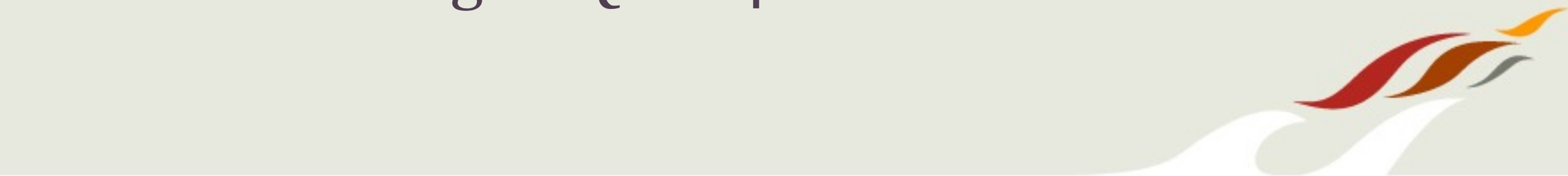
PostgreSQL 8.4 / Prolog

- Want to play along?
- PostgreSQL 8.4
  - git clone git://git.postgresql.org/git/postgresql.git
  - http://www.postgresql.org/docs/current/static/anoncvs.html
  - ./configure; make; make install;
- Pagila Sample Database
  - http://pgfoundry.org/frs/?group_id=1000150&release_id=998
- Alternatively
  - http://omniti.com/is/hiring

OmniTI

No More Waiting

A Guide to PostgreSQL 8.4



OmniTI / Presentation / Robert Treat

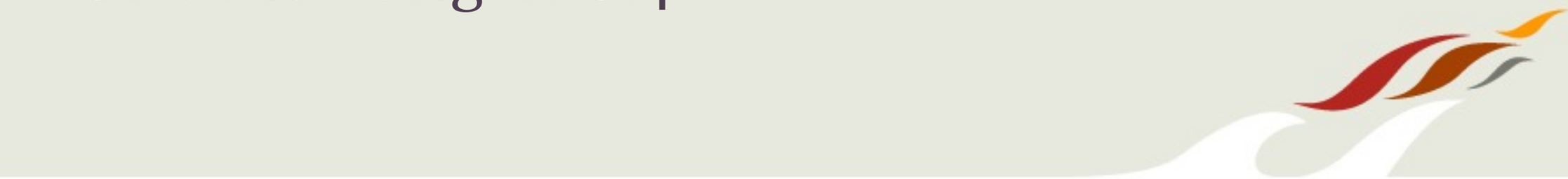
Still A Little More Waiting

A Guide to PostgreSQL 8.4

Presentation / Robert Treat

Still A Little More Waiting

A Guide to Postgres 8.4



Presentation / Robert Treat

Postgres 8.4

- Performance
- Administration
- PSQL
- Development
- Procedures

BUT FIRST!

Postgres 8.4 / Recap

Postgres 8.4 / Recap

- 8.3 Released (2008-02-04)

Postgres 8.4 / Recap

- 8.3 Released (2008-02-04)
- Commit Fests (March,May,July,September)

Postgres 8.4 / Recap

- 8.3 Released (2008-02-04)
- Commit Fests (March,May,July,September)
- Feature Freeze (November)

Postgres 8.4 / Recap

- 8.3 Released (2008-02-04)
- Commit Fests (March,May,July,September)
- Feature Freeze (November)
- Exodus (December, January, February)

Postgres 8.4 / Recap

- 8.3 Released (2008-02-04)
- Commit Fests (March,May,July,September)
- Feature Freeze (November)
- Exodus (December, January, February)
- Beta (April)

Postgres 8.4 / Recap

- 8.3 Released (2008-02-04)
- Commit Fests (March,May,July,September)
- Feature Freeze (November)
- Exodus (December, January, February)
- Beta (April)
- Release (June? July!)

Postgres 8.4

- Performance
- Administration
- PSQL
- Development
- Procedures

Postgres 8.4 / Perf / Visibility Maps

- Table = 1 or more files
- File -> 8K pages
- Keep track of pages that changed!
- Improved Vacuum Performance
  - reduced cpu
  - reduced i/o

Postgres 8.4 / Perf / Visibility Maps

- Table = 1 or more files
- File -> 8K pages
- Keep track of pages that changed!
- Improved Vacuum Performance
  - reduced cpu
  - reduced i/o
    - Selena Deckelmann - Friday @ 11:30 - Vacuum Strategy

OmniTI

Postgres 8.4 / Perf / Default Stats Target

Postgres 8.4 / Perf / Default Stats Target

- default_statistics_target = 100
- removed from tuning guide!

Postgres 8.4 / Perf / Default Stats Target

- default_statistics_target = 100
- removed from tuning guide!
- recent benchmarks have shown trouble :-(

Postgres 8.4 / Perf / Wait, there's more!

Postgres 8.4 / Perf / Wait, there's more!

- improved optimizer statistics calculations
- improved statistics for full-text columns
- new semi-join executor method
- new anti-join executor method
- improve performance of text_position()
- reduced i/o frequency of stats info
- constraint_exclusion = 'partition' option
- i/o read-ahead for bitmap scans
- Hash Methods for DISTINCT / UNION / INTERSECT / EXCEPTION
- New GUC - cursor_tuple_fraction
- improved build speed for hash indexes
- improved access speed for has indexes
- reduced memory for trigger handling
- pg_restore -j (multi-workers)

OmniTI

Postgres 8.4 / Perf / Wait, there's more!

Postgres 8.4

- Performance
- Administration
- PSQL
- Development
- Procedures

OmniTI

Postgres 8.4 / Admin / Free Space Map

- New free-space-map implementation
- No more shared memory, now kept on disk
  - remove GUC max_fsm_pages (former tuning guide option)
  - remove GUC max_fsm_relations (former tuning guide option)
- add GUC vacuum_freeze_table_age
  - ignore visibility map, scan whole table
    - advanced relfrozenxid

OmniTI

Postgres 8.4 / Admin / Termination

- 8.0 -> select pg_cancel_backend(pid)
- 8.4 -> select pg_terminate_backend(pid)

Postgres 8.4 / Admin / Stats Temp File

- pgstat.stat, stores statistics information
- hi i/o load
  - very busy servers
  - large schema size
- NEW GUC stats_temp_directory
  - point it at swap/memory filesystem

Postgres 8.4 / Admin / Column Privileges

- You can now grant access to specific columns

```
pagila=# grant select (first_name, last_name) on actor_info to amber;
GRANT
pagila=# set session authorization amber;
SET
pagila=> select * from actor_info;
ERROR:  permission denied for relation actor_info
STATEMENT:  select * from actor_info;
ERROR:  permission denied for relation actor_info
pagila=> select first_name, last_name from actor_info limit 2;
 first_name | last_name
------------+-----------
 PENELOPE   | GUINESS
 NICK       | WAHLBERG
(2 rows)
pagila=> \z actor_info
                            Access privileges
 Schema |    Name    | Type | Access privileges | Column access privileges
--------+------------+------+-------------------+--------------------------
 public | actor_info | view |                   | first_name:
                                                :   amber=r/postgres
                                                : last_name:
                                                :   amber=r/postgres
```

Postgres 8.4 / Admin / Column Privileges

- You can now grant access to specific columns

```
pagila=# grant select (first_name, last_name) on actor_info to amber;
GRANT
pagila=# set session authorization amber;
SET
pagila=> select * from actor_info;
ERROR:  permission denied for relation actor_info
STATEMENT:  select * from actor_info;
ERROR:  permission denied for relation actor_info
pagila=> select first_name, last_name from actor_info limit 2;
 first_name | last_name
------------+-----------
 PENELOPE   | GUINESS
 NICK       | WAHLBERG
(2 rows)
pagila=> \z actor_info
                              Access privileges
 Schema |    Name    | Type | Access privileges | Column access privileges
--------+------------+------+-------------------+--------------------------
 public | actor_info | view |                   | first_name:
                                                :   amber=r/postgres
                                                : last_name:
                                                :   amber=r/postgres
```

- Stephen Frost - You Missed It! - Postgres Access Controls

OmniTI

Postgres 8.4 / Admin / Statement Stats

- Contrib Module, track statistics
- load shared library in postgresql.conf, load schema in database
- tracks per user, database, query

```
pagila=# select pg_stat_statements_reset();
 pg_stat_statements_reset
--------------------------

(1 row)

pagila=# select * from pg_stat_statements;
 userid | dbid  |                query                | calls | total_time | rows
--------+-------+-------------------------------------+-------+------------+------
     10 | 16384 | select pg_stat_statements_reset(); |     1 |   0.000113 |    1
(1 row)
```

OmniTI

Postgres 8.4 / Admin / Auto Explain

- Contrib Module, explain plan for slow queries
- Can be loaded per session, or in postgresql.conf

```
pagila=# load 'auto_explain';
LOAD
pagila=# set auto_explain.log_min_duration = 2000;
SET
pagila=# select pg_sleep(3);
pg_sleep
----------

(1 row)

~~
LOG:  duration: 3000.196 ms  plan:
        Result  (cost=0.00..0.01 rows=1 width=0)
STATEMENT:  select pg_sleep(3);
```

Postgres 8.4 / Admin / Timed pgbench

- Contrib Module, simple benchmarking tool
- can now limit by time, rather than transactions
- Greg Smith - Thurs @ 1:30 - Database Hardware Benchmarking

```
robert$ bin/pgbench -p 5484 -T 60 pagila
starting vacuum...end.
transaction type: TPC-B (sort of)
scaling factor: 1
query mode: simple
number of clients: 1
duration: 60 s
number of transactions actually processed: 30774
tps = 512.908446 (including connections establishing)
tps = 512.956459 (excluding connections establishing)
```

OmniTI

Postgres 8.4 / Admin / Explain Verbose

- Now shows column output

```
pagila=# explain analyze verbose select first_name, last_name, title from film_actor join actor using (actor_id) join film
using (film_id);
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=83.00..314.83 rows=5462 width=28) (actual time=1.311..9.809 rows=5462 loops=1)
   Output: actor.first_name, actor.last_name, film.title
   Hash Cond: (film_actor.actor_id = actor.actor_id)
   ->  Hash Join  (cost=76.50..233.22 rows=5462 width=17) (actual time=1.065..5.818 rows=5462 loops=1)
         Output: film_actor.actor_id, film.title
         Hash Cond: (film_actor.film_id = film.film_id)
         ->  Seq Scan on film_actor  (cost=0.00..81.62 rows=5462 width=4) (actual time=0.008..1.082 rows=5462 loops=1)
               Output: film_actor.actor_id, film_actor.film_id, film_actor.last_update
         ->  Hash  (cost=64.00..64.00 rows=1000 width=19) (actual time=1.046..1.046 rows=1000 loops=1)
               Output: film.title, film.film_id
               ->  Seq Scan on film  (cost=0.00..64.00 rows=1000 width=19) (actual time=0.003..0.488 rows=1000 loops=1)
                     Output: film.title, film.film_id
   ->  Hash  (cost=4.00..4.00 rows=200 width=17) (actual time=0.238..0.238 rows=200 loops=1)
         Output: actor.first_name, actor.last_name, actor.actor_id
         ->  Seq Scan on actor  (cost=0.00..4.00 rows=200 width=17) (actual time=0.011..0.072 rows=200 loops=1)
               Output: actor.first_name, actor.last_name, actor.actor_id
```

OmniTI

Postgres 8.4 / Admin / Explain Verbose

- See, select * is evil :-)

```
pagila=# explain analyze verbose select * from film_actor join actor using (actor_id) join film using (film_id);
     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=83.00..314.83 rows=5462 width=423) (actual time=1.974..15.409 rows=5462 loops=1)
   Output: film.film_id, actor.actor_id, film_actor.last_update, actor.first_name, actor.last_name, actor.last_update,
film.title, film.description, film.release_year, film.language_id, film.original_language_id, film.rental_duration, film.rental_rate,
film.length, film.replacement_cost, film.rating, film.last_update, film.special_features, film.fulltext
   Hash Cond: (film_actor.actor_id = actor.actor_id)
   ->  Hash Join  (cost=76.50..233.22 rows=5462 width=400) (actual time=1.806..9.730 rows=5462 loops=1)
         Output: film_actor.last_update, film_actor.actor_id, film.film_id, film.title, film.description, film.release_year,
film.language_id, film.original_language_id, film.rental_duration, film.rental_rate, film.length, film.replacement_cost,
film.rating, film.last_update, film.special_features, film.fulltext
         Hash Cond: (film_actor.film_id = film.film_id)
         ->  Seq Scan on film_actor  (cost=0.00..81.62 rows=5462 width=12) (actual time=0.007..1.263 rows=5462 loops=1)
               Output: film_actor.actor_id, film_actor.film_id, film_actor.last_update
         ->  Hash  (cost=64.00..64.00 rows=1000 width=390) (actual time=1.787..1.787 rows=1000 loops=1)
               Output: film.film_id, film.title, film.description, film.release_year, film.language_id, film.original_language_id,
film.rental_duration, film.rental_rate, film.length, film.replacement_cost, film.rating, film.last_update, film.special_features,
film.fulltext
               ->  Seq Scan on film  (cost=0.00..64.00 rows=1000 width=390) (actual time=0.003..0.429 rows=1000 loops=1)
                     Output: film.film_id, film.title, film.description, film.release_year, film.language_id, film.original_language_id,
film.rental_duration, film.rental_rate, film.length, film.replacement_cost, film.rating, film.last_update, film.special_features,
film.fulltext
   ->  Hash  (cost=4.00..4.00 rows=200 width=25) (actual time=0.157..0.157 rows=200 loops=1)
         Output: actor.actor_id, actor.first_name, actor.last_name, actor.last_update
         ->  Seq Scan on actor  (cost=0.00..4.00 rows=200 width=25) (actual time=0.009..0.049 rows=200 loops=1)
               Output: actor.actor_id, actor.first_name, actor.last_name, actor.last_update
```

OmniTI

Postgres 8.4 / Admin / Wait, there's more!

Postgres 8.4 / Admin / Wait, there's more!

- report all queries involved in a deadlock error
- Database Level LC_Collate/ LC_Type
- ALTER SEQUENCE RESTART command
- TRUNCATE TABLE RESTART IDENTITY
- multi-column GIN indexes
- pg_conf_load_time() function
- Add columns to end of view with ALTER VIEW
- GUC track_activity_query_size
- Improved support for krb5, gssapi, sspi
- Read only GUC segment_size
- Read only GUC wal_block_size
- Read only GUC wal_segment_size
- Allow static logfile name
- Track Function Stats
- Pre-Parse pg_hba.conf before reload
- improved SSL certificate handling
- pg_settings now shows available options for guc with defined set

OmniTI

Postgres 8.4 / Admin / Wait, there's more!

Postgres 8.4

- Performance
- Administration
- PSQL
- Development
- Procedures

Postgres 8.4 / PSQL / Function Editing

- \df function_name
- \df+ function_name
  - shows function definition
- \ef function_name
  - opens fully formed function definition in $editor

```
CREATE OR REPLACE FUNCTION public.last_updated()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN
    NEW.last_update = CURRENT_TIMESTAMP;
    RETURN NEW;
END $function$
```

OmniTI

Postgres 8.4 / PSQL / List Your Objects

- \dX shows only user objects
- \df list user functions
- \dXS includes system objects
- \dfS list all functions

```
pagila=# \df
                                  List of functions
 Schema |             Name              | Result data type | Argument data types
--------+-------------------------------+------------------+---------------------------------------
 public | _group_concat                 | text             | text, text
 public | film_in_stock                 | SETOF integer    | p_film_id integer, p_store_id integer,
                                                           : OUT p_film_count integer
 public | film_not_in_stock             | SETOF integer    | p_film_id integer, p_store_id integer,
                                                          : OUT p_film_count integer
 public | get_customer_balance          | numeric          | p_customer_id integer,
                                                           : p_effective_date timestamp
 public | inventory_held_by_customer | integer          | p_inventory_id integer
 public | inventory_in_stock            | boolean          | p_inventory_id integer
 public | last_day                      | date             | timestamp without time zone
 public | last_updated                  | trigger          |
 public | rewards_report                | SETOF customer   | min_monthly_purchases integer,
                                                           : min_dollar_amount_purchased numeric
(9 rows)
```

Postgres 8.4 / PSQL / Enum Values

- Display enum options when displaying enum types

```
pagila=# \dT+ mpaa_rating
                                 List of data types
Schema |    Name     | Internal name | Size | Elements | Description
-------+-------------+---------------+------+----------+-------------
public | mpaa_rating | mpaa_rating   | 4    | G        |
                                                : PG
                                                : PG-13
                                                : R
                                                : NC-17
```

Postgres 8.4 / PSQL / Table Size

```
pagila=# \dt+
                          List of relations
 Schema |       Name       | Type  |  Owner   |    Size    | Description
--------+------------------+-------+----------+------------+-------------
 public | actor            | table | postgres | 16 kB      |
 public | address          | table | postgres | 56 kB      |
 public | category         | table | postgres | 8192 bytes |
 public | city             | table | postgres | 32 kB      |
 public | country          | table | postgres | 8192 bytes |
 public | customer         | table | postgres | 64 kB      |
 public | film             | table | postgres | 432 kB     |
 public | film_actor       | table | postgres | 216 kB     |
 public | film_category    | table | postgres | 40 kB      |
 public | inventory        | table | postgres | 200 kB     |
 public | language         | table | postgres | 8192 bytes |
 public | payment          | table | postgres | 0 bytes    |
 public | payment_p2007_01 | table | postgres | 72 kB      |
 public | payment_p2007_02 | table | postgres | 136 kB     |
 public | payment_p2007_03 | table | postgres | 336 kB     |
 public | payment_p2007_04 | table | postgres | 400 kB     |
 public | payment_p2007_05 | table | postgres | 16 kB      |
 public | payment_p2007_06 | table | postgres | 0 bytes    |
 public | rental           | table | postgres | 1072 kB    |
 public | staff            | table | postgres | 8192 bytes |
 public | store            | table | postgres | 8192 bytes |
```

OmniTI

Postgres 8.4 / PSQL / Foreign Keys

- Referencing tables now displayed in \d

```
pagila=# \d store
                    Table "public.store"
      Column       |     Type     |              Modifiers
-------------------+--------------+-----------------------------------
 store_id          | integer      | not null default nextval('store_store_id_seq')
 manager_staff_id  | smallint     | not null
 address_id        | smallint     | not null
 last_update       | timestamp    | not null default now()
Indexes:
    "store_pkey" PRIMARY KEY, btree (store_id)
    "idx_unq_manager_staff_id" UNIQUE, btree (manager_staff_id)
Foreign-key constraints:
    "store_address_id_fkey" FOREIGN KEY (address_id) REFERENCES address(address_id)
    "store_manager_staff_id_fkey" FOREIGN KEY (manager_staff_id) REFERENCES staff(staff_id)
Referenced by:
  "customer_store_id_fkey" IN customer FOREIGN KEY (store_id)
     REFERENCES store(store_id) ON UPDATE CASCADE ON DELETE RESTRICT
  "inventory_store_id_fkey" IN inventory FOREIGN KEY (store_id)
     REFERENCES store(store_id) ON UPDATE CASCADE ON DELETE RESTRICT
  "staff_store_id_fkey" IN staff FOREIGN KEY (store_id) REFERENCES store(store_id)
Triggers:
    last_updated BEFORE UPDATE ON store FOR EACH ROW EXECUTE PROCEDURE last_updated()
```

Postgres 8.4 / PSQL / Wait, there's more!

Postgres 8.4 / PSQL / Wait, there's more!

- Improved handling of long lines and tab characters
- Better control of \timing
- Improved tab completion support for tables in multiple schemas
- Column storage type display
- Add tablespace and database size information to \l
- Show sequence details in \d output

Postgres 8.4 / PSQL / Wait, there's more!

Postgres 8.4

- Performance
- Administration
- PSQL
- Development
- Procedures

Postgres 8.4 / Dev / WITH

- Common Table Expression (aka CTE, WITH queries)
- Declare WITH before query
- WITH computed once, before rest of query

```
pagila=# with epic_films as (select film_id, array_agg(first_name ||' '|| last_name) as featuring from film join film_actor using (film_id) join actor using (actor_id) group by film_id )
select * from epic_films where array_upper(featuring,1) > 12 ;

 film_id |                                  featuring
---------+----------------------------------------------------------------------------------------------------
     606 | {"HELEN VOIGHT","KEVIN BLOOM","TIM HACKMAN","GOLDIE BRODY","ANGELINA ASTAIRE","SPENCER PECK","WALTER TORN","SIDNEY CROWE
","GINA DEGENERES","RUSSELL BACALL","DAN STREEP","ROCK DUKAKIS","AUDREY BAILEY"}
     714 | {"JENNIFER DAVIS","LUCILLE TRACY","BURT DUKAKIS","REESE KILMER","CARMEN HUNT","JUDE CRUISE","ANGELA HUDSON","SPENCER DEP
P","HARRISON BALE","HARVEY HOPE","NICK DEGENERES","DEBBIE AKROYD","THORA TEMPLE"}
     508 | {"WOODY HOFFMAN","VAL BOLGER","REESE KILMER","JULIA BARRYMORE","MENA TEMPLE","CHRISTIAN NEESON","BURT POSEY","SCARLETT D
AMON","WALTER TORN","CAMERON ZELLWEGER","LUCILLE DEE","FAY WINSLET","JAYNE NOLTE","MENA HOPPER","JULIA ZELLWEGER"}
     146 | {"JOHNNY LOLLOBRIGIDA","LUCILLE TRACY","ELVIS MARX","SISSY SOBIESKI","VAL BOLGER","SUSAN DAVIS","RUSSELL TEMPLE","AL GAR
LAND","NICK DEGENERES","OLYMPIA PFEIFFER","LISA MONROE","HUMPHREY GARLAND","ROCK DUKAKIS"}
     249 | {"NICK WAHLBERG","JODIE DEGENERES","CARMEN HUNT","CAMERON WRAY","MICHELLE MCCONAUGHEY","SEAN WILLIAMS","BEN WILLIS","GRE
G CHAPLIN","MORGAN HOPKINS","DARYL CRAWFORD","MORGAN WILLIAMS","IAN TANDY","REESE WEST"}
      87 | {"ED CHASE","JENNIFER DAVIS","UMA WOOD","FRED COSTNER","KIRSTEN PALTROW","SANDRA PECK","DAN HARRIS","RAY JOHANSSON","KEN
NETH PESCI","CHRIS BRIDGES","WARREN JACKMAN","HUMPHREY GARLAND","AUDREY BAILEY"}
     188 | {"SISSY SOBIESKI","WOODY JOLIE","MEG HAWKE","RUSSELL BACALL","MORGAN MCDORMAND","ALBERT NOLTE","CATE HARRIS","RUSSELL TE
MPLE","VIVIEN BASINGER","HARVEY HOPE","WILL WILSON","ALAN DREYFUSS","GENE MCKELLEN"}
(7 rows)
```

Postgres 8.4 / Dev / WITH Recursive

```
pagila=# WITH recursive Fib (i, j) AS (
      VALUES (0, 1)
      UNION ALL
      SELECT (i + j), (i + j) + j
      FROM Fib
      WHERE (i + j) < 1000
)
SELECT i FROM Fib
UNION ALL
SELECT j FROM Fib ORDER BY i;
  i
------
   0
   1
   1
   2
   3
   5
   8
  13
  21
  34
  55
  89
 144
 233
 377
 610
 987
1597
(18 rows)
```

OmniTI

Postgres 8.4 / Dev / WITH Recursive

```
pagila=# WITH recursive Fib (i, j) AS (
      VALUES (0, 1)
      UNION ALL
      SELECT (i + j), (i + j) + j
      FROM Fib
      WHERE (i + j) < 1000
)
SELECT i FROM Fib
UNION ALL
SELECT j FROM Fib ORDER BY i;
  i
------
   0
   1
   1
   2
   3
   5
   8
  13
  21
  34
  55
  89
 144
 233
 377
 610
 987
1597
(18 rows)
```

- Greg Stark - Friday @ 3:00 - Intro to Recursive Queries

OmniTI

Beyond Simple SQL

rank()

```
SELECT * FROM (select c1.first_name, c1.last_name, c1.store_id,
p1.total, (select 1 + count(*) from customer c2 join (select
customer_id, sum(amount) as total from only payment group by
customer_id) p2 using (customer_id) where c2.store_id = c1.store_id
and p2.total > p1.total) as rank from customer c1 join (select
customer_id, sum(amount) as total from only payment group by
customer_id) p1 using (customer_id) ) x WHERE x.rank <= 3 ORDER BY
x.store_id, x.rank;
 first_name | last_name | store_id | total  | rank
------------+-----------+----------+--------+------
 ELEANOR    | HUNT      |        1 | 216.54 |    1
 CLARA      | SHAW      |        1 | 195.58 |    2
 TOMMY      | COLLAZO   |        1 | 186.62 |    3
 KARL       | SEAL      |        2 | 221.55 |    1
 MARION     | SNYDER    |        2 | 194.61 |    2
 RHONDA     | KENNEDY   |        2 | 194.61 |    2
(6 rows)
```

OmniTI

Postgres 8.4 / Dev / Rank / New

```
pagila=# select * from  (
with cte as (
    select first_name, last_name, store_id, sum(amount) as total
    from payment join customer using (customer_id) group by
    first_name, last_name, store_id
)
    select first_name, last_name, store_id, total,
    rank() over (partition by store_id order by total desc)
    from cte ) x where rank <= 3;

 first_name | last_name | store_id | total  | rank
------------+-----------+----------+--------+------
 ELEANOR    | HUNT      |        1 | 216.54 |    1
 CLARA      | SHAW      |        1 | 195.58 |    2
 TOMMY      | COLLAZO   |        1 | 186.62 |    3
 KARL       | SEAL      |        2 | 221.55 |    1
 MARION     | SNYDER    |        2 | 194.61 |    2
 RHONDA     | KENNEDY   |        2 | 194.61 |    2
(6 rows)
```

Postgres 8.4 / Dev / Rank / New

```
pagila=# select * from  (
with cte as (
    select first_name, last_name, store_id, sum(amount) as total
    from payment join customer using (customer_id) group by
    first_name, last_name, store_id
)
    select first_name, last_name, store_id, total,
    rank() over (partition by store_id order by total desc)
    from cte ) x where rank <= 3;

 first_name | last_name | store_id | total  | rank
------------+-----------+----------+--------+------
 ELEANOR    | HUNT      |        1 | 216.54 |    1
 CLARA      | SHAW      |        1 | 195.58 |    2
 TOMMY      | COLLAZO   |        1 | 186.62 |    3
 KARL       | SEAL      |        2 | 221.55 |    1
 MARION     | SNYDER    |        2 | 194.61 |    2
 RHONDA     | KENNEDY   |        2 | 194.61 |    2
(6 rows)
```

- Fetter/Harada - Thursday @ 4:00 - Window Functions

OmniTI

Postgres 8.4 / Dev / Limit

- LIMIT can be based on a subquery

```
pagila=# SELECT title FROM film ORDER BY random()
pagila-#  LIMIT (SELECT count(*)/10 FROM film WHERE rating = 'G');
        title
---------------------
 NONE SPIKING
 PULP BEVERLY
 SUPERFLY TRIP
 LOCK REAR
 STATE WASTELAND
 FEUD FROGMEN
 FEVER EMPIRE
 LEATHERNECKS DWARFS
 RANGE MOONWALKER
 CHOCOLAT HARRY
 MODERN DORADO
 SHREK LICENSE
 NUTS TIES
 EMPIRE MALKOVICH
 LOSE INCH
 DADDY PITTSBURGH
 DOZEN LION
(17 rows)
```

Postgres 8.4 / Dev / FTS Partial Match

- Searching partial words and word stems together

```
pagila=# select title, description from film where fulltext @@ to_tsquery('dog:*')
         except
         select title, description from film where fulltext @@ to_tsquery('dog');

    title     |                          description
--------------+---------------------------------------------------------------
 ARABIA DOGMA | A Touching Epistle of a Madman And a Mad Cow who must Defeat a
              : Student in Nigeria
 DOGMA FAMILY | A Brilliant Character Study of a Database Administrator And a
              : Monkey who must Succumb a Astronaut in New Orleans
(2 rows)
```

Postgres 8.4 / Dev / FTS Partial Match

- Searching partial words and word stems together

```
pagila=# select title, description from film where fulltext @@ to_tsquery('dog:*')
         except
         select title, description from film where fulltext @@ to_tsquery('dog');

    title     |                          description
--------------+---------------------------------------------------------------
 ARABIA DOGMA | A Touching Epistle of a Madman And a Mad Cow who must Defeat a
              : Student in Nigeria
 DOGMA FAMILY | A Brilliant Character Study of a Database Administrator And a
              : Monkey who must Succumb a Astronaut in New Orleans
(2 rows)
```

- Oleg/Teodor - Right Now - In The Other Room

OmniTI

Postgres 8.4 / Dev / Table Command

- Select all columns for output
- SQL Spec Command

```
pagila=# table language;
 language_id |     name       |     last_update
-------------+----------------+-----------------------
           1 | English        | 2006-02-15 10:02:19
           2 | Italian        | 2006-02-15 10:02:19
           3 | Japanese       | 2006-02-15 10:02:19
           4 | Mandarin       | 2006-02-15 10:02:19
           5 | French         | 2006-02-15 10:02:19
           6 | German         | 2006-02-15 10:02:19
```

OmniTI

Postgres 8.4 / Dev / Table Command

- Select all columns for output
- SQL Spec Command

```
pagila=# table language;
 language_id |     name       |     last_update
-------------+----------------+-----------------------
           1 | English        | 2006-02-15 10:02:19
           2 | Italian        | 2006-02-15 10:02:19
           3 | Japanese       | 2006-02-15 10:02:19
           4 | Mandarin       | 2006-02-15 10:02:19
           5 | French         | 2006-02-15 10:02:19
           6 | German         | 2006-02-15 10:02:19
```

- Evil, but catchy

OmniTI

- Time based generate_series()

```
pagila=# select * from generate_series(
         now() - '5 minutes'::interval,
         now(),
         '50 seconds'::interval);
         generate_series
-------------------------------
 2009-04-03 12:49:40.209094-04
 2009-04-03 12:50:30.209094-04
 2009-04-03 12:51:20.209094-04
 2009-04-03 12:52:10.209094-04
 2009-04-03 12:53:00.209094-04
 2009-04-03 12:53:50.209094-04
 2009-04-03 12:54:40.209094-04
(7 rows)
```

Postgres 8.4 / Dev / Wait, there's more!

Postgres 8.4 / Dev / Wait, there's more!

- make column alias keyword "as" optional (per sql spec)
- sql standard interval handling
- Support for statement level triggers for TRUNCATE command
- suppress_redundent_updates() trigger
- Case Insensitive Text module
- generate_subscripts() function
- array_aggregate() function
- unnest() function

Postgres 8.4 / Dev / Wait, there's more!

Postgres 8.4

- Performance
- Administration
- PSQL
- Development
- Procedures

Postgres 8.4 / Proc / CASE Control

- CASE as a control structure
- Simplify code

```
pagila=# CREATE FUNCTION kidsafe(v_title text) RETURNS text
AS $$
BEGIN
    CASE (select rating from film where title = v_title)
        WHEN 'NC-17','R' THEN RETURN 'no';
        WHEN 'PG-13' THEN RETURN 'maybe';
        WHEN 'PG','G' THEN RETURN 'yes';
    END CASE;
END;
$$ LANGUAGE plpgsql;

pagila=# select kidsafe('EVE RESURRECTION'), kidsafe('SUNRISE LEAGUE'), kidsafe('AIRPORT POLLOCK'); kidsafe |
kidsafe | kidsafe ---------+---------+---------
 yes     | maybe   | no
(1 row)
```

OmniTI

Postgres 8.4 / Proc / Variable Args

- Allows passing 1..n arguments
- Declare argument “variadic”
- Available for all languages (handled by parser)

```
pagila=# create or replace function twist
(variadic v_ints int[], out v_param int)
returns setof int as $$
declare v_i int;
begin
        for v_i in select generate_subscripts(v_ints, 1) loop
                v_param := v_i; return next;
        end loop;
        return;
end; $$ language plpgsql;
CREATE FUNCTION
pagila=# select twist(1); twist
-------
      1
(1 row)

pagila=# select twist(1,2,3); twist
-------
      1
      2
      3
(3 rows)
```

Postgres 8.4 / Proc / Default Args

- Function arguments can now have default values

```
pagila=# create or replace function fancy_last_day
(timestamp default current_timestamp::timestamp)
returns date
immutable
language sql
as $$
SELECT CASE
       WHEN EXTRACT(MONTH FROM $1) = 12 THEN
            (((EXTRACT(YEAR FROM $1) + 1) operator(pg_catalog.||) '-01-01')::date - INTERVAL '1 day')::date
       ELSE
            ((EXTRACT(YEAR FROM $1) operator(pg_catalog.||) '-' operator(pg_catalog.||)
            (EXTRACT(MONTH FROM $1) + 1) operator(pg_catalog.||) '-01')::date - INTERVAL '1 day')::date
       END
$$;
CREATE FUNCTION

pagila=# select fancy_last_day('2112-12-21 21:12:21.12'::timestamp);
-[ RECORD 1 ]-----+-----------
fancy_last_day | 2112-12-31

pagila=# select fancy_last_day();
-[ RECORD 1 ]-----+-----------
fancy_last_day | 2009-04-30
```

Postgres 8.4 / Proc / Wait...

Postgres 8.4 / Proc / Wait...

- RETURNS TABLE for plpgsql functions
- Support HINT, DETAIL, and SQLSTATE in RAISE command for plpgsql
- RETURN QUERY EXECUTE support in plpgsql
- EXECUTE USING for plpgsql
- allow srf functions to be called in select clause for plpgsql
- quote_nullable()

Postgres 8.4 / Proc / Wait...

Postgres 8.4 / Upgrades

- dump / restore garbage
  - pg_migrator (please test)
- more than two dozen incompatibilities
  - read the release notes!

Postgres 8.4 / More Info

- http://www.depesz.com/
- http://wiki.postgresql.org/
- http://www.xzilla.net
- http://omniti.com/is/hiring
- PGCon, Ottawa, Canada, May 19th - 22nd
  - This is you!

Thank you for listening.

OmniTI / Presentation