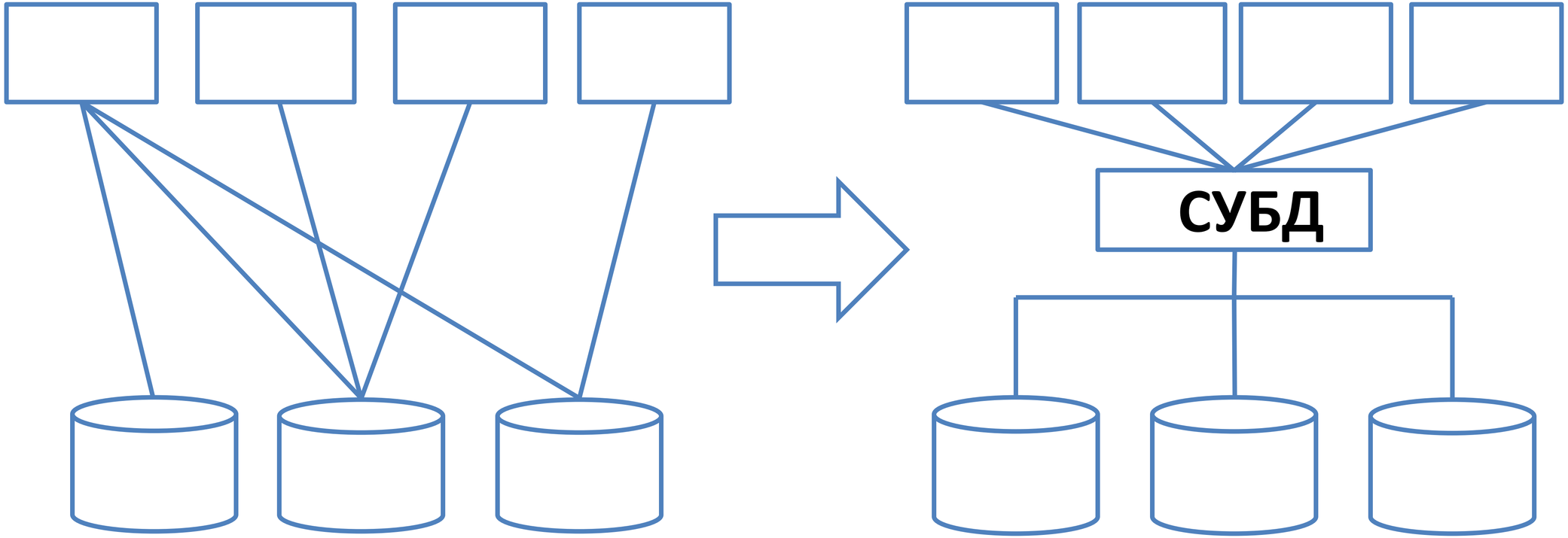
Основы технологий баз данных

Б. А. Новиков

Предыдущие серии

краткое содержание

Централизация управления общими данными нескольких приложений



Демонстрационная база данных PostgreSQL

https://postgrespro.ru/education/demodb

Ключевые свойства моделей данных

- Методы идентификации объектов внутри и вне модели
- Поиск и связывание объектов
- Массовая или штучная обработка объектов данных

Postgres PROFESSIONAL

Теоретическая реляционная модель данных

- Домены, отношения, операции
- Функциональные зависимости и ключи
- Естественная идентификация по значениям атрибутов
- Идентификация определяется функциональными зависимостями
- Связи вычисляются динамически при выполнении операций соединения
- Ассоциативный доступ к данным по значениям (операция селекции)

Postgres PROFESSIONAL

Другие модели данных

- Сущность-связь
- Объектно-ориентированные
- Объектно-реляционные
- Сетевая и иерархическая
- Слабоструктурированные
- Тернарная
- Бинарная

Языки запросов реляционных СУБД

Реляционная алгебра

- Операции
  - Селекция (фильтрация), проекция
  - (Декартово) произведение, соединение, эквисоединение, полусоединение, антисоединение
  - Объединение, пересечение, разность
- (Алгебраические) выражения

Реляционные исчисления

- Предикаты, построенные с помощью логических операций и кванторов
- Не допускается рекурсия

- Все реляционные языки эквивалентны по выразительной мощности

Мощность языков запросов

- Поисковая алгебра

  Включают фильтрацию и теоретико-множественные операции (объединение, пересечение и разность)

- Алгебра с полусоединениями

  Поисковые с полусоединениями

- Реляционные языки

  Все реляционные операции без рекурсии

- Рекурсивные языки с ограничением отрицания

  Datalog

- Языки, эквивалентные машине Тьюринга

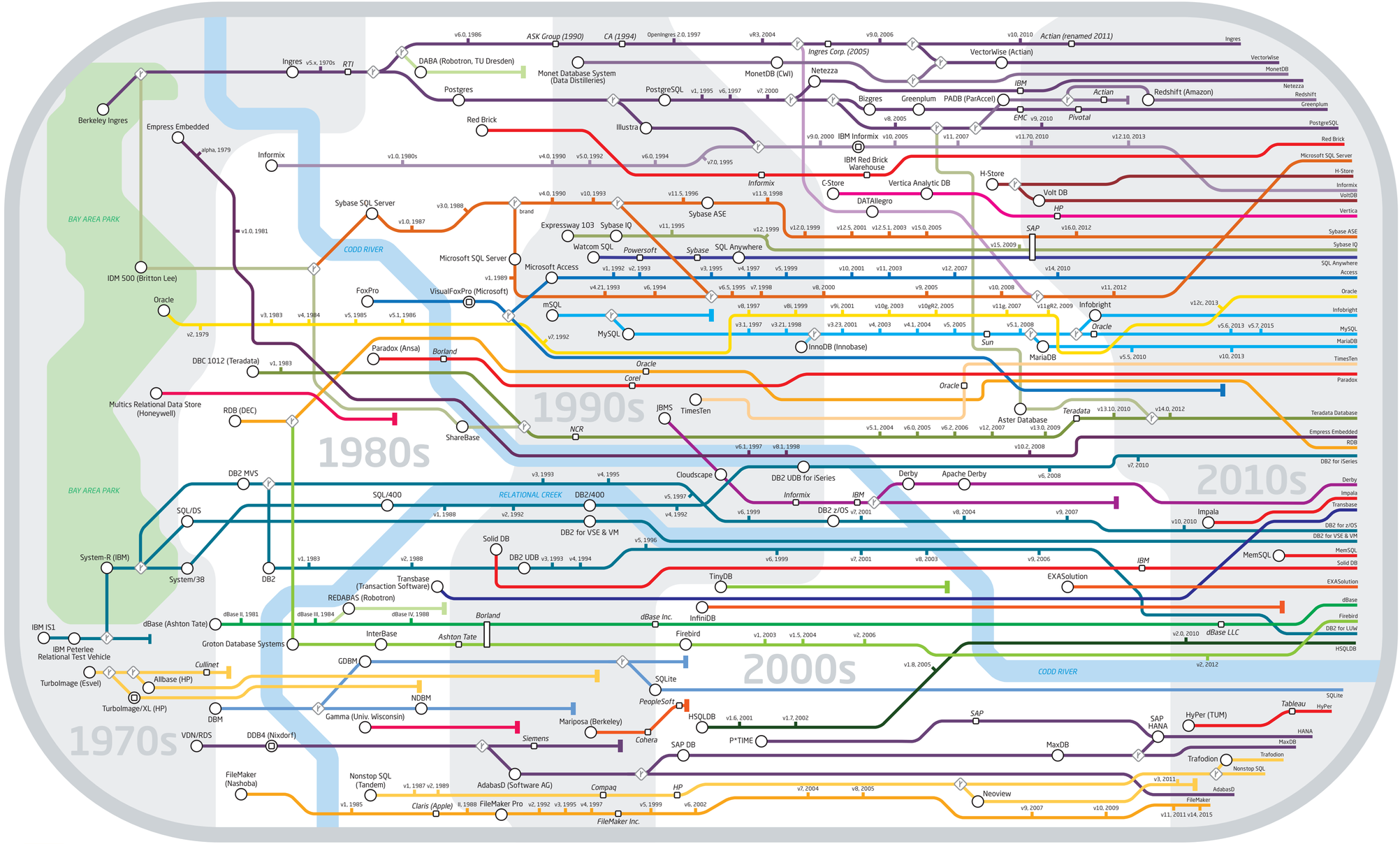
Postgres PROFESSIONAL

Декларативные и императивные языки

- Запись на императивном языке (программирования) задает действия, необходимые для получения результата
- Запрос (или программа) на декларативном языке задает условия, которым должен удовлетворять результат, но не способ получения результата
- Декларативные языки предоставляют
  - больше возможностей для выбора оптимального метода вычисления
  - более выразительную запись, близкую к спецификациям
- Императивные языки проще для программистов
- Реляционное исчисление декларативно, алгебра императивна, но допускает оптимизацию

Postgres PROFESSIONAL

Генеалогия реляционных СУБД



https://hpi.de/naumann/projects/rdbms-genealogy.html

PROFESSIONAL Postgres

Декларативный язык запросов SQL

Происхождение

- IBM System R: sequel, 1975–1984

  sequel <= SEQueL <= Structured English Query Language

- SQL 1986 — первый стандарт

- 1989, 1992, 1999, 2003, . . .

Синтаксис

- Зарезервированные ключевые слова (несколько десятков)
  - SELECT, FROM, WHERE, INSERT, UPDATE, DELETE, . . .
- Регистр (почти всегда) игнорируется
- Константы: 123, 12.3, ‘Текстовые строки’, . . .
- Имена: FLIGHTS, Bookings, flights.departure_airport, “SELECT”, “Имя”
- Операторы, предложения, выражения

Postgres PROFESSIONAL

Типы данных SQL

Числовые типы

- Целые (smallint, integer, bigint, int)
- Вещественные (real, double precision, float, float8)
- Десятичные (decimal, numeric)
- Типы serial
- Самое важное: money

```
demo=# SELECT 1234.56::int AS "Целое число";
 Целое число
--------------
         1235
(1 row)

demo=# SELECT 1234.5678::money;
   money
-----------
 $1,234.57
(1 row)
```

Даты, время, интервалы времени

- timestamp
- timestamp with time zone
- date
- time
- time with time zone
- interval

Текстовые строки

- varchar (*length*)
- char (*length*)
- text
- byte

Язык описания данных SQL DDL

Операторы DDL

- Схема базы данных состоит из операторов DDL
- ОПЕРАТОР тип-объекта-БД имя-объекта-БД уточняющие-предложения
- CREATE
- DROP
- ALTER

Создание, изменение и уничтожение таблиц

```
demo=# DROP TABLE IF EXISTS fixed_precision;
NOTICE:  table "fixed_precision" does not exist, skipping
DROP TABLE

demo=# CREATE TABLE fixed_precision (
demo(#     amount  numeric(7,2),
demo(#     percent numeric(7,2));
CREATE TABLE

demo=# ALTER TABLE fixed_precision ADD COLUMN name text;
ALTER TABLE
```

Операторы манипулирования данными SQL

Операторы манипулирования

- SELECT . . . FROM . . . WHERE . . .
- INSERT INTO table-name VALUES . . .
- UPDATE table-name SET . . . WHERE . . .
- DELETE FROM table-name WHERE . . .

Любой результат — отношение

```
demo=# SELECT current_date AS today, current_time AS right_now;
   today     |     right_now
-------------+--------------------
 2018-09-01 | 15:08:49.259903+03
(1 row)
```

Десятичная арифметика с ограниченной точностью

```
demo=# INSERT INTO fixed_precision (amount) VALUES (2.13), (3.14);
INSERT 0 2
demo=# UPDATE fixed_precision SET percent = amount * 0.03;
UPDATE 2
demo=# SELECT * FROM fixed_precision;
 amount | percent | name
--------+---------+------
   2.13 |    0.06 |
   3.14 |    0.09 |
(2 rows)
demo=# SELECT sum(amount) * 0.03, sum(percent) FROM fixed_precision;
 ?column? | sum
----------+------
   0.1581 | 0.15
(1 row)
```

Формат оператора SELECT

`SELECT`	список скалярных выражений
`FROM`	список табличных выражений
`WHERE`	предикат (условие)
`GROUP BY`	скалярных выражений
`HAVING`	еще один предикат
`ORDER BY`	еще один список скалярных выражений

Почти грамматика

- Скалярное выражение:

  константа, атрибут,
  арифметическое выражение,
  скалярная функция, **(select…)** –
  возвращающий скалярное значение

- Табличное выражение:

  таблица, представление, операция
  над таблицами, **(select…)**

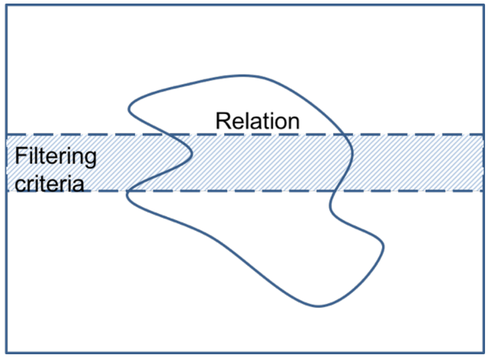
- Предикат:

  логическое выражение,
  составленное из простых условий
  над скалярными выражениями

Postgres PROFESSIONAL

Алгебраические операции: селекция

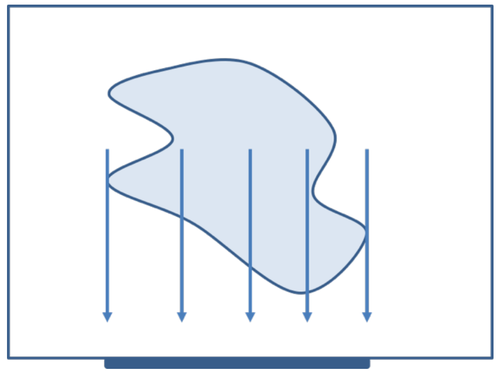
```
demo=# SELECT title, credits
FROM courses
WHERE credits > 8;
     title      | credits
----------------+---------
 Анализ данных |      10
(1 row)
```



Алгебраические операции: проекция

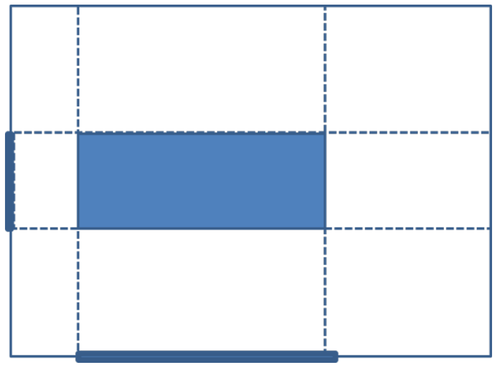
```
demo=# SELECT * FROM students;
 stud_id |  name   | start_year
---------+---------+------------
    1451 | Анна    |       2014
    1432 | Виктор  |       2014
    1556 | Нина    |       2015
(3 rows)
```

```
demo=# SELECT DISTINCT
start_year FROM students;
start_year
------------
2015
2014
(2 rows)
```



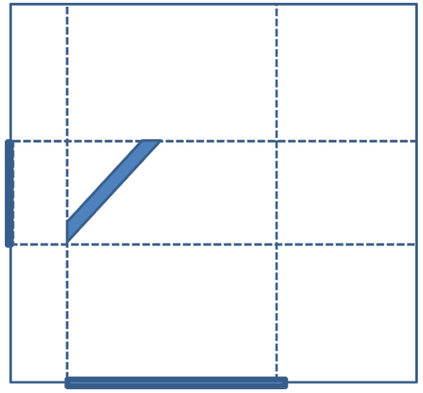
Произведение

```
demo=# SELECT * FROM courses, exams;
 course_no |     title      | credits | stud_id | course_no | exam_date  | grade
-----------+----------------+---------+---------+-----------+------------+-------
 CS301     | Базы данных    |       5 |    1451 | CS301     | 2016-05-25 |     5
 CS305     | Анализ данных  |      10 |    1451 | CS301     | 2016-05-25 |     5
 CS301     | Базы данных    |       5 |    1556 | CS301     | 2017-05-23 |     5
 CS305     | Анализ данных  |      10 |    1556 | CS301     | 2017-05-23 |     5
 CS301     | Базы данных    |       5 |    1451 | CS305     | 2016-05-25 |     5
 CS305     | Анализ данных  |      10 |    1451 | CS305     | 2016-05-25 |     5
 CS301     | Базы данных    |       5 |    1432 | CS305     | 2016-05-25 |     4
 CS305     | Анализ данных  |      10 |    1432 | CS305     | 2016-05-25 |     4
(8 rows)
```



Соединение по условию равенства значений атрибутов

```
demo=# SELECT * FROM courses, exams
WHERE courses.course_no = exams.course_no;
 course_no |     title      | credits | stud_id | course_no | exam_date  | grade
-----------+----------------+---------+---------+-----------+------------+-------
 CS301     | Базы данных    |       5 |    1451 | CS301     | 2016-05-25 |     5
 CS301     | Базы данных    |       5 |    1556 | CS301     | 2017-05-23 |     5
 CS305     | Анализ данных  |      10 |    1451 | CS305     | 2016-05-25 |     5
 CS305     | Анализ данных  |      10 |    1432 | CS305     | 2016-05-25 |     4
(4 rows)
```



Алгебраическая запись соединения

```
demo=# SELECT *
FROM courses
    JOIN exams ON courses.course_no = exams.course_no;
 course_no |     title      | credits | stud_id | course_no | exam_date  | grade
-----------+----------------+---------+---------+-----------+------------+-------
 CS301     | Базы данных    |       5 |    1451 | CS301     | 2016-05-25 |     5
 CS301     | Базы данных    |       5 |    1556 | CS301     | 2017-05-23 |     5
 CS305     | Анализ данных  |      10 |    1451 | CS305     | 2016-05-25 |     5
 CS305     | Анализ данных  |      10 |    1432 | CS305     | 2016-05-25 |     4
(4 rows)
```

Левое внешнее соединение

```
demo=# SELECT students.name, exams.grade
FROM students
    LEFT JOIN exams ON students.stud_id = exams.stud_id
                    AND course_no = 'CS305';
  name   | grade
--------+-------
 Анна    |     5
 Виктор  |     4
 Нина    |
(3 rows)
```

```
demo=# SELECT students.name, exams.grade
FROM students
    LEFT JOIN exams ON students.stud_id = exams.stud_id
WHERE course_no = 'CS305';
```

Другие варианты соединения

- Полное внешнее соединение
  - Объединение левого и правого внешних соединений
- Полусоединение
  - Проекция соединения на первый аргумент
- Антисоединение
  - Разность между первым аргументом и результатом полусоединения

Куда пропала внешняя схема?

- Операторы DDL сочетают описание концептуальной схемы и схемы хранения
- Роль внешней схемы выполняют списки выбираемых значений в операторе SELECT и в других операторах

Postgres PROFESSIONAL

Псевдонимы для табличных выражений

```
demo=# SELECT f.flight_no,
    f.departure_airport AS dep_airport,
    dep.city AS dep_city,
    f.arrival_airport AS arr_airport,
    arr.city AS arr_city
FROM flights f
    JOIN airports dep ON f.departure_airport = dep.airport_code
    JOIN airports arr ON f.arrival_airport = arr.airport_code
WHERE f.status = 'Departed'
  AND f.scheduled_arrival < bookings.now();
 flight_no | dep_airport |  dep_city   | arr_airport |    arr_city
-----------+-------------+-------------+-------------+----------------
 PG0496    | SVO         | Москва      | JOK         | Йошкар-Ола
 PG0574    | OVB         | Новосибирск | HMA         | Ханты-Мансийск
 PG0304    | SGC         | Сургут      | SVX         | Екатеринбург
(3 rows)
```

Подзапросы в списке SELECT

```
demo=# SELECT
    f.flight_no,
    f.departure_airport AS dep_airport,
    ( SELECT city
      FROM airports
      WHERE airport_code = f.departure_airport
    ) AS dep_city,
    f.arrival_airport AS arr_airport,
    ( SELECT city
      FROM airports
      WHERE airport_code = f.arrival_airport
    ) AS arr_city
FROM flights f
WHERE f.status = 'Departed'
  AND f.scheduled_arrival < bookings.now();
 flight_no | dep_airport |  dep_city   | arr_airport |    arr_city
-----------+-------------+-------------+-------------+----------------
 PG0496    | SVO         | Москва      | JOK         | Йошкар-Ола
 PG0574    | OVB         | Новосибирск | HMA         | Ханты-Мансийск
 PG0304    | SGC         | Сургут      | SVX         | Екатеринбург
(3 rows)
```

Осторожно, дублирование!

```
demo=# SELECT
    ( SELECT courses.course_no FROM courses
      WHERE courses.course_no  = exams.course_no
    ) AS course_no,
    ( SELECT courses.title FROM courses
      WHERE courses.course_no  = exams.course_no
    ) AS title,
    exams.exam_date,
    exams.stud_id,
    exams.grade
FROM exams;
```

Вложенные подзапросы в WHERE

```
demo=# SELECT stud_id, grade, course_no
FROM exams
WHERE (SELECT start_year
       FROM students
       WHERE students.stud_id = exams.stud_id) > 2014;
 stud_id | grade | course_no
---------+-------+-----------
    1556 |     5 | CS301
(1 row)
```

IN, NOT IN

```
demo=# SELECT name, start_year
FROM students
WHERE stud_id IN (SELECT stud_id FROM exams WHERE course_no = 'CS305');
  name   | start_year
--------+------------
 Виктор |       2014
 Анна   |       2014
(2 rows)

demo=# SELECT name, start_year
FROM students
WHERE stud_id NOT IN (SELECT stud_id FROM exams WHERE grade < 5);
 name | start_year
------+------------
 Анна |       2014
 Нина |       2015
(2 rows)
```

EXISTS

```
demo=# SELECT name, start_year
FROM students
WHERE NOT EXISTS (
    SELECT stud_id
    FROM exams
    WHERE grade < 5 AND exams.stud_id = students.stud_id
);
 name | start_year
------+------------
 Анна |       2014
 Нина |       2015
(2 rows)
```

Алгебра или исчисление?

- Табличные выражения можно рассматривать как вариант алгебры
- Реляционная часть оператора SELECT выглядит как запрос в исчислении
- Возможны разные стили использования или их комбинация
- Возможны эквивалентные преобразования из одной формы в другую
- «Не так уж важно, есть жизнь на Марсе или нет»

Postgres PROFESSIONAL

Упорядочивание результата

```
demo=# SELECT *
FROM exams
ORDER BY grade, stud_id, course_no;
 stud_id | course_no | exam_date  | grade
---------+-----------+------------+-------
    1432 | CS305     | 2016-05-25 |     4
    1451 | CS301     | 2016-05-25 |     5
    1451 | CS305     | 2016-05-25 |     5
    1556 | CS301     | 2017-05-23 |     5
(4 rows)
```

Агрегирование (группировка)

```
demo=# SELECT count(*), count(DISTINCT stud_id), avg(grade)
FROM exams;
 count | count |        avg
-------+-------+--------------------
     4 |     3 | 4.7500000000000000
(1 row)

demo=# SELECT courses.title, count(*), avg(exams.grade)
FROM exams
    JOIN courses ON exams.course_no = courses.course_no
GROUP BY courses.title;
     title      | count |        avg
----------------+-------+--------------------
 Базы данных    |     2 | 5.0000000000000000
 Анализ данных  |     2 | 4.5000000000000000
(2 rows)
```

Нелогичность SQL

```
demo=# SELECT count(*) FROM flights WHERE scheduled_departure > ‘2022-Sep-19';
 count
-------
     0
(1 row)

demo=# SELECT count(*) FROM flights WHERE scheduled_departure > '2022-Sep-19'
demo-# GROUP BY departure_airport;
 count
-------
(0 rows)
```

Фильтрация по результату агрегирования

```
demo=# SELECT count(*), departure_airport FROM flights
demo-# GROUP BY departure_airport HAVING count(*) > 6000
demo-# ORDER BY 1 DESC;
 count | departure_airport
-------+-------------------
 20875 | DME
 19347 | SVO
 12332 | LED
 11145 | VKO
  6846 | OVB
(5 rows)
```

Теоретико-множественное объединение

```
demo=# SELECT * FROM aircrafts WHERE range > 4500
UNION
SELECT * FROM aircrafts WHERE range < 7500;
 aircraft_code |        model        | range
---------------+---------------------+-------
 CR2           | Бомбардье CRJ-200   |  2700
 CN1           | Сессна 208 Караван  |  1200
 733           | Боинг 737-300       |  4200
 319           | Аэробус A319-100    |  6700
 321           | Аэробус A321-200    |  5600
 320           | Аэробус A320-200    |  5700
 773           | Боинг 777-300       | 11100
 763           | Боинг 767-300       |  7900
 SU9           | Сухой Суперджет-100 |  3000
(9 rows)
```

```
demo=# SELECT * FROM aircrafts WHERE range > 4500
UNION ALL
SELECT * FROM aircrafts WHERE range < 7500;
 aircraft_code |        model        | range
---------------+---------------------+-------
 773           | Боинг 777-300       | 11100
 763           | Боинг 767-300       |  7900
 320           | Аэробус A320-200    |  5700
 321           | Аэробус A321-200    |  5600
 319           | Аэробус A319-100    |  6700
 SU9           | Сухой Суперджет-100 |  3000
 320           | Аэробус A320-200    |  5700
 321           | Аэробус A321-200    |  5600
 319           | Аэробус A319-100    |  6700
 733           | Боинг 737-300       |  4200
 CN1           | Сессна 208 Караван  |  1200
 CR2           | Бомбардье CRJ-200   |  2700
(12 rows)
```

Пересечение и разность

```
demo=# SELECT * FROM aircrafts WHERE range > 4500
INTERSECT
SELECT * FROM aircrafts WHERE range < 7500;
 aircraft_code |      model       | range
---------------+------------------+-------
 321           | Аэробус A321-200 |  5600
 320           | Аэробус A320-200 |  5700
 319           | Аэробус A319-100 |  6700
(3 rows)
```

```
demo=# SELECT * FROM aircrafts WHERE range > 4500
EXCEPT
SELECT * FROM aircrafts WHERE range < 7500;
 aircraft_code |     model      | range
---------------+----------------+-------
 773           | Боинг 777-300 | 11100
 763           | Боинг 767-300 |  7900
(2 rows)
```

Ограничения целостности

Ограничения целостности определяются

- При создании таблицы для отдельных атрибутов или в предложении constraints

```
demo=# CREATE TABLE courses (
    course_no varchar(30),
    title     text NOT NULL,
    credits   integer,
    CONSTRAINT course_pkey PRIMARY KEY (course_no)
);
CREATE TABLE
```

- Оператором ALTER TABLE ADD CONSTRAINT . . .

Типы ограничений целостности

- NOT NULL
- CHECK (условие)
- UNIQUE
- PRIMARY KEY (список атрибутов)
- FOREIGN KEY (список атрибутов) REFERENCES таблица (список атрибутов)

Модификация базы данных

Операторы

- Операции манипулирования данными (DML)
  - INSERT
  - UPDATE
  - DELETE
  - COPY (PostgreSQL)
- Модификация схемы (DDL)
  - CREATE
  - DROP
  - ALTER
  - TRUNCATE

Postgres PROFESSIONAL

Вставка новых строк

- Вставка значений:

```
INSERT INTO t_person (список колонок)
VALUES (список скалярных значений)
...
```

- Множественное добавление:

```
INSERT INTO t_person (список колонок)
(SELECT ...
 FROM ...
 WHERE ...)
```

Удаление строк

```
DELETE t_table_name WHERE предикат
```

- И снова предикат может быть сложным
- Каскадные удаления — можно предусмотреть в описании данных

  ON DELETE CASCADE

Изменение существующих строк

```
UPDATE t_tablename SET
   колонка = значение,
   ...
WHERE предикат
```

- Как и в других операторах, выражения и предикат могут быть сложными подзапросами

Многообразие выразительных средств SQL

Встроенные функции и операторы

- Преобразование типов
- Элементарные функции (тригонометрические, экспонента, логарифм, …)
- Гиперболические функции
- Работа со строками
- Даты, время, интервалы
- Обработка встроенных сложных типов (XML, JSON, …)
- (PostgreSQL) Функции, процедуры и операторы, определяемые пользователем

Postgres PROFESSIONAL

Условные выражения

```
(CASE
 WHEN предикат1 THEN значение1
 WHEN предикат2 THEN значение2
 WHEN предикат3 THEN значение3
 ...
 ELSE значение
 END)
```

Псевдонимы для табличных выражений

```
CREATE TABLE par_child(
  parent text,
  child text,
  CONSTRAINT PRIMARY KEY (parent, child));

SELECT
  g1.parent AS grandparent,
  g2.child  AS grandchild
FROM par_child g1
  JOIN par_child g2 ON g1.child = g2.parent;
```

Представления (Views)

```
CREATE VIEW airports
AS
SELECT ml.airport_code,
    ml.airport_name ->> lang()
          AS airport_name,
    ml.city ->> lang() AS city,
    ml.coordinates,
    ml.timezone
FROM airports_data ml;
```

- логические объекты схемы базы данных
- Могут использоваться в запросах как таблицы
- Определение вычисляемых объектов
- Выделение часто встречающихся подзапросов
- Не вычисляются до выполнения запроса

Именованные подзапросы: общие табличные выражения

```
WITH sub_expr AS
  ( подзапрос или оператор )
SELECT ...
FROM ..., sub_expr
...
WHERE ... IN (SELECT ... FROM sub_expr ...)
...
```

Postgres PROFESSIONAL

Вывод результатов модификации

```
INSERT INTO ...
  SELECT ...
  RETURNING список скалярных выражений;

UPDATE ... SET ...
  WHERE ...
  RETURNING список скалярных выражений;

DELETE FROM ...
  WHERE ...
  RETURNING список скалярных выражений;
```

https://postgrespro.ru/education/university/dbtech