Range Types
In Your Application

Jeff Davis
pgsql@j-davis.com

Goals

- Improved application functionality
- Better Performance
- Easier to use and less error-prone

Quick Introduction

- A Range Type represents a range of an ordinary type
- NUMRANGE: range of NUMERICs
- DATERANGE: range of DATEs
- TSTZRANGE: range of TIMESTAMPTZs

What is a Range?

- “1pm until 4pm today” is a range
- “3.1 – 7.7” is a range
- “192.168.1.10 through .20” is a range
- Can be discrete
  - INTRANGE, DATERANGE
- Or continuous
  - TSTZRANGE, NUMRANGE

Functions/Operators

- Contains “@>”
- Overlaps “&&”
- Intersection “*”
- Union “+”
- Many more...

Example

```
SELECT contains(
   range(1.7, 90.1),
   3.3 -- scalar
);
-- returns TRUE

SELECT overlaps(
   '[-2, -1]'::numrange,
   range(6.2) -- singleton range
);
-- returns FALSE
```

Inclusive/Exclusive Bounds

- Does '[1.1, 2.2)' include the point 2.2?
- “[“ and “]” mean “inclusive”
- And “(“ and “)” mean “exclusive”
- Answer: No.
- Range(1.1, 2.2) constructor function uses inclusive-exclusive form
  - Other constructors exist

Scheduling Example - Schema

```
CREATE TABLE reservation
(
  user_id TEXT,
  room_id INT,
  during DATERANGE
);
```

Scheduling Example - Code

```
import psycopg2
conn = psycopg2.connect(
  'host=/tmp dbname=postgres user=jdavis')
cur = conn.cursor()
cur.execute('''
    INSERT INTO reservation
    VALUES(%s, %s, %s)
  ''',
  ('bill', 456, '[2013-04-07, 2013-04-10)'))

# ...
```

Scheduling Example - Code

```
cur.execute('''
    SELECT
      user_id, room_id,
      lower(during),upper(during)
    FROM reservation
  ''')

print(cur.fetchone())
```

Problem: Overlapping Reservations

- What if two people try to reserve the same room for overlapping dates?
- If the range was identical, we could use UNIQUE
- But for overlapping, we need something better.
- Ideas?

Solution: Overlapping Reservations

```
CREATE EXTENSION btree_gist;

ALTER TABLE reservation ADD
  EXCLUDE USING gist
    (room_id WITH =,
     during  WITH &&);
```

Solution Continued

- Should also prevent users from reserving different rooms for overlapping dates
  - Can't be in two places at once
- Solution is similar

Queries - DEMO

- Which rooms are occupied on April 10th, 2013?
- Which users are present at the same time as Bill?
- How many total room-days are reserved?

Compare to non-range queries

DEMO

Conclusion

- Don't constrain yourself to representing individual points only
  - Especially not when it comes to time!
- Simplify queries and schema
- Solve the “non-overlapping” problem
  - Especially for scheduling!
- Benefit from range indexing