PostgreSQL Day Italy 2007

Migrating a community platform from Mysql to PostgreSQL

Andreas 'ads' Scherbaum

http://ads.wars-nicht.de/blog/

PGP: 9F67 73D3 43AA B30E CA8F 56E5 3002 8D24 4813 B5FE

06th July 2007
Table of contents

1. Overview
2. Introducing platforms
3. Ways to migrate - Exchange database
4. Ways to migrate - Rewrite application
5. Statistics
6. End
Two web platforms migrated from MySQL to PostgreSQL

Why?
- No need, if your platform scales well
- No need, if you don’t have database trouble

Reasons:
- ”Usual” MySQL problems like stability, data integrity, performance with many parallel requests
- Schemas available (different databases migrated into one single DB)
- Stored Procedures possible
- Datatypes like GIS available
Overview

- Two web platforms migrated from MySQL to PostgreSQL

Why?
- No need, if your platform scales well
- No need, if you don’t have database trouble

Reasons:
- "Usual" MySQL problems like stability, data integrity, performance with many parallel requests
- Schemas available (different databases migrated into one single DB)
- Stored Procedures possible
- Datatypes like GIS available
Two web platforms migrated from MySQL to PostgreSQL

Why?
- No need, if your platform scales well
- No need, if you don’t have database trouble

Reasons:
- "Usual" MySQL problems like stability, data integrity, performance with many parallel requests
- Schemas available (different databases migrated into one single DB)
- Stored Procedures possible
- Datatypes like GIS available
Two web platforms migrated from MySQL to PostgreSQL

Why?
- No need, if your platform scales well
- No need, if you don’t have database trouble

Reasons:
- ”Usual” MySQL problems like stability, data integrity, performance with many parallel requests
- Schemas available (different databases migrated into one single DB)
- Stored Procedures possible
- Datatypes like GIS available
Overview

Two web platforms migrated from MySQL to PostgreSQL

Why?
- No need, if your platform scales well
- No need, if you don’t have database trouble

Reasons:
- ”Usual” MySQL problems like stability, data integrity, performance with many parallel requests
- Schemas available (different databases migrated into one single DB)
- Stored Procedures possible
- Datatypes like GIS available
Overview

Two web platforms migrated from MySQL to PostgreSQL

Why?
- No need, if your platform scales well
- No need, if you don’t have database trouble

Reasons:
- ”Usual” MySQL problems like stability, data integrity, performance with many parallel requests
- Schemas available (different databases migrated into one single DB)
- Stored Procedures possible
- Datatypes like GIS available
PostgreSQL

UniHelp.de

- Student platform
- 11,000 students from Magdeburg
- Full rewrite, migration from MySQL to PostgreSQL
- Most functionality has been taken over, some new features
- Will be Open Source soon

Urbanite.de

- Community platform
- 17,000 users from around Magdeburg
- Database migration from MySQL to PostgreSQL only
- Name change from nachtlebenmd.de to urbanite.de

Andreas 'ads' Scherbaum

Migrating a community platform from MySQL to PostgreSQL
UniHelp.de

- Student platform
- 11,000 students from Magdeburg
- Full rewrite, migration from MySQL to PostgreSQL
- Most functionality has been taken over, some new features
- Will be Open Source soon

Urbanite.de

- Community platform
- 17,000 users from around Magdeburg
- Database migration from MySQL to PostgreSQL only
- Name change from nachtlebenmd.de to urbanite.de

PostgreSQL

Andreas 'ads' Scherbaum
http://ads.wars-nicht.de/blog/

Migrating a community platform from MySQL to PostgreSQL
Preparation:

- Does the website itself work with the old and new database (MySQL specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values
- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

Preparation:

- Does the website itself work with the old and new database (MySQL specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values
- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

- Preparation:
  - Does the website itself work with the old and new database (MySQL specific functions used)?
    - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
    - PHP does not recognize PostgreSQL boolean values
  - MySQL-specific options used in SQL?
    - Aggregate functions need rewrite
    - JOINs (especially OUTER JOINs) need adaption
    - ORDER BY RAND() does not work in PostgreSQL
    - Handling of NULL values is different
    - Handling of date, datetime and timestamp fields is different
    - Handling of binary data (Images, Blobs)
Exchange database

**Preparation:**

- Does the website itself work with the old and new database (MySQL specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values

- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

Preparation:

- Does the website itself work with the old and new database (MySQL specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values

- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

**Preparation:**

- Does the website itself work with the old and new database (MySql specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values
- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

Preparation:

- Does the website itself work with the old and new database (Mysql specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values

- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

- Preparation:
  - Does the website itself work with the old and new database (Mysql specific functions used)?
    - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
    - PHP does not recognize PostgreSQL boolean values
  - MySQL-specific options used in SQL?
    - Aggregate functions need rewrite
    - JOINs (especially OUTER JOINs) need adaption
    - ORDER BY RAND() does not work in PostgreSQL
    - Handling of NULL values is different
    - Handling of date, datetime and timestamp fields is different
    - Handling of binary data (Images, Blobs)
Exchange database

- Preparation:
  - Does the website itself work with the old and new database (MySQL specific functions used)?
    - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
    - PHP does not recognize PostgreSQL boolean values
  - MySQL-specific options used in SQL?
    - Aggregate functions need rewrite
    - JOINs (especially OUTER JOINs) need adaption
    - ORDER BY RAND() does not work in PostgreSQL
    - Handling of NULL values is different
    - Handling of date, datetime and timestamp fields is different
    - Handling of binary data (Images, Blobs)
Exchange database

Preparation:

- Does the website itself work with the old and new database (Mysql specific functions used)?
  - AdoDB was used in urbanite.de, so almost no MySQL specific options in PHP
  - PHP does not recognize PostgreSQL boolean values
- MySQL-specific options used in SQL?
  - Aggregate functions need rewrite
  - JOINs (especially OUTER JOINs) need adaption
  - ORDER BY RAND() does not work in PostgreSQL
  - Handling of NULL values is different
  - Handling of date, datetime and timestamp fields is different
  - Handling of binary data (Images, Blobs)
Exchange database

- **Realization:**
  - Try to rewrite a database dump with `mysql2pgsql`
  - In `urbanite`:
    - Table definition dumped and handled with `mysql2pgsql` plus post rework
    - PHP scripts used to export/import data from one database into another
    - Handling of timestamps and boolean values with this scripts
    - Binary data was exported back into files

Andreas 'ads' Scherbaum
http://ads.wars-nicht.de/blog/
Migrating a community platform from Mysql to PostgreSQL
Realization:

- Try to rewrite a database dump with mysql2pgsql
- In urbanite:
  - Table definition dumped and handled with mysql2pgsql plus post rework
  - PHP scripts used to export/import data from one database into another
  - Handling of timestamps and boolean values with this scripts
  - Binary data was exported back into files
Exchange database

Realization:
- Try to rewrite a database dump with mysql2pgsql
- In urbanite:
  - Table definition dumped and handled with mysql2pgsql plus post rework
  - PHP scripts used to export/import data from one database into another
  - Handling of timestamps and boolean values with this scripts
  - Binary data was exported back into files
Modify web application

- Handling of JOINs:

Get number of locations in a category:

MySQL:

```sql
SELECT c.*, COUNT(cl.location_id) AS location_count
FROM categories c
LEFT OUTER JOIN categories_locations cl
ON .category_id=c.id
GROUP BY c.id;
```

PostgreSQL:

```sql
SELECT c.*, COUNT(cl.location_id) AS location_count
FROM categories c
LEFT OUTER JOIN categories_locations cl
ON cl.category_id=c.id
GROUP BY c.id, c.parent_id, c.name, c.link_name, c.left_nr,
c.right_nr, c.level, c.is_important;
```

PostgreSQL returns identically values in c.*, if the result is grouped by the Primary Key.

Andreas 'ads' Scherbaum

http://ads.wars-nicht.de/blog/

Migrating a community platform from Mysql to PostgreSQL
Modify web application

Handling of JOINs:

Get number of locations in a category:

**MySQL:**

```sql
SELECT c.*, COUNT(cl.location_id) AS location_count
FROM categories c
LEFT OUTER JOIN categories_locations cl
  ON .category_id=c.id
GROUP BY c.id;
```

**PostgreSQL:**

```sql
SELECT c.*, COUNT(cl.location_id) AS location_count
FROM categories c
LEFT OUTER JOIN categories_locations cl
  ON cl.category_id=c.id
GROUP BY c.id, c.parent_id, c.name, c.link_name, c.left_nr,
  c.right_nr, c.level, c.is_important;
```

PostgreSQL returns identically values in c.*, if the result is grouped by the Primary Key.
Modify web application

- Handling of JOINs:

Get number of locations in a category:

MySQL:

```sql
SELECT c.*, COUNT(cl.location_id) AS location_count
FROM categories c
LEFT OUTER JOIN categories_locations cl
  ON .category_id=c.id
GROUP BY c.id;
```

PostgreSQL:

```sql
SELECT c.*, COUNT(cl.location_id) AS location_count
FROM categories c
LEFT OUTER JOIN categories_locations cl
  ON cl.category_id=c.id
GROUP BY c.id, c.parent_id, c.name, c.link_name, c.left_nr,
        c.right_nr, c.level, c.is_important;
```

PostgreSQL returns identically values in c.*, if the result is grouped by the Primary Key.
Modify web application

- Handling of ORDER BY RAND():
  - Not mentioned in the standard
  - MySQL uses "ORDER BY RAND()"
  - PostgreSQL requires the column to appear in the column list
  - "SELECT ..., RANDOM() as srandom ... ORDER BY srandom"
Modify web application

- Handling of ORDER BY RAND():
  - Not mentioned in the standard
  - MySQL uses "ORDER BY RAND()"
  - PostgreSQL requires the column to appear in the column list
  - "SELECT ..., RANDOM() as srandom ... ORDER BY srandom"
Modify web application

- Handling of ORDER BY RAND():
  - Not mentioned in the standard
  - MySQL uses "ORDER BY RAND()"
  - PostgreSQL requires the column to appear in the column list
  - "SELECT ..., RANDOM() as srandom ... ORDER BY srandom"
Modify web application

Handling of boolean values:

test=# \d bool_test
   Table "public.bool_test"
   Column | Type    | Modifiers
----------+---------+-----------
test     | boolean |

while ($row = pg_fetch_array($exec_query)) {
   print $row['test'] . ': ' . ($row['test']) ? " it’s true ...
" : " it’s false ...
";
}

ads@iridium:/home/ads > php5 version.php
   f: it’s true ...
   t: it’s true ...

Andreas 'ads' Scherbaum
Migrating a community platform from Mysql to PostgreSQL

http://ads.wars-nicht.de/blog/
Modify web application

- Handling of boolean values:

```
test=# \d bool_test
   Table "public.bool_test"
   Column | Type   | Modifiers
----------+---------+-----------
test     | boolean |

  test=# SELECT test FROM bool_test;
  test
     -----
    t
    f
  (2 rows)

  while ($row = pg_fetch_array($exec_query)) {
    print $row['test'] . ':' . ($row['test']) ? " it's true ...
    : " it's false ...
  }
```

ads@iridium:/home/ads > php5 version.php
  f: it’s true ...
  t: it’s true ...

Andreas 'ads' Scherbaum
http://ads.wars-nicht.de/blog/

Migrating a community platform from Mysql to PostgreSQL
Modify web application

Handling of boolean values:

test=# \d bool_test
    Table "public.bool_test"
    Column | Type    | Modifiers
    ------+---------+-----------
    test  | boolean | |

test=# SELECT test FROM bool_test;
    test
    ------
    t
    f
    (2 rows)

while ($row = pg_fetch_array($exec_query)) {
    print $row['test'] . ': ' . ($row['test']) ? " it’s true ...
" : " it’s false ...
";
}

ads@iridium:/home/ads > php5 version.php
    f: it’s true ...
    t: it’s true ...
Modify web application

Handling of boolean values:

test=# \d bool_test
   Table "public.bool_test"
   Column    | Type     | Modifiers
---------------+-----------+-----------
test          | boolean   |           

test=# SELECT test FROM bool_test;
   test
------
t  
f
(2 rows)

while ($row = pg_fetch_array($exec_query)) {
   print $row['test'] . ': ' . ($row['test']) ? " it’s true ...
      " : " it’s false ...
      

ads@iridium:/home/ads > php5 version.php
   f: it’s true ...
   t: it’s true ...

Andreas 'ads' Scherbaum http://ads.wars-nicht.de/blog/

Migrating a community platform from Mysql to PostgreSQL
Modify web application

- Handling of boolean values:
  - In MySQL, a boolean is just an alias for an integer
  - PHP has no problem to distinguish ‘0’ and ‘1’
  - In PostgreSQL the values are ‘f’ and ‘t’
  - libpq tells PHP the column type in PQftype
  - PHP does not care and just creates a normal variable
  - You have to rewrite every statement like:
    ```php
    if (!$row['columnname']) ...
    ```
Modify web application

- Handling of boolean values:
- In MySQL, a boolean is just an alias for an integer
- PHP has no problem to distinguish '0' and '1'
- In PostgreSQL the values are 'f' and 't'
- libpq tells PHP the column type in PQftype
- PHP does not care and just creates a normal variable
- You have to rewrite every statement like:
  if (!$row['columnname']) ...
Modify web application

- Handling of boolean values:
  - In MySQL, a boolean is just an alias for an integer
  - PHP has no problem to distinguish '0' and '1'
  - In PostgreSQL the values are 'f' and 't'
  - libpq tells PHP the column type in PQftype
  - PHP does not care and just creates a normal variable
  - You have to rewrite every statement like:
    if (!$row['columnname']) ...

Andreas 'ads' Scherbaum
http://ads.wars-nicht.de/blog/
Migrating a community platform from Mysql to PostgreSQL
Modify web application

- Handling of NULL values:
  - MySQL sorts NULL values first
  - PostgreSQL sorts NULL values last
  - No error messages, but different results!
Modify web application

- Handling of NULL values:
  - MySQL sorts NULL values first
  - PostgreSQL sorts NULL values last
- No error messages, but different results!
Modify web application

- Handling of NULL values:
  - MySQL sorts NULL values first
  - PostgreSQL sorts NULL values last
  - No error messages, but different results!
Modify web application

- Handling of date, datetime and timestamp fields:
  - Different function names, syntax and results
  - Timestamp in Mysql results in an integer (20070706150421)
  - Timestamp in PostgreSQL results in a string (2007-07-06 15:04:21.398557+02)
  - Different result types:
  - Return value of date operations in PG is 'interval'
  - Website areas like 'Events' or 'Calendar' need a lot of attention/debugging
Modify web application

- Handling of date, datetime and timestamp fields:
  - Different function names, syntax and results
    - Timestamp in Mysql results in an integer (20070706150421)
    - Timestamp in PostgreSQL results in a string (2007-07-06 15:04:21.398557+02)
  - Different result types:
    - Return value of date operations in PG is 'interval'
    - Website areas like 'Events' or 'Calendar' need a lot of attention/debugging
Modify web application

- Handling of date, datetime and timestamp fields:
- Different function names, syntax and results
- Timestamp in Mysql results in an integer (20070706150421)
- Timestamp in PostgreSQL results in a string (2007-07-06 15:04:21.398557+02)
- Different result types:
- Return value of date operations in PG is 'interval'
- Website areas like 'Events' or 'Calendar' need a lot of attention/debugging
Modify web application

- Handling of date, datetime and timestamp fields:
- Different function names, syntax and results
- Timestamp in Mysql results in an integer (20070706150421)
- Timestamp in PostgreSQL results in a string (2007-07-06 15:04:21.398557+02)
- Different result types:
- Return value of date operations in PG is 'interval'
- Website areas like 'Events' or 'Calendar' need a lot of attention/debugging
Modify web application

- Handling of date, datetime and timestamp fields:
- Different function names, syntax and results
- Timestamp in Mysql results in an integer (20070706150421)
- Timestamp in PostgreSQL results in a string (2007-07-06 15:04:21.398557+02)
- Different result types:
- Return value of date operations in PG is 'interval'
- Website areas like 'Events' or 'Calendar' need a lot of attention/debugging
Modify web application

- Handling of date, datetime and timestamp fields:
- Different function names, syntax and results
- Timestamp in Mysql results in an integer (20070706150421)
- Timestamp in PostgreSQL results in a string (2007-07-06 15:04:21.398557+02)
- Different result types:
- Return value of date operations in PG is 'interval'
- Website areas like 'Events' or 'Calendar' need a lot of attention/debugging
Modify web application

- Handling of binary data:
  - Binary data in a database is overhead!
  - If possible, handle files in the filesystem
  - in urbanite:
    - User pictures, photos etc were moved from DB into FS
Modify web application

- Handling of binary data:
  - Binary data in a database is overhead!
  - If possible, handle files in the filesystem
  - in urbanite:
    - User pictures, photos etc were moved from DB into FS
Modify web application

- Handling of binary data:
- Binary data in a database is overhead!
- If possible, handle files in the filesystem
- in urbanite:
  - User pictures, photos etc were moved from DB into FS
Rewrite application
Rewrite application

Database problems are never the only problems!

Almost every application (which you have to rewrite) is grown from some small piece of code and has serious performance, scalability and coding problems.

Reason: Otherwise no need for a rewrite ;-)
Rewrite application

Database problems are never the only problems!

Almost every application (which you have to rewrite) is grown from some small piece of code and has serious performance, scalability and coding problems.

Reason: Otherwise no need for a rewrite ;-)
Rewrite application

Database problems are never the only problems!

Almost every application (which you have to rewrite) is grown from some small piece of code and has serious performance, scalability and coding problems.

Reason: Otherwise no need for a rewrite ;-)
Rewrite application

**Preparation:**

- Summarize the features, even occasionally used
- Get descriptions for all features
- Use the chance to correct all(!) database mistakes

**UniHelp.de:**

- Requirements: no single line of code or database structure will be reused
Rewrite application

- Preparation:
  - Summarize the features, even occasionally used
  - Get descriptions for all features
  - Use the chance to correct all(!) database mistakes
  - UniHelp.de:
    - Requirements: no single line of code or database structure will be reused
Rewrite application

- **Preparation:**
  - Summarize the features, even occasionally used
  - Get descriptions for all features
  - Use the chance to correct all(!) database mistakes
  - UniHelp.de:
    - Requirements: no single line of code or database structure will be reused
Rewrite application

Preparation:

- Summarize the features, even occasionally used
- Get descriptions for all features
- Use the chance to correct all(!) database mistakes

UniHelp.de:

- Requirements: no single line of code or database structure will be reused
Rewrite application

- **Common problems:**
  - Primary key for users was the username
    - Username was used in friendlist, in URLs, for guestbook, forum entries and more ...
  - Friendlist was a comma-separated list
  - Now imagine a request for changing a username ...
  - Solution for rewrite: use IDs as primary key if possible
Rewrite application

Common problems:
- Primary key for users was the username
  - Username was used in friendlist, in URLs, for guestbook, forum entries and more ...
  - Friendlist was a comma-separated list
  - Now imagine a request for changing a username ...
- Solution for rewrite: use IDs as primary key if possible
Rewrite application

- **Common problems:**
  - Primary key for users was the username
    - Username was used in friendlist, in URLs, for guestbook, forum entries and more ...
  - Friendlist was a comma-separated list
    - Now imagine a request for changing a username ...
  - Solution for rewrite: use IDs as primary key if possible
Rewrite application

- **Common problems:**
  - Primary key for users was the username
    - Username was used in friendlist, in URLs, for guestbook, forum entries and more ...
  - Friendlist was a comma-separated list
  - Now imagine a request for changing a username ...
  - Solution for rewrite: use IDs as primary key if possible
Rewrite application

- **Common problems:**
  - Primary key for users was the username
    - Username was used in friendlist, in URLs, for guestbook, forum entries and more ...
  - Friendlist was a comma-separated list
  - Now imagine a request for changing a username ...
  - **Solution for rewrite:** use IDs as primary key if possible
Rewrite application

- Parse problems:
  - Help your application, split informations
  - Name of uploaded attachment was integrated in the text

```sql
mysql> SELECT id,eintrag FROM unihelp_gaestebuch WHERE id='1957337313';
+------------+---------------------------------------------------------------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>eintrag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1957337313</td>
<td>[cimg] upload/images/1182/1182256950_1152748413_birthdaycake3.jpg [/cimg]</td>
</tr>
<tr>
<td></td>
<td>Happy Birthday</td>
</tr>
</tbody>
</table>
+------------+---------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

- Almost impossible to have an 'edit' function or to reposition the image
Rewrite application

- Parse problems:
  - Help your application, split informations
  - Name of uploaded attachment was integrated in the text

```
mysql> SELECT id,eintrag FROM unihelp_gaestebuch WHERE id='1957337313';
+------------+---------------------------------------------------------------------------+
| id | eintrag |
+------------+---------------------------------------------------------------------------+
| 1957337313 | [cimg] upload/images/1182/1182256950_1152748413_birthdaycake3.jpg [/cimg] |
|    | Happy Birthday |
+------------+---------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

- Almost impossible to have an 'edit' function or to reposition the image
Rewrite application

Scaling problems:

- Scale your application for $bignum$ users
- UniHelp.de (old):
  - All uploaded images/files (for guestbooks and forum) were placed in a single directory
  - ~250 000 files in one directory
- Our workaround: split pathnames by first (4) characters
- Need to rewrite every input/output function for using the new naming schema
- Solution after rewrite: move all files in subdirectories like users/<username> or forum/<threadid>
- Move deleted entries into another table, delete entry and file in a cronjob

Andreas 'ads' Scherbaum http://ads.wars-nicht.de/blog/

Migrating a community platform from Mysql to PostgreSQL
Scaling problems:

- Scale your application for $bignum$ users
- UniHelp.de (old):
  - All uploaded images/files (for guestbooks and forum) were placed in a single directory
  - ~250 000 files in one directory
- Our workaround: split pathnames by first (4) characters
- Need to rewrite every! input/output function for using the new naming schema
- Solution after rewrite: move all files in subdirectories like users/\(<username>\) or forum/\(<threadid>\)
- Move deleted entries into another table, delete entry and file in a cronjob
Rewrite application

Scaling problems:

- Scale your application for $\text{bignum}$ users
- UniHelp.de (old):
  - All uploadeded images/files (for guestbooks and forum) were placed in a single directory
  - $\approx 250,000$ files in one directory
  - Our workaround: split pathnames by first (4) characters
  - Need to rewrite every! input/output function for using the new naming schema
  - Solution after rewrite: move all files in subdirectories like users/\text{<username>} or forum/\text{<threadid>}
  - Move deleted entries into another table, delete entry and file in a cronjob
Rewrite application

- **Scaling problems:**
  - Scale your application for $bignum$ users
  - UniHelp.de (old):
    - All uploaded images/files (for guestbooks and forum) were placed in a single directory
    - ~250 000 files in one directory
  - Our workaround: split pathnames by first (4) characters
  - Need to rewrite every input/output function for using the new naming schema
  - Solution after rewrite: move all files in subdirectories like users/<username> or forum/<threadid>
  - Move deleted entries into another table, delete entry and file in a cronjob
Rewrite application

Scaling problems:

- Scale your application for $bignum users
- UniHelp.de (old):
  - All uploaded images/files (for guestbooks and forum) were placed in a single directory
  - ~250 000 files in one directory
- Our workaround: split pathnames by first (4) characters
- Need to rewrite every! input/output function for using the new naming schema
- Solution after rewrite: move all files in subdirectories like users/$username or forum/$threadid
- Move deleted entries into another table, delete entry and file in a cronjob
Rewrite application

- Scaling problems:
  - Scale your application for $bignum$ users
  - UniHelp.de (old):
    - All uploadeded images/files (for guestbooks and forum) were placed in a single directory
    - ~250 000 files in one directory
  - Our workaround: split pathnames by first (4) characters
  - Need to rewrite every! input/output function for using the new naming schema
  - Solution after rewrite: move all files in subdirectories like users/<username> or forum/<threadid>
  - Move deleted entries into another table, delete entry and file in a cronjob
UniHelp.de:

- 1.7 GB MySQL dump
- ~5.7 M guestbook entries parsed (inline images extracted)
- ~260 000 forum postings in ~30 000 threads parsed and migrated into new forum structure
- ~500 000 uploaded files (21 GB) moved and renamed
- ~13 000 files with course material integrated in the new study system
- ~50 000 thumbnail and preview images created

Time for migration: around 14 hours (Java application)

More features (shoutbox as example), some MySQL workarounds removed (like extra tables for latest guestbook entries)

But better database performance
Statistics

- UniHelp.de:
  - 1.7 GB MySQL dump
  - ~5.7 M guestbook entries parsed (inline images extracted)
  - ~260 000 forum postings in ~30 000 threads parsed and migrated into new forum structure
  - ~500 000 uploaded files (21 GB) moved and renamed
  - ~13 000 files with course material integrated in the new study system
  - ~50 000 thumbnail and preview images created
  - Time for migration: around 14 hours (Java application)
  - More features (shoutbox as example), some MySQL workarounds removed (like extra tables for latest guestbook entries)
  - But better database performance
Statistics

UniHelp.de:

- 1.7 GB MySQL dump
- ~5.7 M guestbook entries parsed (inline images extracted)
- ~260 000 forum postings in ~30 000 threads parsed and migrated into new forum structure
- ~500 000 uploaded files (21 GB) moved and renamed
- ~13 000 files with course material integrated in the new study system
- ~50 000 thumbnail and preview images created

Time for migration: around 14 hours (Java application)
- More features (shoutbox as example), some MySQL workarounds removed (like extra tables for latest guestbook entries)
- But better database performance
Statistics

Urbanite.de:

- Better database performance with PostgreSQL
- No real problems after migration
- ~2.3 M guestbook entries
- ~4 GB images (guestbooks, galleries) with ~40,000 files in ~420 galleries
- ~450 locations in database with ~3200 ratings
Most common problem in both platforms: (auto)vacuum

- Urbanite.de was not usable at all without well-tuned vacuum
- UniHelp.de was noticeable slowed down in a week without vacuum
- But UniHelp.de was also not usable at all without extra tuning for some tables
Most common problem in both platforms: (auto)vacuum

Urbanite.de was not usable at all without well-tuned vacuum

UniHelp.de was noticeable slowed down in a week without vacuum

But UniHelp.de was also not usable at all without extra tuning for some tables
Most common problem in both platforms: (auto)vacuum

Urbanite.de was not usable at all without well-tuned vacuum

UniHelp.de was noticeable slowed down in a week without vacuum

But UniHelp.de was also not usable at all without extra tuning for some tables
Questions?

Andreas 'ads' Scherbaum <andreas@scherbaum.biz>
PostgreSQL User Group Germany