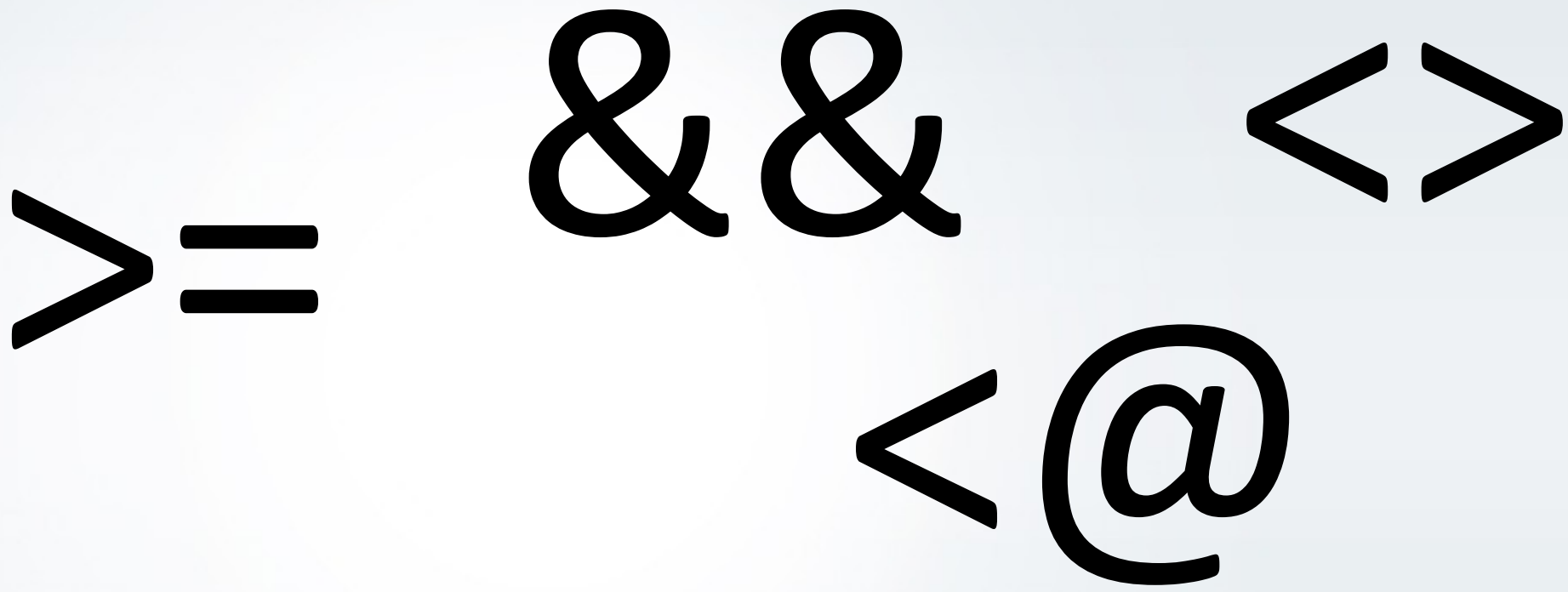


I know greater-than-or-equal-to when I see it!

Noah Misch | 2014-05-22

Layers of Index Support

- Index access methods (`pg_am`)
  - Type-independent; specific to certain index layout
  - `btree, hash, gist, gin, spgist`
- Operator classes (`pg_opclass`)
  - Specific to a data type + index access method
  - Tightly related: operator families (`pg_opfamily`)
  - `int4_ops, text_ops`

What is an operator class?

- In general: ties a data type to an access method
- The case of btree: comparison function and operators

```
CREATE TABLE t (c date PRIMARY KEY);
INSERT INTO t VALUES ('2014-01-01');
INSERT INTO t VALUES ('2015-01-01');
...
-- <(date,date) operator
SELECT * FROM t WHERE c < current_date;
```

What is an operator family?

- Extends operator support to multiple data types
- Relevant for btree and hash only

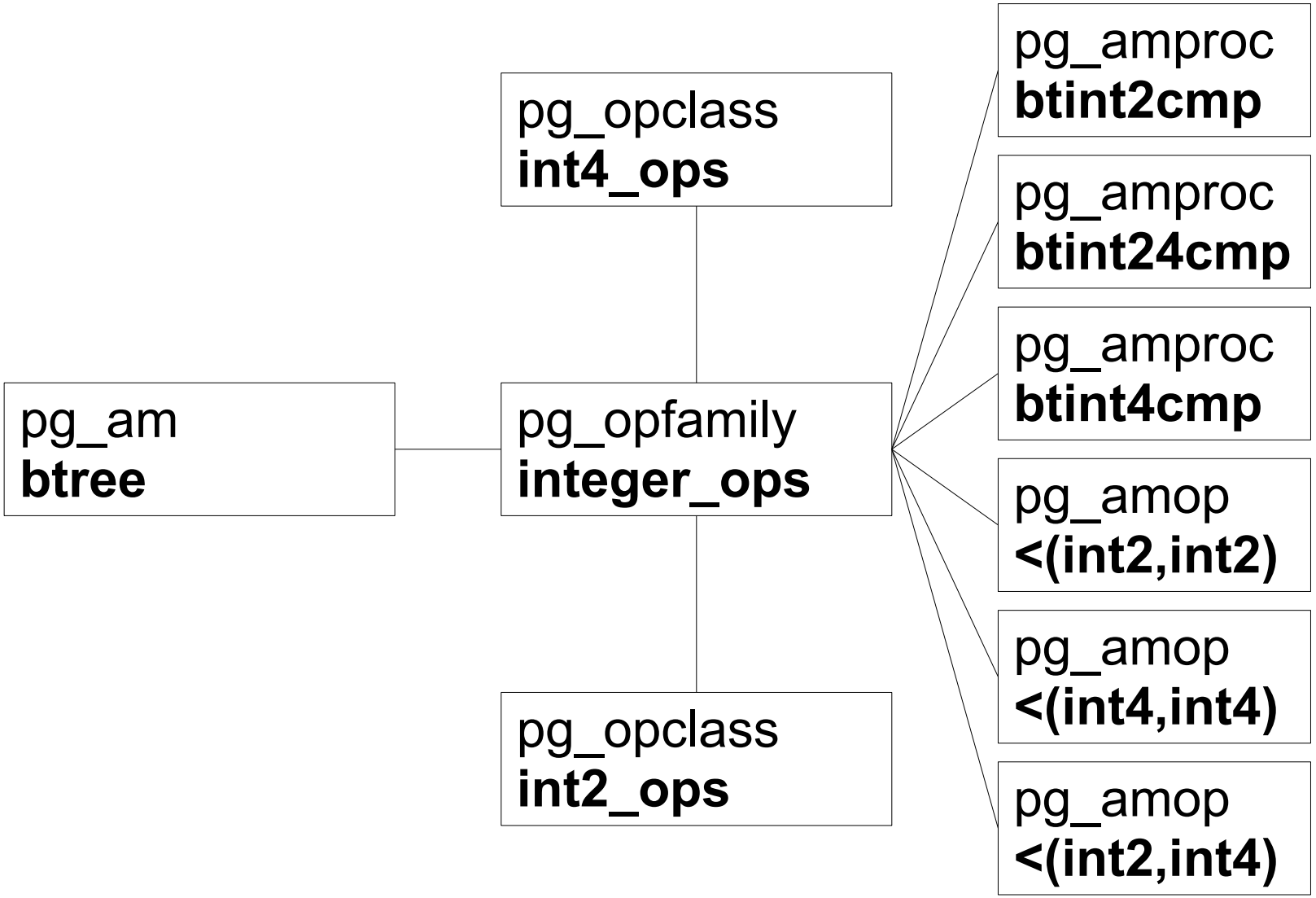
```
CREATE TABLE t (c date PRIMARY KEY);
INSERT INTO t VALUES ('2014-01-01');
INSERT INTO t VALUES ('2015-01-01');
...
-- <(date,timestamptz) operator
SELECT * FROM t WHERE c < now();
```

btree int4_ops walk-through

- `FUNCTION` entries maintain the index
- List of `OPERATOR` qualified to exploit the index
- “equal-sign operator” vs. “equality operator”

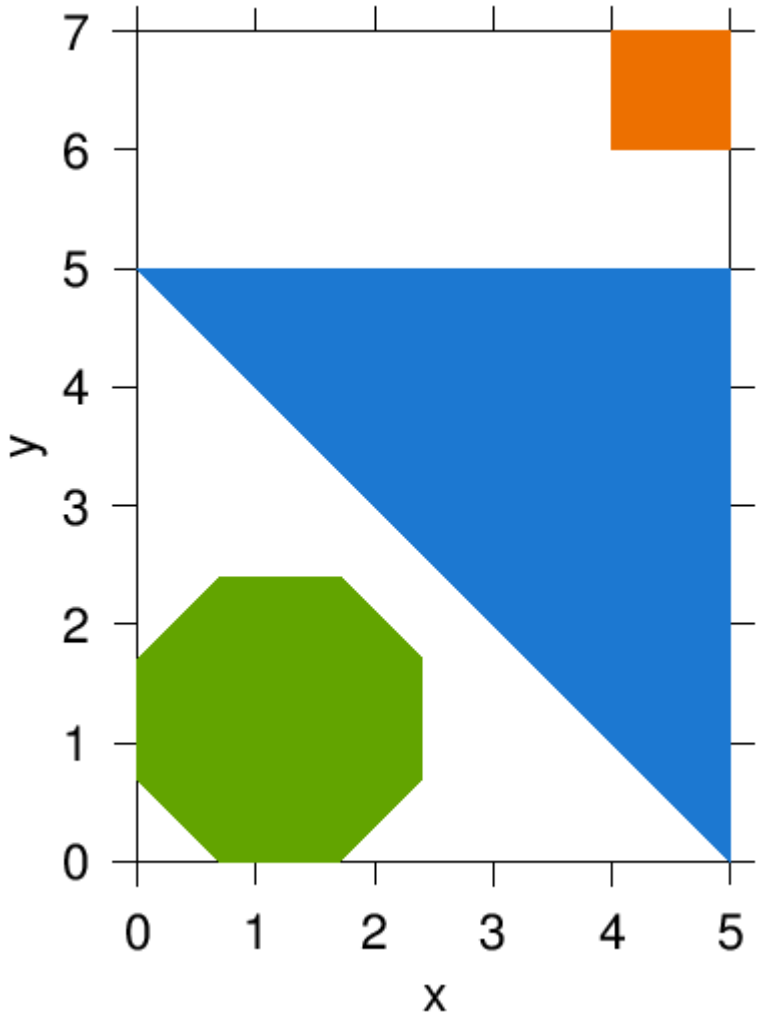
```
CREATE OPERATOR FAMILY integer_ops USING btree;
CREATE OPERATOR CLASS int4_ops
 DEFAULT FOR TYPE integer
    USING btree FAMILY integer_ops AS
 FUNCTION 1 btint4cmp(integer, integer),
 OPERATOR 1 <,
 OPERATOR 2 <=,
 OPERATOR 3 =,
 OPERATOR 4 >=,
 OPERATOR 5 >;
```

System Catalog Representation



Multiple Operator Classes

- `btree`: other sort orders
  - `text_pattern_ops`
- `hash`: not done in practice
- `gin`, `gist`, `spgist`: fruitful opportunities



ORDER BY

```
-- uses btree text_ops
ORDER BY textcol;

-- uses btree text_pattern_ops
ORDER BY textcol USING ~<~;

-- can use e.g. gist_trgm_ops
ORDER BY textcol <-> 'search condition';
```

Equality

- `UNION`
- `GROUP BY, DISTINCT`
- array, composite type comparisons
- Choice of default equality semantics is important

```
[local] test=# SELECT DISTINCT x
FROM unnest(array[1.00, 1.1, 1.0]) t(x);
  x
------
  1.1
 1.00
(2 rows)
```

Equality Surprises

- Operator names like “=” and “<” are not special ...
- … excepting `CASE`, `IN`, `IS DISTINCT FROM`, etc

```
[local] test=# SELECT DISTINCT x
FROM unnest(array['(1,1),(0,0)',
'(2,2),(1,1)']::box[]) t(x);
ERROR:  could not identify an equality
operator for type box
[local] test=# SELECT '(1,1),(0,0)'::box IN
('(2,2),(1,1)'::box);
 ?column?
----------
 t
```

Merge Join

```
[local] test=# SET enable_hashjoin = off;
SET
[local] test=# EXPLAIN (costs off)
SELECT opfmethod, opfname, array_agg(amopopr)
FROM pg_amop ao JOIN pg_opfamily f
ON amopfamily = f.oid GROUP BY 1,2;
                  QUERY PLAN
```

```
 HashAggregate
   Group Key: f.opfmethod, f.opfname
   ->  Merge Join
         Merge Cond: (f.oid = ao.amopfamily)
         ->  Sort
               Sort Key: f.oid
               ->  Seq Scan on pg_opfamily f
         ->  Sort
               Sort Key: ao.amopfamily
               ->  Seq Scan on pg_amop ao
```

Hash Join

```
[local] test=# EXPLAIN (costs off)
SELECT opfmethod, opfname, array_agg(amopopr)
FROM pg_amop ao JOIN pg_opfamily f
ON amopfamily = f.oid GROUP BY 1,2;
                    QUERY PLAN
```

```
 HashAggregate
   Group Key: f.opfmethod, f.opfname
   ->  Hash Join
         Hash Cond: (ao.amopfamily = f.oid)
         ->  Seq Scan on pg_amop ao
         ->  Hash
               ->  Seq Scan on pg_opfamily f
```

Writing Generic Data Type Consumers

- Don't hard-code “=”
- Which equality semantics?
  - btree/hash default equality
  - exact match (output comparison; `record_image_ops`)
- Do look up equality by operator class
  - backend: `TYPECACHE_EQ_OPR`
  - frontend: copy its algorithm
- Not all types have these operations

Implementing Data Types

- Choice of default equality semantics is important
  - Option to omit them entirely (`xml`, `json`, `box`)
- Try to include a default btree operator class
- Default hash operator class is then easy
- Other access methods are situation-specific
  - gin for container-like types
  - gist often starts with the search strategy, not the type

Questions?

Further Reading

- http://www.postgresql.org/docs/current/static/xindex.html
- `contrib/btree_gist`, `contrib/btree_gin`
- Other built-in and `contrib` operator classes
- `ATAddForeignKeyConstraint()`

hash int4_ops

```
CREATE OPERATOR CLASS int4_ops
 DEFAULT FOR TYPE integer
    USING hash FAMILY integer_ops AS
 FUNCTION 1 hashint4(integer),
 OPERATOR 1 =;
```

Array Element Searches: gin _int4_ops

```
CREATE OPERATOR CLASS _int4_ops
 DEFAULT FOR TYPE integer[]
    USING gin FAMILY array_ops AS
 STORAGE integer,
 FUNCTION 1 btint4cmp(integer,integer),
 FUNCTION 2 ginarrayextract(...),
 FUNCTION 3 ginqueryarrayextract(...),
 FUNCTION 4 ginarrayconsistent(...),
 OPERATOR 1 &&(anyarray,anyarray),
 OPERATOR 2 @>(anyarray,anyarray),
 OPERATOR 3 <@(anyarray,anyarray),
 OPERATOR 4 =(anyarray,anyarray);
```