

# Introducing the IBMDB2i Storage Engine for MySQL



Presented by

Scott Klement

<http://www.scottklement.com>

© 2010, Scott Klement

*Why do programmers get Halloween and Christmas mixed-up?  
31 OCT = 25 DEC*

## Objectives Of This Session



- Why is the IBM DB2 for i (IBMDB2I) storage engine so exciting?
- Introduce working in the PASE environment.
- How you can use the IBM DB2 for i storage engine with open source applications
- How you can use tables created from IBM DB2 for i from traditional (native) applications
- (removed) How to install the MySQL and IBMDB2I -- [see articles](#)

*Although the installation is actually the first thing you need to do -- and I originally planned to include it. However, I had too much material for one session -- and since installation only means following step-by-step instructions, I decided to just give you a link.*

## Our Database (DB2 for i)



- Built-in to the IBM i operating system
- Deeply integrated into the OS
- Provides native support for databases (RPG F-spec type support)
- Full support for SQL statements, triggers, procedures, functions, constraints, etc.
- Provides two full SQL query engines
  - Classic Query Engine (CQE)
  - SQL Query Engine (SQE -- faster)
- Today it's called DB2 for i
- For years, had no name... simply "the database"
- Did everything we needed -- it was hard to beat!

*For many years it was all we had...*

3

## Why Would We Want Something Else?



People run computers for the applications they provide.

The ability to download hundreds or thousands of free applications (written in PHP, Java, C, C++, etc) and run them on IBM i is important!

Mainly PHP. Most PHP software using a database was written for MySQL

- PHP certainly supports DB2 (and many other databases.)
- You could go through and change the code to use DB2.
- But that's time consuming and risks introducing bugs.
- Want to do it again every time you upgrade a version?!

How can we download applications in native PHP, have them work with MySQL support, yet still have the benefits of the deeply integrated DB2 database?

4

# What is MySQL?



- most popular open source DB software
- provides a relational database "engine"
  - understands/interprets SQL statements
  - query analyzer, query engine, etc.
- designed from the ground-up for use with a TCP/IP network.
  - ✓ Works equally well when user is on another computer as when on the same computer.
  - ✓ Client interfaces available for all major platforms
- part of the **LAMP** stack – very popular in PHP circles – *and there's a ton of software for it!*
  - **L**inux (we'll substitute IBM i)
  - **A**pache (we already have)
  - **M**ySQL (we now have!)
  - **P**HP (via Zend Core or Zend Platform)

# Take It Seriously!



When you hear "free code", you sometimes don't take it seriously. Just a toy? Fun, but not ready for important production jobs?

*MySQL is not like that! Take MySQL seriously!*

Here is a small sampling of the folks who rely upon MySQL:

Motorola  
Siemens  
Lucent  
Cingular  
Cisco  
Nortel

Leapfrog  
Scholastic  
Ticketmaster  
Adobe  
Sony  
Associated Press

Facebook  
Google  
Craigslist  
Yahoo!

# How Does MySQL Work?



The SQL idea: you tell it "what you want", it figures out how to go about retrieving it.

- You don't tell it which index to use
- You don't tell it how to go about retrieving a record.
- For example: Should it retrieve the records randomly, then sort them? Or retrieve them sorted? *That's its decision, not yours...*

*But, MySQL does NOT actually know how to retrieve the physical disk storage!  
Weird, eh?*

Instead, it uses what's called a "Pluggable Storage Engine".

- modular design
- anyone can write their own method of storing data.

Examples:

- Record-oriented databases (normal) (MyISAM and InnoDB)
- CSV (allows read/write/query, etc on CSV files)
- Memory (for temporary, but very fast, databases)

*NOTE: IBM i supports the above via the IFS interface....*

- For IBM i, IBM provides the "IBM DB2 for i" (IBMDB2i) storage engine 7

# Where Have We Heard That Before?



Can you think of anything else that's modular like that?

Anything else where a core part of the software can be replaced without changing the higher-levels?

How about IBM i?

How about TIMI?

# The IBM DB2 for i Storage Engine



When using IBMDB2I storage engine under MySQL:

- Specify via ENGINE=IBMDB2I on SQL statements, or by setting as the active storage engine (or even the default storage engine)
- MySQL does it's job of interpreting, analyzing, etc, the SQL
- But the IBM DB2 for i storage engine is called to store the data.
- IBMDB2i will store the data in "ordinary" PFs, and will use LFs for indexes, etc.

That means:

- Tables, Indexes and Views must be **created** from MySQL.
- But once created, they can be freely **accessed** from MySQL, DB2 or native interfaces (like RPG F-specs, OPNQRYF, etc)
- Includes reading records (RPG CHAIN, READ, READ or SQL SELECT)
- Writing & updating records (RPG WRITE/UPDATE or SQL INSERT/UPDATE)
- However, stored procedures and functions are still completely separate.

9

## Scenario 1: Free Software on i



- Download free software
  - SugarCRM, ZenCart, MediaWiki, many others.
  - Write custom reports in RPG
  - Integrate data into your existing systems using RPG
  - Fewer barriers, because data is in traditional IBM objects.

Example: You want a CRM software.

- Download SugarCRM (available for free – but also available as commercial with support, etc)
- Install into iAMP stack
- Use IBMDB2I storage engine for tables.
- Now you can write custom reports in RPG if you want.
- You can integrate with (for example) an existing payables package in RPG -- just access the tables normally, they're just PFs and LFs!

10

## Scenario 2: Access IBM i from Outside



At Klement's Sausage we have:

- Existing servers running PHP and MySQL
- Would be very nice if they could view data on IBM i.
  - Track order progress.
  - Write new orders into system
  - Get item/pricing information
- IBM doesn't make DB2 drivers for FreeBSD.
- Previously, we used Java/JDBC, but that was cumbersome because the applications are in PHP.
- Web services work nicely, but aren't as simple to create/deploy.
- Often had to download whole file into MySQL, and use from there.
  - *Not real time!*
  - *Caused errors due to duplicated/changed data.*

After converting tables to be both MySQL & DB2:

- Easy to access via MySQL from the PHP on the FreeBSD boxes.
- Easy to use from RPG (it's still in PFs and LFs)

11

## Scenario 3: Test on Laptop, Deploy to i



- One of the nice features of PHP is that it's cross platform. Many developers have taken to writing PHP software on their laptops.
  - While waiting at the airport
  - While flying (maybe over the Atlantic!)
  - Long distance train trip
- But you are limited as to which parts you could write:
  - Database was on i, and not available on your laptop
  - You could run MySQL on laptop, but you'd have to change the database logic to run on DB2, and therefore have to re-test it.
- With IBMDB2I, you can run MySQL in both places (laptop and i) and still take advantage of native PFs and LFs.
- But still have compatible DB on laptop

Apps are *somewhat database independent*, because MySQL acts as a "database independence layer"

12

# The PASE Environment



MySQL was originally written for a Unix environment. Keeping it as a Unix program means maximum compatibility with code written for the LAMP stack.

To interact with MySQL, you use a Unix environment:

- Scrolling command-line (or "shell") interface.
- Hierarchical directory structure (IFS)
- ASCII character set

The Portable Application Solutions Environment (PASE) provides Unix compatibility on IBM i.

- Run AIX programs with minimal changes (or no changes)
- Use existing AIX compilers to generate the code
- Provides full Unix environment on i

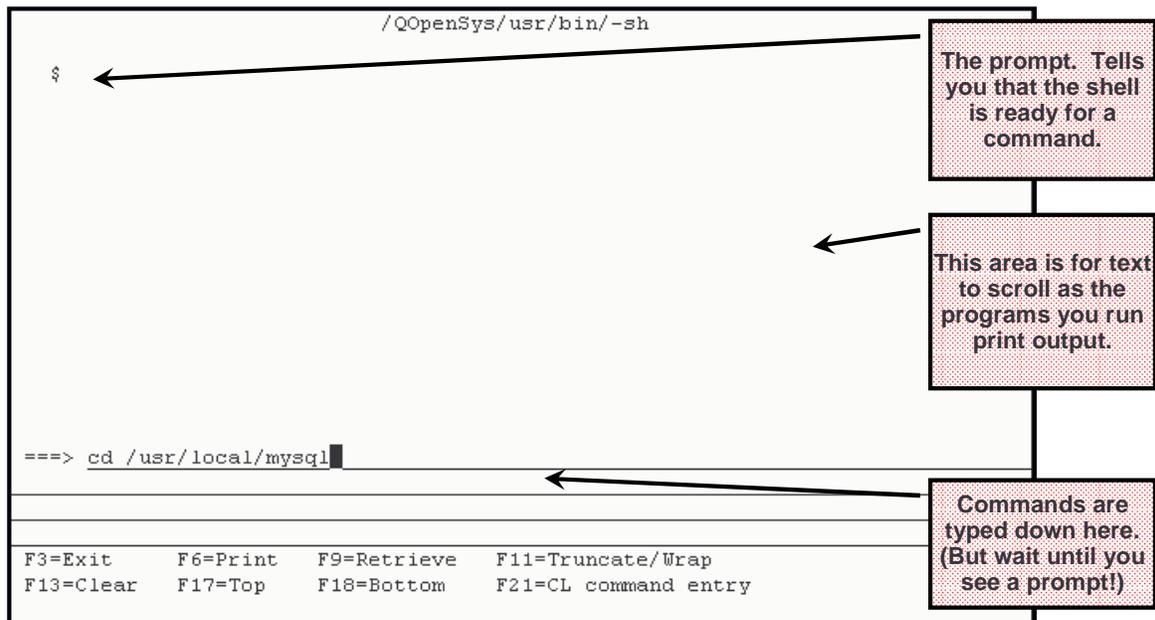
13

# PASE and the Shell



To put yourself at a PASE command line ("shell"), type:

`CALL QP2TERM`



14

# Calling Programs in a Unix Shell



For example, if I typed the following command:

```
cd /tmp
```

- Up to the first space is the program name
- The rest of the line is a series of parameters to be passed to that program, separated by spaces (in this example, there's only one parameter, '/tmp')

Therefore, the preceding Unix command is equivalent to the following syntax at the traditional IBM i command-line:

```
CALL PGM(CD) PARM('/tmp')
```

By contrast, this is calling the program named 'mv', and passing two parameters:

```
mv test.key /home/klemscot/.ssh
```

15

# Quoting Special Shell Characters



There are several characters that have special meanings when typed at a Unix shell.

- Blanks delimit parameters
- Dollar-signs insert variables
- Semi-colons allow more than one command on a line
- Back-slashes mean that the next character is taken literally.
- Ampersands, pipes, greater than, less than all have special meanings
- Characters in quotes do not have special meanings, except:
  - Inside double-quotes ("weak quotes") dollar signs and double quotes
  - Inside single-quotes ("strong quotes") only the single quote itself has a special meaning. Or a backslash if it's followed by a single quote or another backslash.

This example works because the spaces and single quotes do not have special meanings when typed inside weak quotes:

```
cp "Today's Lesson.ppt" "archive/Yesterday's Lesson.ppt"
```

16

# Finding Programs with PATH



To find a program, the PASE shell (like other Unix shells) will search all directories in your PATH environment variable. PATH contains a list of IFS directories to search, separated by colons. Here's an example of setting the PATH from the native environment (prior to calling QP2TERM:)

```
ADDENVVAR ENVVAR(PATH) VALUE('/QOpenSys/usr/bin:/dir1:/dir2')
```

If I typed the following command in PASE:

```
mypgm parm1 parm2 parm3
```

PASE would look for a program named 'mypgm' by searching these IFS paths:

- /QOpenSys/usr/bin/mypgm
- /dir1/mypgm
- /dir2/mypgm

Think of PATH the way you think of library lists. (Except it's only used to locate programs -- \*LIBL are used for files and other objects, too.)

17

# Basic PASE Tools



A few commonly used programs (included with PASE):

<code>cd dirname</code>	Change current working directory to <i>dirname</i>
<code>cp src dest</code>	copy a file ( <i>src</i> ) to another name or directory ( <i>dest</i> )
<code>mv old new</code>	Move / rename <i>old</i> file to <i>new</i> name
<code>pwd</code>	Print working directory
<code>mkdir dirname</code>	Create (make) a new directory name <i>dirname</i>
<code>chmod mode file</code>	Change authorities ( <i>mode</i> ) of <i>file</i>
<code>ls</code>	List files (like 'dir' in MS-DOS, or WRKOBJ)
<code>ls -l</code>	List files in long format (more info about the files)
<code>cat file</code>	Dump the contents of <i>file</i> on the screen
<code>rm file</code>	Delete <i>file</i>
<code>find tree expr</code>	Search for files in <i>tree</i> that match <i>expr</i>

More Info about PASE (as well as QShell) is found in the Information Center under Programming -> Shells and Utilities

18

# MySQL Tools Provided



<b>mysql</b>	Client program, let's you interactively work with MySQL <ul style="list-style-type: none"><li>• Command-line SQL interface.</li><li>• Somewhat similar to STRSQL (but for MySQL)</li><li>• Can be used in a shell script or CL program</li></ul>
<b>mysqladmin</b>	Administration/Configuration Tool <ul style="list-style-type: none"><li>• Start, stop, debug and control the MySQL server</li><li>• Copies securely over a network if prefixed by host nam</li><li>• Change on-the-fly configuraton variables.</li></ul>
<b>mysqldump</b>	Back up the MySQL databases <ul style="list-style-type: none"><li>• Outputs all needed SQL statements to recreate/repopulate the database.</li><li>• To restore, run the SQL statements as a script in the mysql tool</li></ul>
<b>mysqld</b>	MySQL server ("daemon") the program that runs the server <ul style="list-style-type: none"><li>• Listens on a network (TCP) port.</li><li>• Receives the requests from the mysql client(s) and runs them.</li></ul>

19

# Starting/Stopping the MySQL Server



- To start/stop the MySQL server, switch to the dir where you put mysql

```
cd /QOpenSys/usr/local/mysql/mysql
```

- lets you use relative paths starting with that directory!

- To start the mysql server:

```
bin/mysqld_safe --user=mysql
```

- same as /usr/local/mysql/mysql/bin/mysqld\_safe

- To stop the server

```
bin/mysqladmin --user=xx --password=xx shutdown
```

- You need to use a mysql userid/password with authority to end the server.

20

# Using the Client



```
mysql --user=klemscot --password=bigboy
```

- Gives you an interactive SQL interface.
  - Like STRSQL?
  - Or the QShell 'db2' command?

Can also run scripts -- input can be read from a pipe.

```
cat myscript.sql | mysql --user=klemscot --password=bigboy
```

- Can also run scripts (pipe the script to the mysql command)
- Chart showing naming rules

21

# The Client Screen



This is what it looks like in the MySQL client

```

/QOpenSys/usr/bin/-sh

$
> mysql --user=klemscot --password=bigboy
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.1.38 MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

mysql>

====> █

F3=Exit      F6=Print      F9=Retrieve   F11=Truncate/Wrap
F13=Clear    F17=Top       F18=Bottom    F21=CL command entry
```

22

## Examples Using the Client



- Database -- similar to a library or schema.
  - `create database my-database-name;`
- To get a list of databases, type:
  - `show databases;`
- To select a database to be used.
  - `use database-name-here;`
- Tables -- similar to a physical file
- To see a list of tables in your selected database.
  - `show tables;`
- To run an SQL statement, just type it followed by a semicolon
  - `create table mytable (field1 integer, field2 char(20));`
  - `insert into mytable values(1, "Scott Klement");`
  - `select * from mytable;`
  - `alter table mytable engine=ibmdb2i;`

23

## Changing Table to IBMDB2I



In this example, the MediaWiki software was installed to use MySQL, but with the InnoDB storage engine (a traditional MySQL record-oriented database engine). To change one of it's tables to use IBMDB2I, I could do this from MySQL:

```
mysql> use wikidb;  
  
mysql> alter table text ENGINE=IBMDB2I;
```

I would have to repeat the "alter table" step for each table (or write a script to do so)

24

# Gradual Transition



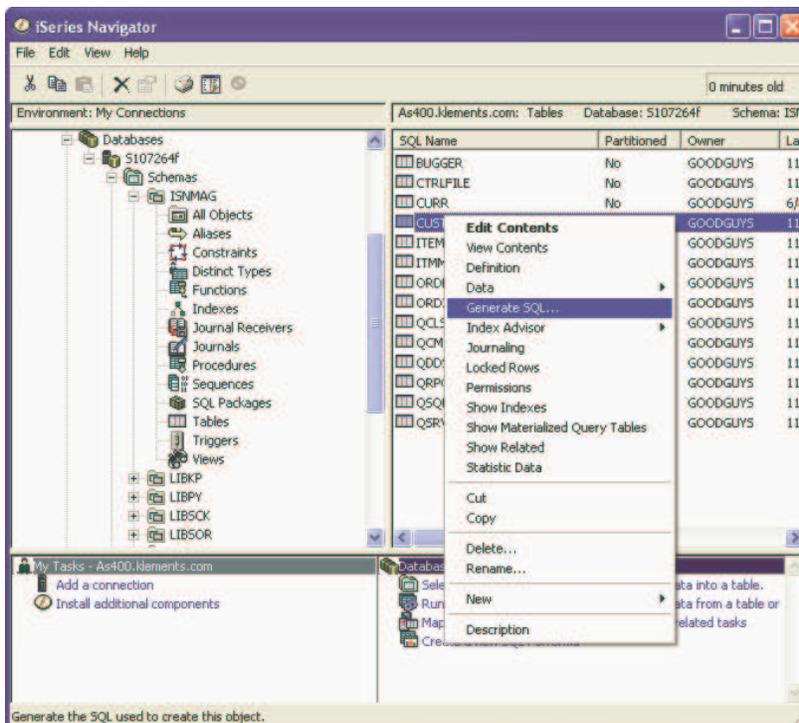
Scenario 1 also illustrates a framework for gradual transition:

- Your apps are pure RPG today.
- You want to convert to PHP
- With IBMDB2I you can transition slowly:
  - Start by converting databases to MySQL with the storage engine
  - Databases are now easy to access from both RPG & PHP
  - Convert one program at a time (over a long period of time) from RPG to PHP
  - PHP code uses MySQL interface.
  - When all is done, your PHP code will be portable, you can now run it (including the database) on Linux or Windows.

*Note: I'm not recommending this. I personally believe that RPG is a better language for business rules than PHP. So it makes sense to keep part of your code in native RPG forever. However, I do know of shops that want to completely transition, and wanted to point out how IBMDB2I can make it easier for them.*

25

## Convert an Existing PF to MySQL (1 of 3)



Converting an existing database to run with MySQL means first you have to create the same table inside MySQL.

To do that, first use iSeries Navigator to retrieve an SQL definition of the file.

26



# File names in MySQL



All of the names in MySQL are case-sensitive (by default.)

- By default, MySQL is installed in /QOpenSys which is a case-sensitive file system.
- MySQL is designed to be used on Unix, where everything is case sensitive.

MySQL also allows names to be much longer than 10 characters.

- Including database names, table names, and column names.

In V5R4, database names are limited to 10 characters (or 8 characters with double-quotes)

- However, table & column names are still able to be long.
- This limit only affects the IBMDB2I storage engine (others work fine.)
- If you try to create a table in a database longer than 10 chars you'll get:

```
ERROR 1005 (HY000): Can't create table
'reallylongname.mytable' (errno: 2516)
```

29

# How They're Represented in DB2



In order to make names case-sensitive in DB2:

- All uppercase names are represented as case-insensitive in DB2
- Regular MySQL (case-sensitive) names are represented as "quoted names" in DB2

Did you know that our traditional library/object system allowed case-sensitive names? It always has. To use this, you put double quotes around the object name.

This is a feature that not many folks use -- but it's always been there.

Example:

```
WRKOBJ "wikidb"/"mytable"
```

When MySQL is in case-insensitive mode (an option you can configure), MySQL will automatically translate all table names to lowercase. Therefore, DB2 will always put the names in quotes (since they're never all-uppercase).

*Seem backward to you?!*

30

## Names Longer Than 8 Characters



When an object name is longer than 8 characters (or 10 if it's all-uppercase) the system needs to create an "alias" name.

For example, I might run the following in MySQL:

```
create table a_really_long_table_name
      ( column1 integer ) ENGINE=IBMDB2I;
```

This creates a table name in MySQL that can't be stored in a normal library. The system will generate a name like this:

```
"a_re0001"
```

There's a file named SYSTABLES that keeps track of the relationships between the long and short names. Therefore, you could find the short name by doing a query from DB2:

```
Select sys_tname from "demo"/SYSTABLES
      where name='a_really_long_table_name'
```

31

## RPG Access Example



- The quotes look a little strange when trying to use the PF from RPG, but it works.

```
* to compile:
*>   OVRDBF FILE(CUSTFILE) TOFILE("demo"/"custfile")
*>   CRTBNDRPG READCUSF
*>   DLTOVR CUSTFILE
*
FCUSTFILE  IF  E          DISK  ExtFile('"demo"/"custfile"')
F          Rename(CUSTFILE: CUSTFILEF)
/free

      read CUSTFILE;
      .
      .
```

You should be able to work with this file exactly the same as any other physical file. (Other than the strange names)

32

# PHP Access Example



```
<?php
$conn = mysql_connect('localhost', 'klemscot', 'bigboy');
if (!$conn) {
    echo "connect failed.\n";
    return 1;
}

mysql_select_db("demo");

$res = mysql_query("select * from custfile");
if (!$res) {
    echo "statement failed.\n";
    return 1;
}

while ($row = mysql_fetch_assoc($res)) {
    echo $row['CUSTNO'] . " " . $row['NAME'] . "\n";
}

mysql_free_result($res);
mysql_close($conn);
?>
```

33

# Links to MySQL Information



From System iNEWS magazine:

MySQL and the DB2 Storage Engine for IBM i (Erwin Early)

<http://systeminetwork.com/article/mysql-and-db2-storage-engine-ibm-i>

Taking Advantage of Open Source for i5/OS (Erwin Early)

<http://systeminetwork.com/article/taking-advantage-open-source-i5os>

How to Use the MySQL IBMDB2I Storage Engine for DB2 on i (Erwin Early)

<http://systeminetwork.com/article/how-use-mysql-ibmdb2i-storage-engine-db2-i>

IBM Redbooks:

Discovering MySQL on IBM i5/OS

<http://www.redbooks.ibm.com/abstracts/sg247398.html>

Using IBM DB2 for i as a Storage Engine of MySQL

<http://www.redbooks.ibm.com/abstracts/SG247705.html>

Has  
Installation  
Info, and  
loads of other  
stuff!

34

# Links to MySQL Information



MySQL: Blessing or Heresy? (Podcast by Susan Gantner)

<http://systeminetwork.com/article/podcast-mysql-blessing-or-heresy>

Official MySQL page for the IBMDB2I storage engine:

[http://solutions.mysql.com/engines/ibm\\_db2\\_storage\\_engine.html](http://solutions.mysql.com/engines/ibm_db2_storage_engine.html)

PHP.NET documentation for the MySQL routines:

<http://php.net/manual/en/book.mysql.php>

35

# This Presentation



You can download a PDF copy of this presentation from:

<http://www.scottklement.com/presentations/>

# Thank you!

36