

ОСНОВЫ ТЕХНОЛОГИЙ БАЗ ДАННЫХ

Б. А. НОВИКОВ

Санкт-Петербургский университет, JetBrains Research

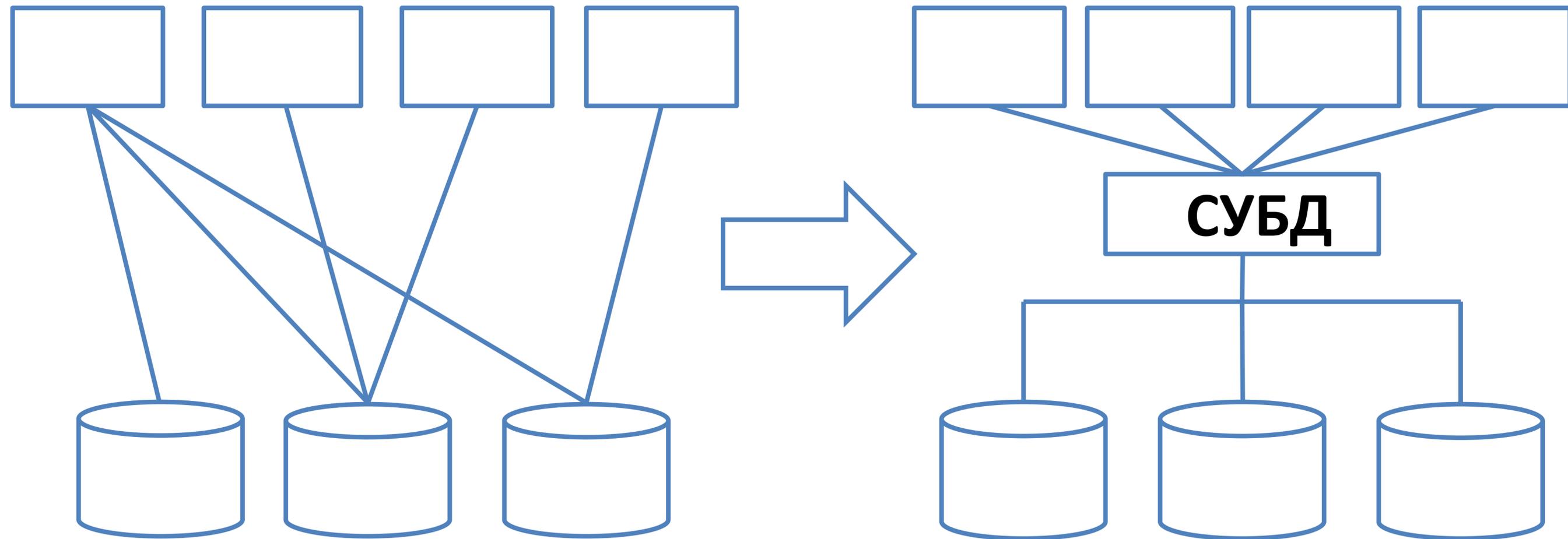


Предыдущие серии

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

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)

Другие модели данных

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

Язык запросов SQL

- Декларативный язык
- Структуры данных и операторы описания данных: CREATE, DROP, ALTER
- Ограничения целостности: NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY
- Операторы: SELECT, INSERT, UPDATE, DELETE
- Реализация операций реляционной алгебры в SQL и другие возможности SQL

Внешние ключи

- FOREIGN KEY — ограничение целостности
- Никак не влияет на семантику запросов на чтение данных
- Соединение (JOIN) никак не связано с понятием внешнего ключа, хотя часто ключом соединения является внешний ключ
- Ограничения целостности никак не влияет на эффективность операции соединения
- Миф о влиянии возник потому, что для внешних ключей часто создаются индексы

Дубликаты



Дискуссия о дубликатах в САСМ

SELECT ALL или SELECT DISTINCT?

+

Дубликаты могут быть необходимы для правильного моделирования реальности

Устранение дубликатов может приводить к снижению эффективности выполнения запросов

Отношения с дубликатами можно описать в теории мультимножеств

-

Реляционная модель работает с множествами и не допускает дубликатов

Введение дубликатов не дает новой информации

Введение дубликатов делает некорректными соотношения между операциями реляционной алгебры

Неэквивалентные запросы

```
CREATE TABLE p(  
  pn varchar(10),  
  pname varchar(10)  
);  
CREATE TABLE sp(  
  sn varchar(10),  
  pn varchar(10)  
);  
INSERT INTO p VALUES('p1', 'screw');  
INSERT INTO p VALUES('p1', 'screw');  
INSERT INTO p VALUES('p1', 'screw');  
INSERT INTO p VALUES('p2', 'screw');  
INSERT INTO sp VALUES('s1', 'p1');  
INSERT INTO sp VALUES('s1', 'p1');  
INSERT INTO sp VALUES('s1', 'p2');  
COMMIT;
```

```
SELECT pn FROM p WHERE pname = 'screw'  
OR pn IN (SELECT pn FROM sp WHERE sn='s1');
```

```
SELECT pn FROM sp  
WHERE sn='s1' OR pn IN (  
  SELECT pn FROM p WHERE pname='screw');
```

```
SELECT p.pn FROM p, sp  
WHERE (sp.sn='s1' AND sp.pn = p.pn) OR p.pname='screw';
```

```
SELECT pn FROM p WHERE pname='screw'  
UNION ALL SELECT pn FROM sp WHERE sn='s1';
```

```
SELECT DISTINCT pn FROM p WHERE pname='screw'  
UNION ALL SELECT pn FROM sp WHERE sn='s1';
```

Безопасные запросы в SQL

- Все хранимые таблицы должны иметь первичные ключи
- Использование DISTINCT гарантирует уникальность выбранных строк
- Включение в список выбираемых значений уникального атрибута каждой из соединяемых таблиц
- При соединении по внешнему ключу включение уникального атрибута таблицы, содержащей внешний ключ, гарантирует уникальность строк результата
- Наличие GROUP BY в запросе гарантирует уникальность строк результата

Программирование запросов: Thinking Sets

- Определить (содержательно), какие элементы составляют множество строк результата
- Построить предложения FROM и, возможно, GROUP BY (получить ключ результата)
- Предложение WHERE (выбрать нужные строки)
- Выражения, определяющие колонки результата
- Возможно, применять эти шаги рекурсивно для построения подзапросов

Языки запросов: итоги

- Декларативные языки позволяют компактно записывать требования к результатам и допускают высокоэффективное выполнение запросов
- Языки запросов теоретической реляционной модели можно выразить конструкциями языка SQL
- Семантика SQL отличается от реляционной из-за неопределенных значений и разрешения дубликатов
- Многообразие средств SQL делает этот язык удобным для самых разнообразных классов приложений

Структуры хранения и выполнение запросов



Схема хранения включает

- Табличные пространства
- Таблицы
- Индексы
- Дополнительные структуры
- Кластеры
- Материализованные представления

Табличные пространства

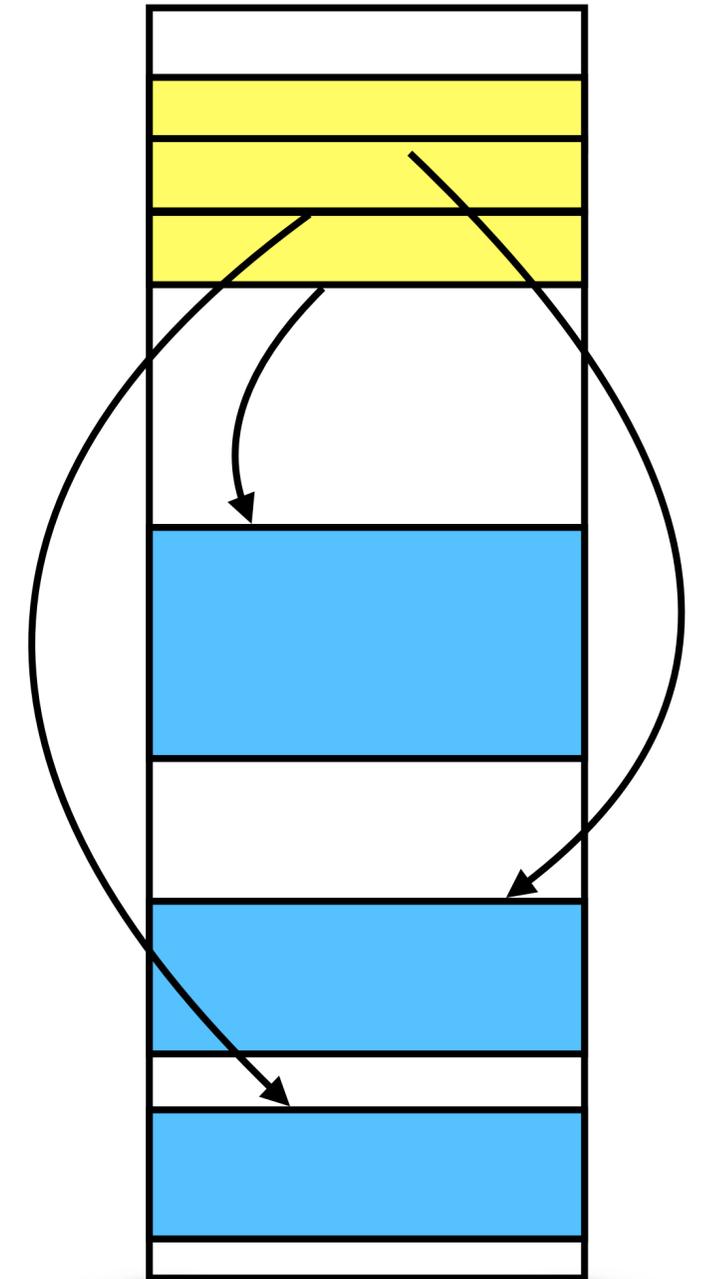
- Определяют абстракцию диска для СУБД
- Могут состоять из нескольких файлов ОС и занимать несколько дисковых устройств
- Экстенты для хранения таблиц и других хранимых объектов выделяются внутри табличных пространств
- Динамическое изменение размеров таблиц
- Распределение внешней памяти может использовать файловую систему
- Соотношение табличных пространств и схем

Адреса и ссылки

- Абсолютный адрес на диске в настоящее время в СУБД не используется
 - Эффективный доступ, очень сложное внесение изменений
- Относительная адресация: ID файла, номер блока, смещение в блоке
 - Используется в большинстве СУБД
- Адресация по уникальным ключам
 - Максимальная гибкость, необходимы дополнительные структуры данных

Размещение логических записей

- Логические записи — это строки таблиц, экземпляры объектов, записи в индексах и др.
- Обычно несколько логических записей размещается в одном блоке
- Неизменяемые идентификаторы логических записей
- Изменение размеров записей
- Значения большого размера хранятся иначе



Избыточные структуры хранения

- **Индексы**

- Избыточная структура
- Предназначены для ускорения поиска
- Прозрачны для приложения

- **Материализованные представления**

- Избыточная структура
- Предназначена для ускорения поиска
- Видна приложению как таблица

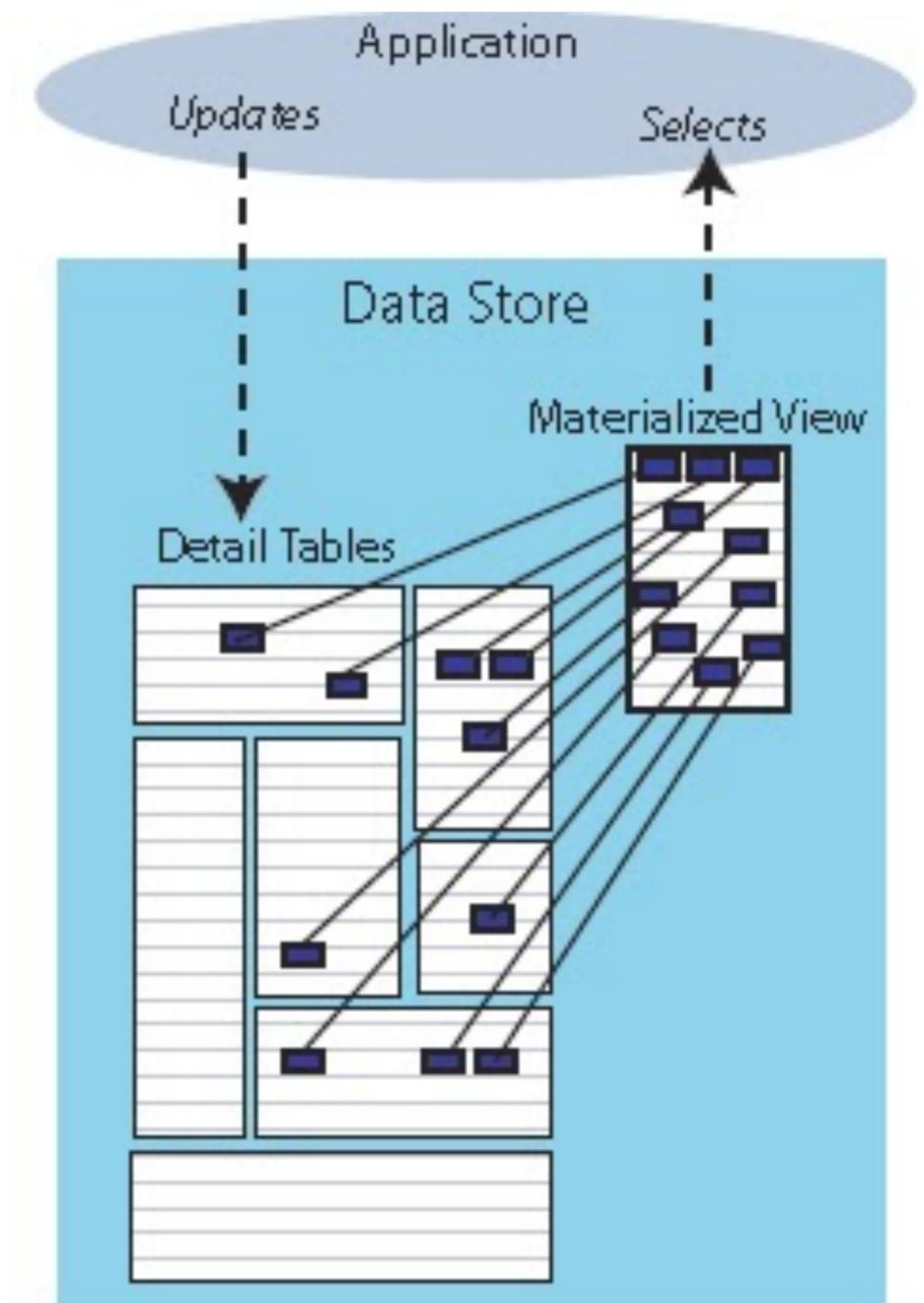
Представления (Views)

```
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

Материализованные представления

- Описываются запросом (SELECT) и хранятся как обычные таблицы
- Обновление: немедленное или по расписанию
- Не каждое материализованное представление можно обновить немедленно
- Дополняющие (incremental) обновления или полная замена хранимых данных



Так называемая денормализация

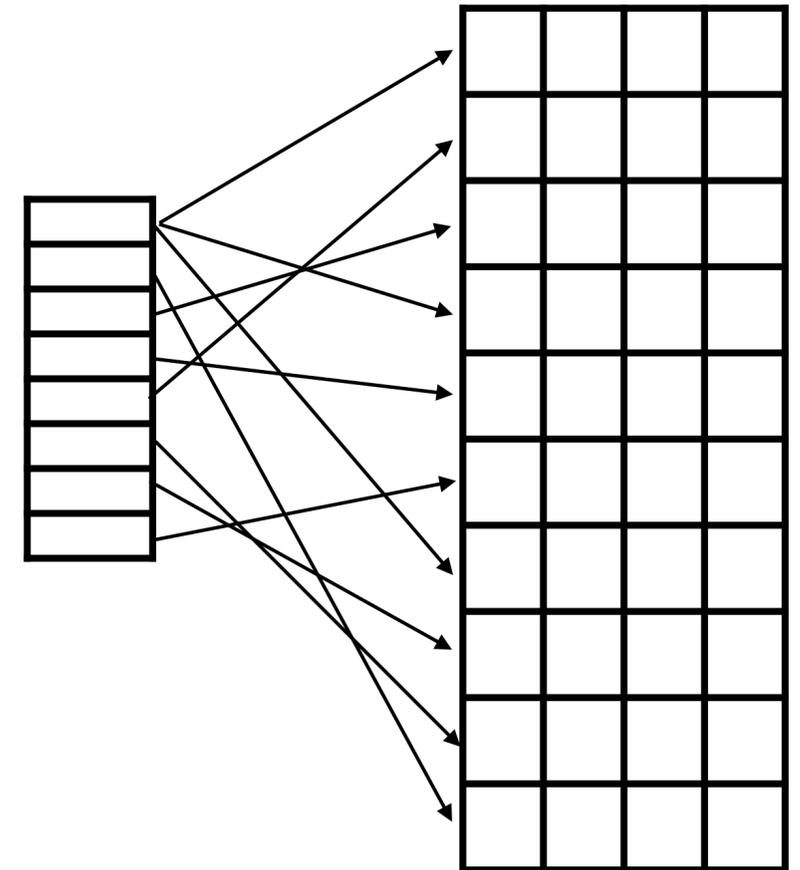
- На жаргоне денормализацией называют отказ от нормализации с целью ускорения работы системы
- Нормализация – понятие логического уровня, а скорость работы зависит от уровня хранения
- Следует использовать материализованные представления в качестве «денормализованных» таблиц

Индексы



Структура индекса

- Логическая запись индекса состоит из
 - значения атрибута (индексного ключа)
 - списка ссылок на логические записи, содержащие это значение
- Индексы по первичным ключам и уникальные индексы содержат только одну ссылку в каждой индексной записи



B-деревья

- Динамическая структура:
не деградирует при модификациях,
не требует реорганизации
- Поиск по диапазону значений ключа
- Особенно эффективны при
небольшой доле выбираемых строк



Просмотр таблиц или поиск по индексам?

- Без индексов
 - пропорционально числу строк в таблице N
- Поиск по индексу
 - пропорционально $\log N + m$
 - m = количество прочитанных строк

	N	$\log_2 N$	уровней в индексе
	10	4	2
	1000	10	2
	1000000	20	4
	1000000000	30	6

Что происходит с запросами

- Синтаксический анализ: преобразование в алгебраическое выражение
- Переписывание: приведение к регулярной структуре
- Оптимизация
 - Логическая оптимизация
 - Выбор алгоритмов и оптимизация по стоимости
- Интерпретация полученного плана выполнения запроса
- Возврат результатов в приложение-клиент

Оптимизация в школьной арифметике

- $ab + ac = a(b + c)$
- Стоимость: $2m + a > m + a$
- Вычисление многочленов по схеме Горнера
- При условии, что стоимость умножения и сложения не зависит от значений
- Это не всегда верно: $99a = 100a + a$

$$\begin{aligned} & a_0 + a_1x + a_2x^2 + a_3x^3 + \dots = \\ & = a_0 + x(a_1 + x(a_2 + x(a_3 + \dots))) \end{aligned}$$

Алгоритмы выполнения алгебраических операций

- Чтение хранимых данных
 - Полный последовательный просмотр
 - Поиск в индексе с просмотром таблицы
 - Чтение только индекса
- Операции соединения, агрегирования, устранения дубликатов
 - Вложенные циклы (NL)
 - Сортировка-слияние (SM)
 - Хеширование (Hash)

Пример запроса

```
SELECT f.flight_no,  
       f.scheduled_departure,  
       count(tf.ticket_no) passengers  
FROM   flights f  
       JOIN ticket_flights tf ON tf.flight_id = f.flight_id  
WHERE  f.departure_airport = 'SV0'  
       AND f.arrival_airport = 'LED'  
       AND f.scheduled_departure BETWEEN  
           bookings.now() - INTERVAL '1 day' AND bookings.now()  
GROUP BY f.flight_no, f.scheduled_departure;
```

План выполнения с полным просмотром

GroupAggregate (cost=183360.24..183360.64 rows=1 width=23)

Group Key: f.flight_no, f.scheduled_departure

-> Sort (cost=183360.24..183360.33 rows=39 width=29)

Sort Key: f.flight_no, f.scheduled_departure

-> Hash Join (cost=7458.52..183359.21 rows=39 width=29)

Hash Cond: (tf.flight_id = f.flight_id)

-> Seq Scan on ticket_flights tf (cost=0.00..153869.12 rows=8392812 width=18)

-> Hash (cost=7458.51..7458.51 rows=1 width=19)

-> **Seq Scan** on flights f (cost=0.00..7458.51 rows=1 width=19)

Filter: ((scheduled_departure <= '2017-08-15 18:00:00+03'::timestamp with

План с поиском по индексу

GroupAggregate (cost=179337.03..179337.43 rows=1 width=23)

Group Key: f.flight_no, f.scheduled_departure

-> Sort (cost=179337.03..179337.13 rows=39 width=29)

Sort Key: f.flight_no, f.scheduled_departure

-> Hash Join (cost=3435.31..179336.00 rows=39 width=29)

Hash Cond: (tf.flight_id = f.flight_id)

-> Seq Scan on ticket_flights tf (cost=0.00..153869.12 rows=8392812 width=18)

-> Hash (cost=3435.30..3435.30 rows=1 width=19)

-> Bitmap Heap Scan on flights f (cost=368.14..3435.30 rows=1 width=19)

Recheck Cond: (departure_airport = 'SVO'::bpchar)

Filter: ((scheduled_departure <= '2017-08-15 18:00:00+03'::timestamp with time zone) AND

-> Bitmap **Index Scan** on flights_departure_airport_idx (cost=0.00..368.14 rows=19696

Index Cond: (departure_airport = 'SVO'::bpchar)



Критерии оптимальности

- Внешние
 - Пропускная способность (?)
 - Время отклика
 - Среднее или максимальное
 - Полный результат или первая строка
- Технические
 - Количество обращений к дискам
 - Суммарное процессоров время
 - Общее время выполнения

Как узнать стоимость выполнения операции?

- Выполнение дает точные результаты
- Оценки на основе статистических характеристик данных
 - Размер занятой памяти (количество блоков)
 - Количество строк
 - Количество разных значений атрибута

SPJ-запросы

- SPJ = Select Project Join
- Операции select и project сокращают объем данных, их следует выполнять как можно раньше
- Если эти операции применяются к промежуточным результатам, их можно выполнить «на лету» при передаче между операциями
- Сложная часть оптимизации: в каком порядке выполнять операции соединения

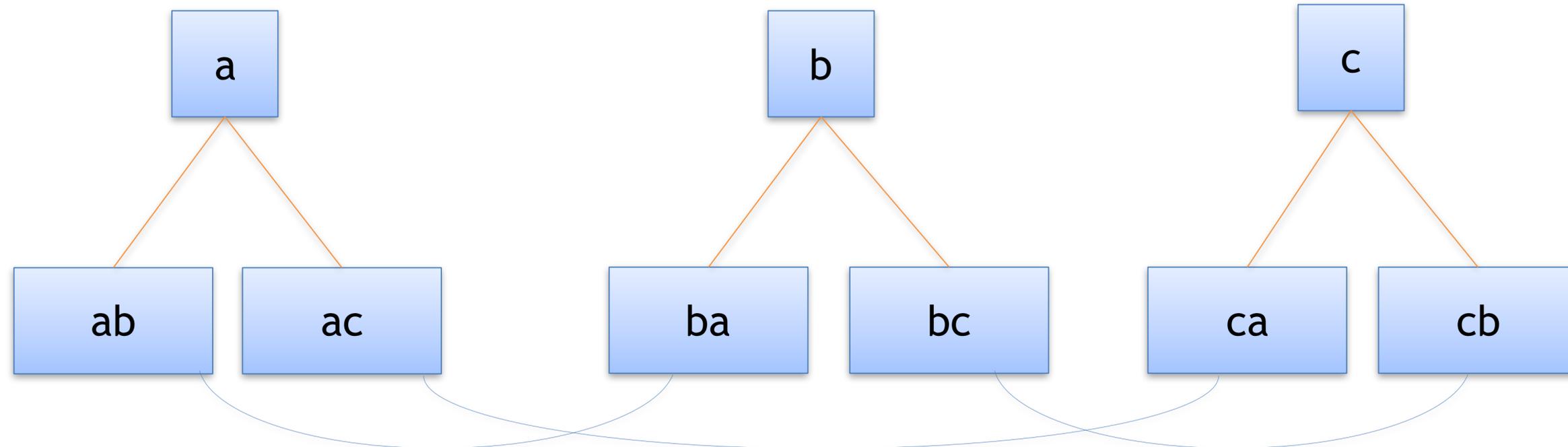
Алгоритм динамического программирования

a

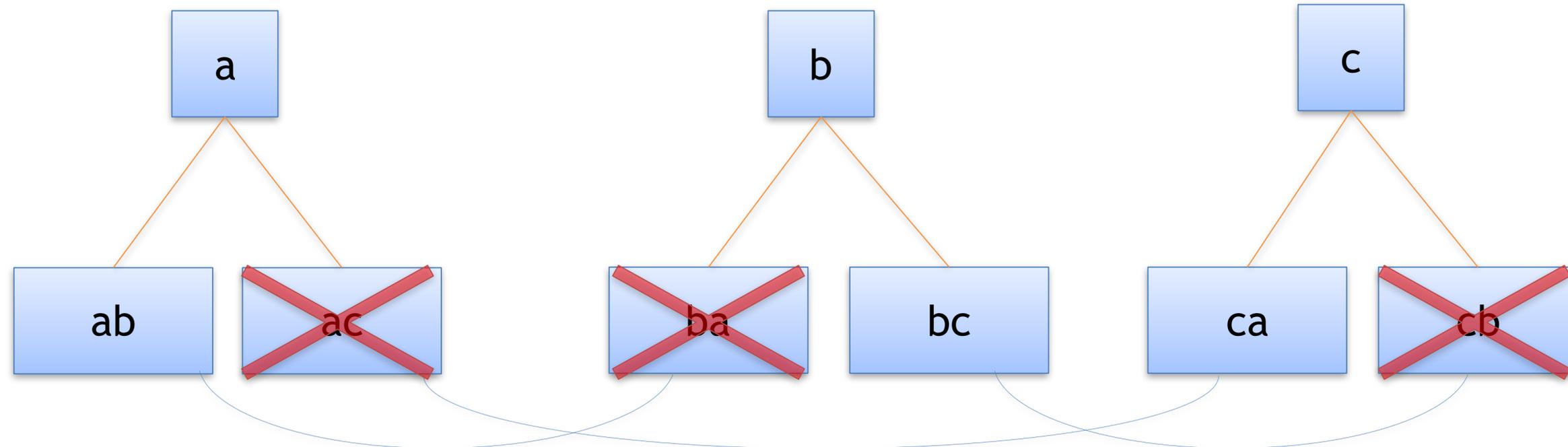
b

c

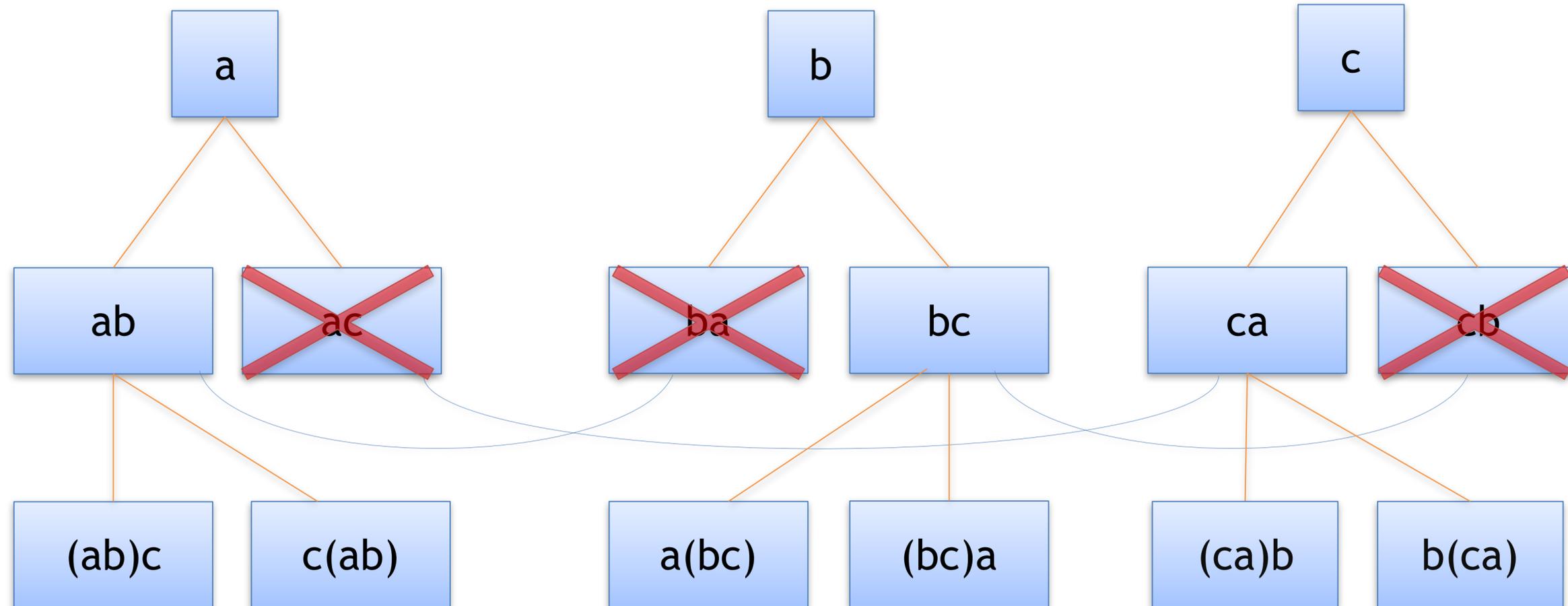
Алгоритм динамического программирования



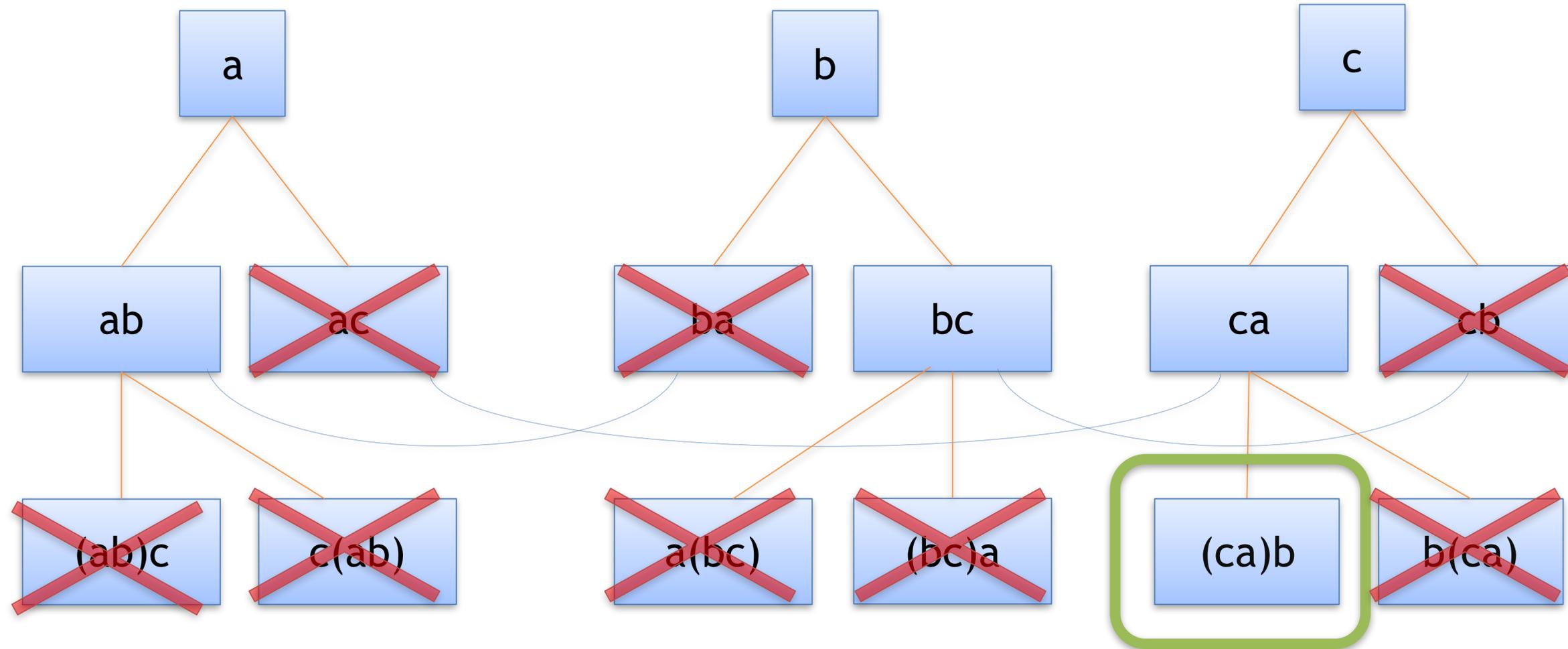
Алгоритм динамического программирования



Алгоритм динамического программирования



Алгоритм динамического программирования



Планы не всегда оптимальны

- Минимизируется оценка стоимости, но не фактическая стоимость
- Высокая сложность делает невозможной точную оптимизацию
- Модели стоимости не могут быть точными
- Оценки статистических характеристик (размеров) промежуточных результатов неточны

Языки запросов: итоги

- Декларативные языки обладают высокой выразительностью
- Запросы описывают, что нужно, но не способ получения
- Оптимизатор выбирает наиболее эффективный способ выполнения запроса с учетом характеристик данных