяют её схему*
- ◆ СУБД сохраняют метаданные в таблицах, которые создаются непосредственно в самой базе данных в разделе *системных таблиц* (принцип единообразия)
- ◆ Для управления схемой базы данных используется язык определения данных (DDL) – подмножество языка SQL

Системные таблицы

Системные таблицы (их также называют системным каталогом) — это хранилище метаданных, то есть информации о структуре базы данных, объектах (таблицах, индексах, последовательностях, процедурах и т.д.), правах доступа и служебных сведениях

Важный момент — системные таблицы соответствуют реляционной модели и для работы с ними используется язык SQL

Некоторые системные таблицы PostgreSQL

pg_class

Информация о таблицах, индексах, последовательностях и других отношениях

pg_attribute

Данные о столбцах всех отношений (таблиц)

pg_index

Информация об индексах

pg_database

Список всех баз данных в кластере

pg_user или pg_roles

Данные о пользователях и ролях

pg_tablespace

Информация о табличных пространствах

pg_stat_activity

Информация об активных подключениях

Операторы DDL

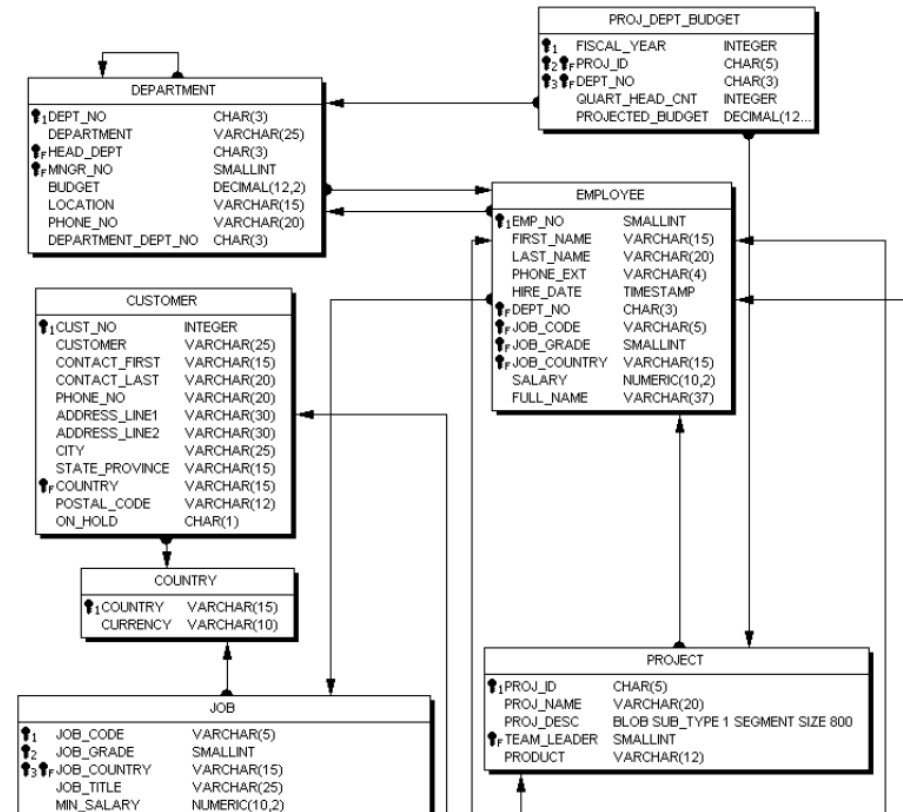
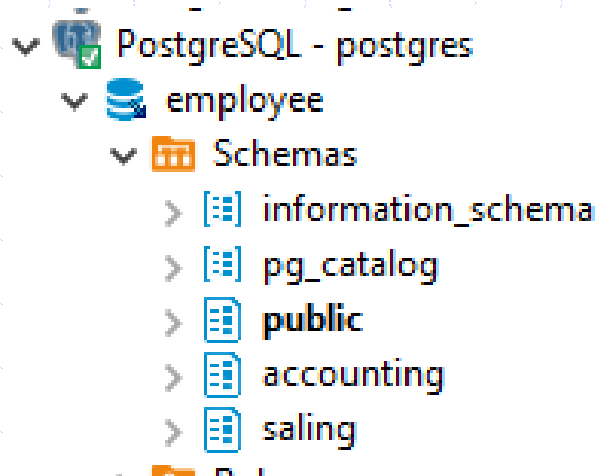
- ◆ Создание объекта схемы CREATE (INSERT в системные таблицы)
- ◆ Изменение объекта схемы ALTER (UPDATE, INSERT и DELETE)
- ◆ Удаление объекта схемы DROP (DELETE и UPDATE)

БД и схема БД

◆ В чем разница?

- Схема – структура БД

- Схема- объект БД



Создание БД (PostgreSQL)

```
CREATE DATABASE <имя> [ WITH ] [owner = <имя пользователя>]  
[<список атрибутов>]
```

CREATE DATABASE вставляет строку в таблицу pg_database — и создаёт базу данных на диске

Право выполнять команду имеют:

- ✓ Администраторы
- ✓ Пользователи с привилегией CREATE DATABASE

Изменение/удаление БД

ALTER DATABASE имя [[WITH] параметр [...]]

изменяет атрибуты базы данных

DROP DATABASE [IF EXISTS] имя [[WITH] (FORCE)]

удаляет базу данных.

Выполнить может только владелец базы данных

Схема

Создать новую схему в текущей базе данных
`CREATE SCHEMA имя_схемы ...`

Изменить схему (имя или владельца)
`ALTER SCHEMA имя_схемы ...`

Удалить схему
`DROP SCHEMA [ IF EXISTS ] имя_схемы ...`

`CREATE SCHEMA students_work AUTHORIZATION postgres;`

Основные объекты схемы

- ◆ DOMAIN
- ◆ FUNCTION
- ◆ PROCEDURE
- ◆ ROLE
- ◆ SEQUENCE
- ◆ TRIGGER
- ◆ USER

- ◆ TABLE
- ◆ CONSTRAINT Ограничения целостности таблиц (связаны с таблицей)
 - NOT NULL
 - CHECK
 - PRIMARY KEY
 - UNIQUE
 - FOREIGN KEY
- ◆ INDEX (связаны с таблицей)
- ◆ VIEW (связаны с таблицами)

Создание таблицы

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ]  
TABLE  
[ IF NOT EXISTS ]  
имя_таблицы ( столбцы [ограничения])
```

Пример

```
CREATE TABLE public.department (  
    dept_no bpchar(3) NOT NULL,  
    department varchar(25) NOT NULL,  
    head_dept bpchar(3) NULL,  
    mngr_no int2 NULL,  
    budget numeric(12, 2) NULL,  
    "location" varchar(15) NULL,  
    phone_no varchar(20) NOT NULL DEFAULT '+7(863)123-45-67'::character varying,  
    CONSTRAINT department_pk PRIMARY KEY (dept_no),  
    CONSTRAINT department_un UNIQUE (department),  
    CONSTRAINT department_fk FOREIGN KEY (head_dept)  
        REFERENCES public.department(dept_no),  
    CONSTRAINT department_fk_mngr FOREIGN KEY (mngr_no)  
        REFERENCES public.employee(emp_no)  
);
```

Изменение таблицы

ALTER TABLE <имя_таблицы > <команды>

Возможные команды:

ADD <добавляемый элемент>

ALTER <изменяемый элемент>

DROP <удаляемый элемент>

RENAME <переименуемый элемент>

Пример

```
CREATE TABLE public.goods_wh (  
    id integer NOT NULL,  
    id_wh character(10) NOT NULL,  
    id_goods character(10) NOT NULL,  
    quantity numeric(15,2)  
);  
  
ALTER TABLE public.goods_wh ALTER COLUMN id ADD GENERATED BY DEFAULT AS IDENTITY (  
    SEQUENCE NAME public.goods_wh_id_seq  
    START WITH 1    INCREMENT BY 1    NO MINVALUE    NO MAXVALUE    CACHE 1  
);  
  
ALTER TABLE public.goods_wh  
    ADD CONSTRAINT goods_wh_pkey PRIMARY KEY (id);  
  
ALTER TABLE public.goods_wh  
    ADD CONSTRAINT goods_wh_id_goods_fkey FOREIGN KEY (id_goods) REFERENCES public.goods(id_goods);  
  
ALTER TABLE public.goods_wh  
    ADD CONSTRAINT goods_wh_id_wh_fkey FOREIGN KEY (id_wh) REFERENCES public.warehouse(id_wh);
```

Типы данных

Целые	smallint (2 байта), integer (4 байта), bigint (8 байт)
Автоинкременты	smallserial, serial, bigserial
С фиксированной точкой	numeric , decimal, money
С плавающей точкой	real, double precision
Строки фиксированной длины	character(n) или char(n)
Строки переменной длины	character varying(n) или varchar(n)
Текст произвольной длины	text
Дата и время	date, time, timestamp, timestamptz, interval
Логические	boolean (true, false, NULL)
Документальные	json, jsonb
Двоичные	bytea
Прочие	uuid, inet, cidr, xml, массивы, геометрические данные, составные типы, диапазоны ...

Ограничения (constraints)

- ◆ Средства контроля за корректностью информации в БД
- ◆ Таблица не может содержать ни одной строки, в которой бы нарушалось ограничение, независимо от момента создания ограничения
- ◆ Попытка добавить новое ограничение, если в таблице уже есть данные, нарушающие его, завершится неудачей
- ◆ Ограничения могут создаваться одновременно с созданием таблицы или после создания
- ◆ Добавление/удаление ограничений через команду `ALTER TABLE`

Ограничение проверки (CHECK)

- ◆ Если ограничение на допустимые значения может быть сформулировано в виде условия, его можно задать с помощью элемента базы данных – ограничения (constraint) CHECK
- ◆ Ограничения CHECK могут быть, как определены сразу при создании таблицы в операторе CREATE TABLE, так и добавлены (изменены или удалены) в операторе ALTER TABLE
- ◆ Ограничения CHECK удобно задавать в доменах, создавая пользовательский тип данных с контролем значений

Пример

```
CREATE TABLE products (  
    product_no integer,  
    name text,  
    price numeric CHECK (price > 0)  
);
```

```
CREATE TABLE products (  
    product_no integer,  
    name text,  
    price numeric  
    CONSTRAINT positive_price  
    CHECK (price > 0) );
```

NULL

- ◆ SQL поддерживает возможность представления отсутствующих данных путем введения признака (NULL–значений)
- ◆ Правило Кодда – для любого типа данных должно быть единообразное представление NULL–значений
- ◆ СУБД резервирует в каждой хранимой строке по одному дополнительному биту на каждый столбец, в котором допустимы NULL–значения.
- ◆ Дополнительный бит служит «индикатором» присутствия или отсутствия содержательного значения в самом поле
- ◆ Стандарт ANSI/ISO и большинство СУБД поддерживают возможность задать для отдельных столбцов в качестве ограничения целостности условие обязательности данных (NOT NULL)
- ◆ Как правило, СУБД не позволяют изменять это условие для таблиц с помощью оператора ALTER TABLE

Пример

```
CREATE TABLE products (  
    product_no integer NOT NULL,  
    name text NOT NULL, -- CHECK (name IS NOT NULL)  
    type text NULL,  
    price numeric );
```

Особенности NULL

- ◆ Большинство СУБД значение арифметического или символьного выражения, среди операндов которого встретилось значение NULL, полагает равным NULL
- ◆ Логическое выражение, среди операндов которого встретилось NULL, принимает значение UNKNOWN – не определено

Всегда unknown !!!

«=NULL» или «<>NULL»

Всегда NULL !!!

« +NULL »

« || NULL »

Предикат проверки

IS NULL

IS NOT NULL

SQL-92

$\text{NULLIF}(v1, v2) = \begin{cases} \text{NULL}, & v1=v2 \\ v1, & \text{иначе} \end{cases}$

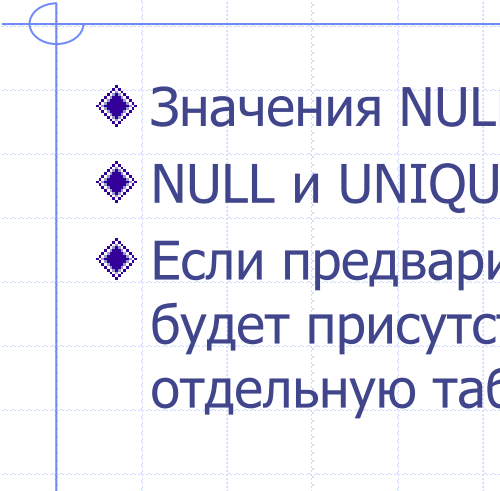
`update RESERV set KOL=NULLIF(KOL,-1);`

`COALESCE(v1,v2,v3,...vn)` – первое NOT NULL из списка или NULL

```
select NAME, COALESCE (STATE,0.0) + COALESCE(COMIS,0.0)
from PAY;
```

`CONCAT(выражение1 [, выражение2, ...])` игнорирует NULL

Агрегатные функции игнорируют NULL

- 
- ◆ Значения NULL недопустимы в столбцах, входящих в первичный ключ
 - ◆ NULL и UNIQUE
 - ◆ Если предварительный анализ показывает, что в большинстве строк постоянно будет присутствовать значение NULL, лучше выделить такой столбец в отдельную таблицу, установив связь с исходной с помощью внешнего ключа

Ограничение уникальности и первичного ключа

Задаются для столбца

column [CONSTRAINT constraint] {PRIMARY KEY | UNIQUE}

Для нескольких столбцов задаются как ограничение таблицы

[CONSTRAINT constraint]

{PRIMARY KEY | UNIQUE} (column [, column ...])

Пример

```
CREATE TABLE products (  
    product_no integer PRIMARY KEY,  
    name text UNIQUE,  
    price numeric );
```

```
CREATE TABLE example (  
    a integer,  
    b integer,  
    c integer,  
    CONSTRAINT must_be_different UNIQUE (a, c) );
```

Требования ограничения первичного ключа

- ◆ Все столбцы, входящие в первичный ключ должны иметь ограничение NOT NULL
- ◆ При определении ограничения PRIMARY KEY для комбинации значений, входящих в первичный ключ автоматически устанавливается ограничение целостности UNIQUE
- ◆ Кроме того, для каждого первичного ключа автоматически создается индекс, ускоряющий поиск по ключу.
- ◆ Ограничение первичного ключа командой ALTER TABLE будет создано только в том случае, если содержащиеся в таблице данные не противоречат этому ограничению.

Выбор первичного ключа

- ◆ Хотя само ограничение PRIMARY KEY не является ссылочным ограничением, оно обычно является частью ограничения ссылочной целостности
- ◆ Рекомендуется при реализации базы данных заменять составные первичные ключи и ключи, основанные на строковых типах, простыми числовыми ключами
- ◆ Для этого достаточно ввести в таблицу дополнительное целочисленное поле, имеющее смысл: «порядковый номер», «идентификационный номер» или «регистрационный номер»
- ◆ Такой первичный ключ называется *суррогатным*
- ◆ Для упрощения работы с суррогатными первичными ключами, пользователь обычно не должен заниматься вводом их значений. Вместо этого используются автоинкрементные типы данных или заменяющие их механизмы (последовательности, генераторы, триггеры)

Ограничение UNIQUE

◆ Контролирует отсутствие дубликатов

- допускает одно или более значений NULL в столбце, на который наложено ограничение UNIQUE

```
RECREATE TABLE t( x INT, y INT, z INT, UNIQUE(x,y,z));  
INSERT INTO t VALUES( NULL, 1, 1 );  
INSERT INTO t VALUES( NULL, NULL, 1 );  
INSERT INTO t VALUES( NULL, NULL, NULL );  
INSERT INTO t VALUES( NULL, NULL, NULL ); -- Разрешено  
INSERT INTO t VALUES( NULL, NULL, 1 );      -- Запрещено
```

- Чтобы изменить, добавляется NULLS NOT DISTINCT

Ограничение внешнего ключа

При определении столбца

column [CONSTRAINT constraint] REFERENCES other_table (column)

Как ограничение таблицы

[CONSTRAINT constraint]

| **FOREIGN KEY** (col [, col ...]) REFERENCES other_table

в любое из определений ограничения можно добавить поведение

[ON DELETE {NO ACTION | RESTRICT | CASCADE | SET DEFAULT | SET NULL}]

[ON UPDATE {NO ACTION | RESTRICT | CASCADE | SET DEFAULT | SET NULL}]

Пример

```
CREATE TABLE t1 ( a integer PRIMARY KEY,  
                  b integer,  
                  c integer,  
                  d integer REFERENCES other_table1 (pk1),  
                  FOREIGN KEY (b, c) REFERENCES other_table (c1, c2) );
```

Внешний ключ

- ◆ Внешний ключ – это столбец или набор столбцов в одной таблице, которые соответствуют столбцу или набору столбцов определяющих возможный ключ (первичный или unique) другой таблицы
- ◆ Внешний ключ необходим для реализации ссылочной целостности в БД
- ◆ В общем случае внешний ключ реализует необязательную связь один–ко–многим. При этом таблица, на которую ссылается другая таблица, называется родительской. Ссылающаяся таблица называется дочерней
- ◆ Термин «ссылочная целостность» обозначает возможности базы данных защищать себя от получения данных, которые могут послужить источником нарушения связи между двумя таблицами
- ◆ В системах управления реляционными базами данных ссылочная целостность поддерживается посредством ограничения внешнего ключа FOREIGN KEY

Операции, которые могут нарушить ссылочную целостность

- ◆ Добавление новой строки в дочернюю таблицу
- ◆ Изменение строки дочерней таблицы, затрагивающее внешний ключ
- ◆ Изменение строки родительской таблицы, затрагивающее возможный ключ, для которого есть связи из дочерней таблицы
- ◆ Удаление строки родительской таблицы, для которой есть связанные строки в дочерней таблицы

Добавление ограничений в существующую таблицу

- ◆ Добавление/удаление ограничений через команду ALTER TABLE
- ◆ Таблица не может содержать ни одной строки, в которой бы нарушалось ограничение
- ◆ Попытка добавить новое ограничение, если в таблице уже есть данные, нарушающие его, завершится неудачей

```
ALTER TABLE OPERATION  
ADD CONSTRAINT FK_OPERATION_4  
FOREIGN KEY (ID_WH)  
REFERENCES WAREHOUSE (ID_WH);
```

Домены

- ◆ Домен — пользовательский тип данных, основанный на другом нижележащем типе. Он может быть определён с условиями, ограничивающими множество допустимых значений подмножеством значений нижележащего типа. В остальном он ведёт себя как нижележащий тип — например, с доменными типом будут работать любые операторы или функции, применимые к нижележащему типу
- ◆ Назначение доменов – вынести некоторые ограничения целостности над базовым типом данных СУБД на уровень описания схемы базы данных
- ◆ Все столбцы в разных таблицах, которые были определены с одним доменом, будут иметь идентичный тип данных и идентичные ограничения, за исключением тех ограничений, которые будут переопределены на уровне описания таблицы

```
CREATE DOMAIN posint AS integer CHECK (VALUE > 0);
```

Создание домена

CREATE DOMAIN имя домена
[AS] тип данных
[DEFAULT литерал]
[NOT NULL| NULL]
[CHECK (условие)];

ALTER DOMAIN . . .

DROP DOMAIN . . .

Домены

```
create domain CUSTNO  
as integer  
default 9999  
check (VALUE>1000);
```

```
create domain TELNO  
as integer  
check ((value is null) or  
      (value between 100 and 999));
```

Использование доменов

```
create table TC  
(id_cust CUSTNO NOT NULL,  
  cust_tel TELNO,  
  ...)
```

NOT NULL можно только добавить
DEFAULT можно изменить
CHECK можно только усилить