

**Работа со слабоструктурированными
данными в реляционных СУБД.
Реализация типов данных
hstore, intarray, xml в СУБД PostgreSQL**

Самохвалов Николай Александрович
ФУПМ МФТИ
«Постгресмен» <http://postgresmen.ru>

Вместо предисловия...

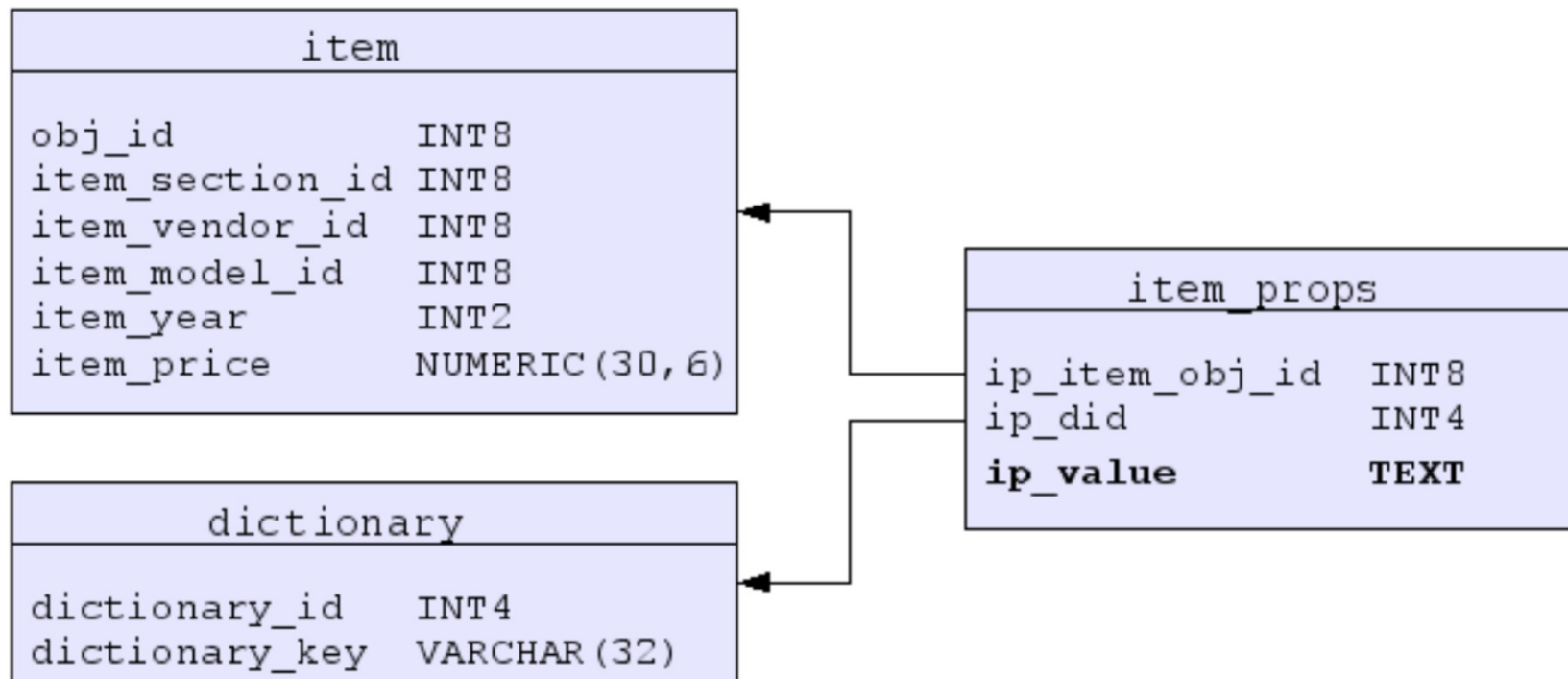
Задача: спроектировать БД каталога разнородных объектов

Авто [+ Подать объявления в этот раздел](#)

Кредиты и ссуды (22)	Комиссионное оформление и страхование (5)
Легковые автомобили (35679)	Прицепы, автодома и фургоны (146)
Новые автомобили	Автобусы (231)
Автомобили с пробегом	Спецтехника (1426)
Малый коммерческий транспорт (2320)	Мотоциклы и мопеды (1505)
Средние и тяжелые грузовики (2039)	Снегоходы (37)
Грузовые прицепы (508)	Автосервис и услуги (780)
Водный транспорт (365)	Сервис и ремонт
Автозапчасти и принадлежности (10507)	Экспертиза и оценка
Для легковых автомобилей	Юридическая помощь
Для коммерческих автомобилей	Автошколы
Для мотоциклов и мопедов	Другое
Диски, шины, колеса	
Для спецтехники и прицепов	
Авто на запчасти	
Средства для ухода	
Другое	
Другое (400)	

IRR.ru, раздел «Авто»

Разнородный каталог: «реляционный» подход №1



Каталог разнородных данных: 1-й способ реализации
(Entity-Attribute-Value, EAV)

Недостатков больше, чем преимуществ:
*3 таблицы, большой объём данных (включая индексы),
необходимость выполнения операции соединения 3-х таблиц*

Разнородный каталог: «реляционный» подход №2

item	
obj_id	INT8
item_section_id	INT8
item_vendor_id	INT8
item_model_id	INT8
item_year	INT2
item_price	NUMERIC(30,6)
item_prop1	INT4
item_prop2	INT4
item_prop3	INT4
item_prop4	INT4
...	
item_prop21	TEXT
item_prop22	TEXT
item_prop23	TEXT
...	
item_prop41	BOOLEAN
...	

Каталог разнородных данных: 2-й способ реализации («широкая таблица»)

Разнородный каталог: «забудем» о 1NF*?

item	
obj_id	INT8
item_section_id	INT8
item_vendor_id	INT8
item_model_id	INT8
item_year	INT2
item_price	NUMERIC(30,6)
item_props	???

int[] ?

anytype[] ?

xml ?

hstore ?

Каталог разнородных данных: 3-й способ реализации (массивы, XML, hstore)

*) 1NF «по Кодду»

ORDBMS PostgreSQL



- Расширяемость (**полноценные** встроенные пользовательские типы данных)
 - использование наряду с основными типами
 - индексная поддержка
 - конкурентный доступ и восстановление после сбоев

Индексы в PostgreSQL

- B-tree
- Hash
- R-tree
- **GiST** (обобщённое поисковое дерево)
- **GIN** (обобщённый обратный индекс)

Индексы в PostgreSQL

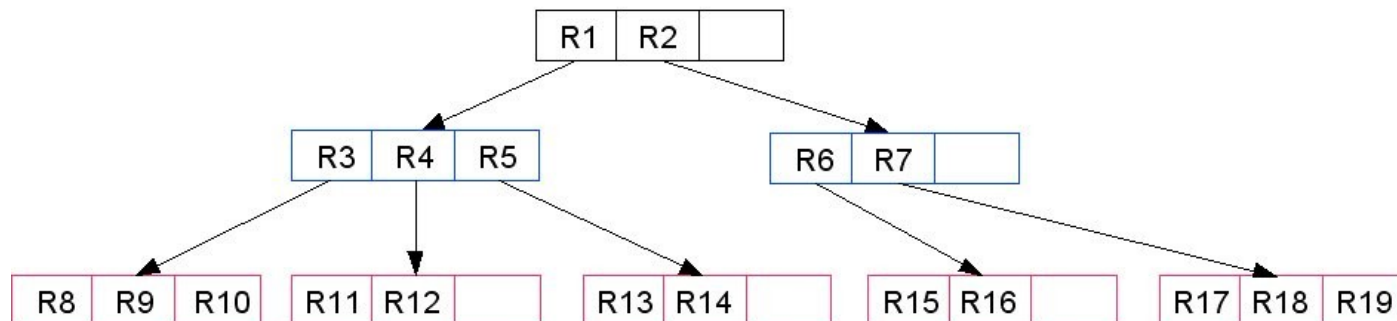
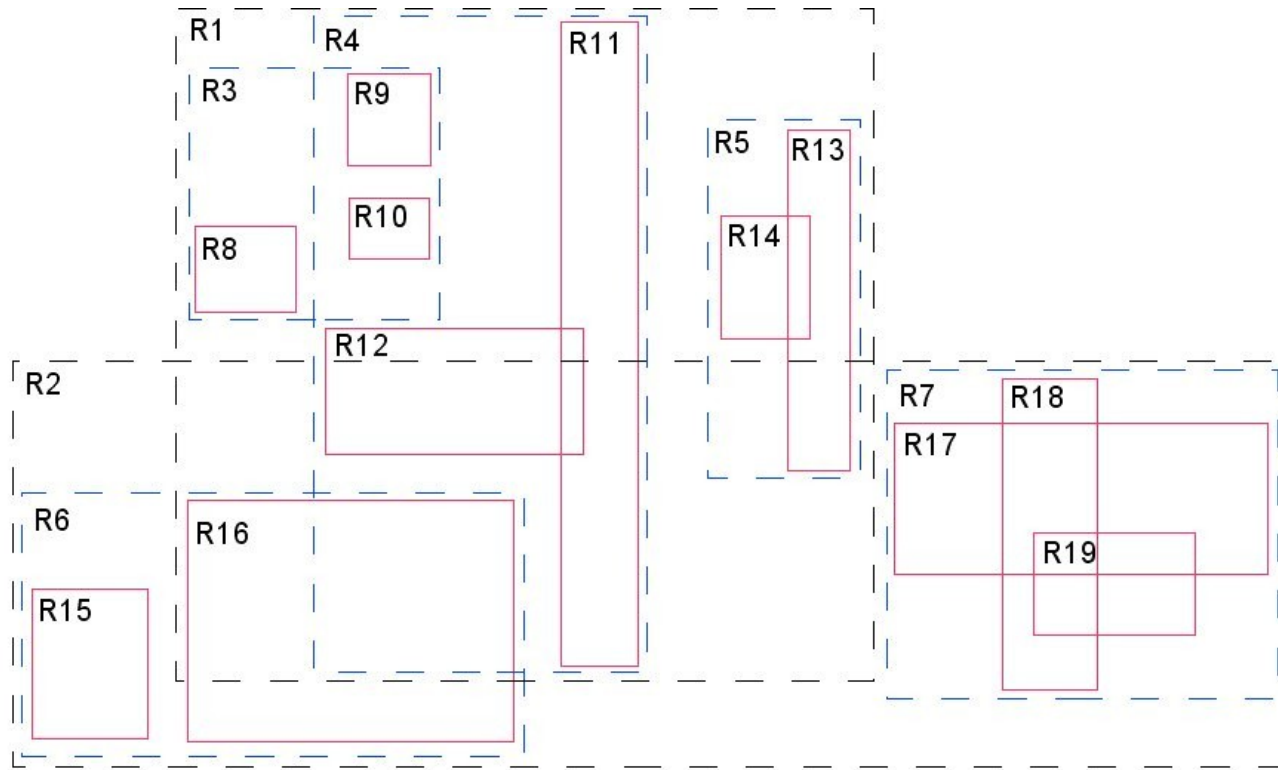
- B-tree
- Hash
- R-tree – теперь тоже GiST!
- **GiST** (обобщённое поисковое дерево)
- **GIN** (обобщённый обратный индекс)

Особенности реализации GiST в PostgreSQL

- Ключи переменной длины
- Композитные ключи
- Оптимизированный INSERT (однопроходный вместо рекурсивного)
- Concurrency & Recovery!

Что такое GiST?

R-tree



A. Guttman. R-tree: A Dynamic Index Structure for Spatial Search. 1984

GiST: Generalized Search Tree

- Каждый узел содержит от kM до M ключей (кроме корня)
- Лист: для всех (p, ptr) выполняется предикат p для данных по указателю ptr
- Внутренний узел: для всех (p, ptr) выполняется предикат p для данных *достижимых* при спуске от ptr
- Дерево сбалансировано
- Основные методы: SEARCH, INSERT, DELETE

Joseph M. Hellerstein, Jeffrey F. Naughton and Avi Pfeffer. Generalized Search Trees for Database Systems. 1995

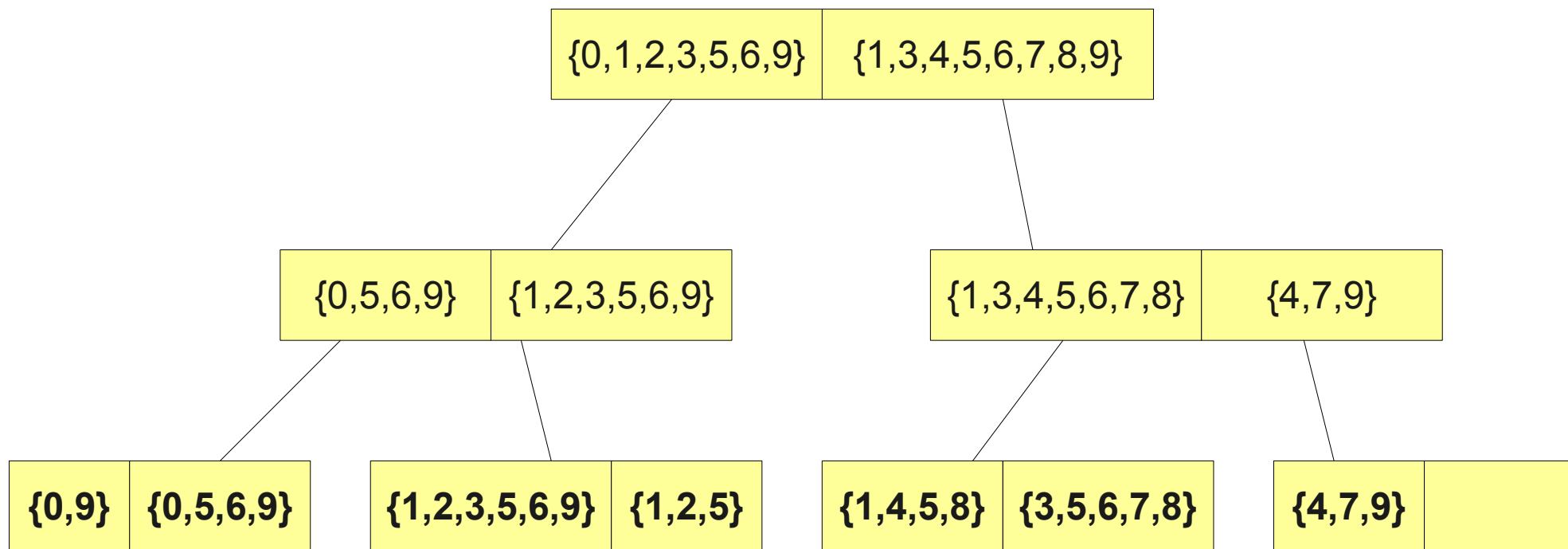
Использование GiST: 7 интерфейсных функций

1. `bool equal(Datum a, Datum b)`
2. `float * penalty(GISTENTRY *origentry, GISTENTRY *newentry, float *result)`
3. `Datum union(GistEntryVector *entryvec, int *size)`
4. `bool consistent(GISTENTRY *entry, Datum query, StrategyNumber strategy)`
5. `GIST_SPLITVEC * split(GistEntryVector *entryvec, GIST_SPLITVEC *v)`
6. `GISTENTRY * compress(GISTENTRY * in)`
7. `GISTENTRY * decompress(GISTENTRY * in)`

Олег Бартунов, Фёдор Сигаев. Написание расширений для PostgreSQL с использованием GiST. 2005.

<http://citforum.ru/database/postgres/gist/>

Применение GiST: RD-tree



Примеры: FTS, contrib/intarray, contrib/hstore, contrib/pg_trgm

Что такое GIN?

Логическая структура GIN для hstore

GIN's keys

KEY 'Standard'

KEY 'Localization'

VALUE 'GSM'

....

NULL

Posting lists (ItemPointer's sorted lists)

List of pointer to rows with HStore contains 'Standard' key

List of pointer to rows with HStore contains 'Locale' key

List of pointer to rows with HStore contains 'GSM' value

List of pointer to rows with HStore contains NULL value

GIN: 4 интерфейсных функции

- Datum* **extractValue**(Datum inputValue, uint32* nentries)
- int **compareEntry**(Datum a, Datum b)
- Datum* **extractQuery**(Datum query, uint32* nentries, StrategyNumber n)
- bool **consistent**(bool check[], StrategyNumber n, Datum query)

Реализация GIN в PostgreSQL

- Конкурентный доступ
- Восстановление после сбоев (WAL)
- Поддержка opclasses, определяемых пользователем
- Встроенная реализация для массивов (произвольных!), полнотекстового поиска
- B-trees: ET (entries tree), PT (posting tree; if needed)

Типы данных для работы
со слабоструктурированными
данными: массивы, `hstore`, `xml`

Типы данных: массивы

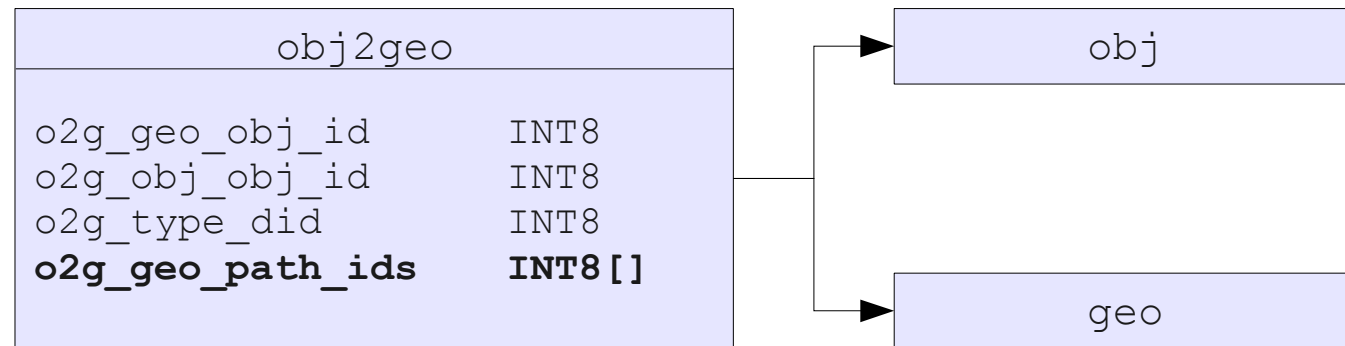
```
CREATE TABLE sal_emp (  
    name            text,  
    pay_by_quarter  integer[],  
    schedule        text[][]  
);
```

- Массивы для произвольных типов (начиная с 8.3: включая композитные типы, user-defined типы, enums): встроенная поддержка **GIN**
- contrib/intarray: **GiST** для int4[] и int8[]
- Произвольная длина (varlena), значение до 1GB
- Многомерные массивы

Использование массивов

(на примере <http://mirtesen.ru>)

- Денормализация



- Гибкая схема БД

person	
obj_id	INT8
obj_status_did	INT8
obj_created	INT8
person_name	VARCHAR(32)
person_email	VARCHAR(255)
...	
person_tags	INT8[]

+ GiST/GIN index

Типы данных: hstore

```
test=# select '"color"=>"red", "age"=>"21"'::hstore;  
          hstore
```

```
-----  
"age"=>"21", "color"=>"red"  
(1 row)
```

```
test=# select '"color"=>"red", "age"=>"21"'::hstore->'age';  
          ?column?
```

```
-----  
21  
(1 row)
```

```
test=# select akeys('a=>1,b=>2'::hstore);  
          akeys
```

```
-----  
{a,b}  
(1 row)
```

Функции и операторы `hstore`

- `akeys(hstore)`, `avals(hstore)` – returns keys/values as an array
- `skeys(hstore)`, `svals(hstore)` – returns keys/values as a set of strings
- `each(hstore)` – returns set of (key,value)
- `exist(hstore, text)`, `defined(hstore, text)` – similarly to Perl (undef is NULL)
- `hstore ? text` – equivalent for `exist()`
- `hstore @> hstore`, `hstore <@ hstore` – contains/contained operation
- `hstore || hstore` - concatenate

Логическая структура GIN для hstore

GIN's keys

KEY 'Standard'

KEY 'Localization'

VALUE 'GSM'

....

NULL

Posting lists (ItemPointer's sorted lists)

List of pointer to rows with HStore contains 'Standard' key

List of pointer to rows with HStore contains 'Locale' key

List of pointer to rows with HStore contains 'GSM' value

List of pointer to rows with HStore contains NULL value

Использование hstore

(на примере <http://mirtesen.ru>)

person	
obj_id	INT8
obj_status_did	INT8
obj_created	INT8
person_name	VARCHAR(32)
person_email	VARCHAR(255)
...	
person_data	hstore

Индексирование `hstore`

- **B-tree:** индексирование проекции

```
CREATE INDEX ... USING BTREE ((person_data->'age'));
```

Плюсы:

- Отличная производительность
- Простой вид запроса
- Запросы с операторами `<`, `<=`, `>`, `>=`
- Можно использовать для сортировки

Минусы:

- Для полноценной индексации требуется 2 индекса (по ключам, по значениям)
- Только один ключ. Для многих ключей — несколько индексов.

Индексирование `hstore`

- **GiST:** `CREATE INDEX ... USING GIST(person_data);`

Плюсы:

- Единый индекс
- Один `indexscan` для обработки нескольких выражений сразу

Минусы:

- При использовании сигнатур (RD-tree) требуется перепроверка найденных значений (`false drops problem`)
- «Скромная» производительность в общем случае
- Только для операции равенства

Индексирование `hstore`

- **GIN:** `CREATE INDEX ... USING GIN(person_data);`

Плюсы:

- Единый индекс
- Один `indexscan` для обработки нескольких выражений сразу
- Очень быстрая операция `exist`
- Отличная масштабируемость

Минусы:

- Требуется перепроверка найденных значений (`false drops problem`) для операции `contains` (но реже, чем в `GiST`)
- Только для операции равенства
- Относительно «медленные» `INSERTs` & `UPDATEs`
- Нет поддержки многоколоночных индексов

PostgreSQL XML type: логический уровень

- Google Summer of Code (лето 2006) + доработка (2006-2007), Nikolay Samokhvalov & Peter Eisentraut
 - XML type (over VARCHAR)
 - SQL/XML publishing functions
 - DTD validation
 - XPath: xpath(), with namespaces support
 - XML simple export functions
- Кроме этого (не требует изменения ядра):
 - XPath functional indexes
 - Full-text search (w/o structure awareness)

Созданные патчи приняты, включены в PostgreSQL 8.3 (сейчас beta2) и постоянно дорабатываются

Samokhvalov, N. XML Support in PostgreSQL. /Spring Young Researcher's Colloquium On Database and Information Systems, SYRCoDIS Moscow, Russia, 2007

Работа над XML-типом в PostgreSQL: Roadmap

логический уровень СУБД
XML type, SQL/XML, DTD,
XPath, functional XPath indexes, GIN

расширение возможностей
XSLT, XSD, RelaxNG, Annotated
schemas decomposition,
advanced FTS, XQuery
...

производительность
?

Производительность xml: задачи

- Выборка всего XML-значения по условию XPath

```
SELECT xmlcol
FROM tbl1
WHERE (xpath('/a/@b', xmlcol))[1] = '123';
```

```
SELECT xmlcol
FROM tbl1
WHERE (xpath('/a[@c="s"]/@b', xmlcol))[1] = '123';
```

- Выборка произвольной части XML-значения

```
SELECT xpath('/a/@b', xmlcol)
FROM tbl1
WHERE id = 111;
```

- Модификация части XML-значения

Индексирование XML-данных (осень 2007)

- B-tree (для выражений, результатом которых является скаляр)
- GIN
- Полнотекстовый индекс (без учёта структуры)

Производительность: базовые идеи

- Cache intermediate results to avoid redundant parsing and XPath evaluation
- Advanced physical storage to speedup access to arbitrary node in XML data
- Use PostgreSQL existing capabilities for full-text search
- Use additional structures/tables/indexes to avoid XPath evaluation at runtime
- Use slices (similar to `array_extract_slice()`) to avoid dealing with entire values (both in `SELECTs` and `UPDATEs`)

Развитие структуры хранения

- 4 bytes: VARLENA header bytes
- 1st TOAST slice (2kb): structure descriptor
- Other slices: raw data

1st TOAST slice – {pathID, startAddr, endAddr}

pathID – short ID of XML path (all paths are stored in database-wide storage table)

startAddr & endAddr – relative addresses of XML element edges, bytes

Развитие структуры хранения

Advantages:

- can skip useless TOAST slices
- can use beginAddr, endAddr as a simple numbering scheme => use GiST/GIN to index

Изучение областей применимости XML и hstore

- Системы документооборота (XML)
- Электронные библиотеки (XML)
- Архивы документов (XML)
- Журнализация операций, регистрация событий (XML, hstore)
- Неоднородный каталог объектов (XML, hstore)
- Прототипирование ПО (XML, hstore)
- Коллектор данных (XML)

Пример 1: журнал действий

Цель: гибкость схемы
(возможность быстро добавлять новые свойства)

action	
action_id	SERIAL
action_type_id	INT4
action_status_id	INT4
action_person_id	INT4
action_data	hstore

Пример 2: каталог разнородных объектов

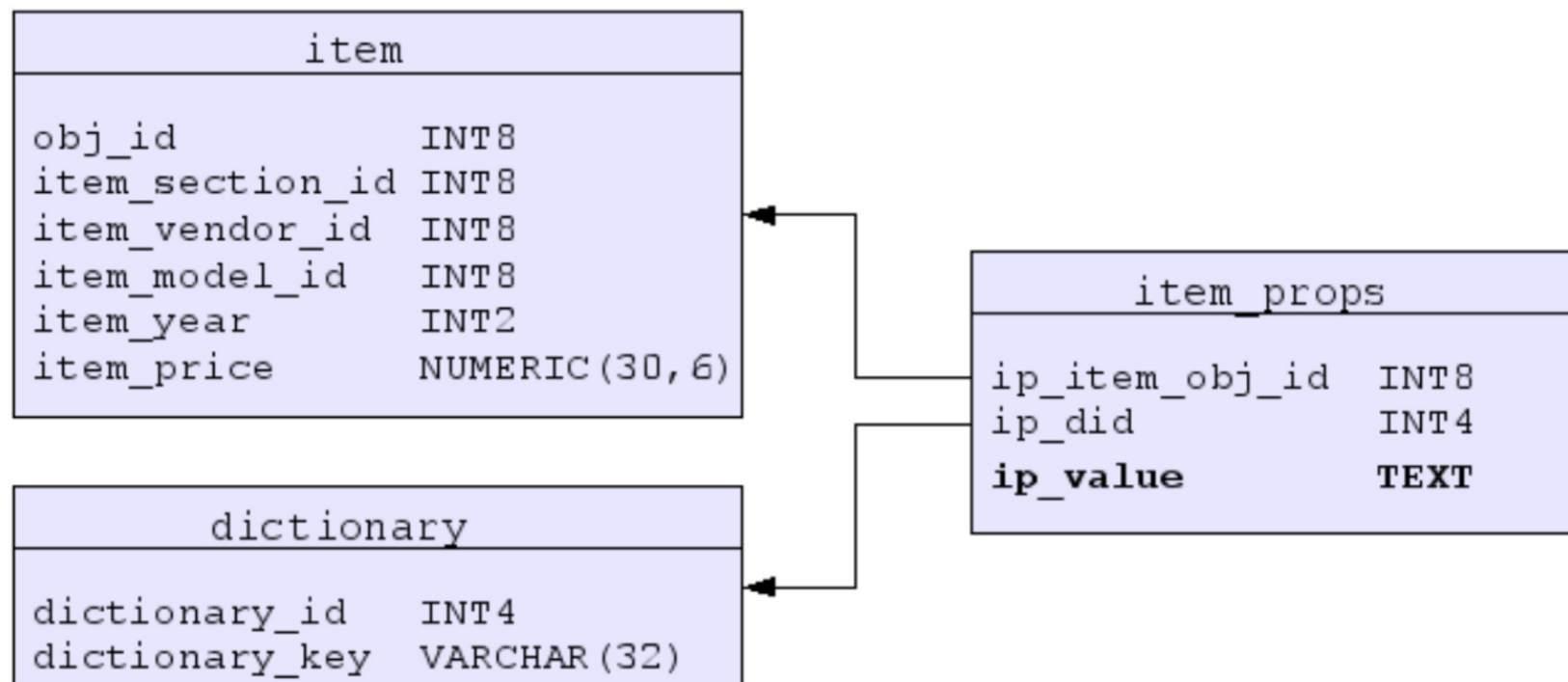
Цель: работа с объектами с различными наборами свойств

АВТО [+ Подать объявления в этот раздел](#)

Кредиты и ссуды (22)	Комиссионное оформление и страхование (5)
Легковые автомобили (35679)	Прицепы, автодома и фургоны (146)
Новые автомобили	Автобусы (231)
Автомобили с пробегом	Спецтехника (1426)
Малый коммерческий транспорт (2320)	Мотоциклы и мопеды (1505)
Средние и тяжелые грузовики (2039)	Снегоходы (37)
Грузовые прицепы (508)	Автосервис и услуги (780)
Водный транспорт (365)	Сервис и ремонт
Автозапчасти и принадлежности (10507)	Экспертиза и оценка
Для легковых автомобилей	Юридическая помощь
Для коммерческих автомобилей	Автошколы
Для мотоциклов и мопедов	Другое
Диски, шины, колеса	
Для спецтехники и прицепов	
Авто на запчасти	
Средства для ухода	
Другое	
Другое (400)	

IRR.ru, раздел «Авто»

Пример 2: разнородный каталог



Каталог разнородных данных: 1-й способ реализации
(Entity-Attribute-Value, EAV)

Пример 2: разнородный каталог

item	
obj_id	INT8
item_section_id	INT8
item_vendor_id	INT8
item_model_id	INT8
item_year	INT2
item_price	NUMERIC(30,6)
item_prop1	INT4
item_prop2	INT4
item_prop3	INT4
item_prop4	INT4
...	
item_prop21	TEXT
item_prop22	TEXT
item_prop23	TEXT
...	
item_prop41	BOOLEAN
...	

Каталог разнородных данных: 2-й способ реализации («широкая таблица»)

Пример 2: разнородный каталог

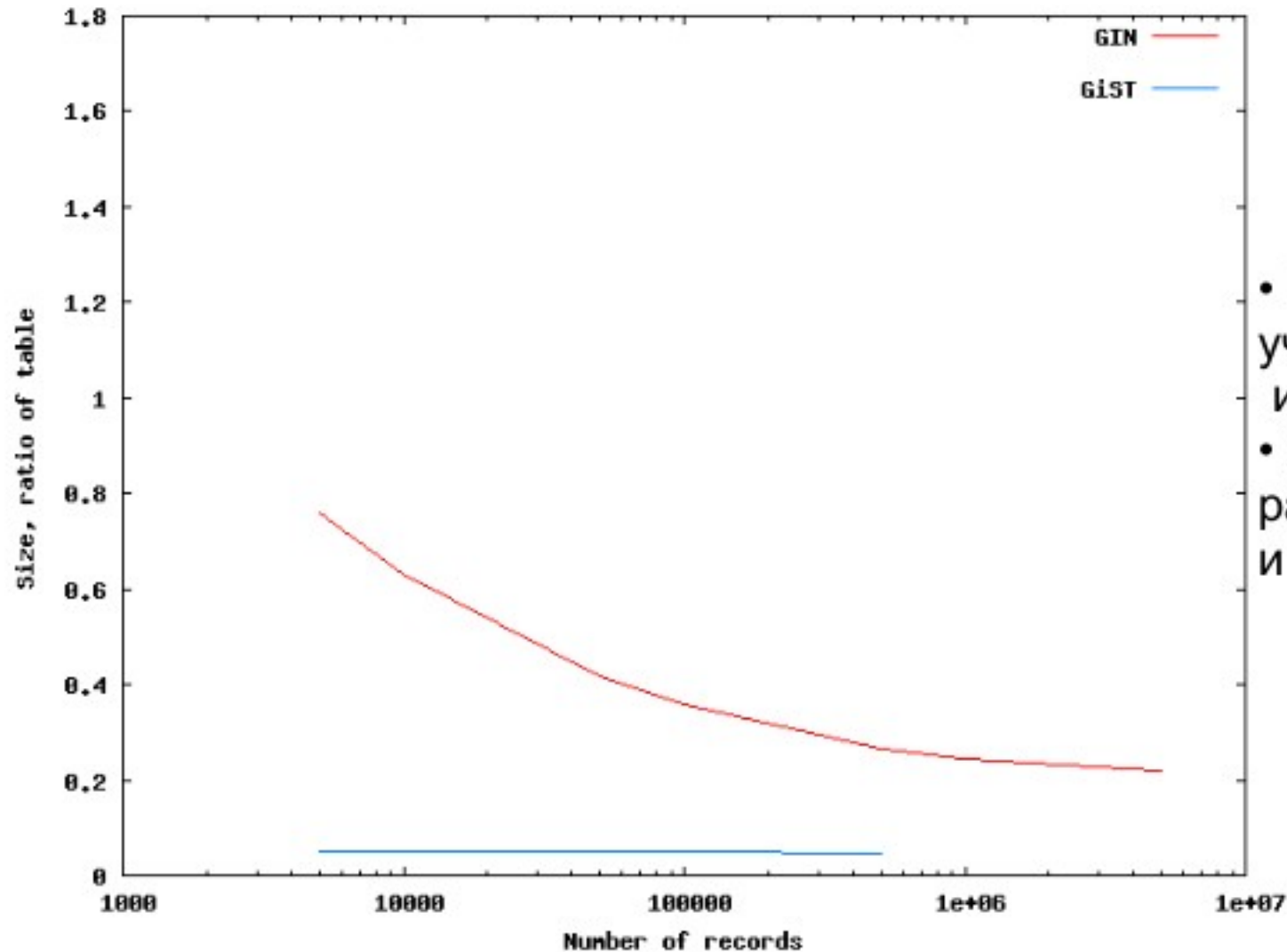
item	
obj_id	INT8
item_section_id	INT8
item_vendor_id	INT8
item_model_id	INT8
item_year	INT2
item_price	NUMERIC(30, 6)
item_props	XML

Каталог разнородных данных: 3-й способ реализации (XML, hstore)

Benchmarkmarks

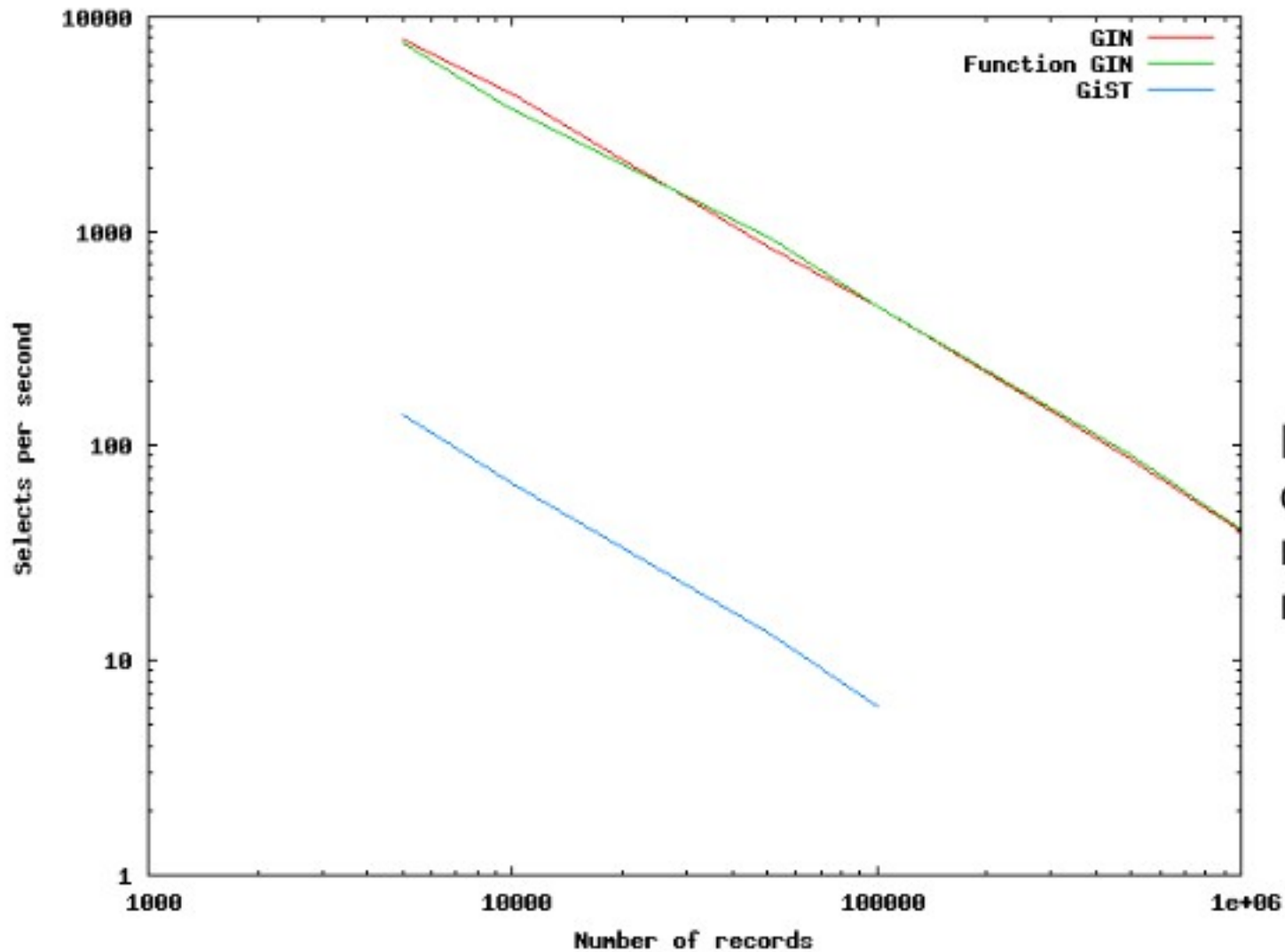
GiST vs GIN

Фёдор Сигаев, Highload-2007



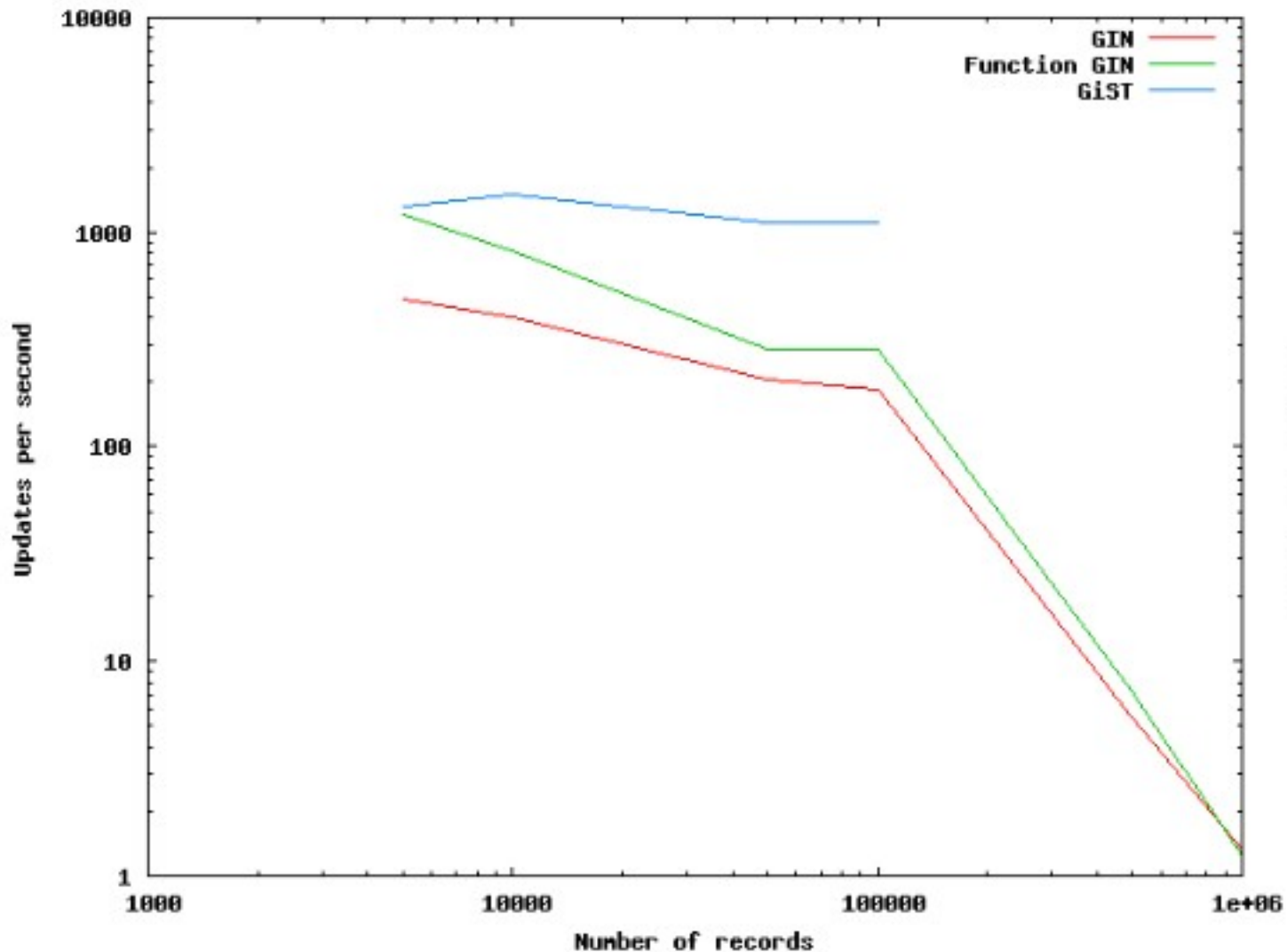
- Размер таблицы учитывает pg_toast и pg_toast_index
- Указано отношение размеров индекса и таблицы

GiST vs GIN: SELECT



Производительность
обоих индексов обратно
пропорциональна
количеству записей.

GiST vs GIN: UPDATE



Производительность GIN при вставке быстро падает из-за большого кол-ва записей в индексе

EAV VS *wide table* VS hstore

на примере данных <http://domnakarte.ru>, 100 тыс. объектов недвижимости;
ноутбук Pentium-M 1.86GHz, 1GB;
shared_buffers = 250MB
work_mem = 20MB

	«широкая таблица»	EAV	hstore (GiST)	hstore (GIN)
Данные, MB	52	231	138	138
Индексов+TOASTed, MB	22	135	6.2	57
Создание индекса, ms	29871	289043	79137	278000
Простая выборка по условию, ms	1.6 / 202*	6.3 / 368*	79 / ~3000*	48 / 130*
Вставка, ms	3.4	5	4.8	8

*) Что делать? Рецепт для web-приложений: «Поднимаем» файлы в файловый кэш ОС:

```
dd ... of=/dev/null
```

Спасибо!

- <http://nikolay.samokhvalov.com>
- <http://postgresmen.ru>
- <http://postgresql.org>
- ns@postgresmen.ru