



Система управления данными для
больших научных проектов

Павел Велихов
НИИСИ РАН

<http://www.scidb.org>



Участники проекта

- ◆ Основатели: Michael Stonebraker, David DeWitt, Jacek Becla, KT Lim
- ◆ Российская команда разработчиков: Павел Велихов, Роман Симаков, Константин Книжник, Артем Смирнов

Michael Stonebraker

- ◆ Ingres
- ◆ Postgres
- ◆ Illustra
- ◆ Cohera
- ◆ Vertica
- ◆ VoltDB



Научные данные

- ◆ “Raw data” (сырые данные) получаем из приборов
- ◆ Научные данные получаются из raw, процедура обработки может меняться
- ◆ Для воспроизводимости результатов сырье данные надо хранить вечно
- ◆ Процедура извлечения научных данных постоянно усложняется (приборы, интеграция, точность)

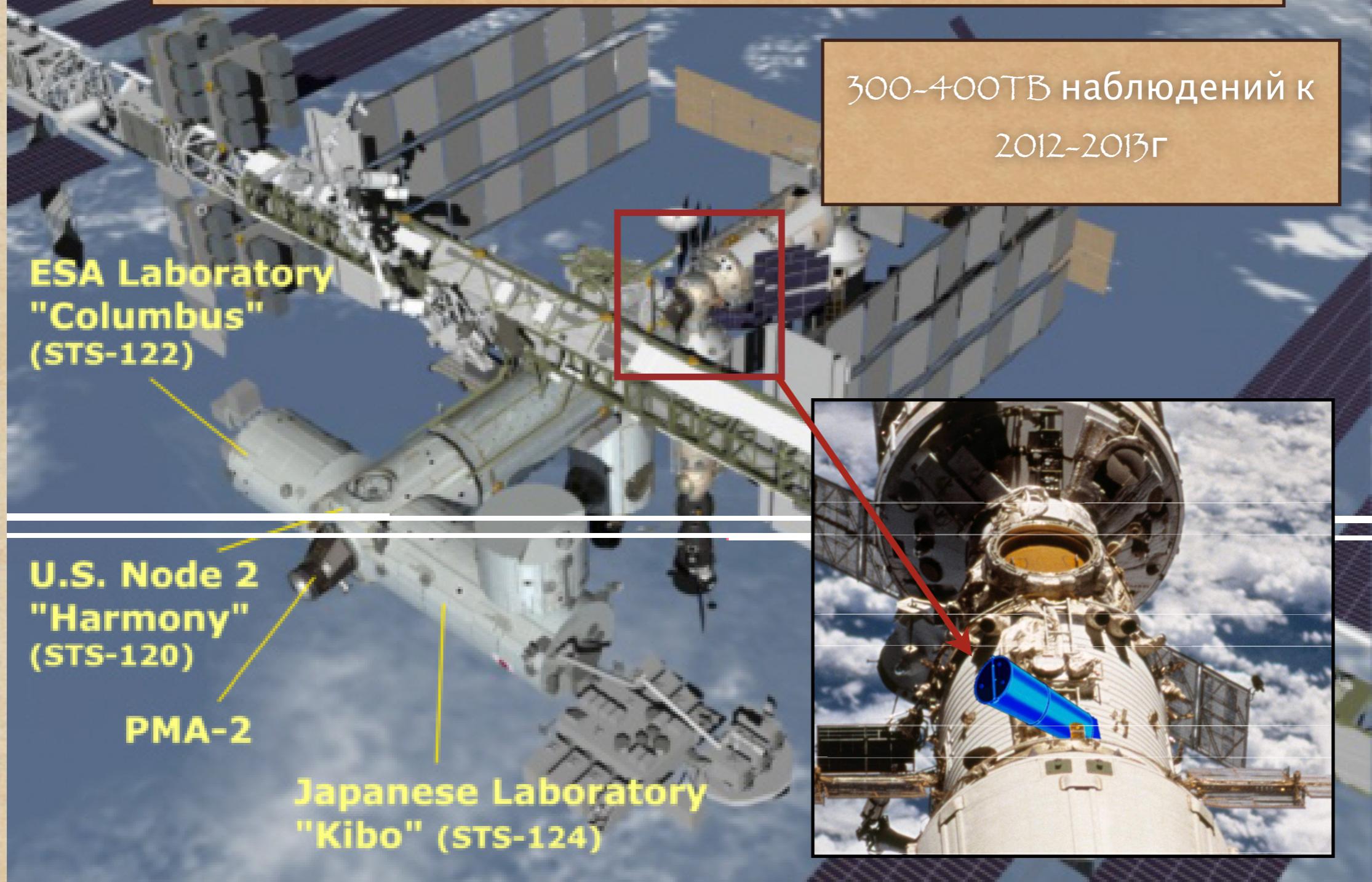
Научные данные (Эволюция)

- ◆ Визуальные наблюдения звезд (каталог Птолемея)
- ◆ Фотографические наблюдения
 - ◆ проявление
 - ◆ выделение объектов, получение метаданных
(яркость, координата, погрешность)
- ◆ ПЗС матрица ~ дополнительные сложности
 - ◆ шумы, неоднородность поля, охлаждение

Большие научные базы

- ◆ LSST - большой обзор неба
 - ◆ 8.4м зеркало, 3.2 Gpx CCD
 - ◆ 49 млрд объектов
 - ◆ 14 Pb данных
- ◆ LHC - Large Hadron Collider
 - ◆ 15 Pb данных ежегодно
 - ◆ 200 центров в 40 странах
- ◆ Результаты численного моделирования (Космология, Нанотехнологии, Генетика,)

ЛИРА ~ орбитальное сканирование неба (ГАИШ МГУ, РКК Энергия)



Обработка сырых данных

- ◆ Специфика приборов
- ◆ Интеграция данных
- ◆ Корреляция данных

Типичный процесс обработки научных данных



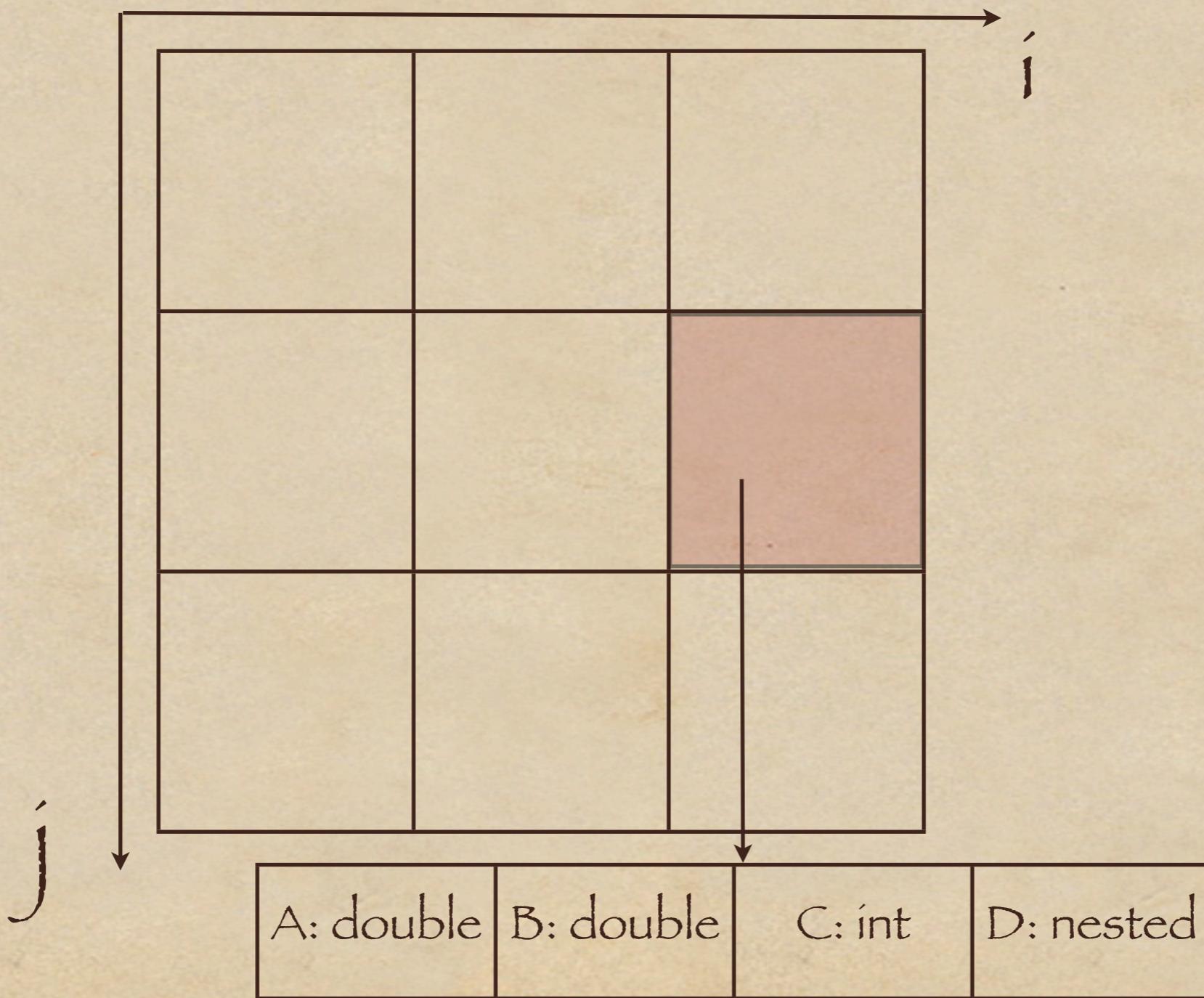
Но зачем нужна еще одна СУБД

- ◆ Большинство научных проектов, где объемы данных внушительные (петабайты) почти не используют СУБД
 - ◆ Эффективность хранения
 - ◆ Производительность
 - ◆ Открытый код для свободного обмена данными

Что требуется от SciDB

- ◆ Эффективное хранение сырых данных
- ◆ Полный цикл анализа данных: data cleaning, feature extraction, data mining, data sharing
- ◆ Версионность и provenance (происхождение) для обеспечения повторяемости результатов
- ◆ Специфические требования для научных данных (погрешность измерений, расширяемость, т.д.)

Модель данных SciDB



Хранение

- ◆ Самая распространенная модель данных в науке – многомерный массив
- ◆ Массив может быть разреженным или/и с неровными краями
- ◆ Каждый элемент массивов может содержать несколько значений
 - ◆ пример: измеренная величина и ее погрешность
- ◆ Матрицы могут быть вложенными

Что тут принципиально нового?

- ◆ Массив можно моделировать реляционной таблицей:
 - ◆ `create table (i,j,A,B,C,D_key)`
- ◆ Также, в современных базах есть тип данных “массив”
- ◆ Но: оба подхода имеют ряд серьезных недостатков

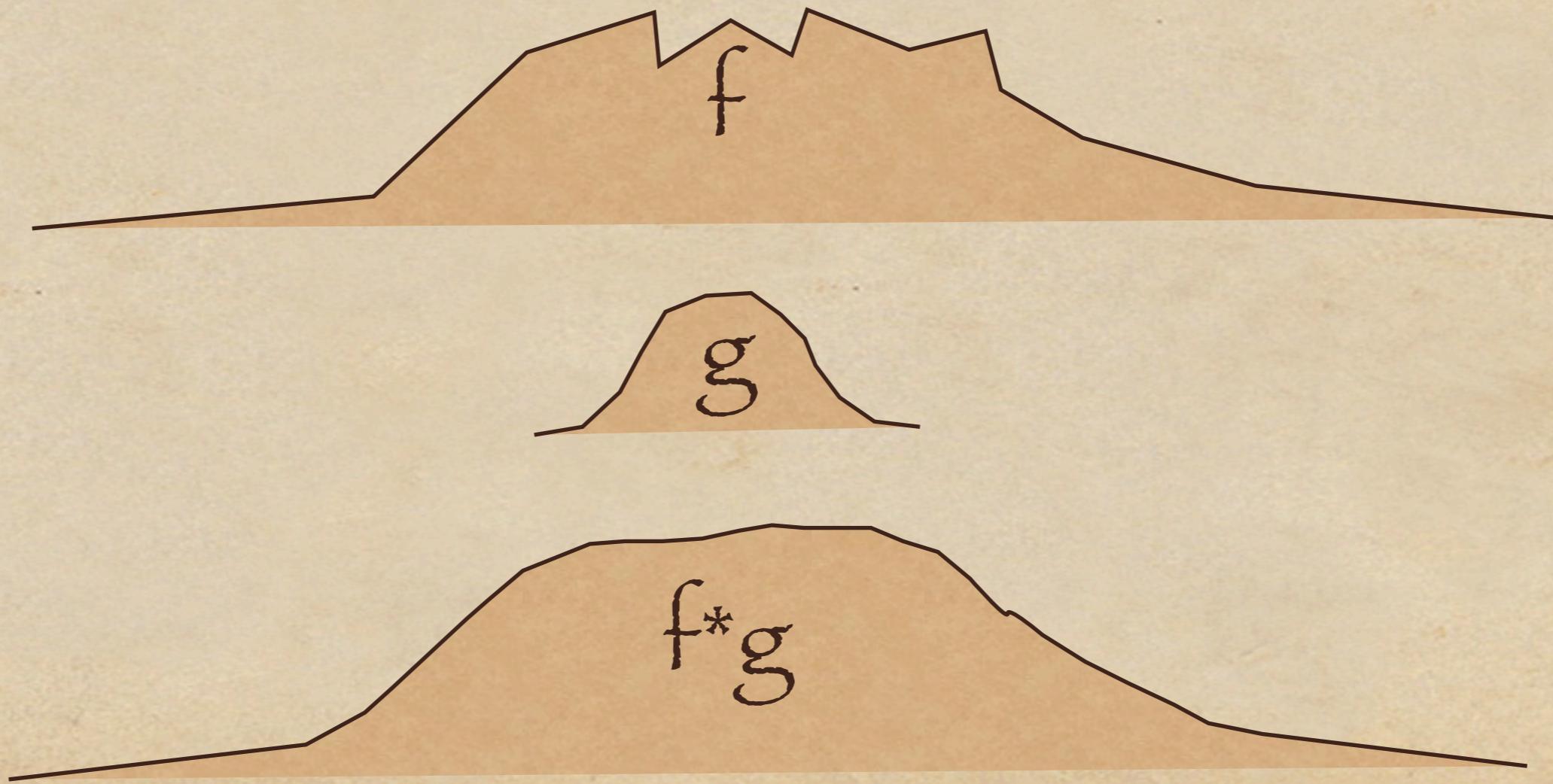
SQL: Массив-таблица

- ◆ Хранение индексов массива.
- ◆ Доступ к подмножествам массива?
- ◆ Вычисления на окрестностях точки?

Используя SQL?

Типичные операции с массивами

- ◆ Свертка: $f^*g(x)$



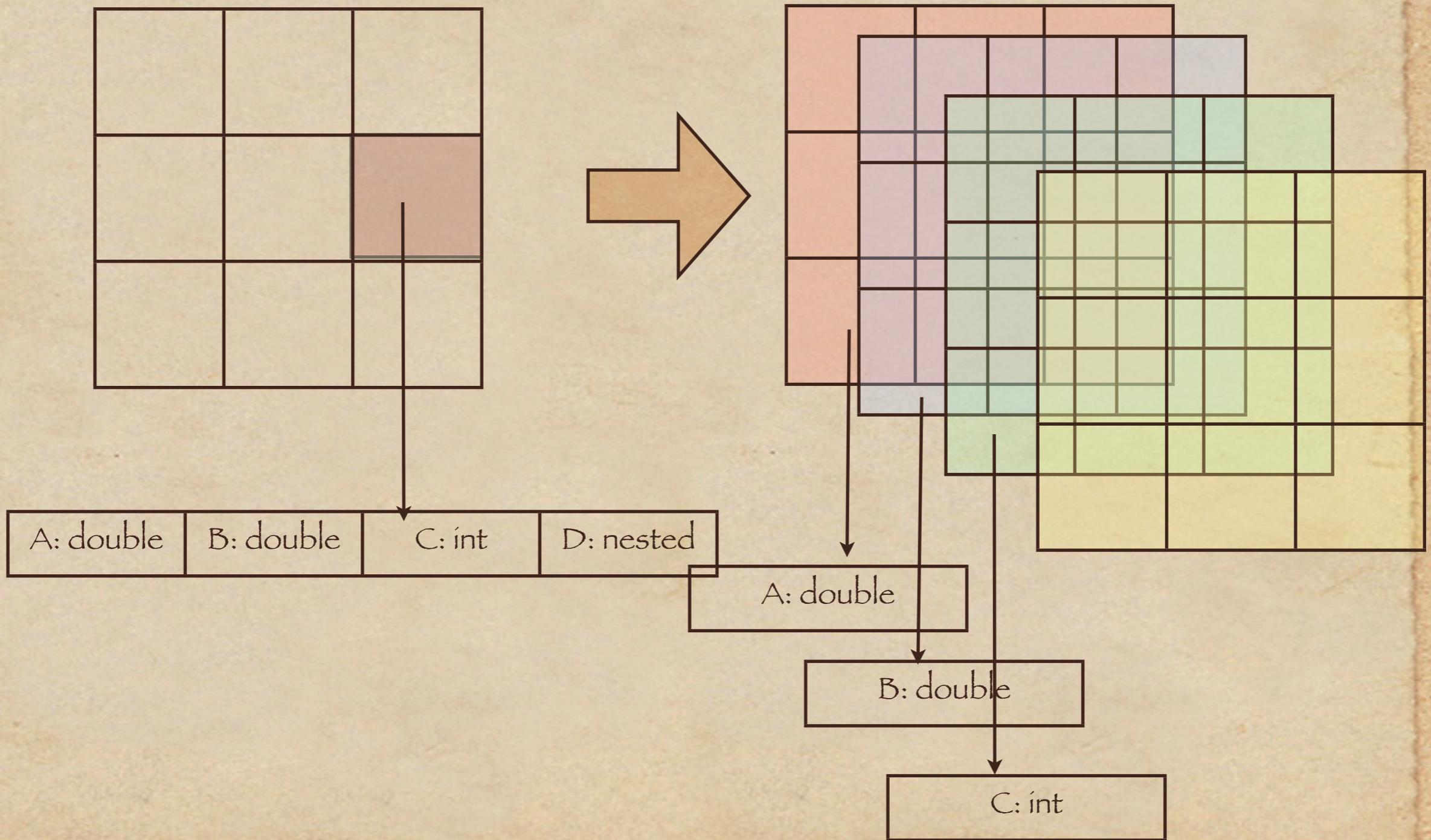
SQL: Массив – тип данных

- ◆ Пропали атрибуты!
- ◆ Язык запросов (*SQL*) с массивами не работает, работать можно только с композицией функций поверх массива!

Хранение данных в SciDB

- ◆ Модель данных – массив
- ◆ Вертикальное хранение с компрессией
- ◆ Горизонтальное разделение данных с перекрытием (для распараллеливания и отказоустойчивости)

Вертикальное хранение



Вертикальное хранение

- ◆ Легко сжимать данные (все атрибуты одного типа)
- ◆ Можно читать с диска только то, что нужно для запроса
- ◆ Отлично подходит для аналитики:
Vertica, Big Table
- ◆ Но не для OLTP!

Транзакционные свойства SciDB

- ◆ Данные не меняются, только добавляются!
- ◆ Таким образом мы избегаем распределенных транзакций
- ◆ Все равно в распределенной системе добиться ACID свойств практически невозможно (что делать, если система развалилась на 2 части и каждая живет своей жизнью?)

CAP теорема

- ◆ (Eric Brewer, UCB 2000): Любая система с распределенными компонентами не может обладать всеми тремя свойствами:
Consistency, Availability, Partition tolerance
- ◆ Partition tolerance – система должна работать при распаде сети, в распределенной системе это неминуемо
- ◆ Приходится выбирать между С и А

BASE (не ACID)

- ◆ Basically Available (потеря части системы не делает систему недоступной)
- ◆ Soft State
- ◆ Eventual Consistency - целостность появится не сразу

Эффективность хранилища

- ◆ Основные накладные расходы современных СУБД
 - ◆ logging
 - ◆ locking
 - ◆ latching
 - ◆ page-based buffer management

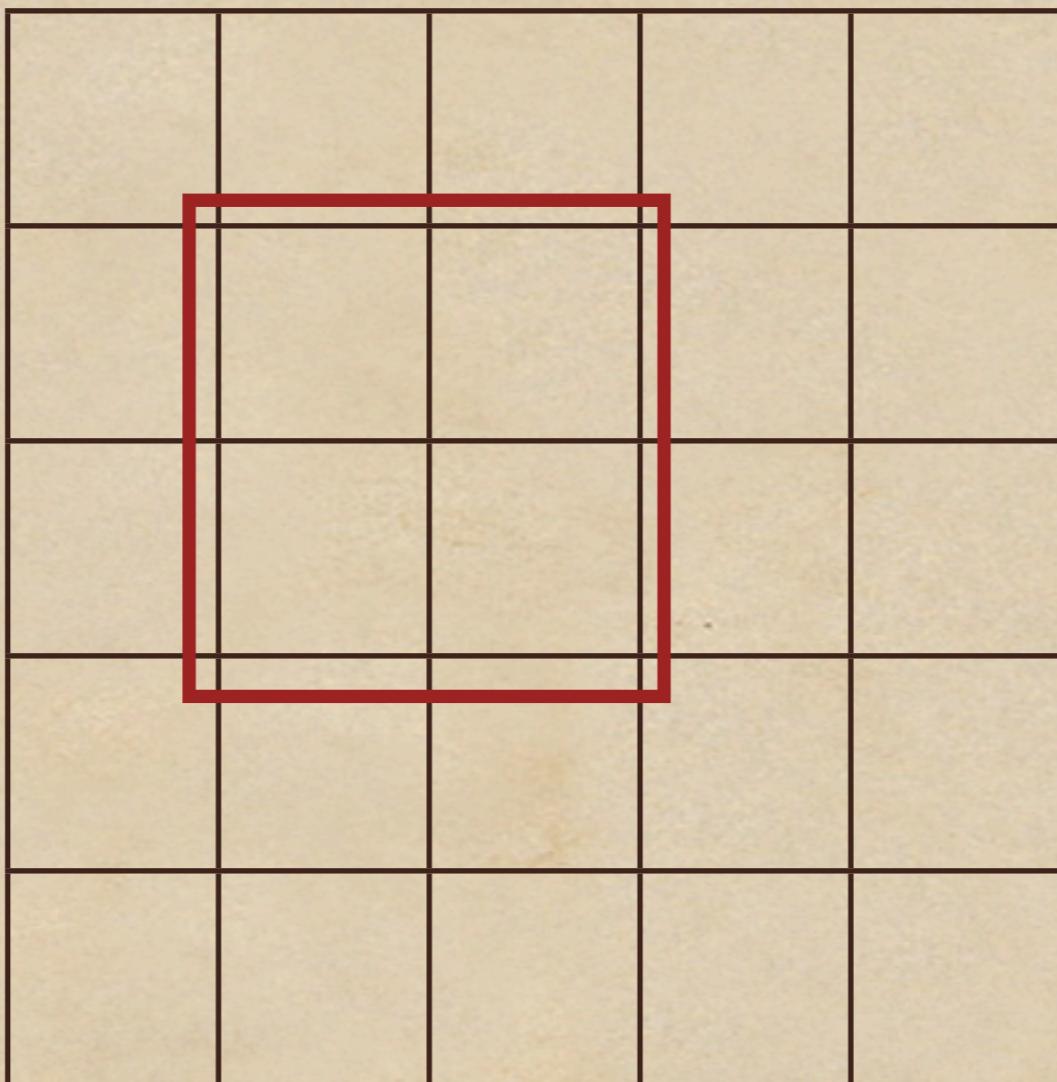
Что требуется от SciDB

- ◆ Эффективное хранение сырых данных
- ◆ Полный цикл анализа данных: data cleaning, feature extraction, data mining, data sharing
- ◆ Версионность и provenance для обеспечения повторяемости результатов
- ◆ Специфические требования для научных данных (uncertainty, т.д.)

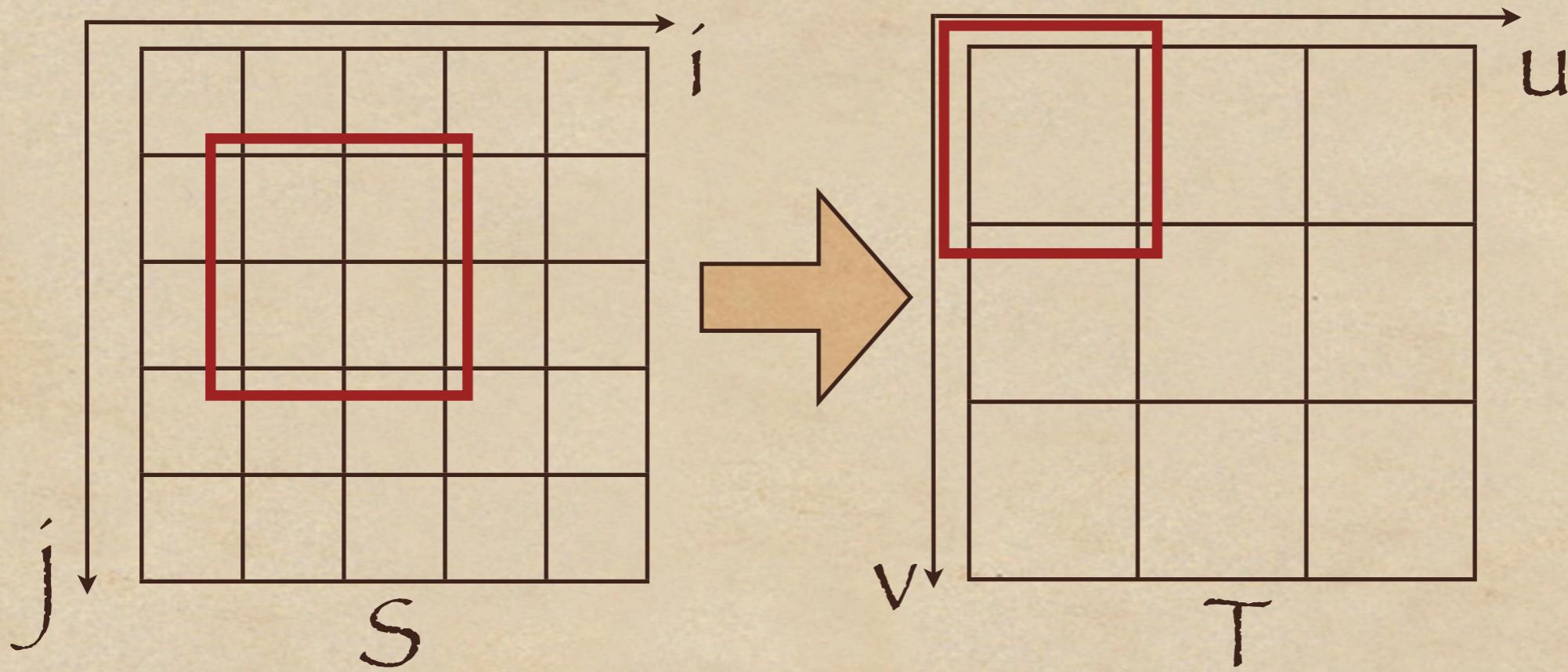
Операции над массивами

- ◆ Алгебра массивов (вместо реляционной)
- ◆ Многие операторы похожи на реляционные - filter, join, group-by
- ◆ Специфические операторы для научных вычислений: subsample, regrid

SciDB Algebra: Subsample



SciDB Algebra: Regrid



map: $(u, v) \rightarrow \{ (i, j) \}$
reduce: $f(\text{map}(u, v)) \rightarrow \text{value}$

Вложенные массивы

- ◆ Как работать с вложенными массивами?
- ◆ Вложенные планы (аля OQL, XQuery)
- ◆ Оператор *Apply* применяет вложенный план к вложенному массиву
- ◆ Полученный результат добавляет как атрибут к массиву

Выполнение запросов в SciDB

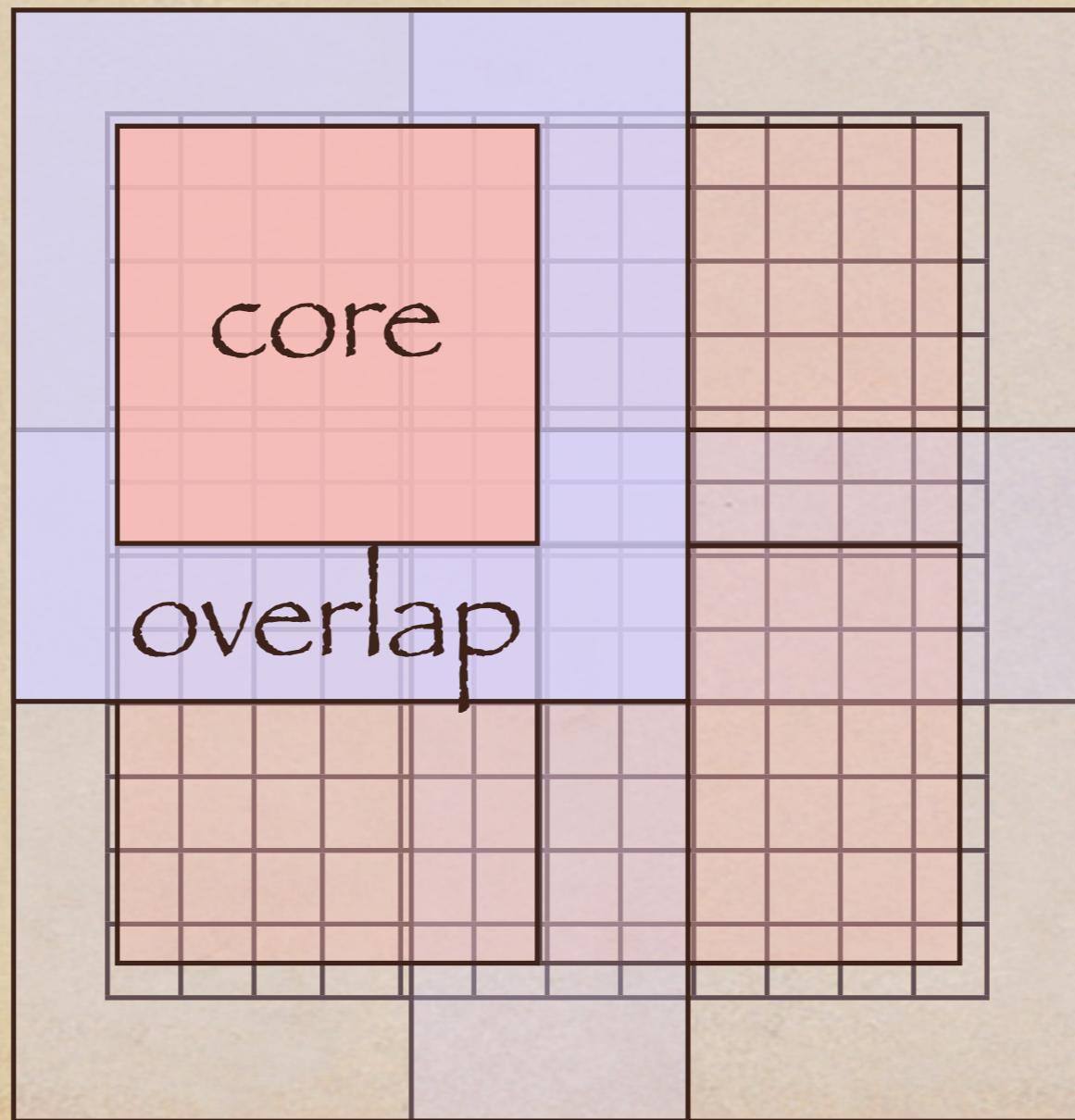
- ◆ Вертикальное хранение с компрессированными чанками
- ◆ Каждый оператор SciDB работает с чанками (аля Vertica), а не с кортежами (Oracle, PostgreSQL,...)

Пример: Subsample

- ◆ Конвейерный (pipeline) интерфейс: `getChunk` (`attr, pos`)
- ◆ Пересылаем вниз `getChunk(attr, pos2)`
 - ◆ $pos2 = pos + offset$
- ◆ Если чанк внутри “окна” оператора, отдаем вверх не распаковывая
- ◆ Распаковываем, обрезаем, передаем вверх

Параллелизм в SciDB

- ◆ Горизонтальное разбиение с перекрытием

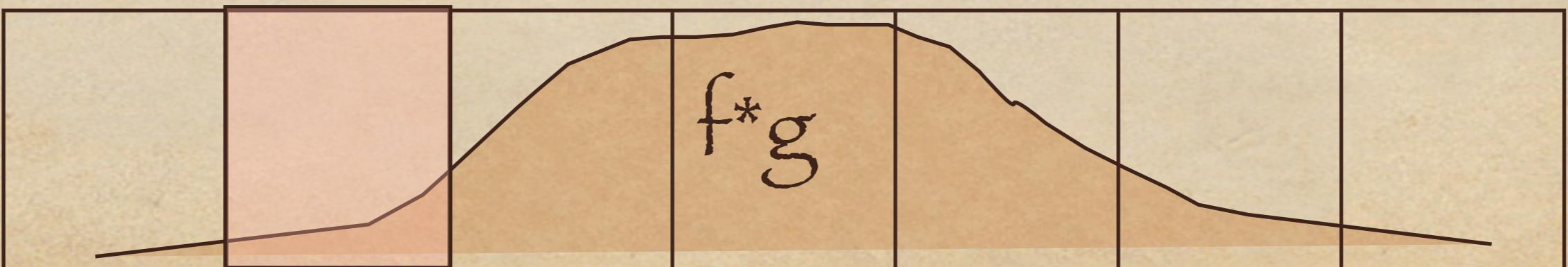
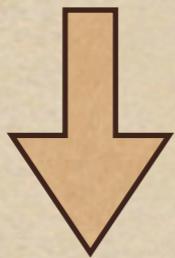
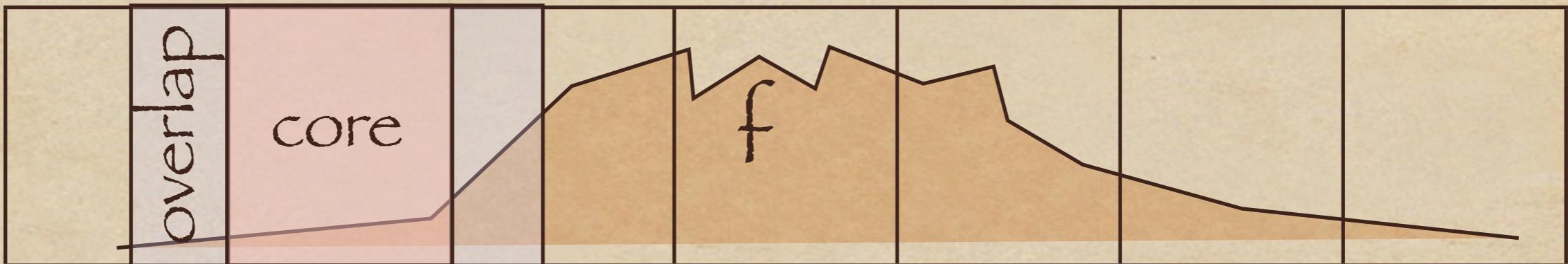


Параллелизм в SciDB

- ◆ Оптимизатор запроса вычисляет, достаточно ли первоначальное разбиение для выполнения запроса
- ◆ Если нет, то в план добавляется оператор *Scatter*, который “докидывает” дополнительные данные

Параллелизм в SciDB

- ◆ Свертка: $f^*g(x)$



Что требуется от SciDB

- ◆ Эффективное хранение сырых данных
- ◆ Полный цикл анализа данных: data cleaning, feature extraction, data mining, data sharing
- ◆ Версионность и provenance для обеспечения повторяемости результатов
- ◆ Специфические требования для научных данных (uncertainty, т.д.)

Дополнительные требования

- ◆ Версионность
 - ◆ изменения данных ≈ новая версия
 - ◆ релиз и последняя версия ≈ быстрый доступ

Дополнительные требования

- ◆ Provenance (происхождение данных) – “откуда взялась эта звезда?”
- ◆ playback
- ◆ отладка анализа данных
- ◆ повторяемость научных данных

Что требуется от SciDB

- ◆ Эффективное хранение сырых данных
- ◆ Полный цикл анализа данных: data cleaning, feature extraction, data mining, data sharing
- ◆ Версионность и provenance для обеспечения повторяемости результатов
- ◆ Специфические требования для научных данных (uncertainty, расширяемость, ...)

Дополнительные требования

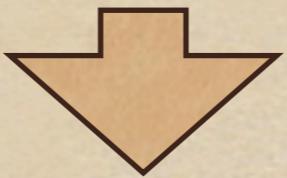
- ◆ Uncertainty - измерения приборов всегда неточные
- ◆ Гауссовское распределение ошибки
- ◆ Частичная поддержка арифметики интервалов в алгебре SciDB (filter,join):
 - ◆ $a - \text{err} < C < a + \text{err}$
 - ◆ $a \approx C$

Дополнительные требования

- ◆ Расширяемость:
 - ◆ User Defined Types: модель PostgreSQL
 - ◆ User Defined Functions: произвольные функции над массивами
 - ◆ Интерфейсы к научным пакетам ПО: R, Matlab.

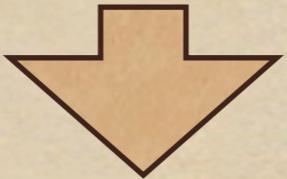
Наша цель:

Сырые данные



SciDB

обработка сырых данных, анализ,
верификация, получение научных
результатов



Повторяемые научные результаты

Пример использования SciDB:

Анализ генома

- ◆ Последовательности нуклеотидов AGCT
- ◆ много N – ошибки прибора
- ◆ контроль качества:
 - ◆ сколько ошибок, где они встречаются, есть ли корреляция с нуклеотидами, какая временная зависимость
- ◆ дополнительные данные о точности сканирования

Коммерческие применения

- ◆ Финансовый анализ
- ◆ GIS (карты, погода, пробки,...)
- ◆ Анализ потока веб-навигации
(eBay)

Статус SciDB

- ◆ Проект с открытым кодом
- ◆ Разработка ведется с начала 2009
- ◆ Демо прототипа: сентябрь 2009
VLDB 09
- ◆ Альфа версия: SciDB v1, Q2 2010

Интересные исследовательские вопросы

- ◆ Разбиение по гиперкубам (curse of dimensionality)
- ◆ Отказоустойчивость
- ◆ Оптимизация запросов
 - ◆ докачивание и кэширование данных
(materialized views)
 - ◆ упорядоченная модель данных
 - ◆ параллельная обработка вложенных массивов

- ◆ Спасибо за внимание
- ◆ Вопросы & Ответы