# МЕТОДЫ ВНЕДРЕНИЯ ФРАГМЕНТНОГО ПАРАЛЛЕЛИЗМА В ПОСЛЕДОВАТЕЛЬНУЮ СУБД С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата физико-математических наук

**К.С. Пан**

Научный руководитель:
ЦЫМБЛЕР Михаил Леонидович,
кандидат физ.-мат. наук, доцент

# Актуальность

- Сверхбольшие данные
- Сверхбольшие реляционные базы данных
- Параллельные СУБД, фрагментный параллелизм
- Коммерческие параллельные СУБД — высокая стоимость или специфические аппаратно-программные платформы
- Свободные СУБД — не реализуют фрагментный параллелизм

# Цель диссертационной работы

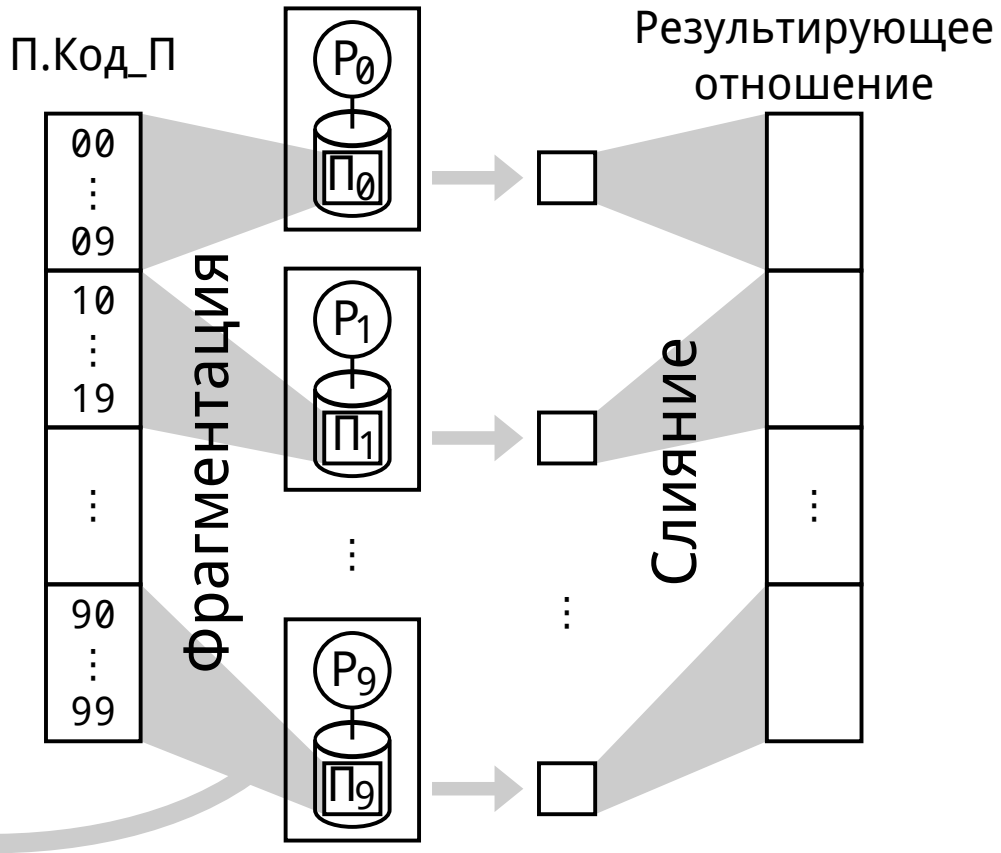
**Цель работы:** разработка методов, архитектурных подходов и алгоритмов, обеспечивающих внедрение фрагментного параллелизма в имеющиеся последовательные СУБД, свободно распространяемые на уровне исходных кодов, а также проверка разработанных методов путем их применения для решения задач аналитической и оперативной обработки сверхбольших баз данных.

# Основные задачи

1. Разработать методы внедрения фрагментного параллелизма в последовательную СУБД с открытым исходным кодом.
2. На основе разработанных методов предложить архитектурные подходы и алгоритмы, реализующие фрагментный параллелизм в рамках последовательной СУБД с открытым исходным кодом. Выполнить параллелизацию одной из свободных последовательных СУБД.
3. Исследовать эффективность предложенных подходов применительно к решению задач классов OLAP и OLTP, связанных с обработкой сверхбольших баз данных.

# Фрагментный параллелизм

$$\Pi_i = \{t | t \in \Pi, \phi(t) = i\}$$
$$i = 0, \ldots, 9$$



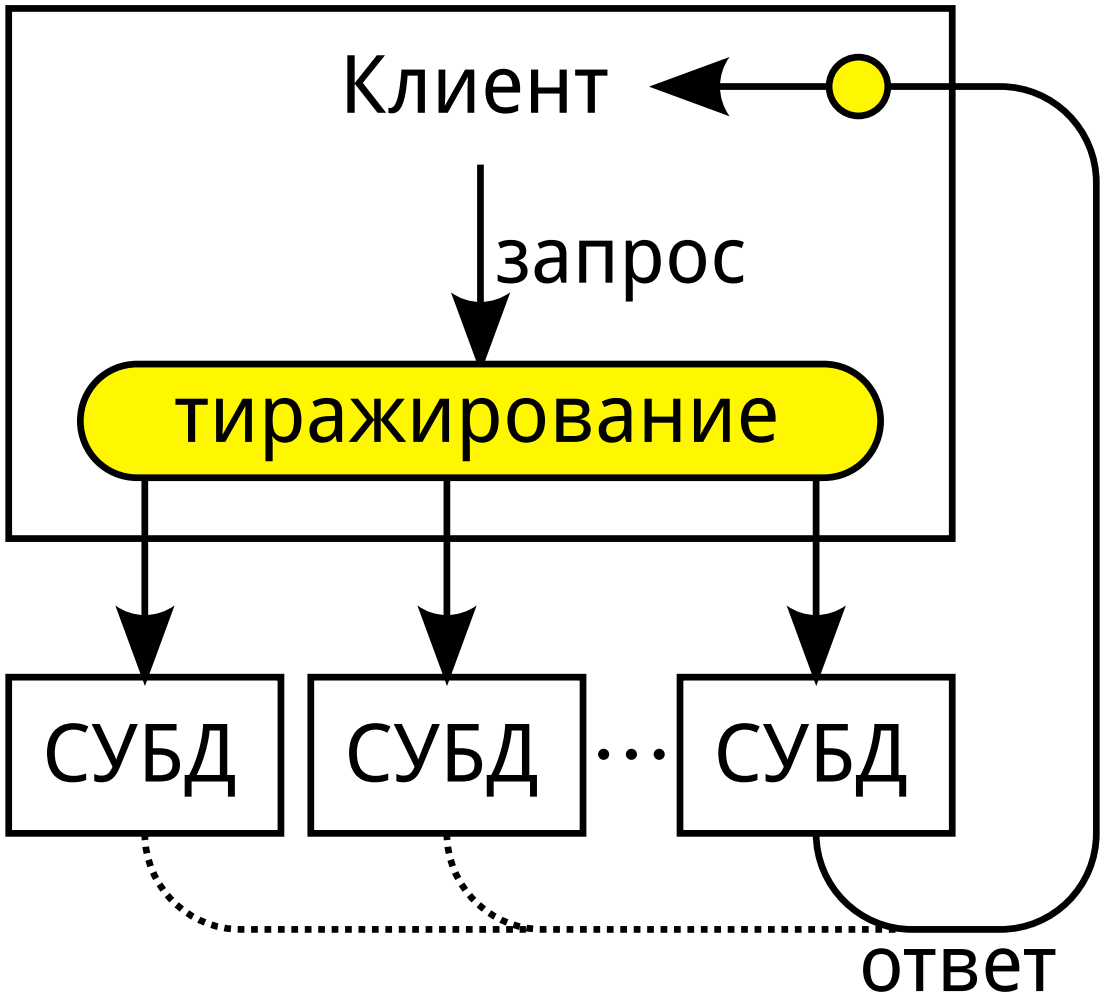

**Функция фрагментации**

$$\phi(t) = (t.\text{Код\_П} \text{ div } 10) \bmod 10$$

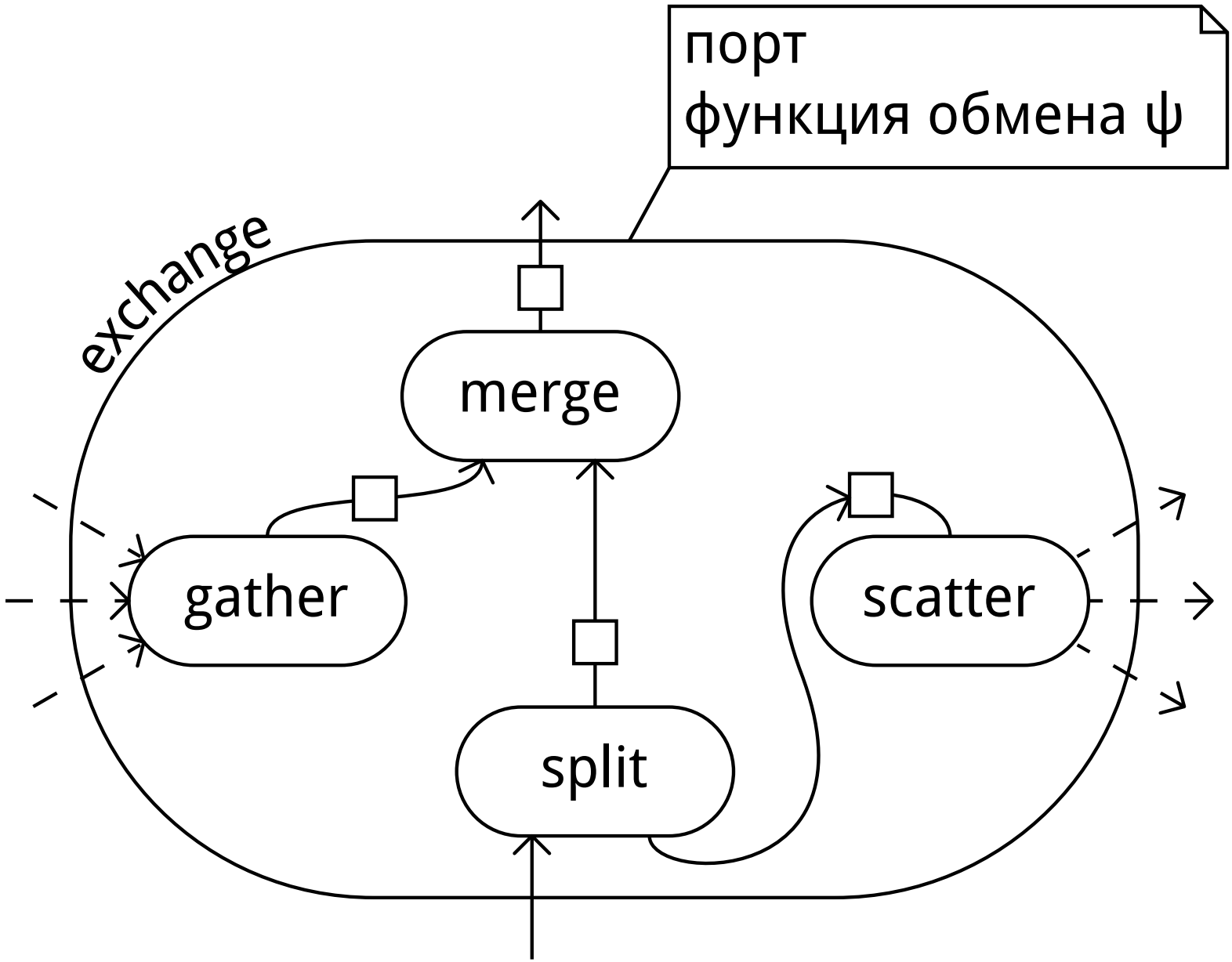
# Методы внедрения фрагментного параллелизма

1. Тиражирование запроса
2. Использование оператора обмена
3. Построение параллельного плана запроса
4. Обработка запросов на изменение данных
5. Хранение метаданных о фрагментации
6. Портирование приложений последовательной СУБД
7. Модификация исходных текстов последовательной СУБД

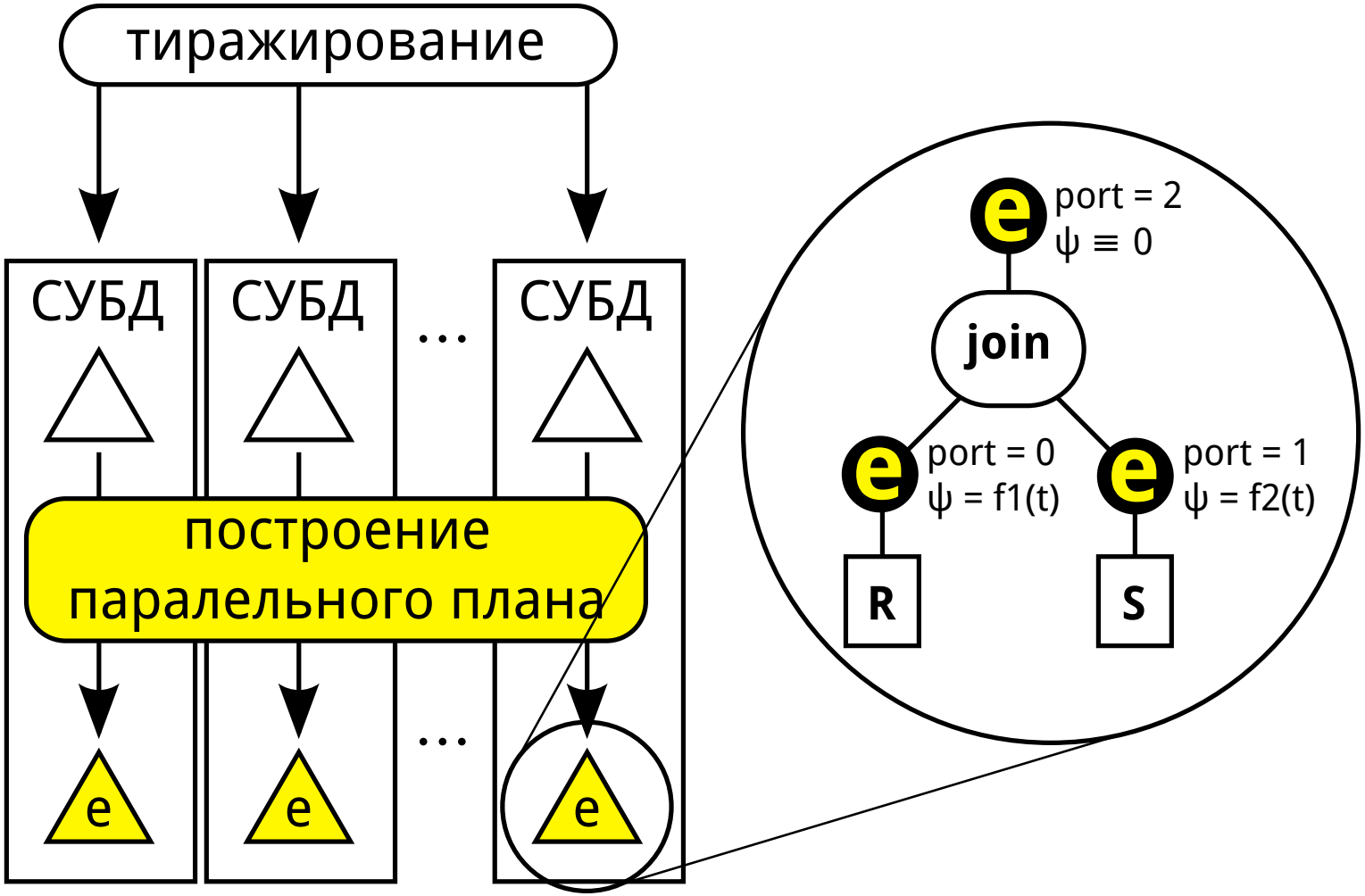
# Тиражирование запроса



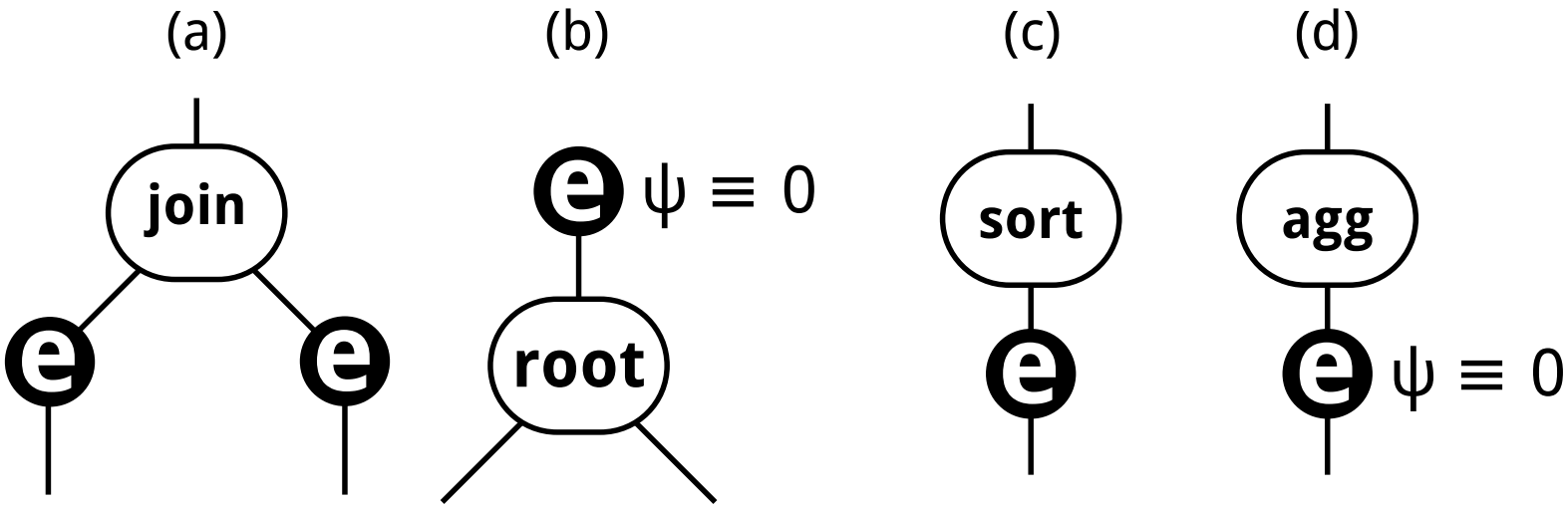
# Оператор обмена (exchange)



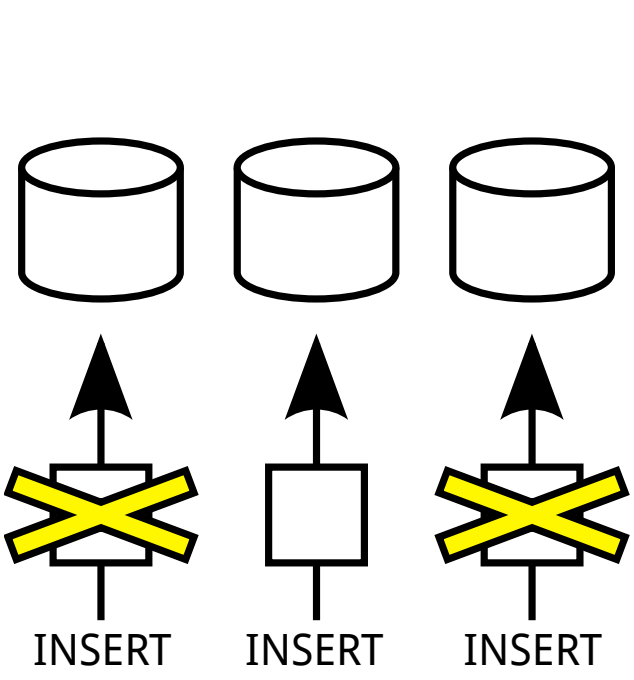
# Построение параллельного плана запроса

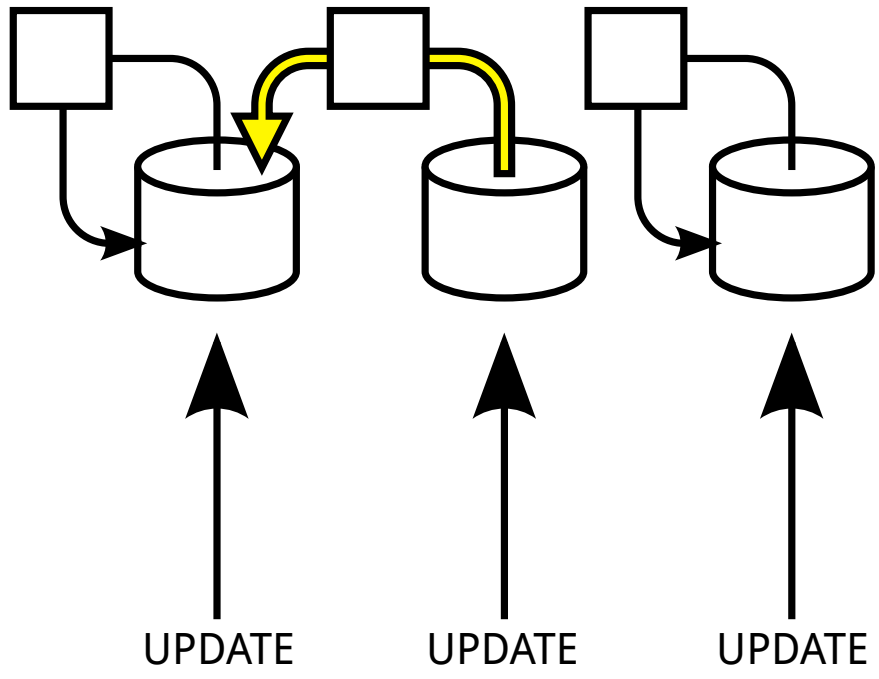


# Построение параллельного плана запроса



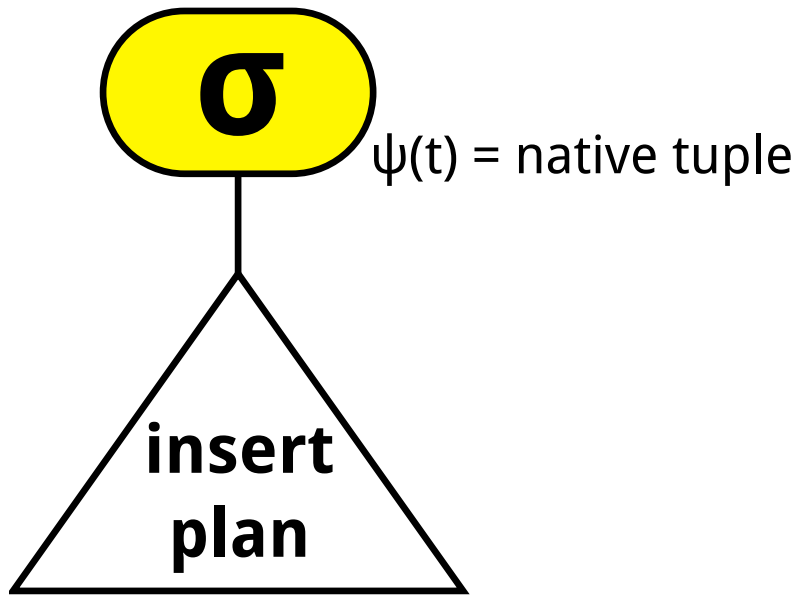
# Обработка запросов на изменение данных



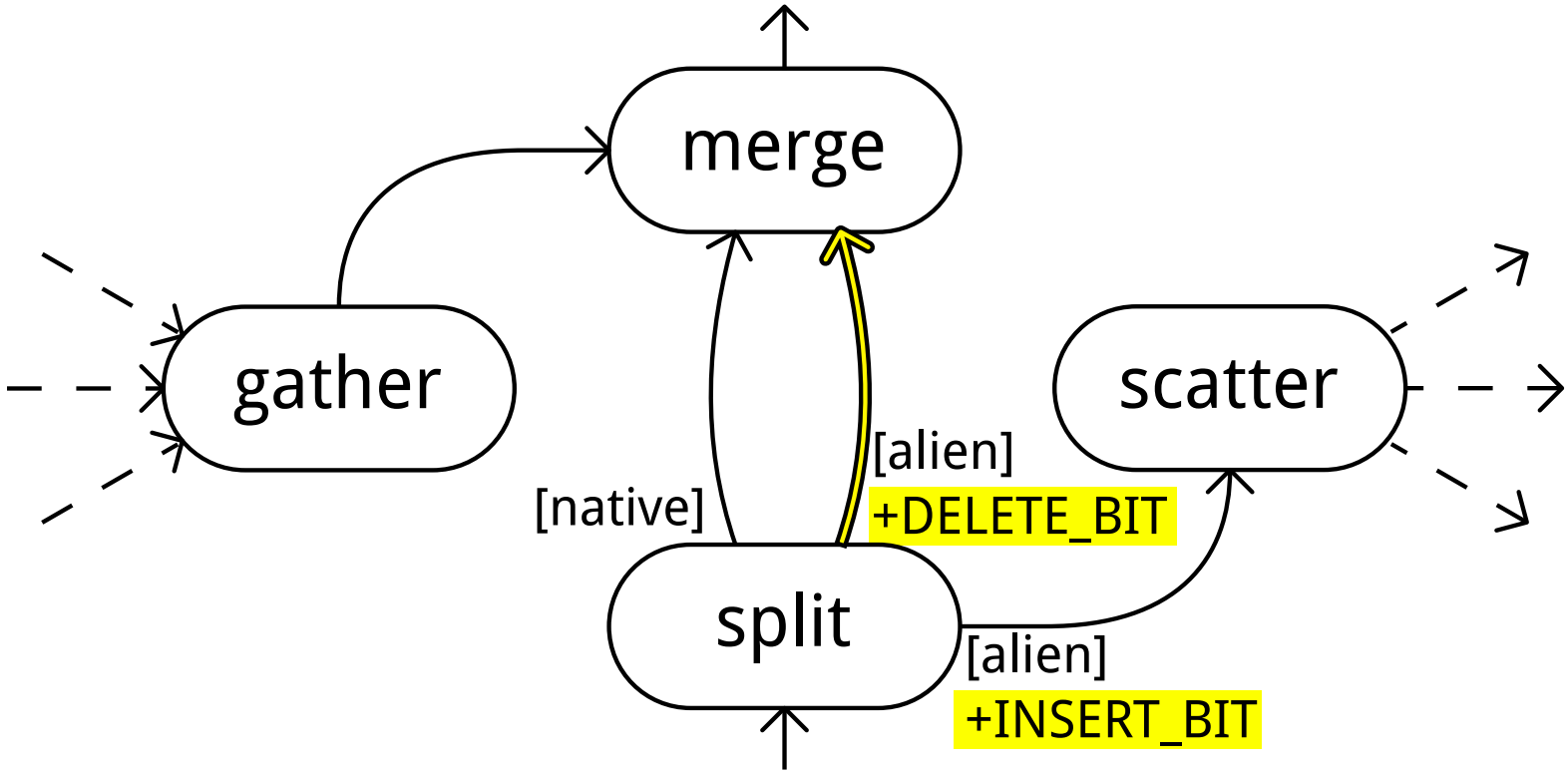



# Обработка запросов INSERT

Дополнительная операция выборки в корне плана запроса обеспечивает отсечение «чужих» кортежей.



# Обработка запросов UPDATE




Модифицированный алгоритм exchange позволяет перемещать кортежи, которые в результате обновления стали «чужими» (если $\varphi(t') \neq \varphi(t)$, то на узле $\varphi(t)$ кортеж $t$ удаляется, а на узле $\varphi(t')$ вставляется $t'$).

# Хранение метаданных о фрагментации

Добавление в язык баз данных средств определения функции фрагментации для таблиц.

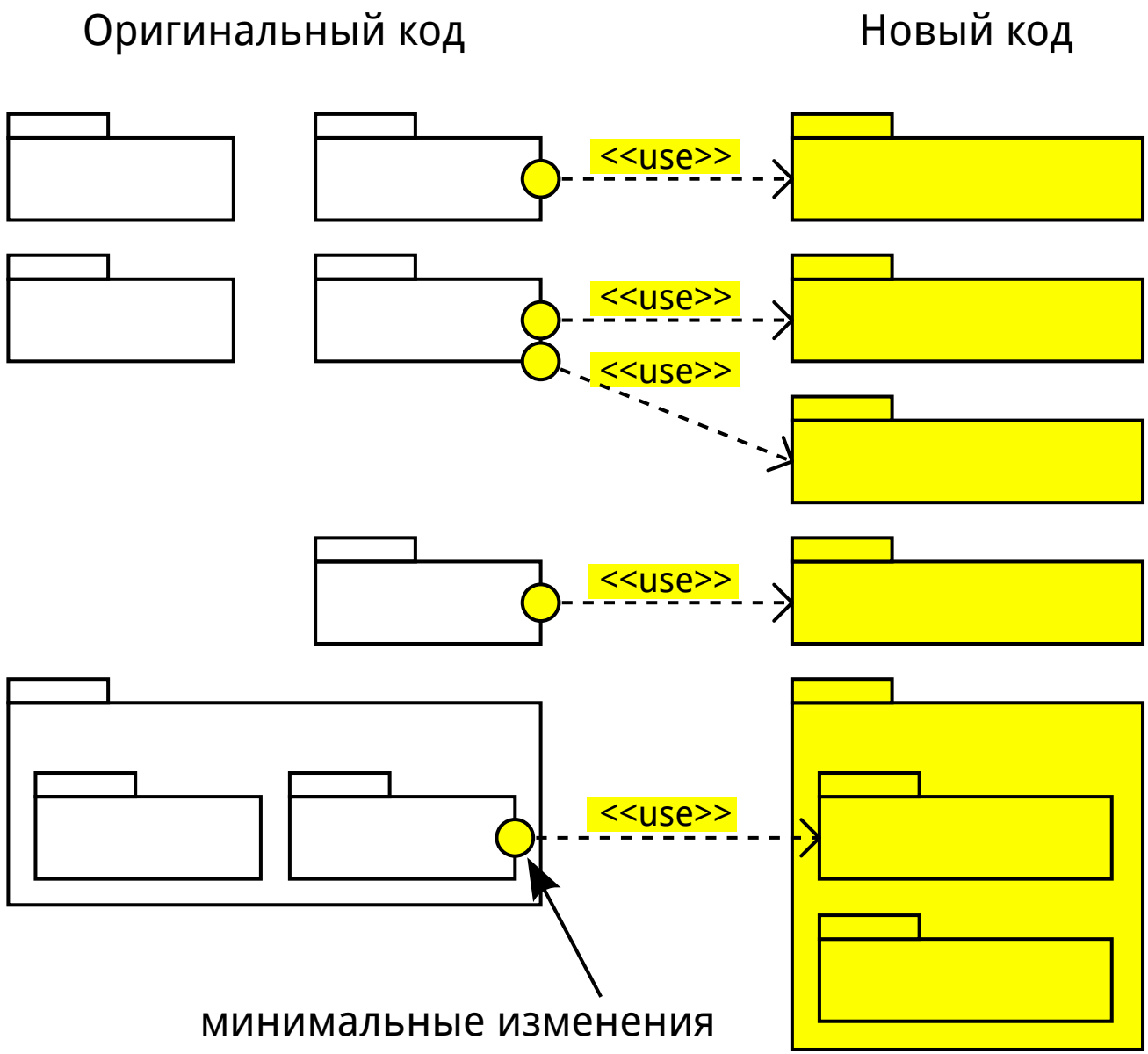
```
create table Person (
    id int,
    name char(30),
    gender char(1),
    birth date,
) with (функция фрагментации);
```

# Портирование приложений последовательной СУБД

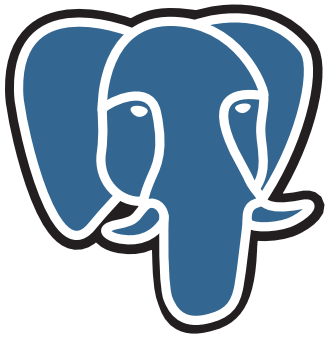

**СУБД**


**Параллельная СУБД**

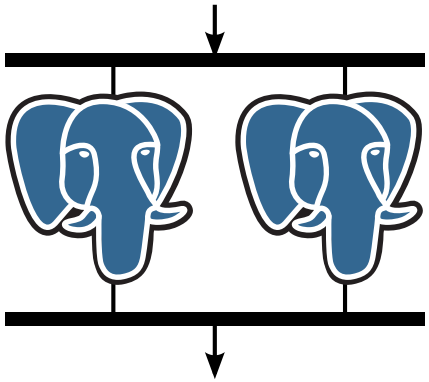
# Модификация исходных текстов последовательной СУБД



# Реализация методов



PostgreSQL

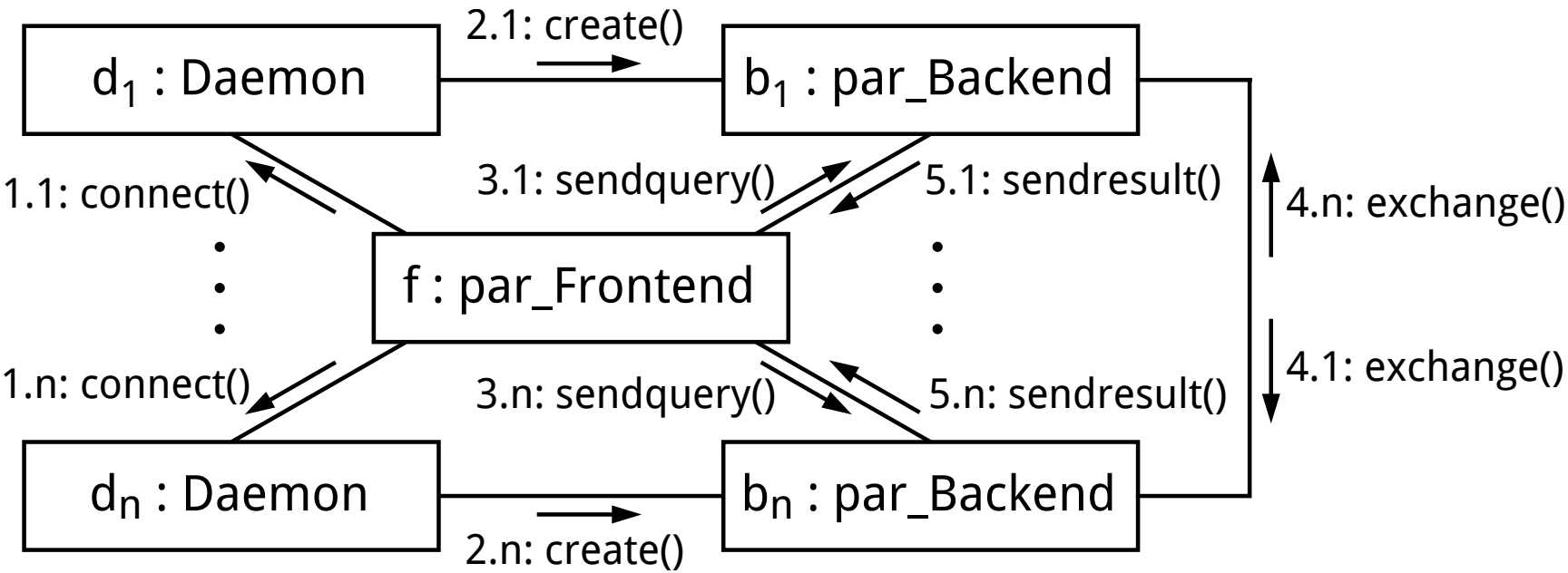


PargreSQL

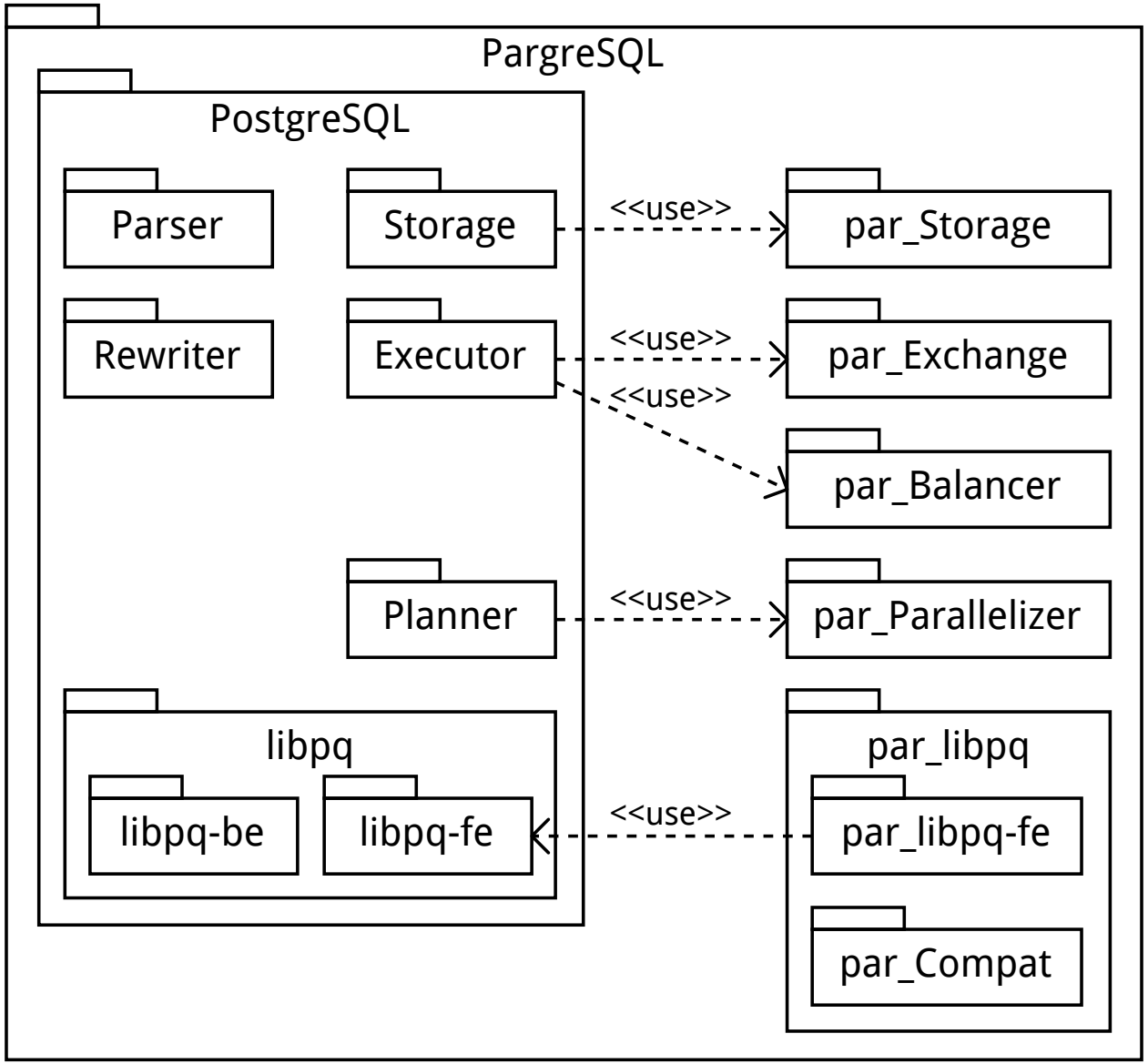
# PostgreSQL


- Свободная объектно-реляционная СУБД с открытым исходным кодом для различных ОС
- Разрабатывается группой энтузиастов с 1995 г. (ответвление проекта POSTGRES М. Стоунбрейкера)
- Основные особенности
  - Поддержка SQL:2011 и ACID-транзакций
  - Хранимые процедуры SQL, pgSQL, Perl, Python, C и др.
  - Макс размеры: таблица – 32 Тб, поле – 1 Гб
  - Качественная документация исходного кода
- Надежная альтернатива коммерческим СУБД
- Широкое применение
  - Корпорации: Apple, Sun, Cisco, Fujitsu, Red Hat, ...
  - Госструктуры: U.S. State Department, United Nations Industrial Development Organization, ...

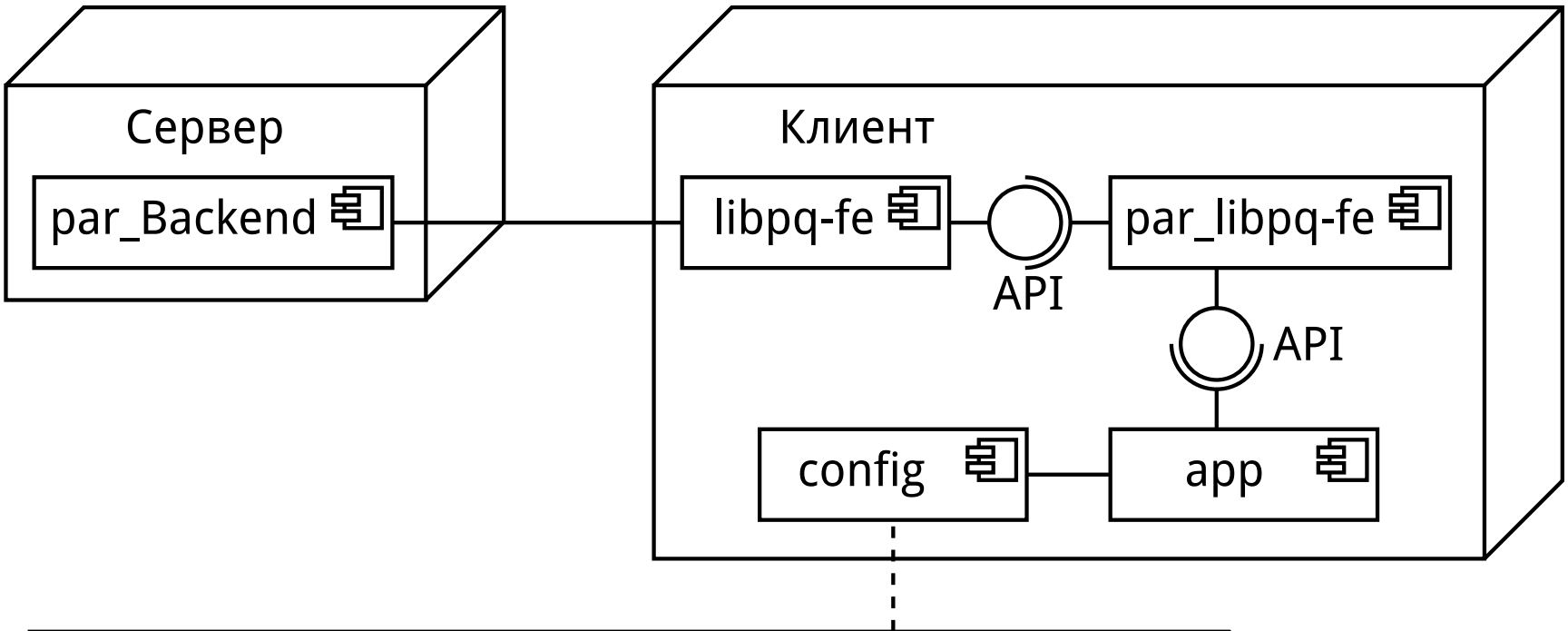
# Клиент-серверное взаимодействие



# Архитектура PargreSQL



# Тиражирование

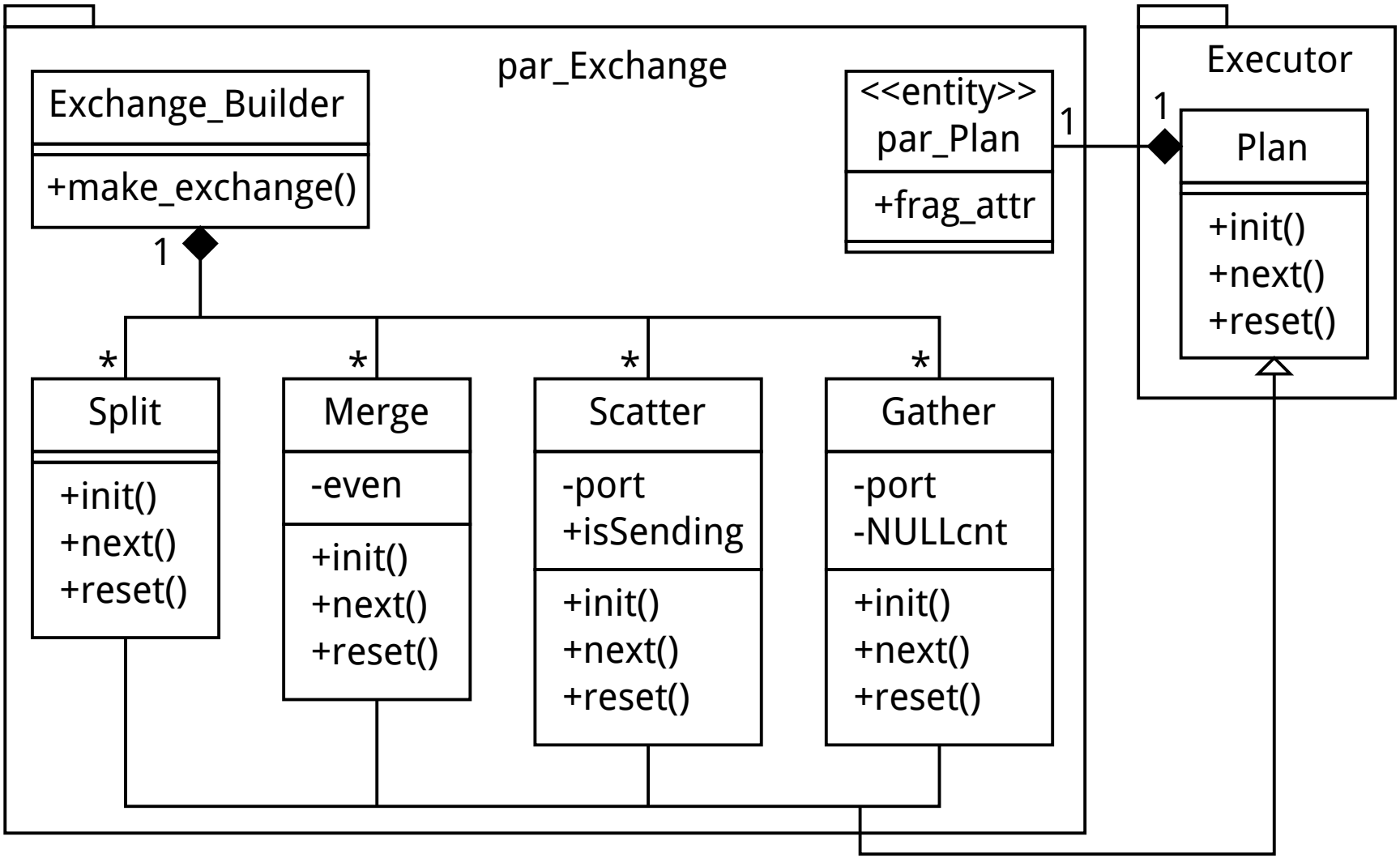



```
# Список строк подключения к экземплярам СУБД
# (по одной строке на каждый узел кластера)
dbname=postgres hostaddr=10.4.5.204 port=5432
dbname=postgres hostaddr=10.4.5.205 port=5432
```

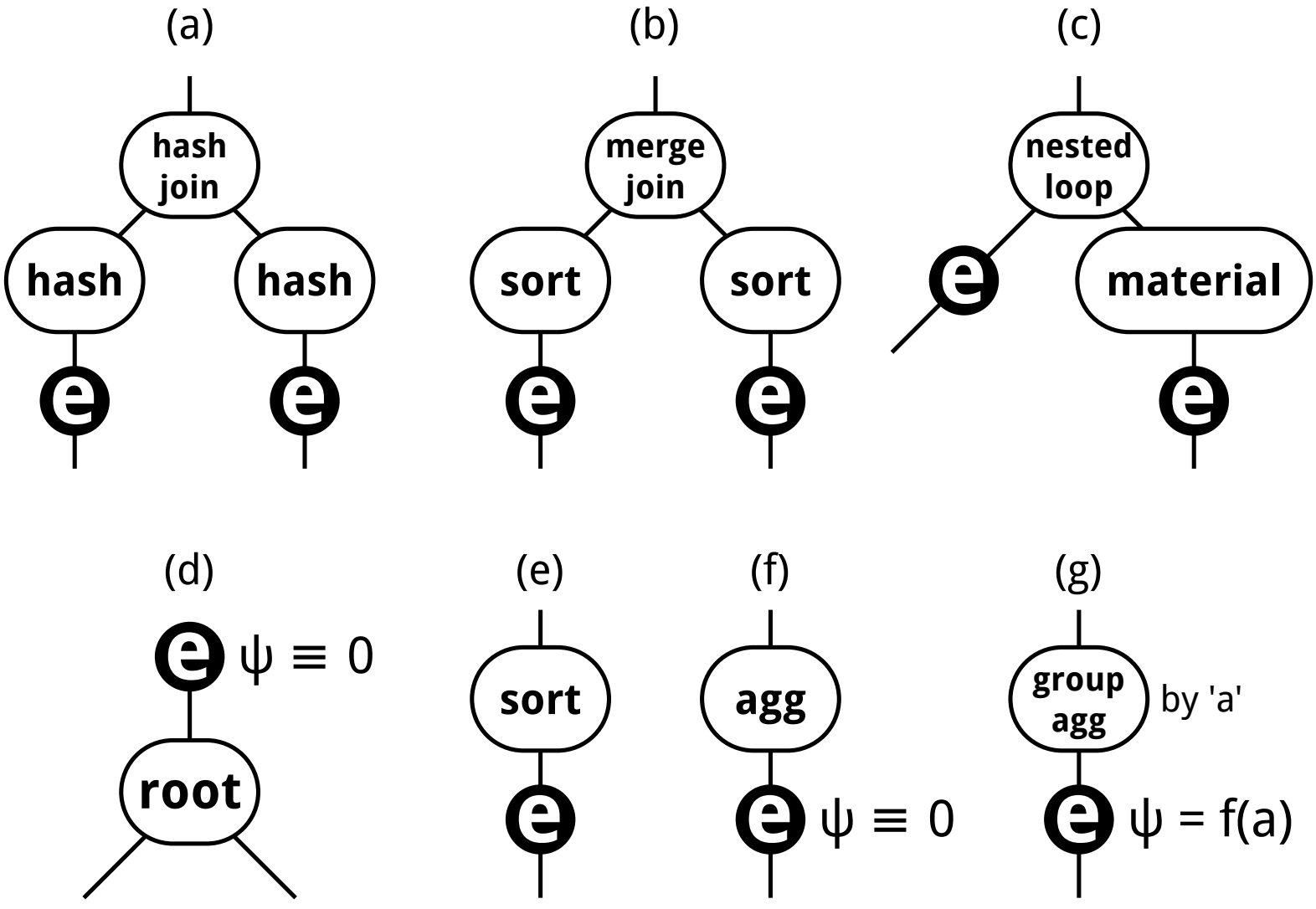
# Портирование

```
//        оригинальная    параллельная
//        библиотека      библиотека
#define PGconn            par_PGconn
#define PQconnectdb(X)    par_PQconnectdb()
#define PQfinish(X)       par_PQfinish(X)
#define PQstatus(X)       par_PQstatus(X)
#define PQexec(X,Y)       par_PQexec(X,Y)
```

# Реализация exchange



# Построение параллельного плана запроса



# Запросы на определение функции фрагментации

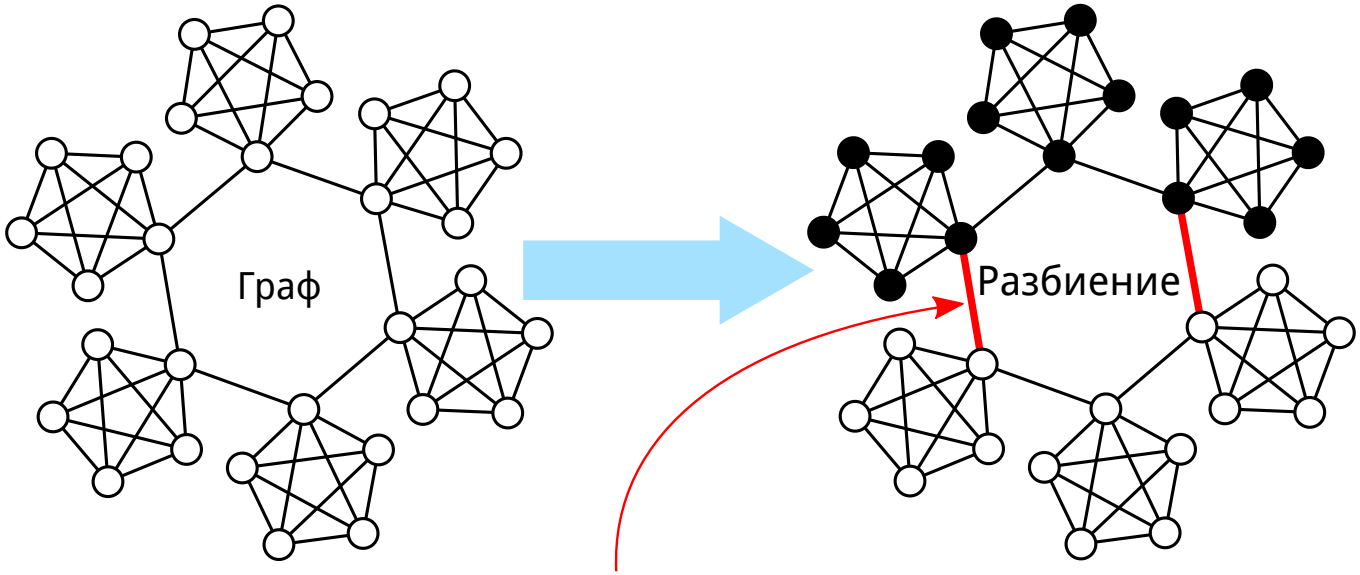
```
create table Person (
    id int,
    name varchar(30),
    gender char(1),
    birth date
) with (fragattr = id);
```

Определит функцию фрагментации
для таблицы Person как $\varphi(t) = t.id \mod n$,
где $n$ — количество вычислительных узлов.

# Проверка методов

- Задачи класса OLAP
  - разбиение сверхбольших графов
  - ускорение и расширяемость СУБД
- Задачи класса OLTP
  - тесты консорциума TPC (Transaction Processing Council)

# Разбиение сверхбольших графов

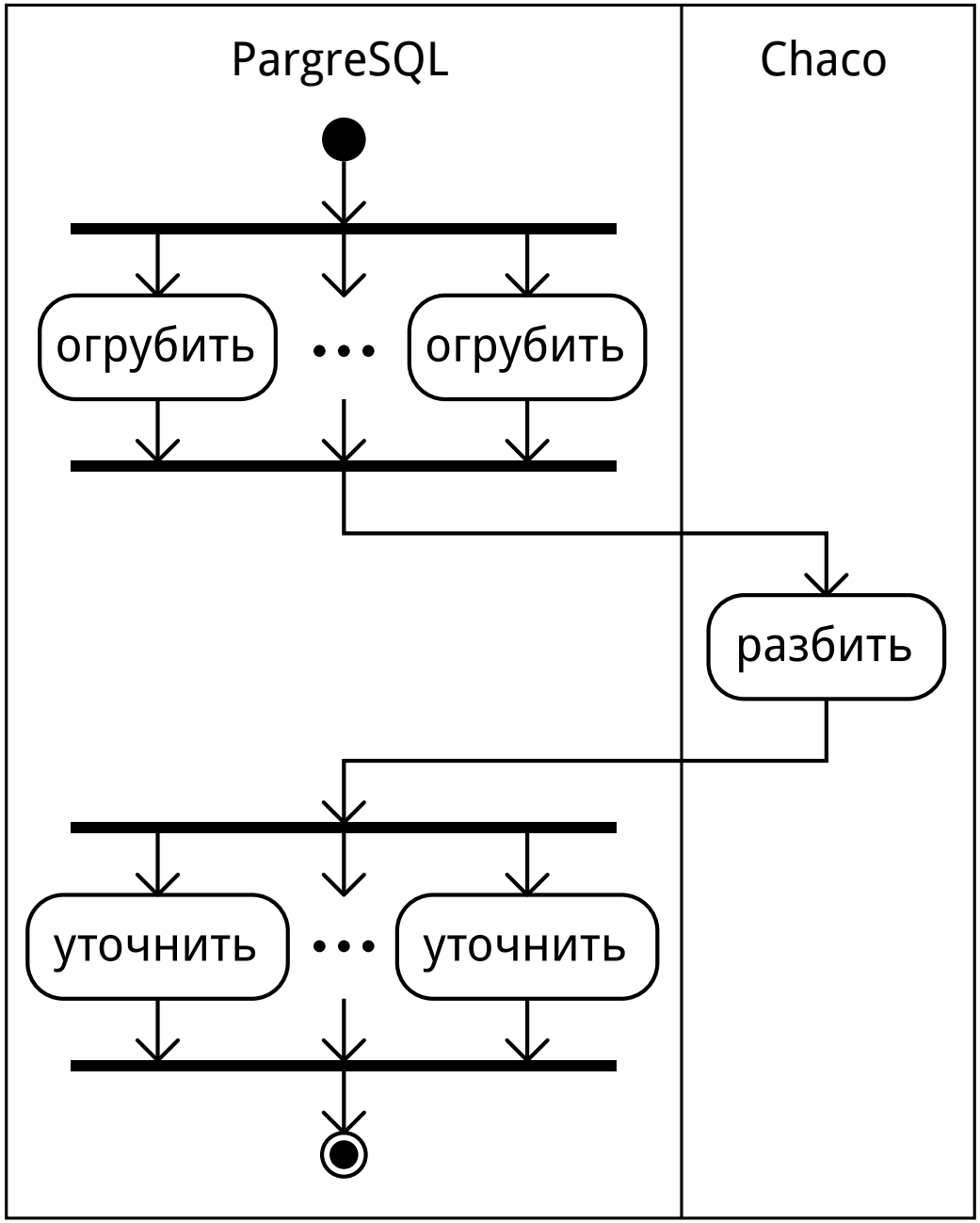


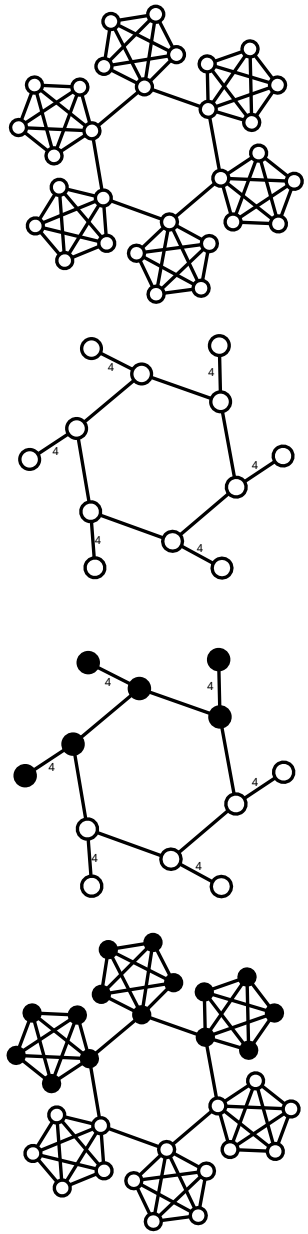

$$G = (N, E) \qquad \bigcup_{i=1}^{p} N_i = N$$

$$\forall i \neq j \quad N_i \cap N_j = \varnothing$$

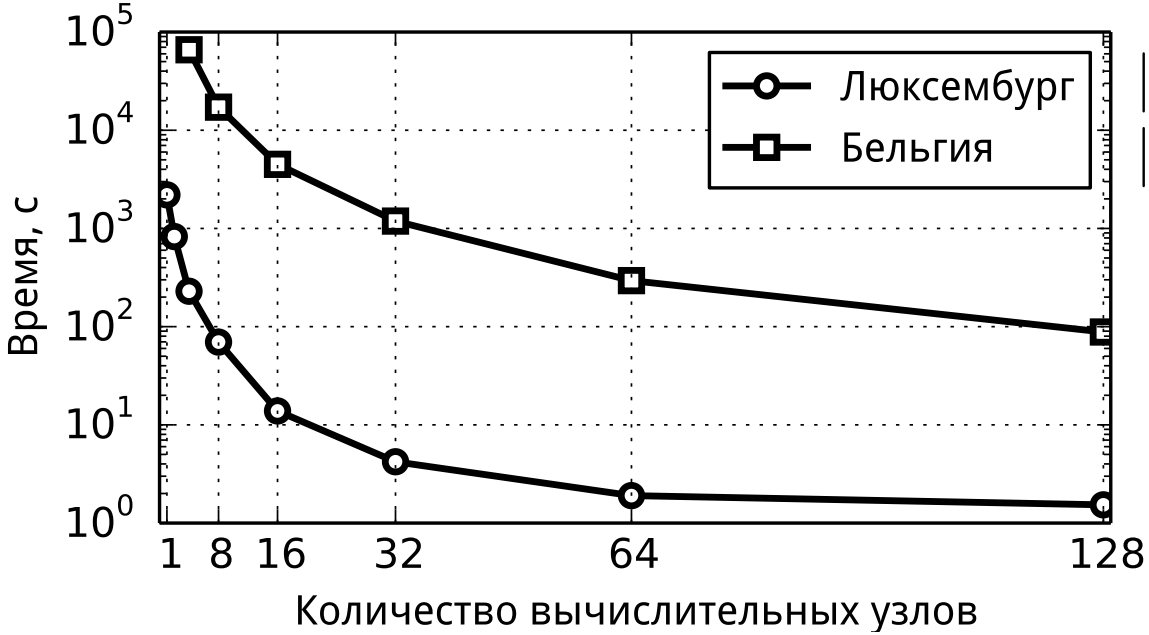
$$\forall i \quad |N_i| \approx \frac{|N|}{p}, \qquad \sum_{u,v:P(u)\neq P(v)} W(u,v) \to \min$$

# Разбиение сверхбольших графов

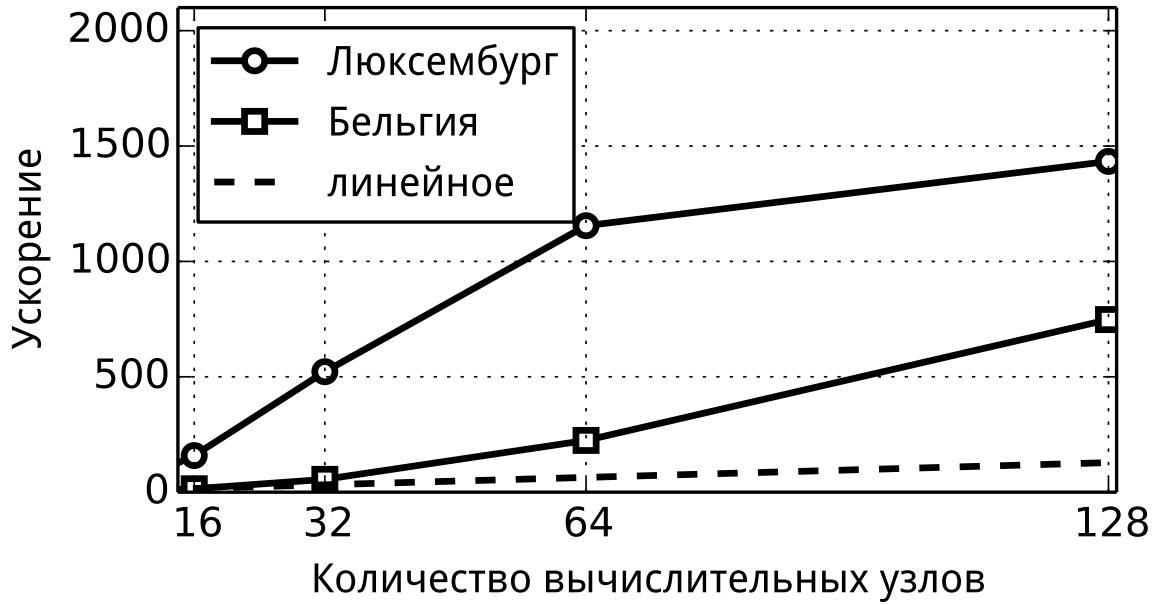


# Разбиение сверхбольших графов

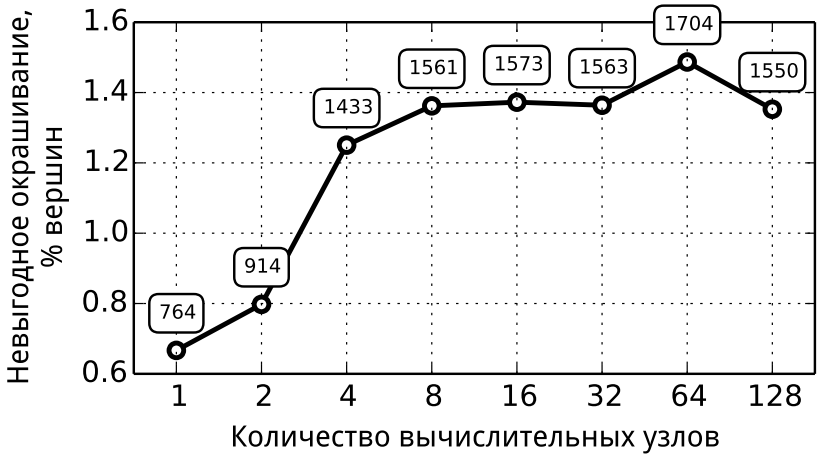



$|N| = 10^5$
$|N| = 10^6$

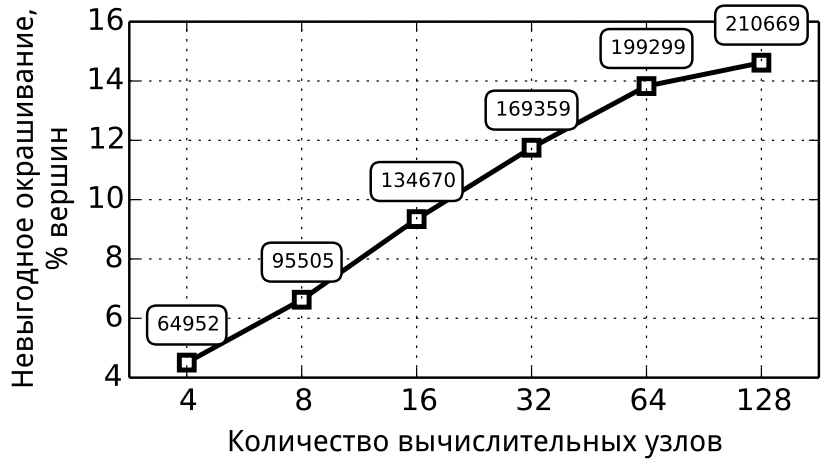


# Разбиение сверхбольших графов

Люксембург ($|N| = 10^5$)

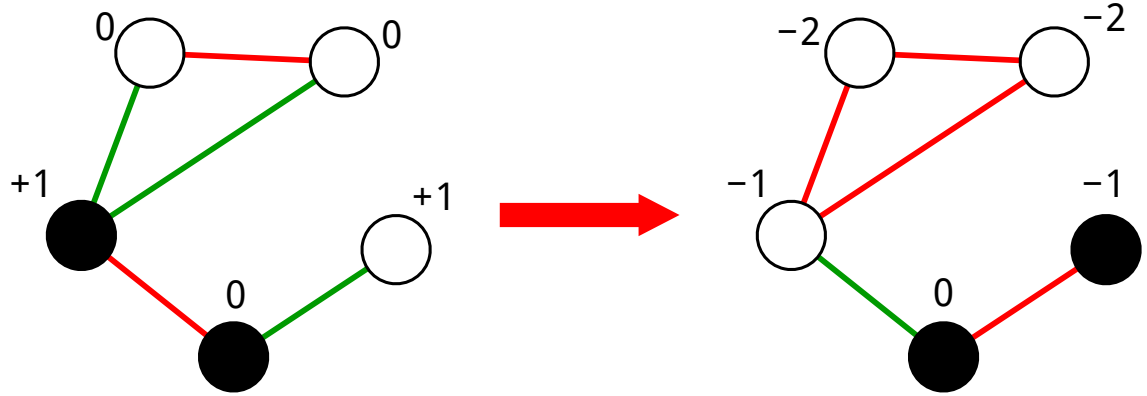



Бельгия ($|N| = 10^6$)



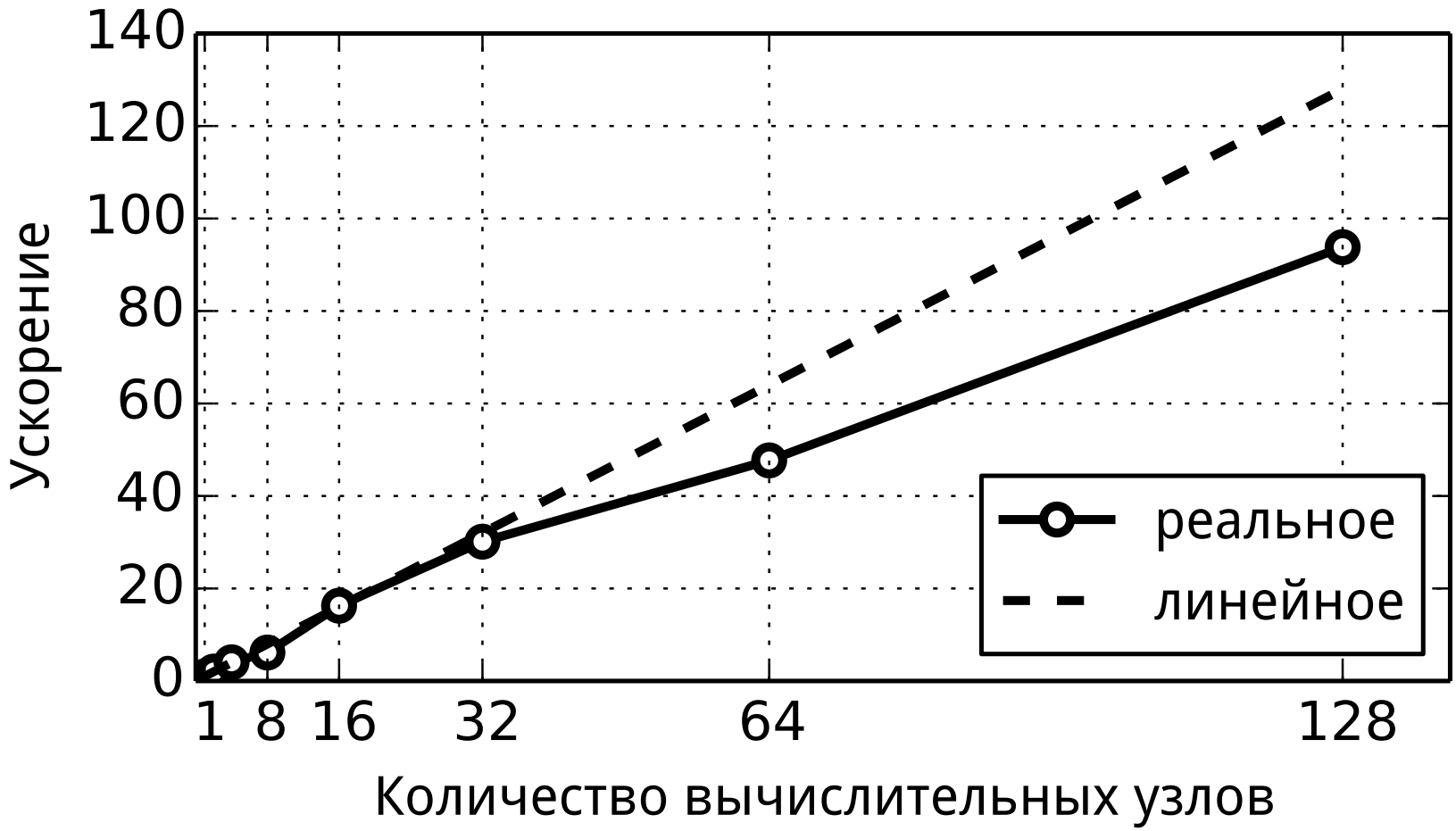

Случайное разбиение дает 30% невыгодно окрашенных вершин.

$$\text{gain}(v) = \sum_{v,u \in E, P(v) \neq P(u)} W(u,v) - \sum_{v,u \in E, P(v) = P(u)} W(u,v)$$



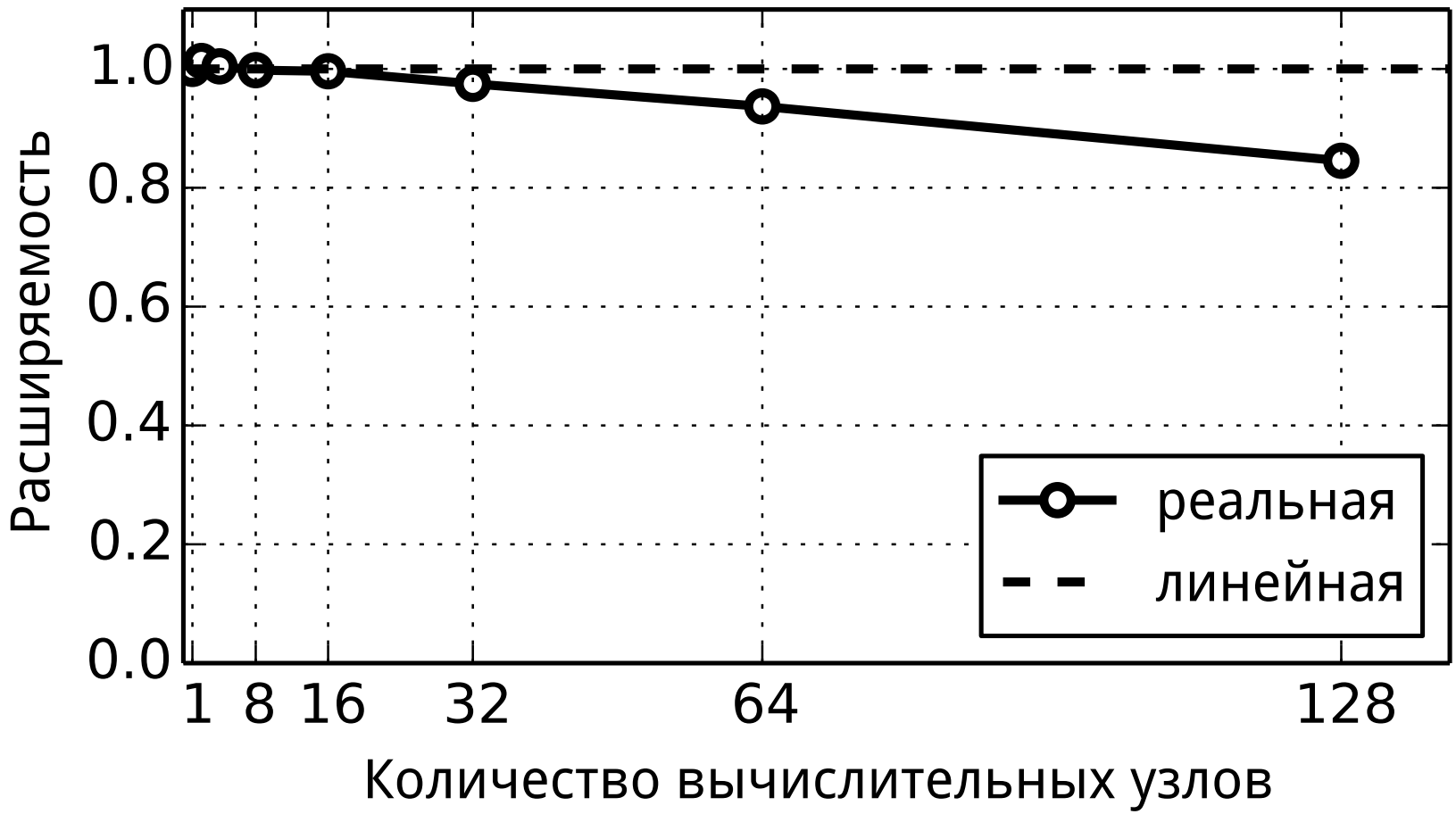
# Ускорение параллельной СУБД

$$R \bowtie S, \qquad |R| = 3 \cdot 10^8, \quad |S| = 7.5 \cdot 10^6$$



# Расширяемость параллельной СУБД

$$R \bowtie S, \qquad |R_i| = i \cdot 12 \cdot 10^6, \quad |S_i| = i \cdot 0.3 \cdot 10^6$$



# Производительность на тесте TPC-C (моделирование складского учета)

| К-во клиентов | tpm-C ↓ |
|---|---|
| 29 | 2202531 |
| 26 | 2197183 |
| 30 | 2195122 |
| 32 | 2194285 |
| 27 | 2189189 |
| 31 | 2188235 |
| 28 | 2181818 |
| 25 | 2173913 |

| К-во клиентов | tpm-C ↓ |
|---|---|
| 24 | 2165413 |
| 23 | 2156250 |
| 22 | 2146341 |
| 20 | 2068965 |
| 19 | 2054054 |
| 18 | 2037735 |
| 21 | 2016000 |
| 17 | 1961538 |

| К-во клиентов | tpm-C ↓ |
|---|---|
| 16 | 1882353 |
| 15 | 1747572 |
| 14 | 1647058 |
| 13 | 1529411 |
| 12 | 1358490 |
| 11 | 1346938 |
| 10 | 1290322 |
| 9 | 1270588 |

| К-во клиентов | tpm-C ↓ |
|---|---|
| 8 | 1156626 |
| 7 | 1150684 |
| 5 | 857142 |
| 6 | 847058 |
| 4 | 657534 |
| 3 | 444444 |
| 2 | 328767 |
| 1 | 150000 |

# Производительность на тесте TPC-C

| № п/п | Кластер | СУБД | К-во узлов | К-во клиентов | tpm-C |
|---|---|---|---|---|---|
| 1 | SPARC SuperCluster with T3-4 Servers | Oracle 11g R2 Enterprise Edition w/RAC w/Partitioning | 108 | 81 | 30 249 688 |
| 2 | IBM Power 780 Server Model 9179-MHB | IBM DB2 9.7 | 24 | 96 | 10 366 254 |
| 3 | Sun SPARC Enterprise T5440 Server Cluster | Oracle 11g Enterprise Edition w/RAC w/Partitioning | 48 | 24 | 7 646 486 |
|  | Торнадо ЮУрГУ | PargreSQL | 12 | 29 | 2 202 531 |
| 4 | HP Integrity rx5670 Cluster Itanium2/1.5 GHz-64p | Oracle 10g Enterprise Edition | 64 | 80 | 1 184 893 |

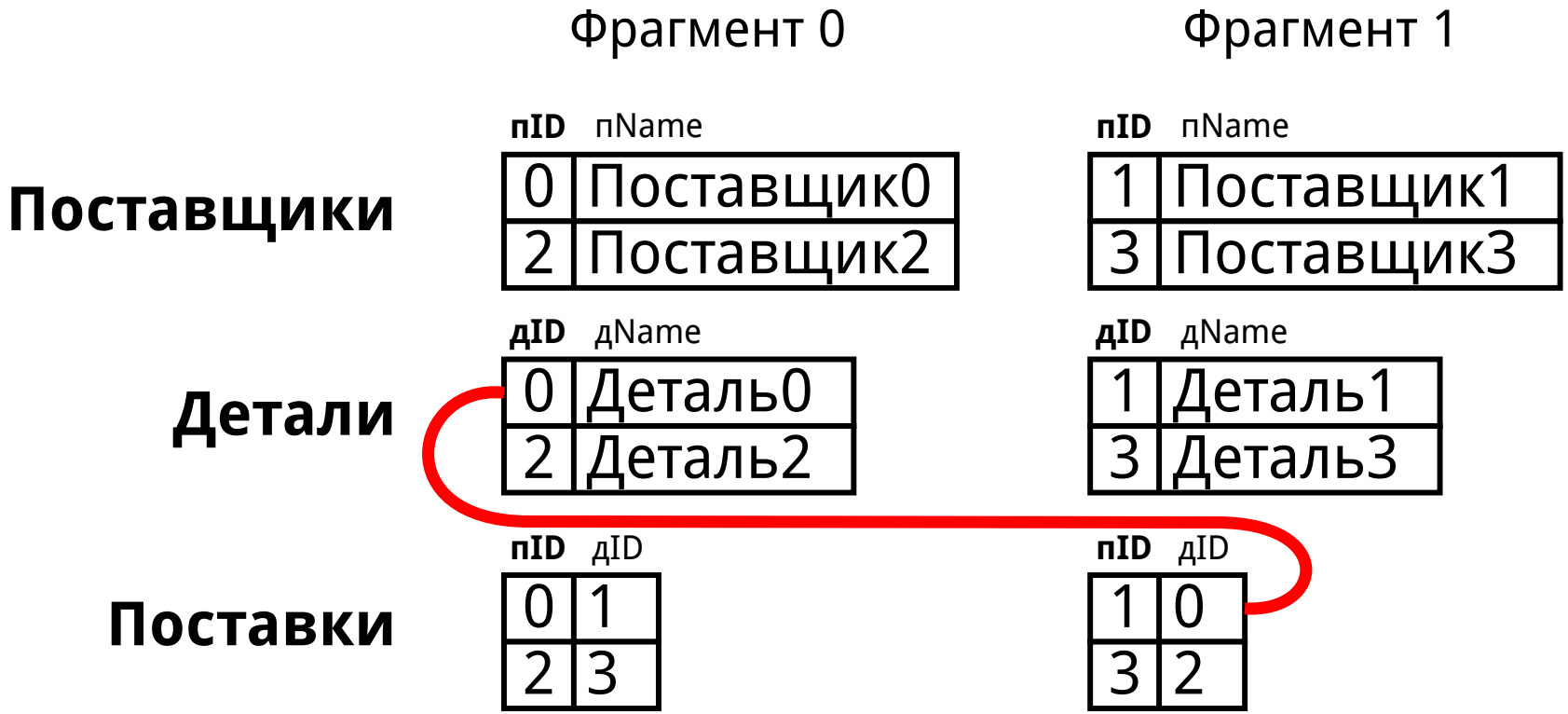
# Основные результаты, выносимые на защиту

1. Разработаны новые методы внедрения фрагментного параллелизма в свободную СУБД с открытым исходным кодом.
2. На основе разработанных методов предложены архитектурные подходы и алгоритмы, реализующие фрагментный параллелизм в рамках последовательной СУБД с открытым исходным кодом, на базе которых выполнена параллелизация СУБД PostgreSQL.
3. Разработан новый алгоритм разбиения сверхбольших графов, состоящих из миллионов вершин и ребер, ориентированный на реляционные СУБД с фрагментным параллелизмом.
4. Проведены вычислительные эксперименты с СУБД PargreSQL, которые показали успешное применение данной СУБД для решения задач классов OLAP и OLTP, связанных с обработкой сверхбольших баз данных.

# Применение методов к другим СУБД

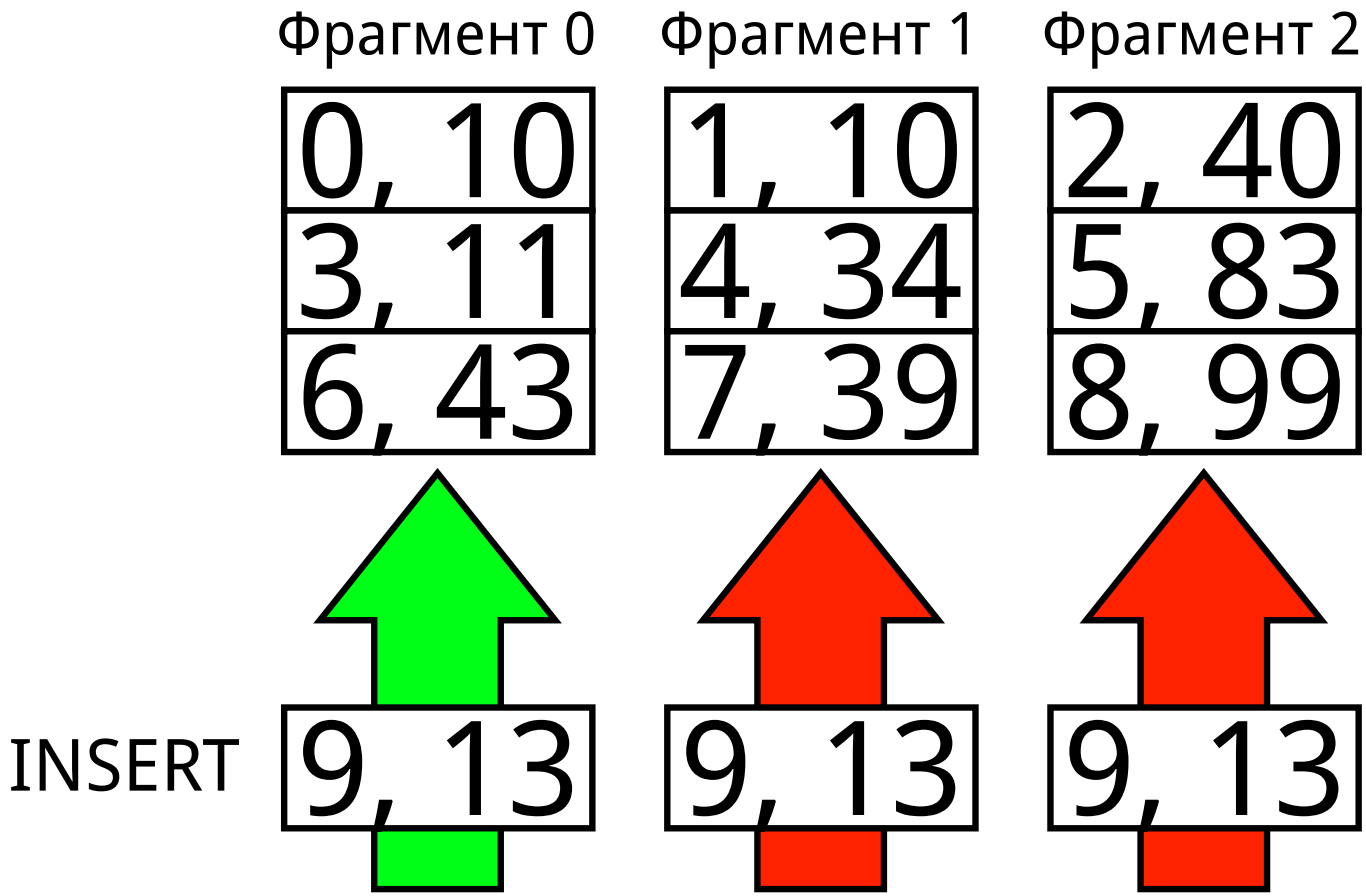
- Реляционные СУБД основаны на реляционной алгебре, поэтому применимы методы обмена и построения параллельного плана.
- Любая СУБД поддерживает словарь данных, поэтому применим метод хранения метаданных.
- Любая СУБД предполагает наличие прикладной библиотеки, поэтому применимы методы тиражирования и портирования.

# Зачем нужны обмены?

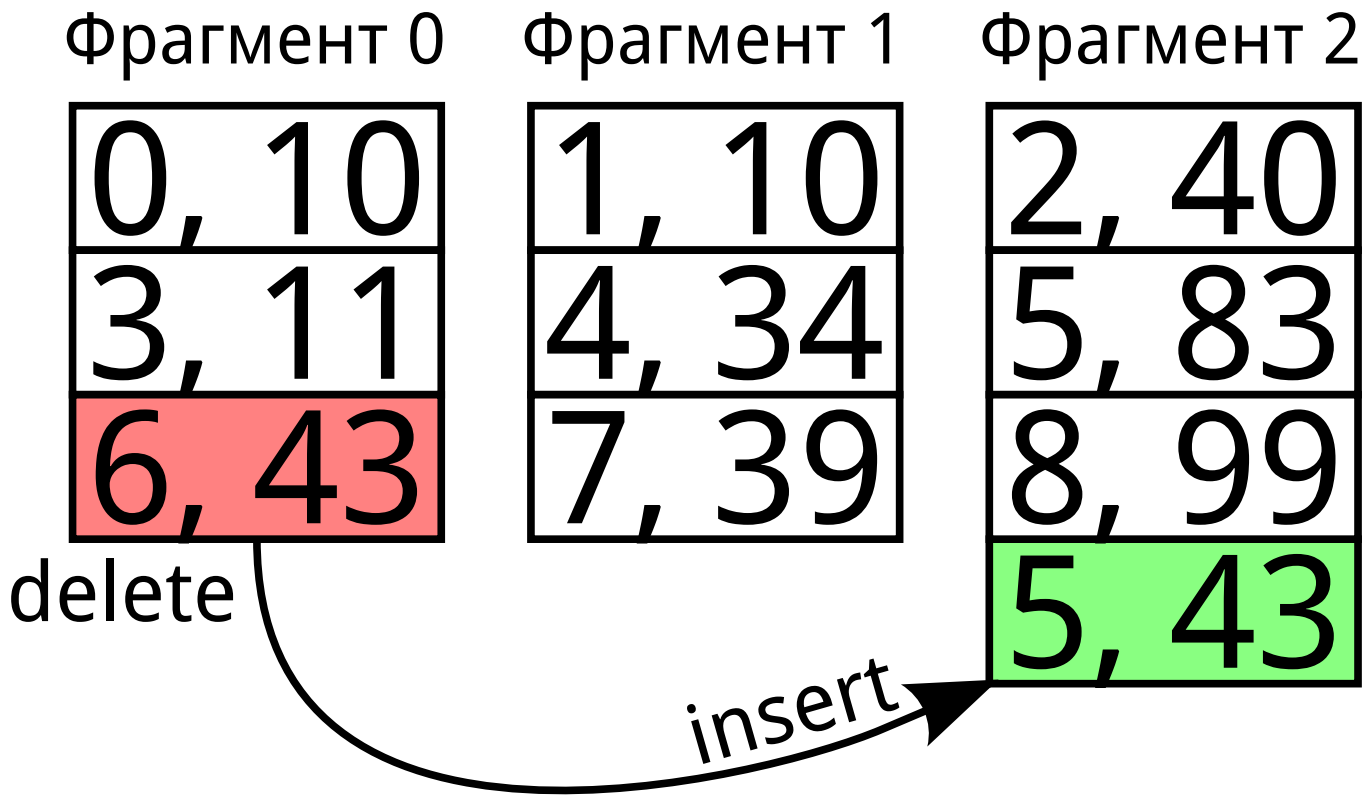



```
select пName, дName
from Поставщики, Детали, Поставки
where Детали.дID = Поставки.дID
and Поставщики.пID = Поставки.пID;
```

# Пример вставки кортежа



# Пример обновления кортежа




UPDATE t SET a = 5 WHERE a = 6;