



Infobright Evaluation Report

**Infobright Community Edition version 3.3.2-beta
Calpont InfiniDB Community Edition 1.5**

Percona Inc, 02 August 2010

Version 1.0 – Authored by Justin Swanhart

Table of Contents

Summary.....	3
Aspects evaluated.....	3
Software used.....	3
Evaluation Results	4
DDL and datatype support	4
Loading Data.....	4
Compression and data size.....	5
Ease of installation and security evaluation.....	5
Ability to query over large data sets.....	6
Query Time Chart.....	6
Conclusions.....	7
Appendix – Query Times and Query list.....	8
q#1.....	9
q#2.....	9
q#3.....	9
q#4.....	10
q#5.....	11
q#6.....	12
q#7.....	13
q#8.....	14
q#9.....	15
q#10.....	15
q#11.....	16
q#12.....	16
q#13.....	16
q#14.....	16
q#15.....	17
q#16.....	17
q#17.....	17
q#18.....	17
q#19.....	17
q#20.....	17
q#21.....	17
q#22.....	18
q#23.....	18
q#24.....	18
q#25.....	19
q#26.....	19
q#27.....	20
q#28.....	21
q#29.....	22

Summary

In accordance with the Infobright “statement of work”, Percona evaluated the functional characteristics of Infobright Community Edition.

Our work was performed in two different series of tests. The first test was an evaluation of the ICE functionality via a set of tests designed to execute the features which Infobright most wanted to test. These tests were provided to Todd and a number of bug reports were filed based on the results of the testing. These tests were done on a small synthetic dataset consisting of only 256 rows.

The second set of testing was done on a 900GB data set provided by Infobright. This data was loaded into both Infobright CE and InfiniDB CE. A series of 29 test queries, the bulk of which were provided by Infobright were executed against both MySQL databases using the 'mysqltest' MySQL testing binary.

This report is focused on the second phase of testing, comparing ICE with InfiniDB.

Aspects evaluated

- DDL and datatype support
- Time to set up the database and load the data
- Size of the loaded database on disk and compression
- Ease of installation and security
- Ability to query over large data sets without errors or crashes
- 29 queries over the carsales database

Software used

- Infobright Community Edition 3.3.2-beta – LOAD DATA INFILE for loading
- InfiniDB 1.5 GA – cpimport utility for loading
- mysqltest 5.1.45 (not distributed with InfiniDB binaries)
- PHP and bash scripts – wrappers for tests and for timing the data loading

Evaluation Results

DDL and datatype support

Infobright supports a number of MySQL data types which are not supported by InfiniDB. The data types below list the type supported by ICE, and what datatype was used on InfiniDB as substitute:

- Year – smallint was used instead
- Time – char(10) was used instead
- Tinytext – varchar(255)
- NOT NULL – NULL was used instead

InfiniDB does not support comments in DDL statements. All 'lookup' comments had to be stripped out of DDL before the table could be created in InfiniDB.

Neither database gives very good error messages when an unsupported datatype is used. The end user experience could be improved by listing the column name and data type that was not supported. Ideally each unsupported feature could be enumerated in warnings so that all the problems could be identified at once.

Loading Data

Infobright CE supports the MySQL “LOAD DATA INFILE” syntax, but it does not do so in a way that is compatible with the default MySQL settings, which can be confusing. I extracted each of the compressed files provided by Infobright and loading them individually with LOAD DATA INFILE.

InfiniDB has a custom loader which is invoked using the 'cpimport' utility. This utility uses a 'job file' system and the jobs must be created by another utility called 'colxml'. Overall the complexity of loading is much higher on InfiniDB than on Infobright.

In both cases, the end user experience might be better if the loader was more compatible with MySQL, including support for escape characters and optional text enclosures.

Loading Times

Infobright Community 3.3.2-beta	28.07 hours total / avg of 1110 seconds per file
InfiniDB 1.5	19.01 hours total / avg of 764 seconds per file

Compression and data size

Infobright Community Edition compresses data significantly when compared to the size of the input data. InfiniDB does not appear to compress data. In fact, the size on disk of the InfiniDB database is slightly larger than the size of the source material.

Size of software and data on disk (905GB source data)

Infobright Community 3.3.2-beta	122GB
InfiniDB 1.5	934GB

Ease of installation and security evaluation

Since both database engines are distributed in both binary and source form. I tested the installation of the databases from RPM files. Both databases had to be installed as the 'root' user. Both databases install their files into a sub=directory of /usr/local. There were additional scripts to run to install the database on InfiniDB, but ICE worked out of the box. InfiniDB uses the standard MySQL port of 3306, but ICE runs on an alternate port by default which can be confusing.

Since ICE supports LOAD DATA INFILE, there is no additional security work which needs to be configured to give end users the ability to load data into the database. Any MySQL user with adequate permissions can load files. The ICE loader can read from named pipes.

While InfiniDB supports loading data with LOAD DATA INFILE, I found the performance to be unacceptable and instead chose to use the combination of tools: colxml and cpimport. The first utility creates an XML "job" file, and the second uses said files to load data. I was unable to get the loading utility to support loading more than one file into a table in a single job file. I used a symlink, a shell script and a hand crafted job file to load each file into the database. To further complicate matters, the job files must be placed into a particular directory, which complicates security from a unix perspective. InfiniDB requires the use of terminal aliases by default, which is inconvenient.

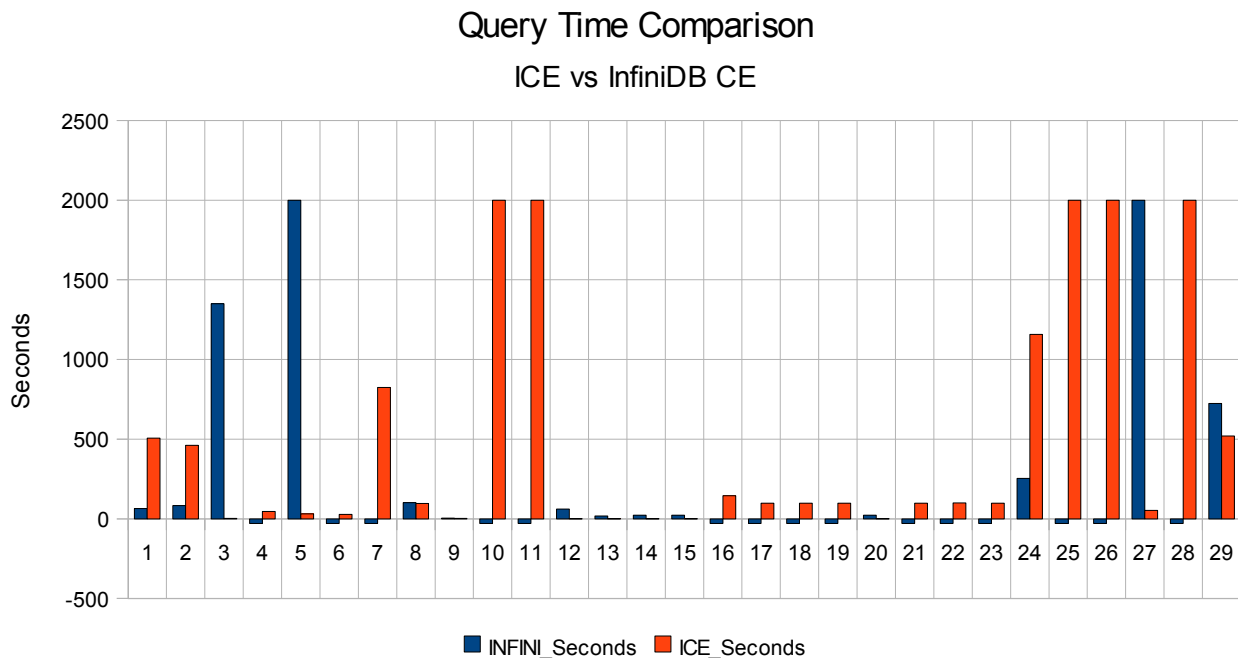
Comparison (all points are out of five)

	Security	Ease of loading/use
Infobright Community 3.3.2-beta	5	4
InfiniDB 1.5	3	3

Ability to query over large data sets

ICE was able to successfully execute all 29 of the test queries which were executed against it. InfiniDB could not execute a large number of the queries. InfiniDB does not support all of the MySQL aggregate functions, and it requires non-default values to run certain queries, or errors such as 'not enough memory for aggregation' can occur. Overall ICE appears to be able to run substantially more queries than InfiniDB due to two factors. First, ICE supports more data types and second, ICE supports more aggregate functions. The results on InfiniDB are questionable, as it gives wrong results for even a simple SELECT COUNT(*) with no where clause.

Query Time Chart



Note:

Queries that could not be run on InfiniDB appear as negative values on the chart.

Some queries may have taken longer than 2000 seconds. Please see chart below.

Conclusions

Overall, on the given data set and for the given queries, ICE performed better and could execute more queries than InfiniDB. Query results differed from ICE and InfiniDB due to bugs, data type differences and/or precision differences. In my opinion ICE is easier to use, performs better and returns more accurate results than InfiniDB at the current time.

We have not evaluated the newest release of ICE, version 3.4, but we look forward to doing so in the next round of testing.

Appendix – Query Times and Query list

Query	ICE_TIME	INFINIDB_TIME
1	505.96	64.82
2	460.39	83.07
3	2.06	1349.66
4	45.97	0
5	31.57	2087.88
6	27.75	0
7	824.36	0
8	96.39	101.15
9	1.9	4.73
10	3667.92	0
11	8105.44	0
12	0.9	60.49
13	0.36	18.42
14	0.25	22.45
15	0.24	22.45
16	144.4	0
17	98.38	0
18	98.6	0
19	97.56	0
20	0.35	23.05
21	97.94	0
22	98.7	0
23	98.16	0
24	1156.59	253.66
25	4320.64	0
26	4036.14	0
27	53.08	2194.49
28	3809.02	0
29	519.16	723.07

Time is in seconds

zero time means an error prevented the query from completing

Query List

q#1

```
select dim_dates.trans_year as c0,  
       sum(sales_commission) as m0  
from fact_sales, dim_dates  
where fact_sales.trans_date=dim_dates.trans_date  
and dim_dates.trans_year=2005  
group by c0;
```

q#2

```
select dim_dates.trans_year as c0,  
       sum(sales_commission) as m0  
from dim_dates, fact_sales  
where fact_sales.trans_date=dim_dates.trans_date  
and dim_dates.trans_year=2005  
group by c0;
```

q#3

```
select count(distinct(car_colour)) from fact_sales;
```

q#4

```
-- sorted_result
select
  d.dealer_name,
  sum(f.dlr_trans_amt)
from
  fact_sales f,
  dim_dealers d,
  dim_msa md
where
  f.dealer_id = d.dealer_id
and
  f.trans_date in
  (
    select
      td.trans_date
    from
      dim_dates td
    where
      td.trans_month='JANUARY'
    and
      td.trans_year=2001
  )
and
  f.msa_id = md.msa_id
and
  md.msa_name in ('GREEN BAY', 'AMARILLO', 'GREELEY', 'ERIE', 'OKLAHOMA
CITY', 'ROCHESTER', 'COLUMBIA', 'DURHAM', 'AKRON', 'CHATTANOOGA')
group by
  d.dealer_name;
```

q#5

```
-- sorted_result
select
  d.dealer_name,
  sum(f.dlr_trans_amt) as sale
from
  fact_sales f,
  dim_dealers d,
  dim_cars md
where
  f.dealer_id = d.dealer_id
and
  f.trans_date in
  (
    select
      td.trans_date
    from
      dim_dates td
    where
      td.trans_month='JANUARY'
    and
      td.trans_year=2001
  )
and
  md.make_id = f.make_id
and
  md.make_name = 'ACURA'
group by
  d.dealer_name;
```

q#6

```
select
  d.dealer_name,
  sum(f.dlr_trans_amt) as total
from
  fact_sales f,
  dim_dealers d,
  dim_cars md,
  dim_msa mkt
where
  f.dealer_id = d.dealer_id
and
  f.trans_date in
  (
    select
      td.trans_date
    from
      dim_dates td
    where
      td.trans_month='JANUARY'
    and
      td.trans_year=2001
  )
and
  md.make_id = f.make_id
and
  mkt.msa_id = f.msa_id
and
  mkt.msa_name in ('GREEN BAY', 'AMARILLO', 'GREELEY', 'ERIE', 'OKLAHOMA
CITY', 'ROCHESTER', 'COLUMBIA', 'DURHAM', 'AKRON', 'CHATTANOOGA')
and
  md.make_name = 'ACURA'
group by
  d.dealer_name;
```

q#7

```
select
  d.dealer_name,
  sum(f.dlr_trans_amt) as total
from
  fact_sales f,
  dim_dealers d,
  dim_cars md,
  dim_msa mkt
where
  f.dealer_id = d.dealer_id
and
  f.trans_date in
  (
    select
      td.trans_date
    from
      dim_dates td
    where
      td.trans_year = 2001
  )
and
  md.make_id = f.make_id
and
  mkt.msa_id = f.msa_id
and
  mkt.msa_name in ('GREEN BAY', 'AMARILLO', 'GREELEY', 'ERIE', 'OKLAHOMA
CITY', 'ROCHESTER', 'COLUMBIA', 'DURHAM', 'AKRON', 'CHATANOOGA')
and
  md.make_name = 'ACURA'
group by
  d.dealer_name;
```

q#8

```
-- sorted_result
select
  d.dealer_name,
  sum(f.dlr_trans_amt) as total
from
  fact_sales f,
  dim_dealers d
where
  f.dealer_id = d.dealer_id
and
  f.trans_date in
  (
    select
      td.trans_date
    from
      dim_dates td
    where
      td.trans_year=2001
    and
      td.trans_month in ('JANUARY','FEBRUARY','MARCH')
  )
and
  d.dealer_state in ('OH','NY','MI','CA','NV')
group by
  d.dealer_name;
```

q#9

```
select
    d.dealer_name,
    sum(f.dlr_trans_amt),
    sum(f.sales_commission),
    sum(f.sales_discount)
from
    fact_sales f,
    dim_dealers d,
    dim_cars c
where
    f.trans_date between '2001-01-01' and '2001-01-01'
and
    c.make_id = f.make_id
and
    d.dealer_id = f.dealer_id
and
    c.make_name = 'ACURA'
group by
    d.dealer_name;
```

q#10

```
select dealer_id, msa_id, dlr_trans_amt from fact_sales order by 1, 2, 3 limit 2000;
```

q#11

```
SELECT car_year,
       car_colour,
       sales_person,
       count(DISTINCT dim_dealers.dealer_id) AS dealer_cnt
FROM fact_sales, dim_dealers
WHERE fact_sales.dealer_id = dim_dealers.dealer_id
      AND car_colour <> 'YELLOW'
      AND sales_person NOT LIKE 'RA%'
      AND car_year IN (2000, 2003, 2005)
      AND fact_sales.make_id NOT IN (SELECT make_id
                                     FROM dim_cars
                                     WHERE model_name LIKE 'E%X%')
GROUP BY car_year, car_colour, sales_person
ORDER BY dealer_cnt DESC, car_year, car_colour, sales_person;
```

q#12

```
select AVG(sales_commission)                                from fact_sales where trans_year=2002;
```

q#13

```
select COUNT(sales_commission)                            from fact_sales where trans_year=2002;
```

q#14

```
select MAX(sales_commission)                              from fact_sales where trans_year=2002;
```

q#15

```
select MIN(sales_commission) from fact_sales where trans_year=2002;
```

q#16

```
select STD(sales_commission) from fact_sales where trans_year=2002;
```

q#17

```
select STDDEV_POP(sales_commission) from fact_sales where trans_year=2002;
```

q#18

```
select STDDEV_SAMP(sales_commission) from fact_sales where trans_year=2002;
```

q#19

```
select STDDEV(sales_commission) from fact_sales where trans_year=2002;
```

q#20

```
select SUM(sales_commission) from fact_sales where trans_year=2002;
```

q#21

```
select VAR_POP(sales_commission) from fact_sales where trans_year=2002;
```

q#22

```
select VAR_SAMP(sales_commission)          from fact_sales where trans_year=2002;
```

q#23

```
select VARIANCE(sales_commission)         from fact_sales where trans_year=2002;
```

q#24

```
-- sorted_result
select
  d.dealer_state,
  sum(f.dlr_trans_amt)
from
  fact_sales f,
  dim_dealers d,
  dim_dates dd
where
  f.dealer_id = d.dealer_id
and
  f.trans_date = dd.trans_date
and dd.trans_year = 2005
group by
  d.dealer_state;
```

q#25

```
select AVG(sales_commission),
COUNT(sales_commission),
MAX(sales_commission),
MIN(sales_commission),
STD(sales_commission),
STDDEV_POP(sales_commission),
STDDEV_SAMP(sales_commission),
STDDEV(sales_commission),
SUM(sales_commission),
VAR_POP(sales_commission),
VAR_SAMP(sales_commission),
VARIANCE(sales_commission),
COUNT(0) cnt_1,
COUNT(*) cnt_2
from fact_sales;
```

q#26

```
-- sorted_result
select year(f.trans_date), dealer_name, sum( sales_commission * dlr_trans_amt )
  from fact_sales f
  join dim_dealers using (dealer_id)
where
  (trans_year in (2004,2005))
group by year(f.trans_date),
         dealer_name;
```

q#27

```
select
  d.dealer_name,
  sum(f.dlr_trans_amt) as sale
from
  fact_sales f,
  dim_dealers d,
  dim_cars md,
  dim_dates dd
where
  f.dealer_id = d.dealer_id
and
  f.trans_date = dd.trans_date
and
  dd.trans_month='JANUARY'
and
  dd.trans_year=2001
and
  md.make_id = f.make_id
and
  md.make_name IN('HONDA','ACURA','OLDSMOBILE')
group by
  d.dealer_name;
```

q#28

```
-- sorted_result
select
  mkt.msa_name,
  md.make_name,
  d.dealer_name,
  sum(f.dlr_trans_amt) as total
from
  fact_sales f,
  dim_dealers d,
  dim_cars md,
  dim_msa mkt,
  dim_dates dd
where
  f.dealer_id = d.dealer_id
and
  f.trans_date = dd.trans_date
and
  dd.trans_year=2004
and
  md.make_id = f.make_id
and
  mkt.msa_id = f.msa_id
and
  mkt.msa_name not in ('GREEN BAY', 'AMARILLO', 'GREELEY', 'ERIE', 'OKLAHOMA
CITY', 'ROCHESTER', 'COLUMBIA', 'DURHAM', 'AKRON', 'CHATTANOOGA')
group by
  1,2,3;
```

q#29

```
select
  d.dealer_name,
  sum(f.dlr_trans_amt) as total
from
  fact_sales f,
  dim_dealers d
where
  f.dealer_id = d.dealer_id
and
  f.trans_date in
  (
    select
      td.trans_date
    from
      dim_dates td
    where
      (td.trans_year=2001 or td.trans_year=2002 or td.trans_year < 2001)
    and
      td.trans_month in ('JANUARY','FEBRUARY','MARCH')
  )
and
  d.dealer_state not in ('OH','NY','MI','CA','NV')
group by
  d.dealer_name;
```