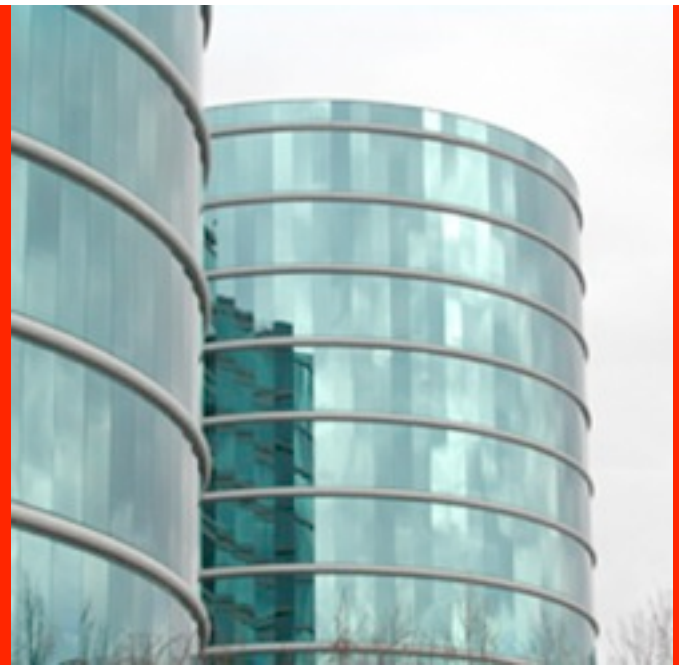


MySQL

ORACLE

Shooting from the hip - MySQL at the command line

Giuseppe Maxia
MySQL Community Team Lead

Shooting from the hip.
MySQL at the command line

```
$ mysql
```

Giuseppe Maxia
MySQL Community Team Lead
twitter: @datacharmer

about me -Giuseppe Maxia

- a.k.a. The Data Charmer
- MySQL Community Team Lead
- Long time hacking with MySQL features
- Formerly, database consultant, designer, coder.
- A passion for QA
- An even greater passion for open source
- ... and community
- Passionate blogger
- http://datacharmer.blogspot.com

Updated presentation slides

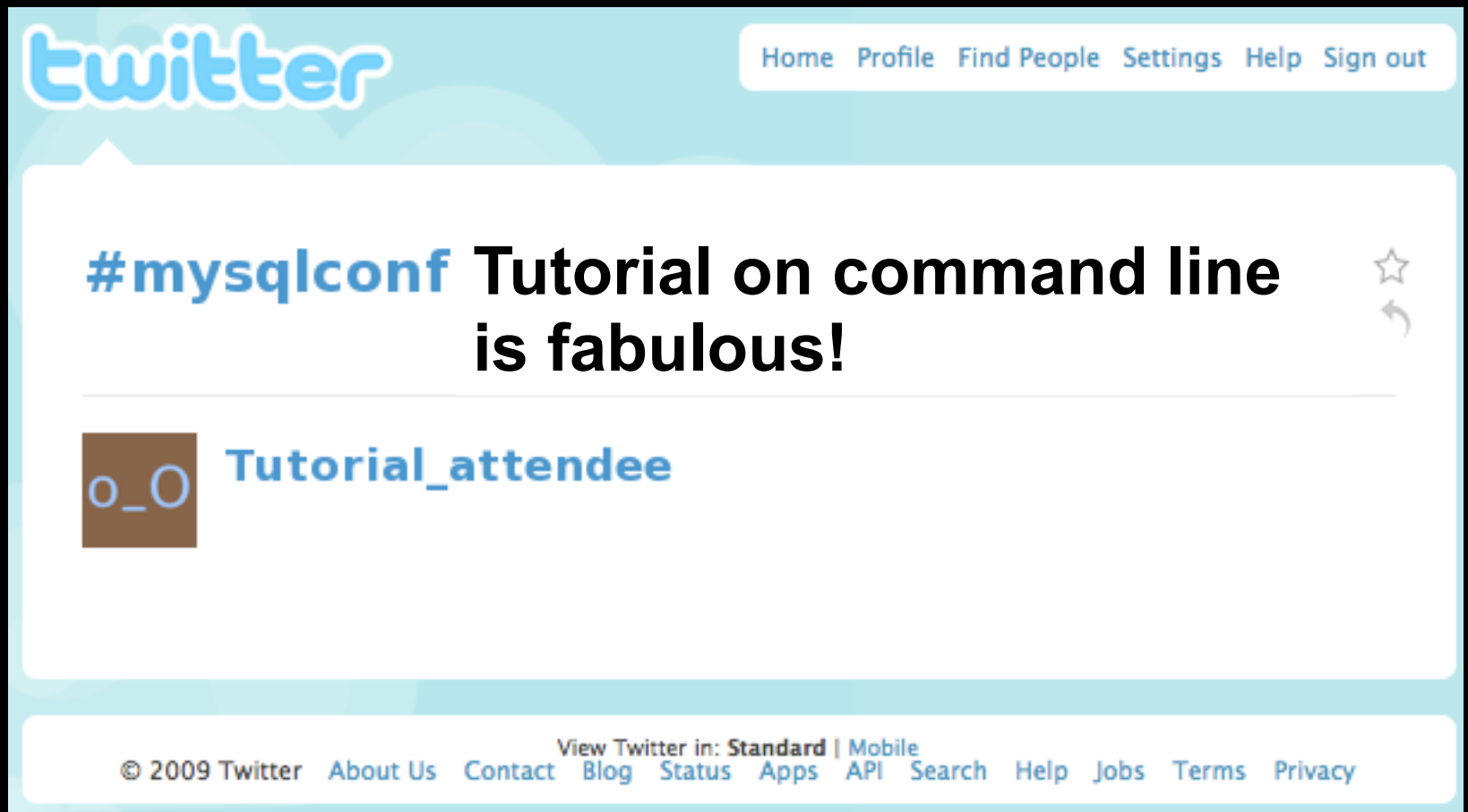
- you will find an up to date copy of these slides at:
- http://tinyurl.com/cli-tut-2010

We love your feedback!

- Tell it to the Twitter community
  - #mysqlconf "your feedback here"
- Twitter me
  - @datacharmer "your feedback here"
- Blog about it (if you already have a blog)
- Find my boss in the corridors and tell him I was great!

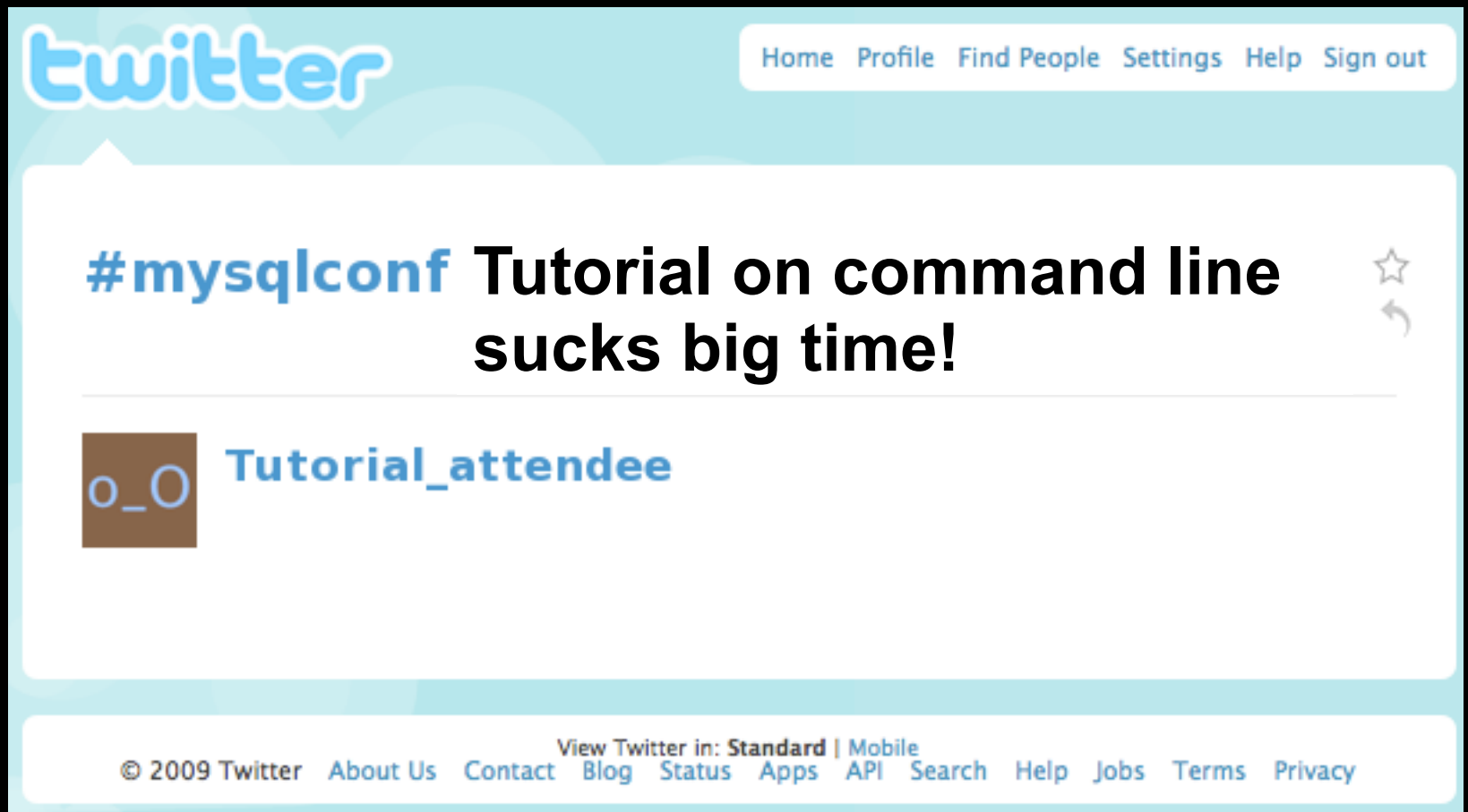
Feedback examples

- Tell it to the Twitter community



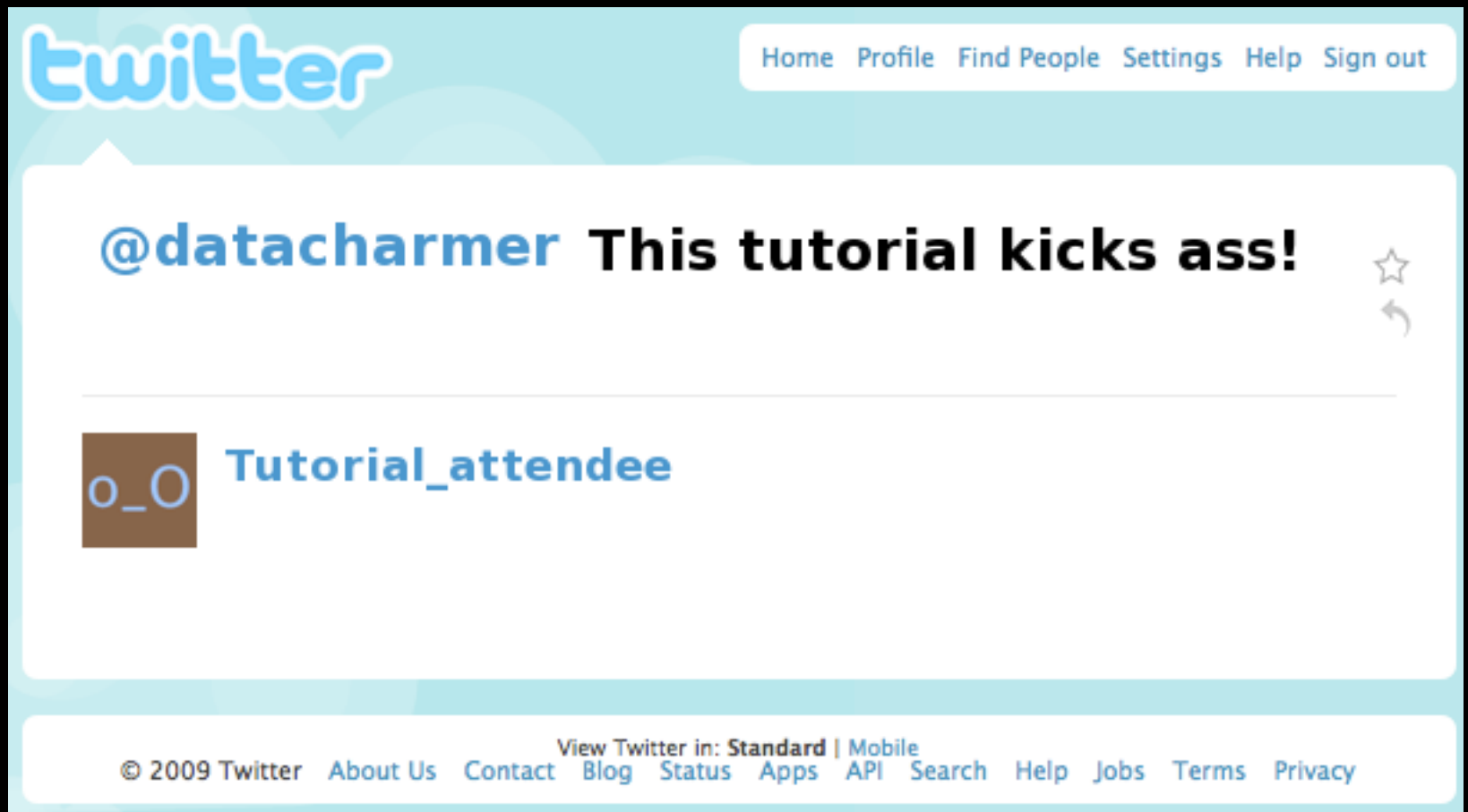
Feedback examples

- Tell it to the Twitter community



Feedback examples

- Twitter me

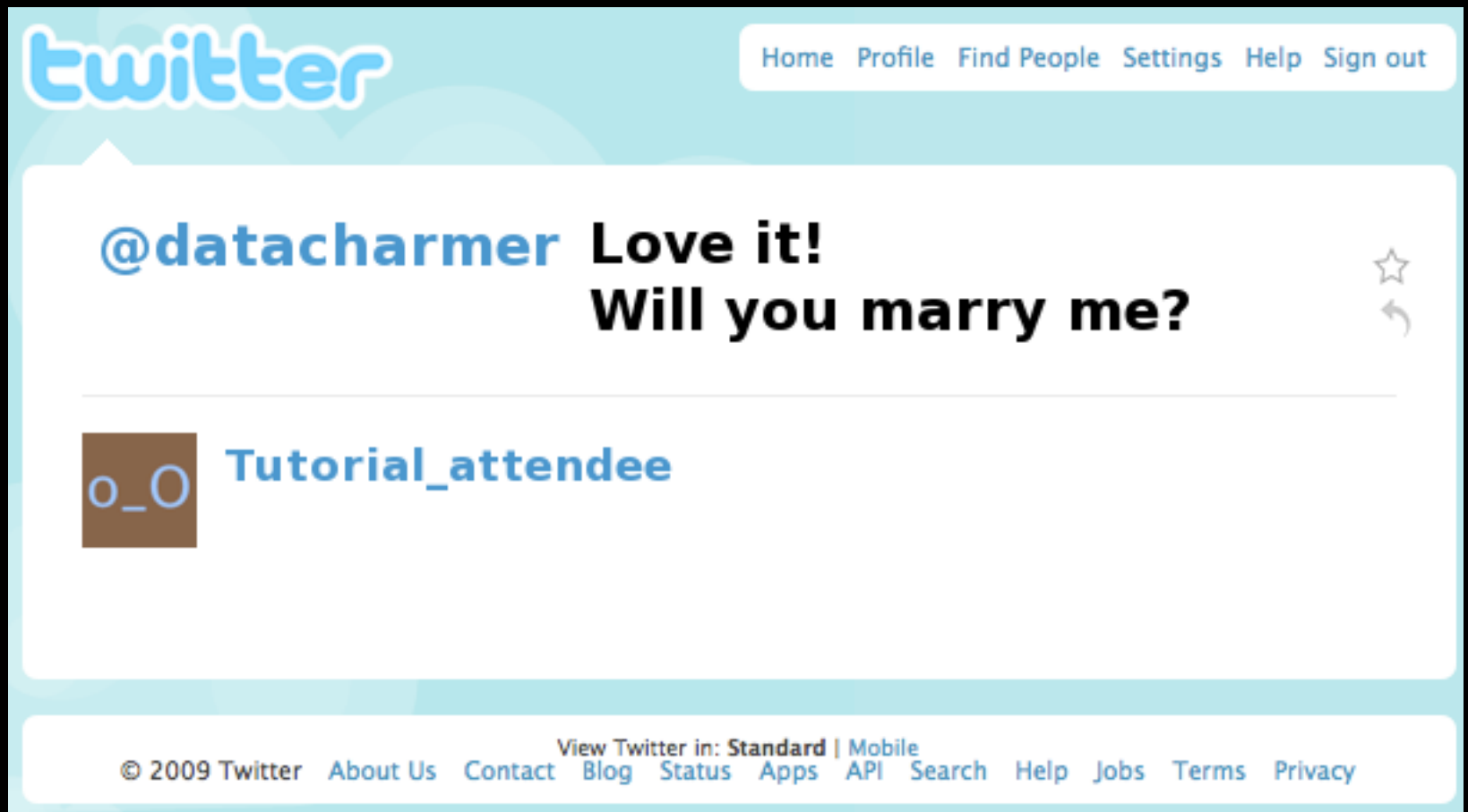


Feedback examples

- Twitter me

Feedback examples

- Twitter me



Feedback examples

- Find my line manager: Kaj Arnö
- Tell him how good you feel about this tutorial

THERE ARE 10 TYPES OF PEOPLE ...

THERE ARE 10 TYPES OF PEOPLE ...

Those who understand binary numbers, and those who don't.

THERE ARE 10 TYPES OF PEOPLE ...

Those who understand binary numbers, and those who don't.

But that's another story.

What this tutorial is about

- Making GUI enthusiasts green with envy;
- Providing tools and tips for command line users;
- Learn a few tricks to make you achieve more and type less.

the command line

WHY

Why the command line

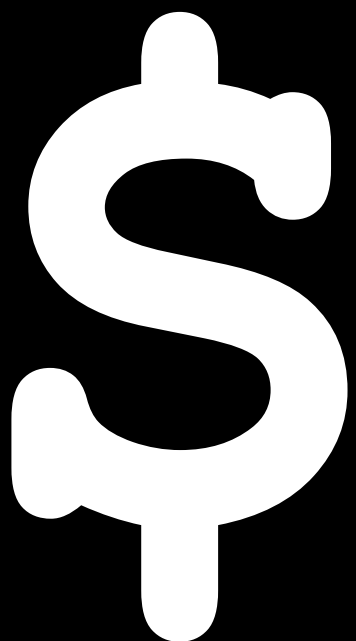
- portability
- efficiency
- fun

the command line

NO
PANIC

Command line panic

Command line panic



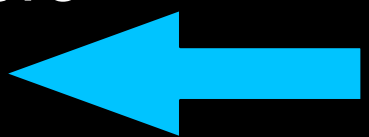
NOW WHAT?

the command line

BASICS

The shell

- It's a command interpreter
- There are many flavors
  - bash
  - zsh
  - ksh
  -



MySQL command line client

- called "mysql"
  - > Takes SQL commands
  - > sends them to the server
  - > gets results
- but it can do much more ...

other tools

- Shell scripts
- Aliases
- Perl
- awk
- sort
- …. and a lot of Unix little helper

in case it wasn't clear

in case it wasn't clear

- command line actions are

in case it wasn't clear

- command line actions are
  - commands (the name of the application to invoke)

in case it wasn't clear

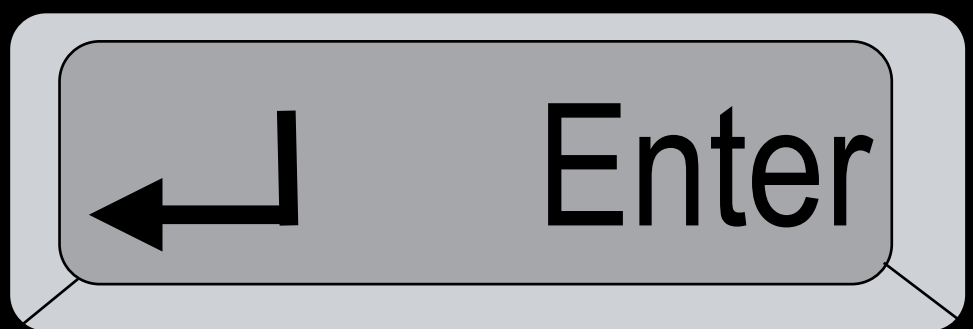
- command line actions are
  - commands (the name of the application to invoke)
  - parameters (optional arguments for the application)

in case it wasn't clear

- command line actions are
  - commands (the name of the application to invoke)
  - parameters (optional arguments for the application)
  - and the ENTER key

in case it wasn't clear

- command line actions are
  - commands (the name of the application to invoke)
  - parameters (optional arguments for the application)
  - and the ENTER key



The Unix philosophy (1)

- One tool for every feature
  - > make one tool that does one thing well
  - > for many features, make many tools
  - > make them work together (see next slide)

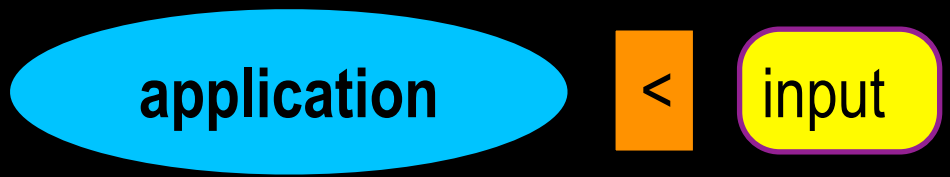
The Unix philosophy (2)

- To make several tool cooperate:
  - pipe the output of one tool to the input of another one.
  - for more actions, pipe more tools

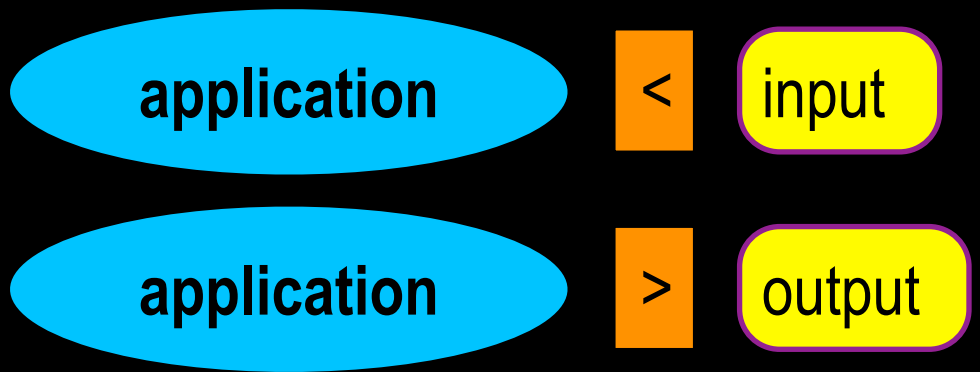
```
$ sort < filename | uniq | nl
```

Unix pipes

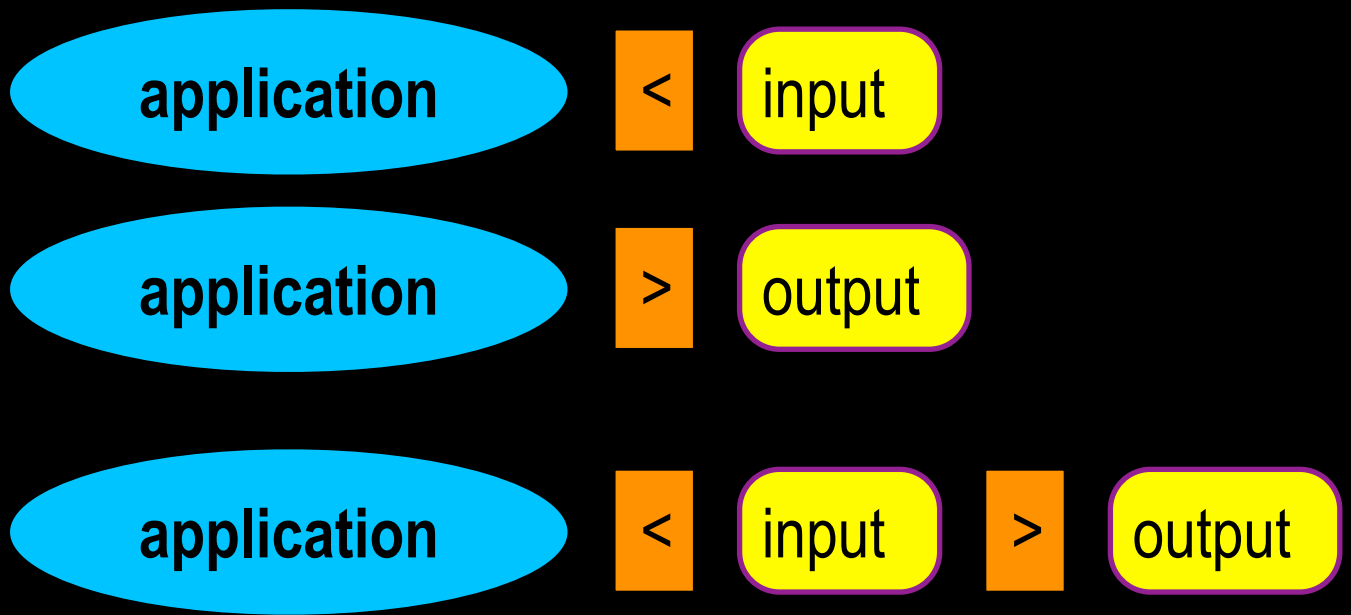
Unix pipes



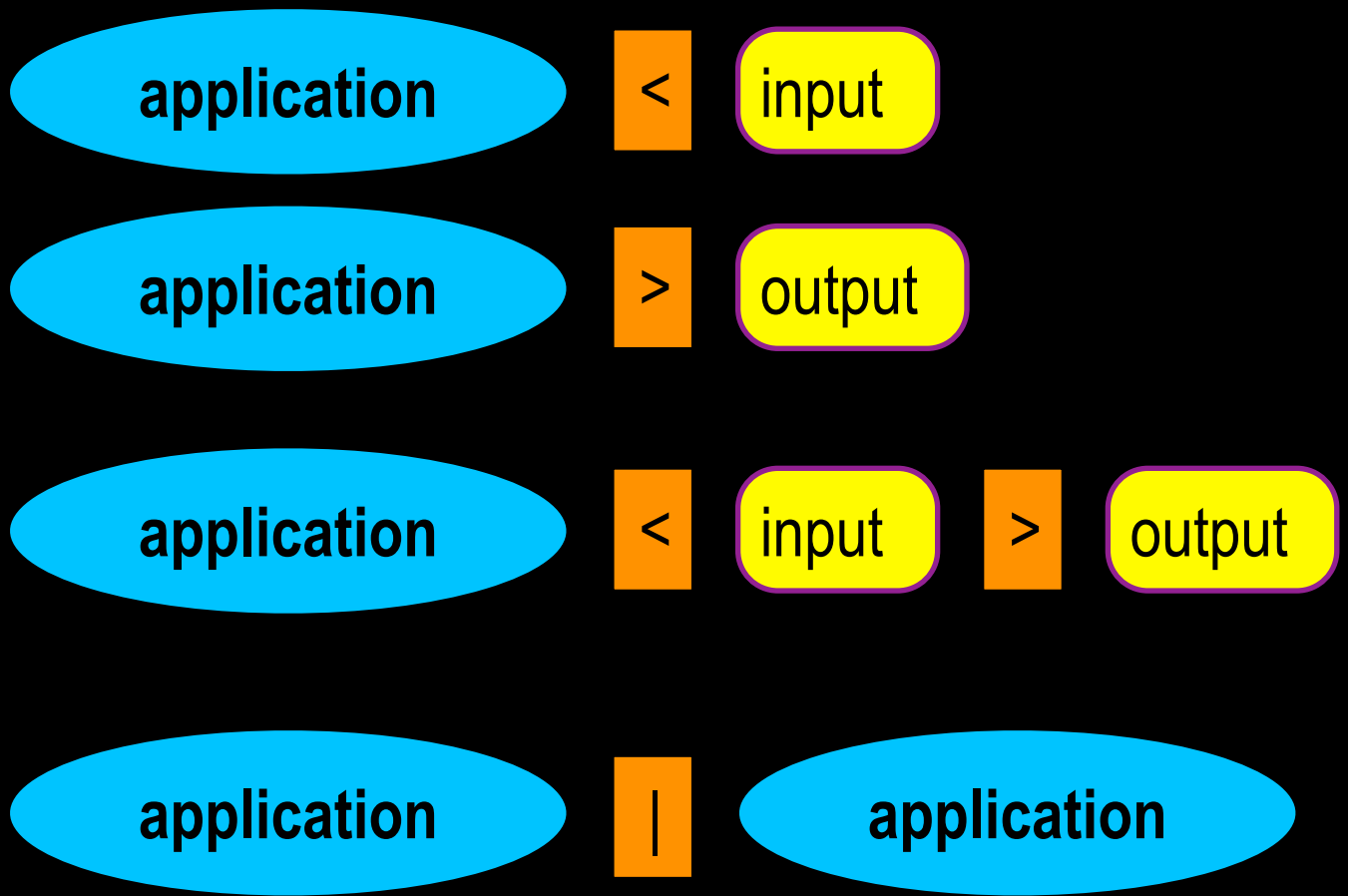
Unix pipes



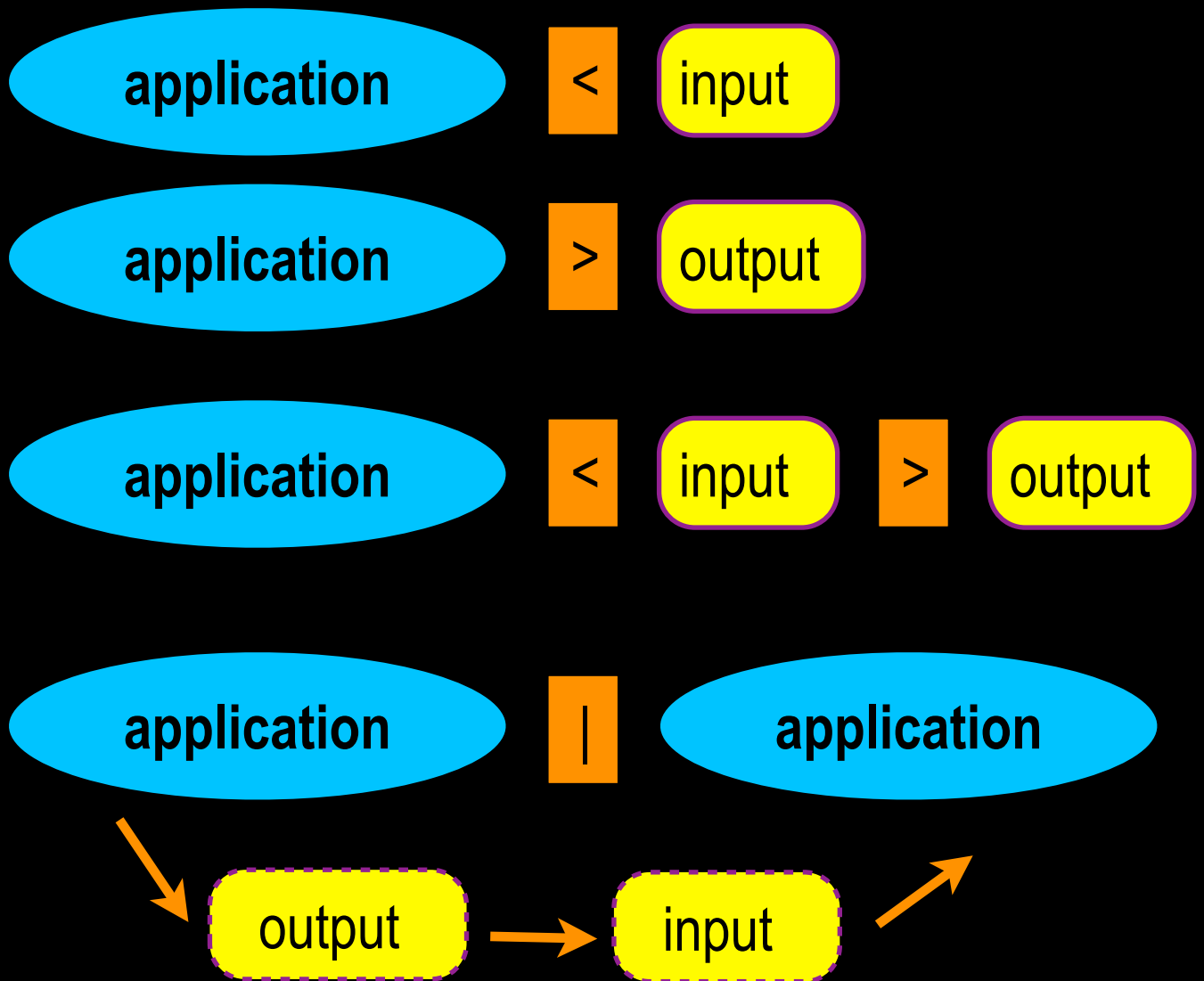
Unix pipes


application
<
input
application
>
output
application
<
input
>
output

Unix pipes

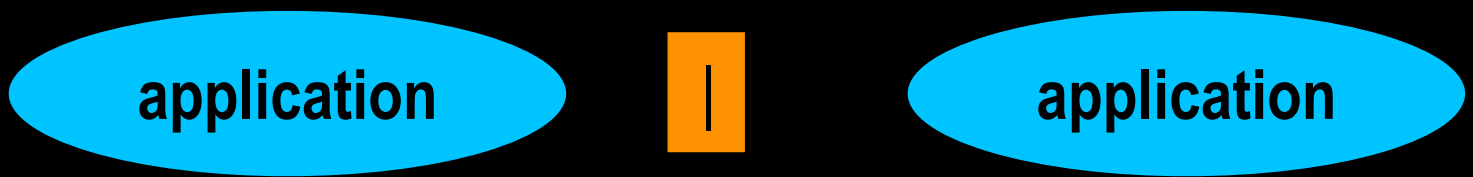

application
<
input
application
>
output
application
<
input
>
output
application
|
application

Unix pipes


application
<
input
application
>
output
application
<
input
>
output
application
|
application
output
input

Unix pipes = parallel processing

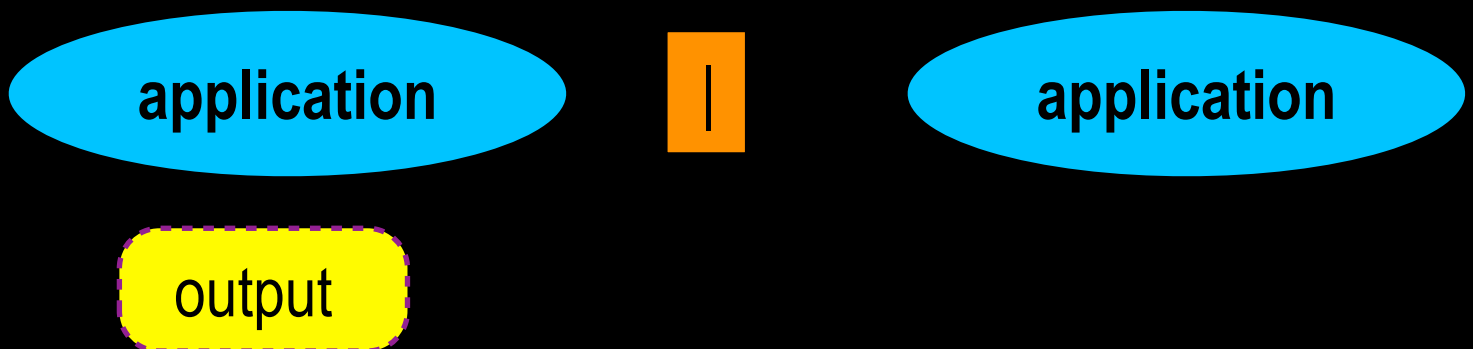
Unix pipes = parallel processing



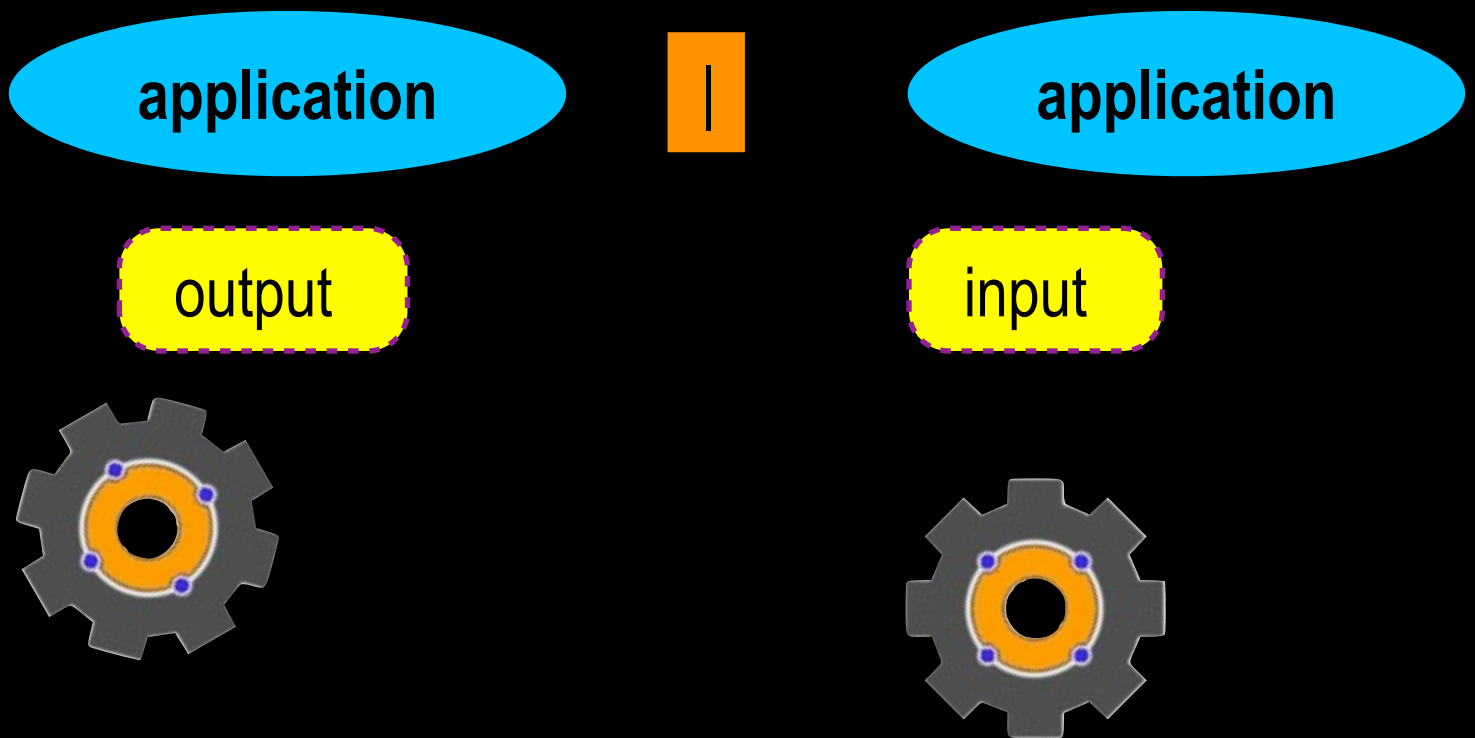
Unix pipes = parallel processing

application | application

Unix pipes = parallel processing



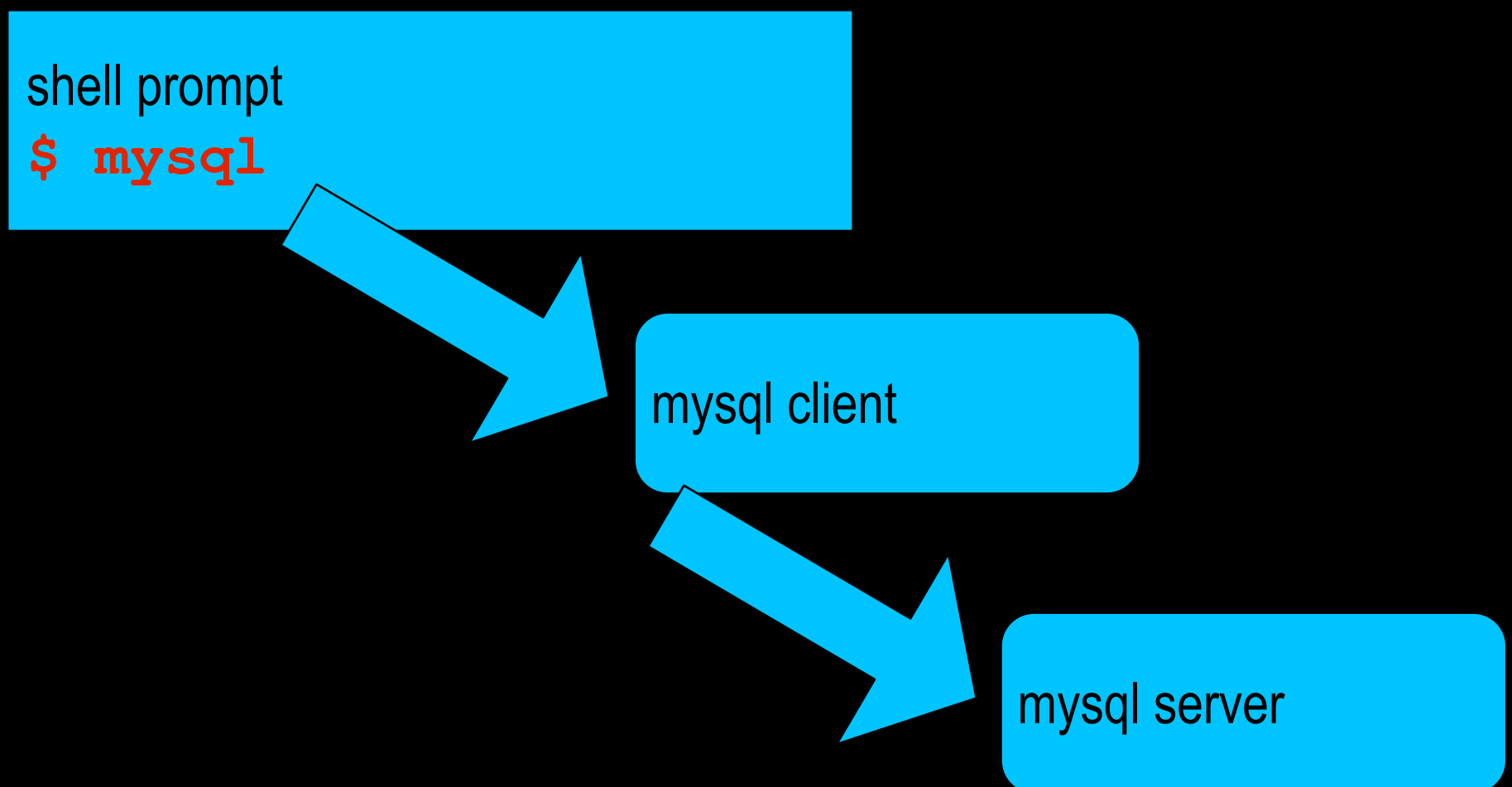
Unix pipes = parallel processing



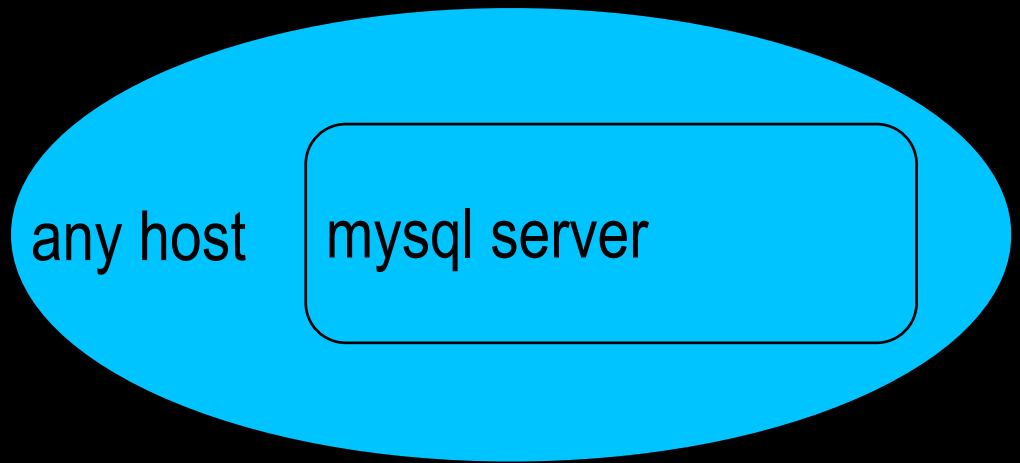
Who gets what

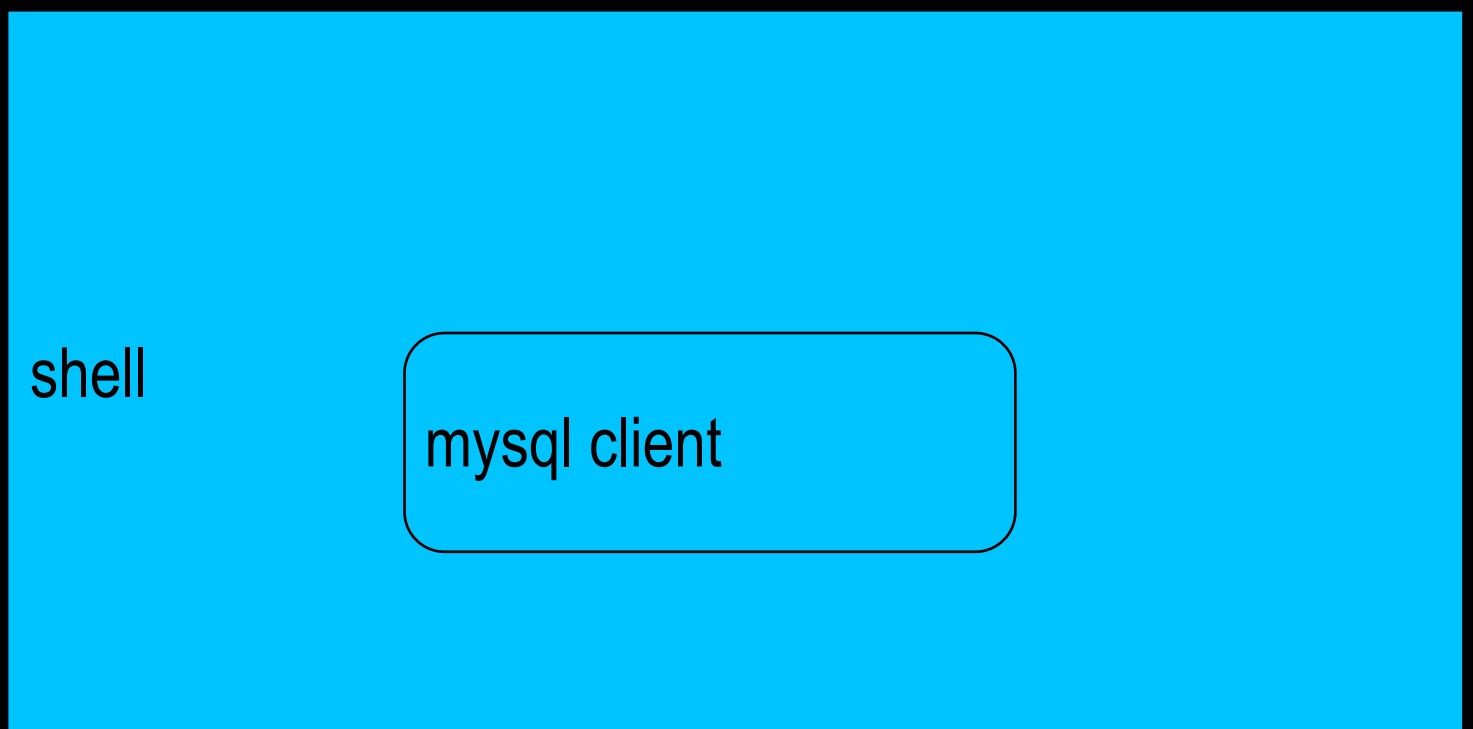
- shell
- application
- database server
- pager

who gets what



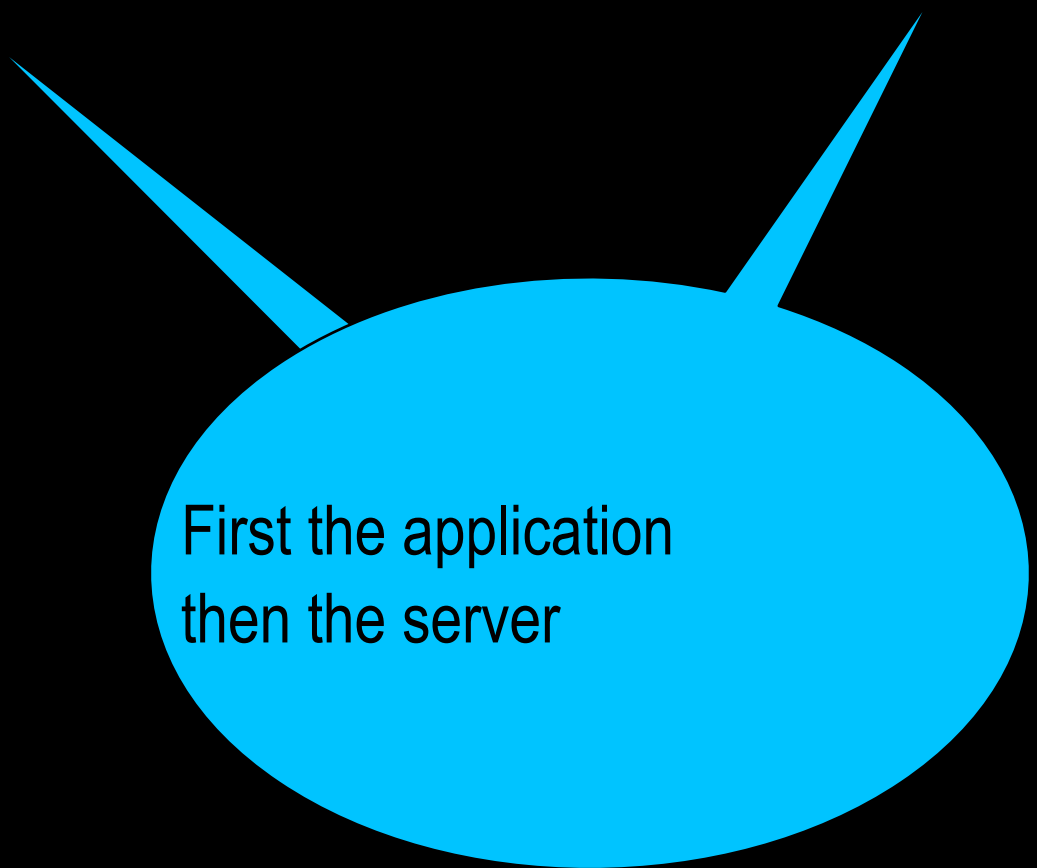
who gets what





who gets what

```
$ mysql -u username -ppassword
```



who gets what

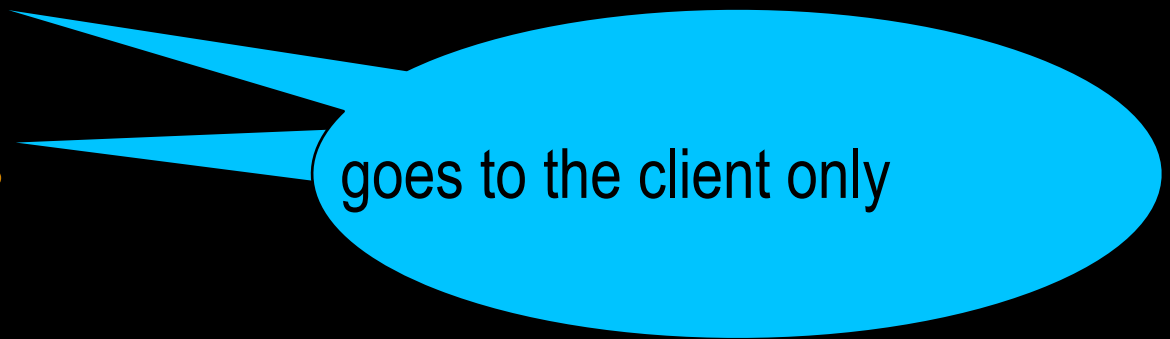
```
$ mysql -u username -ppassword \
  -e "select VERSION()"
```

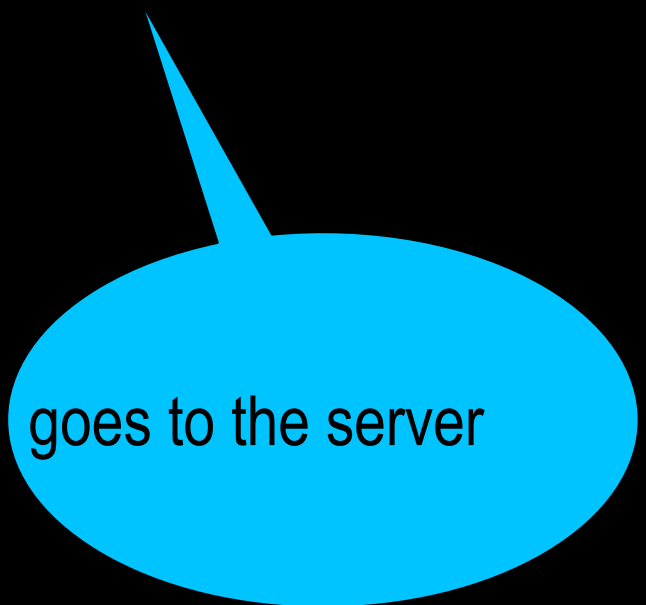
goes to the server

goes to the client, then the server

who gets what

```
mysql> help
mysql> pager less
mysql> select 1;
```





the command line

readline

readline

- Common component
  - shell
  - mysql client
  - many CLI applications

Main cursor movements

Ctrl A Ctrl E	Start/End of line
Meta B Meta F	Back/Forward one word
↑ ↓	previous/next command
Ctrl L	clear screen and take current line on top

Changing text

Meta U	Uppercase word
Meta L	Lowercase word
Ctrl D	delete word
Ctrl K	delete to end of line
Ctrl -	undo

for more tips:

```
$ man readline
```

Some conventions in this talk

```
$ command
```

at the shell

```
mysql> command
```

at the mysql prompt

the Bash shell

bash

components

- commands
- aliases
- variables
- functions

commands

```
$ echo "something"
something
```

aliases

```
$ alias mydate='mysql -e "select NOW()"'
$ mydate
+---------------------+
| now()               |
+---------------------+
| 2010-04-12 07:14:19 |
+---------------------+
```

aliases

```
$ alias myself='mysql'
$ myself -e "select NOW()"'
+---------------------+
| now()               |
+---------------------+
| 2010-04-12 07:14:19 |
+---------------------+
```

variables

```
$ MYBATCH='mysql -N -B'
$ MYQUERY='select curdate()'
$ $MYBATCH -e "$MYQUERY"
2010-04-12
```

nesting variables

```
# t.sh
 CMD="SELECT TABLE_NAME"
 TBL='information_schema.tables'
 WHC1="TABLE_SCHEMA='mysql'"
 WHC2="TABLE_NAME like 'u%' "
 Q1="$CMD FROM $TBL"
 Q2="WHERE $WHC1 AND $WHC2"
 Q="$Q1 $Q2"
 echo $Q
 echo $Q | mysql
```

nesting variables

```
$ sh t.sh
SELECT TABLE_NAME FROM
information_schema.tables WHERE
TABLE_SCHEMA='mysql' AND TABLE_NAME like
'u%'
TABLE_NAME
user
```

the *mysql* command line client

mysql

SQL commands from a file

```
$ mysql < filename.sql
mysql> source filename.sql
```

- executes the commands contained in filename.sql
- commands can be queries understood by the server
- and commands for the client

SQL commands at the command line

```
$ application | mysql
$ echo "SELECT 1" | mysql
$ mysql -e "SELECT 1"
```

- executes the commands from standard input
- commands can be queries understood by the server
- and commands for the client

batch mode

```
$ echo "select curdate()" | mysql
curdate()
2010-04-12
```

table mode

```
$ echo "select curdate()" | mysql -t
+-------------+
| curdate()   |
+-------------+
| 2010-04-12 |
+-------------+
```

table mode

```
$ mysql -e "select curdate()"
+-------------+
| curdate()   |
+-------------+
| 2010-04-12 |
+-------------+
```

Forcing batch mode

```
mysql -B -e 'select curdate()'
curdate()
2010-04-12
```

Forcing batch mode - no header

```
mysql -N -B -e 'select curdate()'
2010-04-12
```

Shell + mysql

```
$ echo "The current server's date is
`mysql -N -B -e 'select curdate()'`."

$ echo "The current server's date is
$(mysql -N -B -e 'select curdate
()')."
```

advanced command line techniques

loops

loops basics

```
$ for N in 1 2 3
> do
> echo $N
> done
1
2
3
```

loops basics

```
$ for N in 1 2 3 ; do echo $N ; done
1
2
3
```

loops basics

```
$ for N in `seq 1 3` ;\
 do echo "this is number $N" ;\
 done

this is number 1
this is number 2
this is number 3
```

loops from file

```
$ head /usr/share/dict/words
A
a
aa
aal
aalii
aam
Aani
aardvark
aardwolf
```

loops from file

```
$ for U in \
`head /usr/share/dict/words ` ; do
perl -le 'print uc shift' $U ; done
A
A
AA
AAL
AALII
AAM
AANI
```

loops from mysql results

```
$  mysql -B -N -e 'show tables from
world'
City
Country
CountryLanguage
```

loops from mysql results

```
$ for T in `mysql -B -N -e 'show
tables from world'` ;\
>     do echo "CREATE TABLE world2.$T
LIKE world.$T;" ; done

CREATE TABLE world2.City LIKE
world.City;
CREATE TABLE world2.Country LIKE
world.Country;
CREATE TABLE world2.CountryLanguage
LIKE world.CountryLanguage;
```

piping loops back into mysql

```
$ for T in `mysql -B -N -e 'show
tables from world'` ;\
>     do echo "CREATE TABLE world2.$T
LIKE world.$T;" ; done | mysql -vv
```

advanced command line techniques

nested loops

doing something for every table in every database

```
$ mysql -e "show databases"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| sakila             |
| world              |
+--------------------+
```

doing something for every table in every database

```
$ mysql -e "show tables from world"
+-----------------+
| Tables_in_world |
+-----------------+
| City            |
| Country         |
| CountryLanguage |
+-----------------+
```

pseudocode

```
for each database
   do
   for each table
      do
      print the table name
```

actual code

```
alias mybatch='mysql -B -N'
for D in `mybatch -e "SHOW SCHEMAS"`
do
  for T in \
  `mybatch -e "SHOW TABLES FROM $D"`
  do
     echo "$D.$T"
  done
done
```

advanced command line techniques

options files

Default option files

```
/etc/my.cnf
/etc/mysql/my.cnf (Debian, Ubuntu)
$BASEDIR/my.cnf
$HOME/.my.cnf
```

Custom option files

```
$ mysql --no-defaults
$ mysql --defaults-file=filename
$ mysql --defaults-extra-file=filename
```

Option files sections

```
[mysqld]
# option for server

[client]
# option for any mysql client

[mysql]
# options specific to 'mysql' CLI app
```

Option files example

```
[client]
user=myname
password=mypassword

[mysql]
prompt="mysql (\h) {\u} [\d] > "
```

Option files: caution

```
# protect your option files!

$ chmod 600 $HOME/.my.cnf

$ ls -l  .my.cnf
-rw-------  1 [...] 10:04  .my.cnf
```

prompt

```
mysql> prompt gm (\h) {\u} [\d] >
PROMPT set to 'gm (\h) {\u} [\d] > '
gm (localhost) {root} [(none)] > use
test
Database changed
gm (localhost) {root} [test] >

# \h -> host
# \u -> user
# \d -> database
```

advanced command line techniques

aliases

An useful alias

```
$ alias mysql_as_root=\
'mysql --
defaults_file=~/.my_as_root.cnf'

mysql_as_root -e "select user()"
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
```

An useful alias

```
#.my_as_root.cnf'
[client]
user=root
password=your_mysql_root_password

[mysql]
prompt="### [\h] {\u} (\d) > "
```

Another useful alias

```
alias mybatch='mysql -B -N '


mybatch -e "select version()"
5.1.45-log
```

advanced command line techniques

functions

functions

- like aliases, but with parameters

sample function

```
function grant_new {
    mysql_as_root -e \
    "GRANT $1 on $2 to $3 identified
by \"$4\" "
}
```

```
$ grant_new SELECT "world.*" john \
  john_secret
```

sample function

```
$ mysql_as_root \
   -e  'show grants for john\G'
*** 1. row ***
Grants for john@%: GRANT USAGE ON *.*
TO 'john'@'%' IDENTIFIED BY PASSWORD
'*73A88CE133741288844E3ABB500F439A9C02C
53A3'
*** 2. row ***
Grants for john@%: GRANT SELECT ON
`world`.* TO 'john'@'%'
```

advanced command line techniques

getting
query
details

which query is being executed?

```
# wrong.sql
# this script has one error
use test ;
drop table if exists t1 ;
create table t1 (id int);
insert into t1 values (a);
```

which query is being executed?

```
$ mysql < wrong.sql
ERROR 1054 (42S22) at line 4: Unknown
column 'a' in 'field list'


# WHERE IS THE ERROR?
```

which query is being executed?

```
$  mysql -v < wrong.sql
---------------
drop table if exists t1
---------------
---------------
create table t1 (id int)
---------------
---------------
insert into t1 values (a)
---------------
ERROR 1054 (42S22) at line 4: Unknown column
'a' in 'field list'
```

getting execution times

```
$ mysql -vv -e \
'select rating,count(*) from sakila.film
group by rating'
+--------+----------+
| rating | count(*) |
+--------+----------+
| G      |      178 |
| PG     |      194 |
| PG-13  |      223 |
| R      |      195 |
| NC-17  |      210 |
+--------+----------+
5 rows in set (0.00 sec)
```

getting execution times (batch mode)

```
$echo 'select rating,count(*) from sakila.film group
by rating' | mysql -vv
--------------
select rating,count(*) from sakila.film group by
rating
--------------


rating  count(*)
G  178
PG 194
PG-13 223
R  195
NC-17 210
5 rows in set
```

getting execution times (batch mode)

```
$ echo 'select rating,count(*) from sakila.film group
by rating' | mysql -vvvB
--------------
select rating,count(*) from sakila.film group by
rating
--------------
+--------+----------+
| rating | count(*) |
+--------+----------+
| G      |      178 |
| PG     |      194 |
| PG-13  |      223 |
| R      |      195 |
| NC-17  |      210 |
+--------+----------+
5 rows in set (0.00 sec)
```

advanced command line techniques

getting distinct results

getting a single value

```
$ mysql -e "show variables like 'port'"
+-----------------+-------+
| Variable_name | Value |
+-----------------+-------+
| port            | 3306  |
+-----------------+-------+

# Here, we only want the port number
```

getting a single value

```
$ mysql -NB -e "show variables like 'port'"
 port            3306


# still two values
```

getting a single value

```
$ mysql -NB \
  -e "show variables like 'port'" \
  | awk '{print $2}'

3306

$ mysql -NB \
  -e "show variables like 'port'" \
 | perl -lane 'print $F[1]'

3306
```

ORACLE

getting a single value: is replication working?

```
$ mysql \
  -e "show slave status \G" \
 | grep -i "slave_io_running"

             Slave_IO_Running: Yes


# too many things.
# We only want a 'yes' or 'no'
```

getting a single value: is replication working?

```
$ mysql \
  -e "show slave status \G" \
  | grep -i "slave_io_running" \
  | awk '{print $2}'

Yes


# yes!
```

advanced command line techniques

combining column names

combining column names

```
$ mysql -N -e 'desc world.City'
+-------------+----------+----+-----+
|          ID |  int(11) | NO | PRI |
|        Name | char(35) | NO |     |
| CountryCode |  char(3) | NO |     |
|    District | char(20) | NO |     |
|  Population |  int(11) | NO |     |
+-------------+----------+----+-----+
```

combining column names

```
# We want:
# SELECT CONCAT_WS('|',
# ID,Name,CountryCode,District,
# Population) from City
```

combining column names (I)

```
echo "select concat_ws('|',
   `mysql -N -e 'desc world.City' | \
    perl -lane 'push @f, $F[0];' \
         -e 'END{print join ",", @f}'`)
from world.City"
```

combining column names (II)

```
echo "select concat_ws('|', `
  mysql -NB -e 'select group_concat
(column_name) from
information_schema.columns where
table_name="City" and
table_schema="world"'`) from
world.City"
```

advanced command line techniques

dumping data

dumping a tab delimited data set

```
# the naive way
$ mysql \
  -e "SELECT *
  into outfile '/tmp/city_data'
from world.City"

# it works only if:
# * /tmp/city_data does not exist
# * /tmp is world writable
```

dumping a tab delimited data set

```
$ mysql -B -N \
  -e "SELECT * from world.City" \
  > /tmp/city_data

# it works always

# if you omit the "-N" it also
# includes a header
```

dumping a CSV

```
$ mysql -B \
  -e "SELECT * from world.City" \
|  perl -F"\t" -lane \
 'print join ",",
  map {s/"/""/g;
      /^[\d.]+$/ ? $_ : qq("$_")}
  @F ' \
 > /tmp/city_data
```

advanced command line techniques

script it

use the mysql.sh script

- The companion code for this tutorial
- http://tinyurl.com/cli-tut-code-2010
- It will save you much typing
- But use it at your own risk!

Usage

- download the script
- load it

```
$ source mysql.sh
```

mysql.sh functions

- **my_batch "query"**
- returns the result of the query.
- Without header
- Without box

```
$ my_batch "SELECT * FROM test.t1"
```

mysql.sh functions

- **my_value "what"**
- returns a single value

- e.g.

```
$ my_value "version()"
$ my_value "@@server_id"
```

mysql.sh functions

- **my_var "what"**
- returns the value of a given variable

- e.g.

```
$ my_var port
$ my_var datadir
```

mysql.sh functions

- **my_status "what"**
- returns the value of a given status item

- e.g.

```
$ my_status com_select
$ my_status connections
```

mysql.sh functions

- **my_dump_dataset "query" [header]**
- returns the values of a given query separated by tabs
- if a header is requested, it will be part of the result

- e.g.

```
$ my_dump_dataset "select *
from world.City"
```

mysql.sh functions

- **my_csv "query" [separator]**
- returns the values of a given query separated by commas, or by a given separator

- e.g.

```
$ my_csv "select * from world.City"
```

inside the command line client

handling output

pager

```
mysql> SELECT * FROM world.City;

rec …
rec …
rec …

4000 + records
unreadable
```

pager

```
mysql> pager less
PAGER set to 'less'
mysql> SELECT * FROM world.City;

# [ records in a manageable window ]

mysql> nopager
PAGER set to stdout

# back to normal
```

More pager tricks

```
mysql> pager vim -
# like 'less', but you can also edit
# the results and eventually save
# them
```

More pager tricks

```
mysql> pager md5sum
mysql> pager sha1sum

# produce a CRC for the result set
# instead of displaying it.
# Useful for data comparison.
# (Thanks to Baron Schwartz)
```

More pager tricks

```
mysql> pager cat > /tmp/t1.txt
mysql> select "one" union select
"two";
mysql> pager cat > /tmp/t2.txt
mysql> select "one" union select
"TWO";
mysql> nopager
mysql> \! vimdiff -o /tmp/t[12].txt

# see the difference between two
# datasets
```

logging sessions

```
mysql> tee mylog.txt

# logs everything that you see on
# the screen.
# your commands, the results,
# warnings, error messages, and so on.

mysql> notee

# back to normal
```

Updated presentation slides

- you will find an up to date copy of these slides at:
- http://tinyurl.com/cli-tut-2010

read more

- http://datacharmer.blogspot.com

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

THANKS!

Question time

Comment on Twitter:

@datacharmer