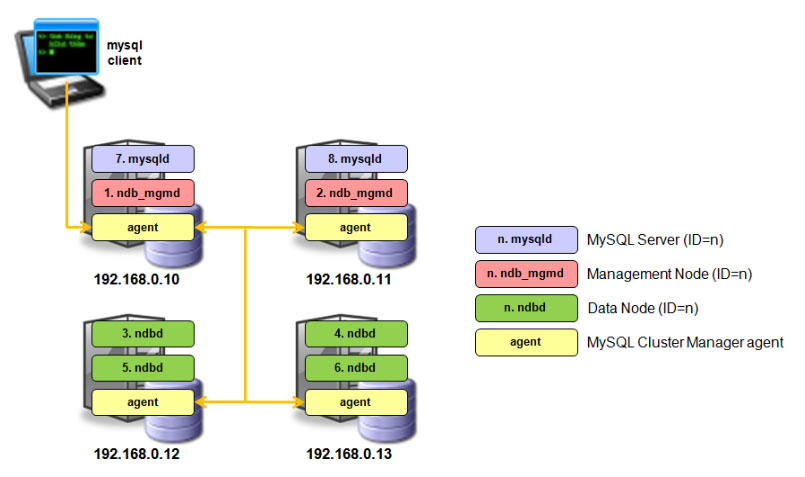
MSc business analytics

Data management & Business intelligence  
assignment 2: MySQl cluster

Chatzimoschou Angeliki

BAPTXXXX



**Professor**: Vassalos Vassilios

**Supervisor**: Kazadeis Ioannis

**Academic Year**: 2015-2016

# Άσκηση 1

1. Δημιουργούμε τον πίνακα PERSON:

CREATE TABLE PERSON (

id BIGINT NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

address VARCHAR(100) NOT NULL,

gender VARCHAR(6) NOT NULL,

email VARCHAR(80) NOT NULL,

PRIMARY KEY (id, last\_name)

)

ENGINE=ndbcluster   
DEFAULT CHARSET=latin1;

Εισάγουμε τις εγγραφές εκτελώντας το script αρχείο “person.sql”:

INSERT INTO ‘PERSON’ VALUES (….)

Ελέγχουμε την κατανομή των εγγραφών:

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='PERSON' AND table\_schema='askisi1\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *246* |
| *p1* |  |  | *253* |
| *p2* |  |  | *234* |
| *p3* |  |  | *267* |

Παρατηρούμε ότι οι εγγραφές έχουν ισοκατανεμηθεί στους 4 datanodes .

1. Δημιουργούμε τον πίνακα PERSON\_2:

CREATE TABLE PERSON\_2 (

id BIGINT NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

address VARCHAR(100) NOT NULL,

gender VARCHAR(6) NOT NULL,

email VARCHAR(80) NOT NULL,

PRIMARY KEY (id,last\_name,gender)

)

ENGINE=ndbcluster   
DEFAULT CHARSET=latin1  
PARTITION by key (gender);

Εισάγουμε τις εγγραφές (αφού δημιουργήσουμε) εκτελώντας το script αρχείο “person\_2.sql”:

INSERT INTO ‘PERSON\_2’ VALUES (….)

Ελέγχουμε την κατανομή των εγγραφών:

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='PERSON\_2' AND table\_schema='askisi1\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *0* |
| *p1* |  |  | *1000* |
| *p2* |  |  | *0* |
| *p3* |  |  | *0* |

Παρατηρούμε ότι όλες οι εγγραφές έχουν κατανεμηθεί σε 1 datanode.

Δημιουργούμε τον πίνακα PERSON\_3:

CREATE TABLE PERSON\_3 (

id BIGINT NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

address VARCHAR(100) NOT NULL,

gender VARCHAR(55) NOT NULL,

email VARCHAR(80) NOT NULL,

PRIMARY KEY (id,last\_name,gender)

)

ENGINE=ndbcluster   
DEFAULT CHARSET=latin1  
PARTITION by key (gender);

Ομοίως εισάγουμε τις εγγραφές εκτελώντας το script αρχείο “person\_3.sql”:

INSERT INTO ‘PERSON\_3’ VALUES (….)

Ελέγχουμε την κατανομή των εγγραφών:

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='PERSON\_3' AND table\_schema='askisi1\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *0* |
| *p1* |  |  | *493* |
| *p2* |  |  | *0* |
| *p3* |  |  | *507* |

Παρατηρούμε ότι όλες οι εγγραφές έχουν κατανεμηθεί ανάλογα με το gender σε 2 data nodes.  
  
Καταλήγοντας από τις δύο δοκιμές μας, μπορούμε να πούμε ότι η διαδικασία που ακολου8εί η hash function της MySQL εξαρτάται από το μέγεθος που θα ορίσουμε στο πεδίο Gender.

# Άσκηση 2

1. Join pushdown Απενεργοποιημένο

set ndb\_join\_pushdown=off;

SELECT COUNT(\*) FROM person JOIN postcode ON person.zipcode=postcode.zip

WHERE latitude>=-13.45 AND latitude<=56.57 AND longitude>=-67.12 AND longitude<=132.22;

RESULTS

COUNT(\*)

194

Duration/Fetch: 59.812 sec / 0.000 sec

Join pushdown Eνεργοποιημένο

set ndb\_join\_pushdown=on;

SELECT COUNT(\*) FROM person JOIN postcode ON person.zipcode=postcode.zip

WHERE latitude>=-13.45 AND latitude<=56.57 AND longitude>=-67.12 AND longitude<=132.22;

RESULTS

COUNT(\*)

194

Duration/Fetch: 1.031 sec / 0.000 sec

Παρατηρούμε ότι υπάρχει μεγάλη διαφορά στους χρόνους ανταπόκρισης όταν το Join Pushdown είναι ενεργοποιημένο. Αυτό συμβαίνει διότι με το Adaptive Query Localization όλη η δουλειά διαμοιράζεται στους data nodes και επιτυγχάνεται παραλληλία.

1. Join pushdown Απενεργοποιημένο

set ndb\_join\_pushdown=off;

SELECT COUNT(\*) FROM person JOIN postcode2 ON person.zipcode=postcode2.zip

WHERE latitude>=-13.45 AND latitude<=56.57 AND longitude>=-67.12 AND longitude<=132.22;

RESULTS

COUNT(\*)

194

Duration/Fetch: 60.016 sec / 0.000 sec

Join pushdown Eνεργοποιημένο

set ndb\_join\_pushdown=on;

SELECT COUNT(\*) FROM person JOIN postcode2 ON person.zipcode=postcode2.zip

WHERE latitude>=-13.45 AND latitude<=56.57 AND longitude>=-67.12 AND longitude<=132.22;

RESULTS

COUNT(\*)

194

Duration/Fetch: 59.703 sec / 0.000 sec

Δεν υπάρχει κάποια αλλαγή στους χρόνους.

Για να διερευνήσουμε γιατί συμβαίνει αυτό θα χρησιμοποιήσουμε τις εντολές EXPLAIN EXTENDED SELECT και SHOW WARNINGS.

set ndb\_join\_pushdown=on;

EXPLAIN EXTENDED SELECT COUNT(\*) FROM person JOIN postcode2 ON person.zipcode=postcode2.zip

WHERE latitude>=-13.45 AND latitude<=56.57 AND longitude>=-67.12 AND longitude<=132.22;

RESULTS

# id, select\_type, table, type, possible\_keys, key, key\_len, ref, rows, filtered, Extra

'1', 'SIMPLE', 'person', 'ALL', NULL, NULL, NULL, NULL, '24961', '100.00', NULL

'1', 'SIMPLE', 'postcode2', 'eq\_ref', 'PRIMARY', 'PRIMARY', '8', 'people.person.zipcode', '1', '100.00', 'Using where with pushed condition ((`people`.`postcode2`.`latitude` >= <cache>(-(13.45))) and (`people`.`postcode2`.`latitude` <= 56.57) and (`people`.`postcode2`.`longitude` >= <cache>(-(67.12))) and (`people`.`postcode2`.`longitude` <= 132.22))'

SHOW WARNINGS;

RESULTS

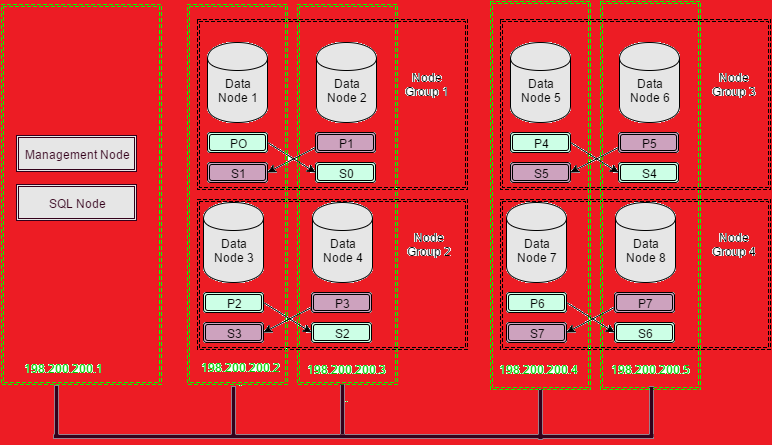
Level, Code, Message

'Note', '1003', 'Can\'t push table \'postcode2\' as child, column \'zip\' does not have same datatype as ref\'ed column \'person.zipcode\''

'Note', '1003', '/\* select#1 \*/ select count(0) AS `COUNT(\*)` from `people`.`person` join `people`.`postcode2` where ((`people`.`postcode2`.`latitude` >= <cache>(-(13.45))) and (`people`.`postcode2`.`latitude` <= 56.57) and (`people`.`postcode2`.`longitude` >= <cache>(-(67.12))) and (`people`.`postcode2`.`longitude` <= 132.22) and (`people`.`person`.`zipcode` = `people`.`postcode2`.`zip`))'

Συμπεραίνουμε ότι η στήλη ‘zip’ έχει διαφορετικό data type στον πίνακα postcode2 από τον πίνακα person.  
Συνεπώς δεν πληρείται μία από τις προϋποθέσεις εφαρμογής της διαδικασίας:  
JOINed columns must have the same data type.

# Άσκηση 3



Στην τοπολογία έχουμε τα εξής:

1 Management Node

1 SQL Node

8 Data Nodes

8 Partitions

2 replicas per partition

8/2 – 4 node groups

Η παρούσα τοπολογία στήθηκε με αυτόν τον τρόπο διότι:

Σύμφωνα με τη θεωρία τα data nodes έχουν μεγάλες απαιτήσεις σε μνήμη, επεξεργαστή και δίσκο ενώ ο SQL node έχει μεγάλες απαιτήσεις σε επεξεργαστή και ανάλογα με τον φόρτο εργασίας και σε μνήμη. Αντιθέτως ο management node έχει γενικά χαμηλές απαιτήσεις.

Έτσι λοιπόν θα πρέπει:

Ο management node να μην είναι μαζί με τα data nodes (split brain)

Ο SQL node να μην είναι μαζί με τα Data Nodes αποφεύγοντας (CPU & RAM competition)

Και τέλος επιτρέπεται ο SQL node να είναι μαζί με τον management node.

ii) Configuration File

[ndbd default]

# Options affecting ndbd processes on all data nodes:

NoOfReplicas=2 # Number of replicas

DataDir=C:/mysql/cluster-data #Directory for each data node’s files

DataMemory=2000M # Memory allocated to data storage

IndexMemory=500M # How much memory to allocate for index storage

[ndb\_mgmd]

# Management process options:

hostname=198.200.200.1 # Hostname or IP address of management node

datadir=C:/mysql/bin/cluster-logs # Directory for MGM node log files

[ndbd]

# Options for data node "1":

hostname=198.200.200.2 # Hostname or IP address

[ndbd]

# Options for data node "2":

hostname=198.200.200.3 # Hostname or IP address

[ndbd]

# Options for data node "3":

hostname=198.200.200.2 # Hostname or IP address

[ndbd]

# Options for data node "4":

hostname=198.200.200.3 # Hostname or IP address

[ndbd]

# Options for data node "5":

hostname=198.200.200.4 # Hostname or IP address

[ndbd]

# Options for data node "6":

hostname=198.200.200.5 # Hostname or IP address

[ndbd]

# Options for data node "7":

hostname=198.200.200.4 # Hostname or IP address

[ndbd]

# Options for data node "8":

hostname=198.200.200.5 # Hostname or IP address

[mysqld]

# SQL node options:

hostname=198.200.200.1 # Hostname or IP address

*iii) Δύο data nodes αποτυγχάνουν. Το cluster παύει να λειτουργεί.*

Αυτό το σενάριο είναι δυνατόν να συμβεί αν τα δύο data nodes ανήκουν στο ίδιο nodegroup.

*Τέσσερις data nodes αποτυγχάνουν, αλλά το cluster συνεχίζει να λειτουργεί κανονικά.*

Αυτό το σενάριο είναι δυνατόν να συμβεί αν και τα τέσσερα data nodes ανήκουν σε διαφορετικά nodegroups.

*Τρεις data nodes σε διαφορετικά nodegroups αποτυγχάνουν. Το cluster παύει να λειτουργεί.* Αυτό το σενάριο δεν είναι δυνατόν να συμβεί γιατί τα τρία data nodes ανήκουν σε διαφορετικά nodegroups κι έτσι το cluster θα συνεχίσει να λειτουργεί.

# Άσκηση 4

1. Δημιουργούμε τους πίνακες EMP, CHILD και BRANCH και εισάγουμε τις εγγραφές:

CREATE TABLE EMP (

EMP\_ID BIGINT NOT NULL AUTO\_INCREMENT,

Last\_Name VARCHAR(50) NOT NULL,

First\_Name VARCHAR(50) NOT NULL,

POS\_ID INT(10) NOT NULL,

Emp\_Year DATETIME NOT NULL,

Branch VARCHAR(10) NOT NULL,

Emp\_Salary DECIMAL(8,2) NOT NULL,

Education INT(5) NOT NULL,

Status VARCHAR(15) NOT NULL,

PRIMARY KEY (EMP\_ID)

) TABLESPACE ts\_1 STORAGE DISK

ENGINE=ndbcluster

DEFAULT CHARSET=latin1;

Εισάγουμε τις εγγραφές εκτελώντας το emp script:

INSERT INTO ‘EMP’ VALUES (…….)

CREATE TABLE CHILD (

EMP\_ID BIGINT NOT NULL AUTO\_INCREMENT,

Child\_LN VARCHAR(50) NOT NULL,

Child\_FN VARCHAR(50) NOT NULL,

Date\_of\_Birth DATE DEFAULT NULL,

PRIMARY KEY (EMP\_ID, Child\_LN, Child\_FN)

) TABLESPACE ts\_1 STORAGE DISK

ENGINE=ndbcluster   
DEFAULT CHARSET=latin1;

Εισάγουμε τις εγγραφές εκτελώντας το CHILD script:

INSERT INTO ‘CHILD’ VALUES (…….)

CREATE TABLE BRANCH (

Branch VARCHAR(10) NOT NULL,

City VARCHAR(25) NOT NULL,

PRIMARY KEY (branch)

) TABLESPACE ts\_1 STORAGE DISK

ENGINE=ndbcluster DEFAULT CHARSET=latin1;

Εισάγουμε τις εγγραφές εκτελώντας το BRANCH script:

INSERT INTO ‘BRANCH’ VALUES (…….)

Ελέγχουμε την κατανομή των εγγραφών στον πίνακα EMP:

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='EMP' AND table\_schema='askisi4\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *24747* |
| *p1* |  |  | *25320* |
| *p2* |  |  | *24932* |
| *p3* |  |  | *25001* |

Ελέγχουμε την κατανομή των εγγραφών στον πίνακα CHILD:

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='CHILD' AND table\_schema='askisi4\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *17672* |
| *p1* |  |  | *17271* |
| *p2* |  |  | *17820* |
| *p3* |  |  | *17237* |

Ελέγχουμε την κατανομή των εγγραφών στον πίνακα BRANCH:

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='BRANCH' AND table\_schema='askisi4\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *2* |
| *p1* |  |  | *1* |
| *p2* |  |  | *1* |
| *p3* |  |  | *0* |

ii)   
 a) ΜΟ Μισθοδοσίας ανά κατάστημα

SELECT BRANCH.BRANCH, BRANCH.CITY, AVG(EMP.EMP\_SALARY) AS AVERAGE\_SALARY

FROM EMP JOIN BRANCH ON EMP.BRANCH= BRANCH.BRANCH

GROUP BY EMP.BRANCH

ORDER BY AVERAGE\_SALARY DESC

**Duration / Fetch** **2.047 sec / 0.000 sec**

RESULTS

*BRANCH CITY AVERAGE\_SALARY*

*PRA Prague 2250.642089*

*ATH Athens 2249.037669*

*BUCH Bucharest 2247.846624*

*BE Berlin 2241.910752*

*b)* ΜΟ Μισθοδοσίας ανά κατάστημα και για υπαλλήλους με ένα παιδί και πάνω

SELECT BRANCH.BRANCH, BRANCH.CITY, AVG(EMP.EMP\_SALARY) AS AVERAGE\_SALARY

FROM EMP JOIN BRANCH ON EMP.BRANCH= BRANCH.BRANCH

WHERE EMP.EMP\_ID IN (SELECT DISTINCT CHILD.EMP\_ID FROM CHILD)   
GROUP BY EMP.BRANCH

ORDER BY AVERAGE\_SALARY DESC

**Duration / Fetch 10938 sec / 0.000 sec**

RESULTS

*BRANCH CITY AVERAGE\_SALARY*

*PRA Prague 2256.895567*

*ATH Athens 2256.123135*

*BUCH Bucharest 2243.642458*

*BE Berlin 2231.792864*

iii)

1. Δημιουργούμε τον EMP\_2 με partition στο BRANCH και έτσι το query θα εκτελεστεί πιο γρήγορα αφού οι εγγραφές ανά BRANCH θα είναι στο ίδιο data node.

CREATE TABLE EMP\_2 (

EMP\_ID BIGINT NOT NULL AUTO\_INCREMENT,

Last\_Name VARCHAR(50) NOT NULL,

First\_Name VARCHAR(50) NOT NULL,

POS\_ID INT(10) NOT NULL,

Emp\_Year DATETIME NOT NULL,

Branch VARCHAR(10) NOT NULL,

Emp\_Salary DECIMAL(8,2) NOT NULL,

Education INT(5) NOT NULL,

Status VARCHAR(15) NOT NULL,

PRIMARY KEY (EMP\_ID,Branch)

) TABLESPACE ts\_1 STORAGE DISK

ENGINE=ndbcluster DEFAULT CHARSET=latin1

PARTITION BY KEY (Branch);

Εισάγουμε τις εγγραφές (αφού δημιουργήσουμε νέο script) εκτελώντας το emp\_2 script :

INSERT INTO ‘EMP\_2’ VALUES (…….)

Ελέγχουμε την κατανομή των εγγραφών

SELECT partition\_name, table\_rows FROM information\_schema.PARTITIONS

WHERE table\_name='EMP\_2' AND table\_schema='askisi4\_35';

RESULTS

*partition\_name, table\_rows*

|  |  |  |  |
| --- | --- | --- | --- |
| *p0* |  |  | *49636* |
| *p1* |  |  | *47015* |
| *p2* |  |  | *3349* |
| *p3* |  |  | *0* |
|  |  |  |  |

Παρατηρούμε ότι οι εγγραφές κατανεμήθηκαν σε 3 partitions.

ΜΟ μισθοδοσίας με χρήση EMP\_2

SELECT BRANCH.Branch, BRANCH.City, AVG(EMP\_2.Emp\_Salary) AS AVERAGE\_SALARY

FROM EMP\_2 JOIN BRANCH ON EMP\_2.Branch=BRANCH.Branch

GROUP BY EMP\_2.Branch

ORDER BY AVERAGE\_SALARY DESC;

RESULTS

*BRANCH CITY AVERAGE\_SALARY*

*PRA Prague 2250.642089*

*ATH Athens 2249.037669*

*BUCH Bucharest 2247.846624*

*BE Berlin 2241.910752*

**Duration / Fetch** **0.316 sec / 0.000 sec**

O χρόνος μειώθηκε από 2.047 sec σε 0.316 sec.

b) Μ.Ο. μισθοδοσίας εργαζομένων με παιδιά με χρήση EMP\_2

SELECT BRANCH.Branch, BRANCH.City, AVG(EMP\_2.Emp\_Salary) AS AVERAGE\_SALARY

FROM EMP\_2 JOIN BRANCH ON EMP\_2.Branch=BRANCH.Branch

WHERE EMP\_2.EMP\_ID IN (SELECT DISTINCT CHILD.EMP\_ID FROM CHILD)

GROUP BY EMP\_2.Branch

ORDER BY AVERAGE\_SALARY DESC;

RESULTS

*BRANCH CITY AVERAGE\_SALARY*

*PRA Prague 2256.895567*

*ATH Athens 2256.123135*

*BUCH Bucharest 2243.642458*

*BE Berlin 2231.792864*

**Duration/Fetch: 113.453 sec / 0.000 sec**

Έχουμε χειρότερο χρόνο με τη χρήση του EMP\_2 με διάρκεια 113.453sec.

Έτσι θα φτιάξουμε τον CHILD\_2 με partition στο EMP\_ID για να έχουμε στο ίδιο data node τους γονείς ενός παιδιού.

CREATE TABLE CHILD\_2 (

EMP\_ID BIGINT NOT NULL AUTO\_INCREMENT,

Child\_LN VARCHAR(50) NOT NULL,

Child\_FN VARCHAR(50) NOT NULL,

Date\_of\_Birth DATE DEFAULT NULL,

PRIMARY KEY (EMP\_ID, Child\_LN, Child\_FN)

) TABLESPACE ts\_1 STORAGE DISK

ENGINE=ndbcluster DEFAULT CHARSET=latin1

Partition by key (EMP\_ID);

Εισάγουμε τις εγγραφές (αφού δημιουργήσουμε νέο script) εκτελώντας το child\_2 script :

INSERT INTO ‘CHILD\_2’ VALUES (…….)

Ελέγχουμε την κατανομή των εγγραφών

SELECT BRANCH.Branch, BRANCH.City, AVG(EMP.Emp\_Salary) AS AVERAGE\_SALARY

FROM EMP JOIN BRANCH ON EMP.Branch=BRANCH.Branch

WHERE EMP.EMP\_ID IN (SELECT DISTINCT CHILD\_2.EMP\_ID FROM CHILD\_2)

GROUP BY EMP.Branch

ORDER BY AVERAGE\_SALARY DESC;

RESULTS

*BRANCH CITY AVERAGE\_SALARY*

*PRA Prague 2256.895567*

*ATH Athens 2256.123135*

*BUCH Bucharest 2243.642458*

*BE Berlin 2231.792864*

**Duration/Fetch: 10.734 sec / 0.000 sec**

Παρατηρούμε ότι ο χρόνος βελτιώθηκε από 10.938sec σε 10,734sec.