# Πρόγραμμα Μεταπτυχιακών Σπουδών
# Προχωρημένα Θέματα Βάσεων Δεδομένων
# Πρακτική εργασία

Λορέντζος Δημήτριος – Σωτήριος,421

Εαρινό 2014

# Δομή παρουσίασης

1. Επιλογή θέματος και περιγραφή.
2. Παρουσίαση σχεσιακού σχήματος και επεξήγηση.
3. Συνοπτική παρουσίαση παραγωγής δεδομένων.
4. Προσεγγίσεις δεδομένων που χρησιμοποιήθηκαν.
5. Παρουσίαση πλάνου εκτέλεσης ερωτημάτων χωρίς και με τη χρήση ευρετηρίων καθώς και επεξήγηση του.
6. Συγκριτική μελέτη των επιδόσεων για τις δύο περιπτώσεις.

# Θέμα & περιγραφή

**Εφαρμογή κρατήσεων και πελατολογίου ξενοδοχειακής μονάδας.**

-Στη παρούσα πρακτική εργασία δημιουργήθηκε μία βάση δεδομένων για τη διαχείριση των πελατών, των κρατήσεων , των δωματίων και των πληρωμών ενός ξενοδοχείου. Η βάση δεδομένων έχει γραφεί στη PostgreSQL με τη βοήθεια του pgAdmin III και έχουν εισαχθεί διάφορες ποσότητες δεδομένων που θα περιγραφούν στη συνέχεια.

-Η εφαρμογή χρησιμοποιείται από τους διαχειριστές συστήματος στην υποδοχή όπου και προσέρχονται οι ενδιαφερόμενοι ώστε να δώσουν τα προσωπικά τους στοιχεία.

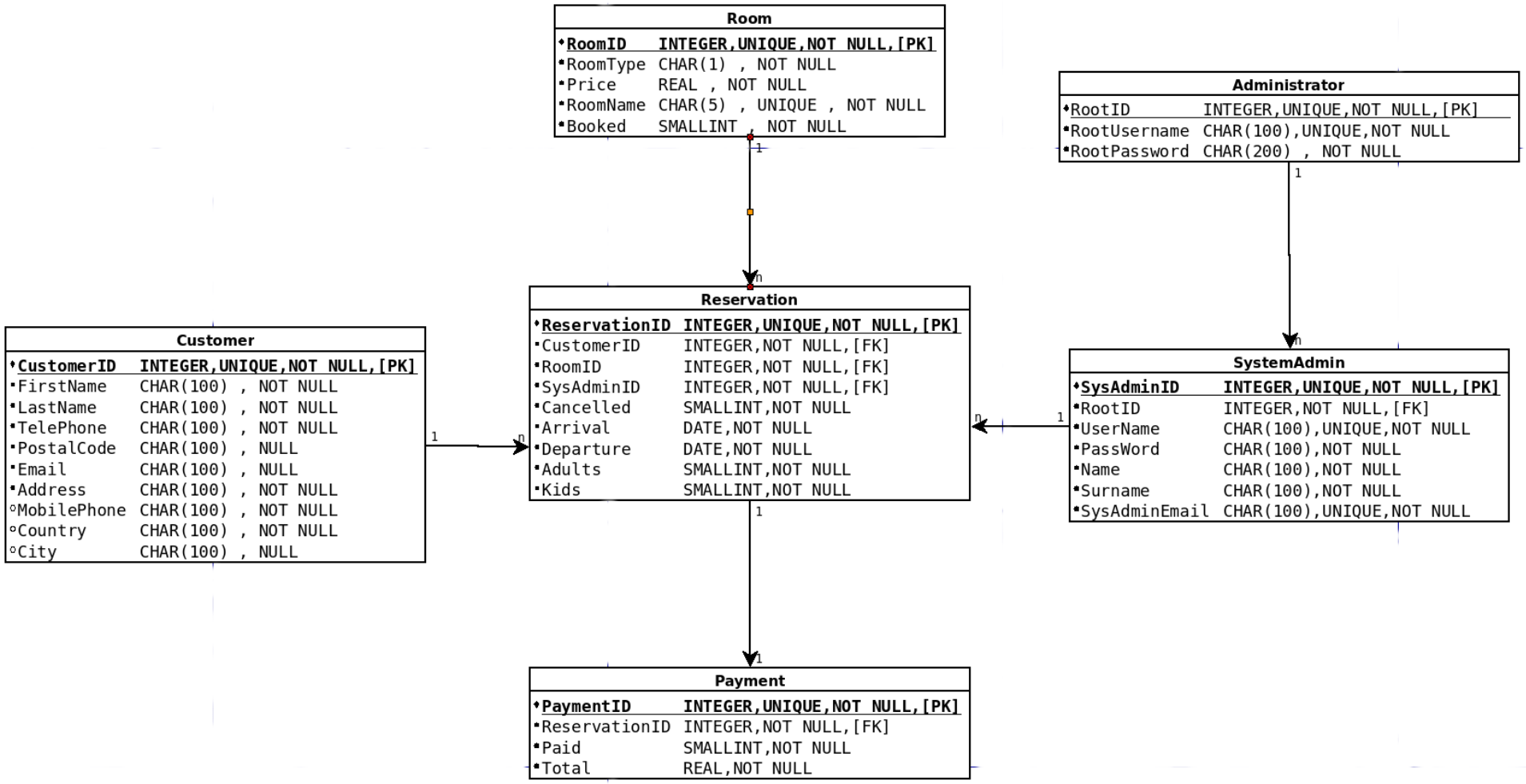
-Αφού καταγραφούν αυτά, κάνουν κράτηση δωματίου για συγκεκριμένο χρονικό διάστημα.

-Πέρα από αυτή την επιλογή η κράτηση μπορεί να καταγραφεί από τους διαχειριστές συστήματος και τηλεφωνικά ,εισάγοντας τα κατάλληλα πεδία στη βάση δεδομένων.

-Οι πελάτες μπορούν επίσης να πληρώσουν για τη διαμονή τους στην αρχή ή στο τέλος αυτής.

-Οι διαχειριστές συστήματος εισάγονται από ένα κεντρικό διαχειριστή.

# Σχεσιακό σχήμα



# Παραγωγή δεδομένων

Έγινε τοπική χρήση του Project Generatedata από το Github: *https://github.com/benkeen/generatedata* για τη παραγωγή των δεδομένων.

DATA SET

| Order | Table Column | Data Type | Examples | Options | Help | Del |
|---|---|---|---|---|---|---|
| 1 | FirstName | Names | Alex (any gender) | Name | ? | |
| 2 | LastName | Names | Smith (surname) | Surname | ? | |
| 3 | PostalCode | Postal / Zip | No examples available. | | ? | |
| 4 | TelePhone | Phone / Fax | Canada (1) | 1-Xxx-Xxx-xxxx | ? | |
| 5 | MobilePhone | Phone / Fax | Canada (2) | (Xxx) Xxx-xxxx | ? | |
| 6 | Email | Email | No examples available. | No options available. | ? | |
| 7 | Country | Country | No examples available. | Limit countries to those selected above | ? | |
| 8 | City | City | No examples available. | No options available. | ? | |
| 9 | Address | Street Address | No examples available. | No options available. | ? | |
| Order | Table Column | Data Type | Examples | Options | Help | Del |

# Προσεγγίσεις δεδομένων

Η εκτέλεση των ερωτημάτων τόσο χωρίς τη χρήση ευρετηρίων όσο και με τη χρήση τους έγινε με βάση τις δύο παρακάτω ποσότητες δεδομένων.

1η προσέγγιση
Πίνακας Administrator:1 εγγραφή
Πίνακας SystemAdmin:10 εγγραφές
Πίνακας Customer:300000 εγγραφές
Πίνακας Payment:300000 εγγραφές
Πίνακας Reservation:300000 εγγραφές
Πίνακας Room:1000 εγγραφές

2η προσέγγιση
Πίνακας Administrator:1 εγγραφή
Πίνακας SystemAdmin:10 εγγραφές
Πίνακας Customer:1000000 εγγραφές
Πίνακας Payment:1000000 εγγραφές
Πίνακας Reservation:1000000 εγγραφές
Πίνακας Room:1000 εγγραφές

# PostgreSQL Query 1 (1/3)

Εμφάνισε τον αριθμό των δίκλινων δωματίων που είναι διαθέσιμα (ομοίως για μονόκλινα,τρίκλινα,τετράκλινα).

```
EXPLAIN ANALYZE SELECT COUNT(Room.RoomID)
FROM Room
WHERE Booked=0 AND RoomType='2';
```

Αρχικά θα εκτελέσουμε το ερώτημα για τα δεδομένα που αναφερθήκαμε στη 1η προσέγγιση χωρίς και με ευρετήρια και στη συνέχεια θα ακολουθήσουμε την ίδια διαδικασία για τη 2η προσέγγιση, για όλα τα ερωτήματα.

# PostgreSQL Query 1 (2/3)

Χωρίς τη χρήση ευρετηρίου:

*Aggregate (cost=**22.33**..**22.34** rows=**1** width=**6**) (actual time=**0.621**..**0.621** rows=**1** loops=**1**)*
*-> Seq Scan on room (cost=0.00..22.00 rows=133 width=6) (actual time=0.032..0.579 rows=132 loops=1)*
*Filter: ((booked = 0) AND (roomtype = '2'::bpchar))*
*Rows Removed by Filter: 868*

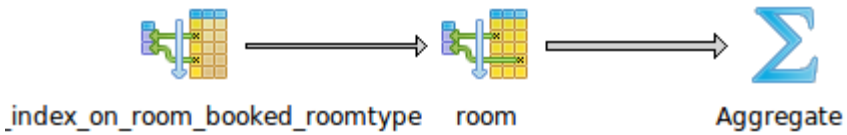
Total runtime: 0.688 ms


# PostgreSQL Query 1 (3/3)

Με τη χρήση ευρετηρίου

```
CREATE INDEX tree_index_on_room_booked_roomtype ON Room
USING btree(RoomType,Booked);
```

*Aggregate (cost=**14.97**..**14.98** rows=**1** width=**6**) (actual time=**0.213**..**0.213** rows=**1** loops=**1**)*
*-> **Bitmap Heap Scan on room** (cost=5.64..14.63 rows=133 width=6) (actual time=0.115..0.187 rows=132 loops=1)*
*Recheck Cond: ((roomtype = '2'::bpchar) AND (booked = 0))*
*-> **Bitmap Index Scan on tree_index_on_room_booked_roomtype** (cost=0.00..5.61 rows=133 width=0) (actual time=0.087..0.087 rows=132 loops=1)*
*Index Cond: ((roomtype = '2'::bpchar) AND (booked = 0))*
Total runtime: 0.285 ms




# PostgreSQL Query 2 (1/5)

Εμφάνισε τον αριθμό των κρατήσεων που έγιναν από την Ελλάδα.

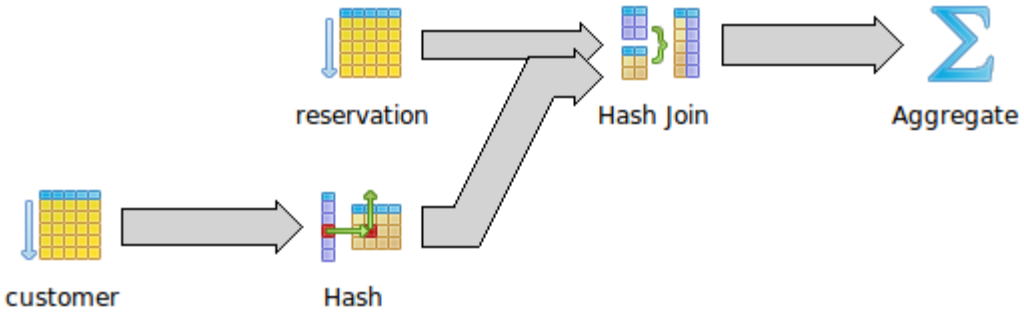
```
EXPLAIN ANALYZE SELECT COUNT(Reservation.ReservationID)
FROM Reservation
INNER JOIN Customer
ON Customer.CustomerID=Reservation.CustomerID
WHERE Country='Greece';
```

# PostgreSQL Query 2 (2/5)

Χωρίς τη χρήση ευρετηρίου:

*Aggregate (cost=**47611.10**..**47611.11** rows=**1** width=**4**) (actual time=**226.435**..**226.435** rows=**1** loops=**1**)*
*-> Hash Join (cost=41265.05..47608.09 rows=1204 width=4) (actual time=168.250..226.326 rows=1159 loops=1)*
*Hash Cond: (reservation.customerid = customer.customerid)*
*-> Seq Scan on reservation (cost=0.00..5206.00 rows=300000 width=8) (actual time=0.005..22.532 rows=300000 loops=1)*
*-> Hash (cost=41250.00..41250.00 rows=1204 width=4) (actual time=168.218..168.218 rows=1159 loops=1)*
*Buckets: 1024 Batches: 1 Memory Usage: 41kB*
*-> Seq Scan on customer (cost=0.00..41250.00 rows=1204 width=4) (actual time=0.072..167.566 rows=1159 loops=1)*
*Filter: (country = 'Greece'::bpchar)*
*Rows Removed by Filter: 298841*

Total runtime: 226.486 ms



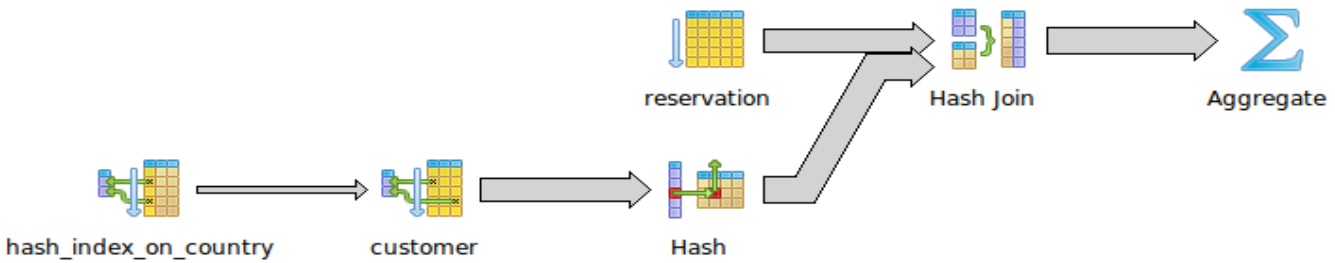

# PostgreSQL Query 2 (3/5)

Με τη χρήση του ευρετηρίου:

```
CREATE INDEX hash_index_on_country ON Customer
USING hash(Country);
```

*Aggregate (cost=**10521.53**..**10521.54** rows=**1** width=**4**) (actual time=**67.543**..**67.543** rows=**1** loops=**1**)*
*-> Hash Join (cost=4175.48..10518.52 rows=1204 width=4) (actual time=2.936..67.414 rows=1159 loops=1)*
*Hash Cond: (reservation.customerid = customer.customerid)*
*-> Seq Scan on reservation (cost=0.00..5206.00 rows=300000 width=8) (actual time=0.004..25.061 rows=300000 loops=1)*
*-> Hash (cost=4160.43..4160.43 rows=1204 width=4) (actual time=2.910..2.910 rows=1159 loops=1)*
*Buckets: 1024 Batches: 1 Memory Usage: 41kB*
*-> Bitmap Heap Scan on customer (cost=37.33..4160.43 rows=1204 width=4) (actual time=0.442..2.593 rows=1159 loops=1)*
*Recheck Cond: (country = 'Greece'::bpchar)*
*-> **Bitmap Index Scan on hash_index_on_country** (cost=0.00..37.03 rows=1204 width=0) (actual time=0.242..0.242 rows=1159 loops=1)*
*Index Cond: (country = 'Greece'::bpchar)*

Total runtime: 67.595 ms

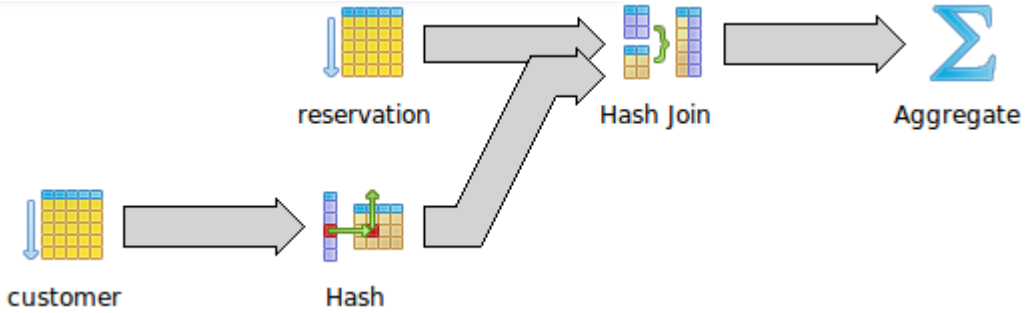


# PostgreSQL Query 2 (4/5)

Χωρίς τη χρήση ευρετηρίου:

*Aggregate (cost=**162453.05**..**162453.06** rows=**1** width=**4**) (actual time=**765.555**..**765.555** rows=**1** loops=**1**)*
*-> Hash Join (cost=137550.02..162443.04 rows=4002 width=4) (actual time=528.639..765.097 rows=3924 loops=1)*
*Hash Cond: (reservation.customerid = customer.customerid)*
*-> Seq Scan on reservation (cost=0.00..17353.00 rows=1000000 width=8) (actual time=0.017..91.777 rows=1000000 loops=1)*
*-> Hash (cost=137500.00..137500.00 rows=4002 width=4) (actual time=528.557..528.557 rows=3924 loops=1)*
*Buckets: 1024 Batches: 1 Memory Usage: 138kB*
*-> Seq Scan on customer (cost=0.00..137500.00 rows=4002 width=4) (actual time=183.898..527.874 rows=3924 loops=1)*
*Filter: (country = 'Greece'::bpchar)*
*Rows Removed by Filter: 996076*

Total runtime: 765.632 ms

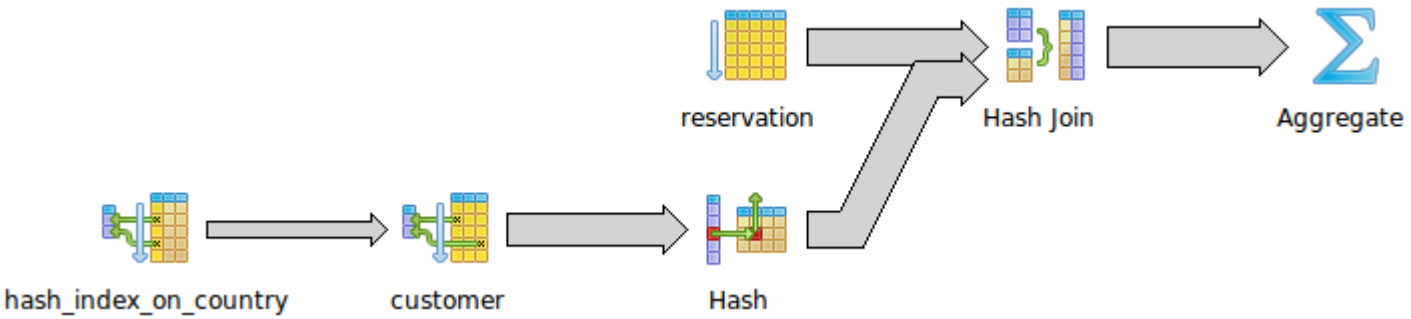



# PostgreSQL Query 2 (5/5)

Με τη χρήση του ευρετηρίου:

*Aggregate (cost=**38841.82**..**38841.83** rows=**1** width=**4**) (actual time=**298.716**..**298.716** rows=**1** loops=**1**)*
*-> Hash Join (cost=13938.62..38831.78 rows=4016 width=4) (actual time=47.515..298.220 rows=3924 loops=1)*
*Hash Cond: (reservation.customerid = customer.customerid)*
*-> Seq Scan on reservation (cost=0.00..17353.00 rows=1000000 width=8) (actual time=0.010..98.011 rows=1000000 loops=1)*
*-> Hash (cost=13888.42..13888.42 rows=4016 width=4) (actual time=47.436..47.436 rows=3924 loops=1)*
*Buckets: 1024 Batches: 1 Memory Usage: 138kB*
*-> Bitmap Heap Scan on customer (cost=135.12..13888.42 rows=4016 width=4) (actual time=8.927..45.438 rows=3924 loops=1)*
*Recheck Cond: (country = 'Greece'::bpchar)*
*-> **Bitmap Index Scan on hash_index_on_country** (cost=0.00..134.12 rows=4016 width=0) (actual time=8.048..8.048 rows=3924 loops=1)*
*Index Cond: (country = 'Greece'::bpchar)*

Total runtime: 298.851 ms




# PostgreSQL Query 3 (1/5)

Εμφάνισε το όνομα, το επώνυμο και το τηλέφωνο των πελατών που ακύρωσαν την κράτησή τους.

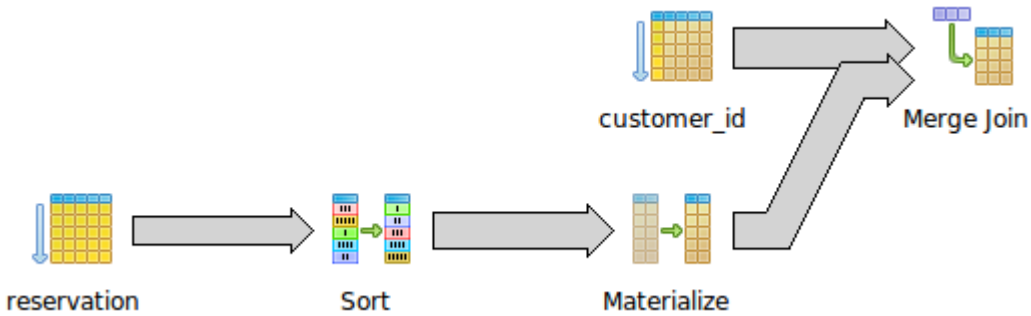
```
EXPLAIN ANALYZE SELECT Customer.FirstName,Customer.LastName,Customer.TelePhone
FROM Customer
INNER JOIN Reservation
ON Customer.CustomerID=Reservation.CustomerID
WHERE Cancelled=1;
```

# PostgreSQL Query 3 (2/5)

Χωρίς τη χρήση ευρετηρίου:

*Merge Join (cost=**20862.68**..**69533.53** rows=**149600** width=**303**) (actual time=**131.947**..**428.967** rows=**149525** loops=**1**)*
*Merge Cond: (customer.customerid = reservation.customerid)*
*-> **Index Scan using customer_id on customer** (cost=0.42..45303.42 rows=300000 width=307) (actual time=0.009..179.370 rows=299996 loops=1)*
*-> Materialize (cost=20862.18..21610.18 rows=149600 width=4) (actual time=131.928..170.868 rows=149525 loops=1)*
*-> Sort (cost=20862.18..21236.18 rows=149600 width=4) (actual time=131.919..152.870 rows=149525 loops=1)*
*Sort Key: reservation.customerid*
*Sort Method: external sort Disk: 2048kB*
*-> Seq Scan on reservation (cost=0.00..5956.00 rows=149600 width=4) (actual time=0.017..55.681 rows=149525 loops=1)*
*Filter: (cancelled = 1)*
*Rows Removed by Filter: 150475*

Total runtime: 436.433 ms



# PostgreSQL Query 3 (3/5)

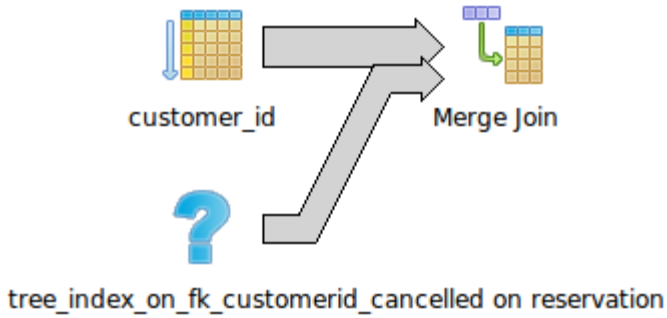
Με χρήση του ευρετηρίου:

```
CREATE INDEX tree_index_on_FK_customer_id_cancelled ON Reservation
USING btree(CustomerID,Cancelled);

CLUSTER Reservation USING tree_index_on_FK_customer_id_cancelled;

CREATE INDEX tree_index_on_fk_customer_id ON reservation
USING btree(CustomerID);
```

*Merge Join (cost=**1.06**..**52210.43** rows=**150380** width=**303**) (actual time=**0.108**..**292.777** rows=**149525** loops=**1**)*
*Merge Cond: (customer.customerid = reservation.customerid)*
*-> **Index Scan using customer_id** on customer (cost=0.42..45303.42 rows=300000 width=307) (actual time=0.013..183.976 rows=299996 loops=1)*
*-> **Index Only Scan using tree_index_on_fk_customerid_cancelled** on reservation (cost=0.42..4277.66 rows=150380 width=4) (actual time=0.086..26.119 rows=149525 loops=1)*
*Index Cond: (cancelled = 1)*
*Heap Fetches: 0*
Total runtime: 299.700 ms

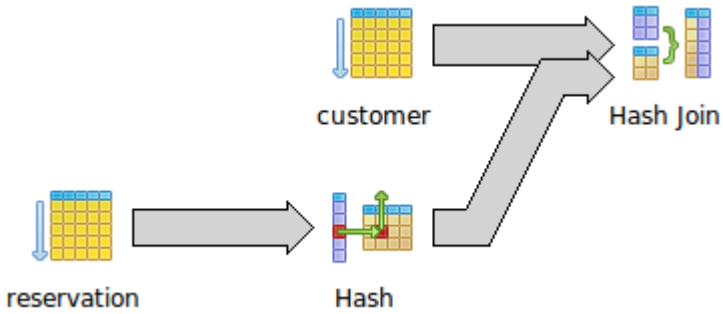



# PostgreSQL Query 3 (4/5)

Χωρίς τη χρήση ευρετηρίου:

*Hash Join (cost=**28063.41**..**259554.74** rows=**500433** width=**303**) (actual time=**266.990**..**1739.909** rows=**499470** loops=**1**)*
*Hash Cond: (customer.customerid = reservation.customerid)*
*-> Seq Scan on customer (cost=0.00..135000.00 rows=1000000 width=307) (actual time=0.013..495.178 rows=1000000 loops=1)*
*-> Hash (cost=19853.00..19853.00 rows=500433 width=4) (actual time=266.415..266.415 rows=499470 loops=1)*
*Buckets: 4096 Batches: 32 Memory Usage: 557kB*
*-> Seq Scan on reservation (cost=0.00..19853.00 rows=500433 width=4) (actual time=0.017..173.487 rows=499470 loops=1)*
*Filter: (cancelled = 1)*
*Rows Removed by Filter: 500530*

Total runtime: 1760.906 ms




# PostgreSQL Query 3 (5/5)

Με τη χρήση του ευρετηρίου:

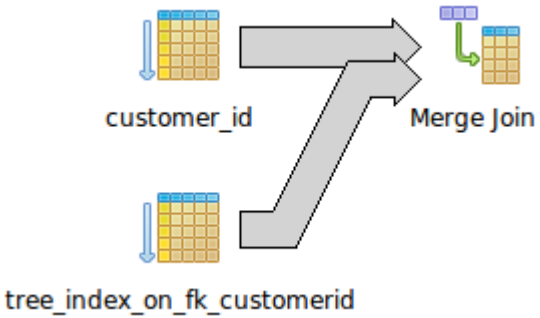
*Merge Join (cost=**1.12..177785.69** rows=**503600** width=**303**) (actual time=**0.047..908.349** rows=**499470** loops=**1**)*

*Merge Cond: (customer.customerid = reservation.customerid)*

*-> **Index Scan using customer_id on customer** (cost=0.42..150983.42 rows=1000000 width=307) (actual time=0.010..538.497 rows=999999 loops=1)*

*-> **Index Scan using tree_index_on_fk_customerid on reservation** (cost=0.42..18007.42 rows=503600 width=4) (actual time=0.032..113.883 rows=499470 loops=1)*

Total runtime: 930.747 ms




# PostgreSQL Query 4 (1/5)

Εμφάνισε το όνομα και το επώνυμο των πελατών που ακύρωσαν την άφιξή τους στο ξενοδοχείο κατά το μήνα Ιανουάριο.

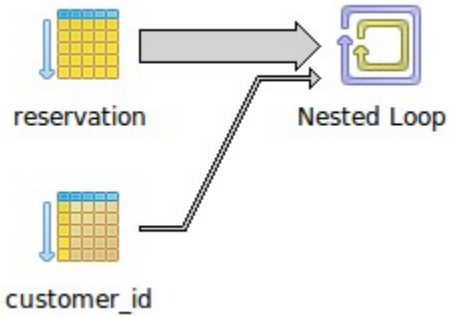
```
EXPLAIN ANALYZE SELECT Customer.FirstName,Customer.LastName
FROM Customer
INNER JOIN Reservation
ON Customer.CustomerID=Reservation.CustomerID
WHERE DATE_PART('month',Arrival)=1 AND Cancelled=1;
```

# PostgreSQL Query 4 (2/5)

Χωρίς τη χρήση ευρετηρίου:

*Nested Loop (cost=**0.42**..**13628.47** rows=**748** width=**202**) (actual time=**0.064**..**114.913** rows=**4739** loops=**1**)*
*-> Seq Scan on reservation (cost=0.00..8206.00 rows=748 width=4) (actual time=0.045..80.395 rows=4739 loops=1)*
*Filter: ((cancelled = 1) AND (date_part('month'::text, (arrival)::timestamp without time zone) = 1::double precision))*
*Rows Removed by Filter: 295261*
*-> **Index Scan using customer_id on customer** (cost=0.42..7.24 rows=1 width=206) (actual time=0.006..0.007 rows=1 loops=4739)*
*Index Cond: (customerid = reservation.customerid)*

Total runtime: 115.314 ms

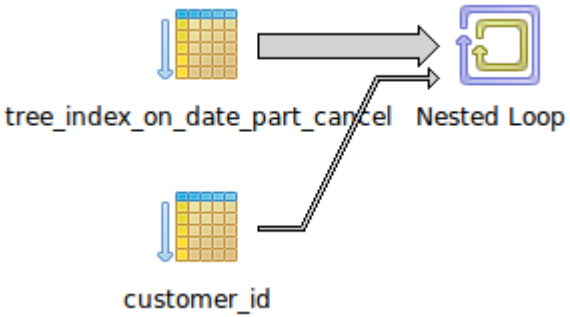



# PostgreSQL Query 4 (3/5)

Με τη χρήση ευρετηρίου:

```
CREATE INDEX tree_index_on_date_part_cancel ON Reservation
USING btree(DATE_PART('month',Arrival),Cancelled) WHERE
DATE_PART('month',Arrival)=1 AND Cancelled=1;
```

*Nested Loop (cost=**0.71**..**6613.92** rows=**752** width=**202**) (actual time=**0.098**..**39.454** rows=**4739** loops=**1**)*
*-> **Index Scan using tree_index_on_date_part_cancel on reservation** (cost=0.29..1161.64 rows=752 width=4) (actual time=0.082..6.159 rows=4739 loops=1)*
*Index Cond: ((date_part('month'::text, (arrival)::timestamp without time zone) = 1::double precision) AND (cancelled = 1))*
*-> **Index Scan using customer_id on customer** (cost=0.42..7.24 rows=1 width=206) (actual time=0.006..0.006 rows=1 loops=4739)*
*Index Cond: (customerid = reservation.customerid)*
Total runtime: 39.886 ms




# PostgreSQL Query 4 (4/5)

Χωρίς τη χρήση ευρετηρίου:

*Nested Loop (cost=**0.42**..**47283.41** rows=**2502** width=**202**) (actual time=**0.084**..**679.508** rows=**80966** loops=**1**)*

*-> Seq Scan on reservation (cost=0.00..27353.00 rows=2502 width=4) (actual time=0.056..272.822 rows=80966 loops=1)*

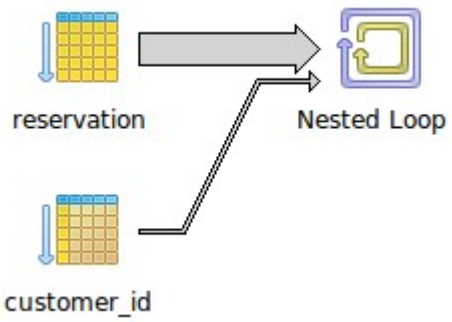
*Filter: ((cancelled = 1) AND (date_part('month'::text, (arrival)::timestamp without time zone) = 1::double precision))*

*Rows Removed by Filter: 919034*

*-> **Index Scan using customer_id on customer** (cost=0.42..7.96 rows=1 width=206) (actual time=0.004..0.005 rows=1 loops=80966)*

*Index Cond: (customerid = reservation.customerid)*

Total runtime: 685.111 ms




# PostgreSQL Query 4 (5/5)

Με τη χρήση ευρετηρίου:

*Nested Loop (cost=**0.85**..**24128.10** rows=**2518** width=**202**) (actual time=**2.793**..**320.837** rows=**80966** loops=**1**)*

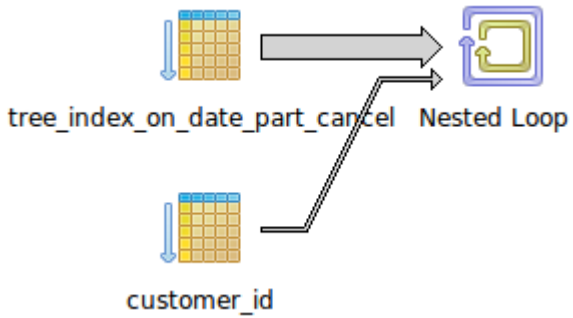
*-> **Index Scan using tree_index_on_date_part_cancel on reservation** (cost=0.42..4070.41 rows=2518 width=4) (actual time=2.776..34.180 rows=80966 loops=1)*

*Index Cond: ((date_part('month'::text, (arrival)::timestamp without time zone) = 1::double precision) AND (cancelled = 1))*

*-> **Index Scan using customer_id on customer** (cost=0.42..7.96 rows=1 width=206) (actual time=0.003..0.003 rows=1 loops=80966)*

*Index Cond: (customerid = reservation.customerid)*

Total runtime: 326.043 ms




# Συμπεράσματα (1/2)

Ολοκληρώνοντας τη παρούσα πρακτική εργασία εξάγουμε ορισμένα συμπεράσματα αλλά και τη σημασία χρήσης ευρετηρίων στην επιλογή του Optimizer για το πιο αποδοτικό πλάνο. Πιο συγκεκριμένα παρατήρησα:

- Την αυτόματη δημιουργία ευρετηρίων για τα πεδία της βάσης που ήταν ορισμένα ως πρωτεύοντα κλειδιά στη περίπτωση μελέτης ερωτημάτων χωρίς τον ορισμό ευρετηρίων. Αυτό γίνεται για να υπάρχει αποδοτική πρόσβαση στα διάφορα ερωτήματα που θα εκτελέσουμε.

- Μπορούμε να χρησιμοποιήσουμε μόνο ένα CLUSTERED Index σε ένα πίνακα λόγω του ότι τα δεδομένα μπορούν να είναι ταξινομημένα μόνο σε μία διάταξη.

- Επίσης παρατήρησα πως η PostgreSQL αξιοποιεί αποδοτικά τα B+ δέντρα μιας και μπορούμε να ορίσουμε ευρετήρια με σύνθετα κλειδιά αναζήτησης σε αντίθεση με τα Hash indexes που μας επιτρέπουν μόνο ένα κλειδί αναζήτησης.

- Τα B+ δέντρα είναι καλά για ερωτήματα διαστήματος αλλά και ελέγχους ισότητας.

- Τα Hash indexes είναι καλύτερα σε ελέγχους ισότητας από τα B+ δέντρα.

# Συμπεράσματα (2/2)

- Ακόμα έγινε ορισμός μερικών ευρετηρίων (Partial Indexes) με τη λογική πως εάν έχουμε 1000000 εγγραφές σε ένα πίνακα και θέλουμε να αποκλείσουμε κάποιες από αυτές είναι καλή επιλογή να δημιουργήσουμε μερικά ευρετήρια λόγω της εξοικονόμησης δίσκου αλλά και για να κάνουμε περισσότερο αποτελεσματική τη σάρωση του ευρετηρίου. Ο κύριος περιοριστικός παράγοντας που έχουν τα μερικά ευρετήρια είναι πως πρέπει να χρησιμοποιήσουμε την ίδια συνθήκη στο WHERE του ερωτήματος μας με αυτή που ορίσαμε στο ευρετήριο ώστε να γίνει αποδοτική η χρήση του.

- Η καλύτερη πρακτική για να βελτιστοποιήσουμε ένα ερώτημα που θέλουμε να εκτελέσουμε είναι να το ορίσουμε με τέτοιο τρόπο ώστε να είναι κατάλληλο με βάση τη συνθήκη που έχουμε ορίσει στο where του ερωτήματος.

- Επιπροσθέτως παρατήρηση τη δυσκολία του να βελτιστοποιήσουμε τη σύζευξη τριών πινάκων μιας και οι διαφορές στα αποτελέσματα με και χωρίς τη χρήση ευρετηρίων ήταν οριακά καλύτερα με τη χρήση ευρετηρίων. Η χρήση ευρετηρίων ήταν πολύ αποδοτική για σύζευξη δύο πινάκων.

- Τέλος το βασικό συμπέρασμα που βγάζουμε από τη παρούσα εργασία είναι η αναγκαιότητα χρήσης ευρετηρίων για βάσεις δεδομένων με πολύ μεγάλο όγκο δεδομένων.

# Αναφορές

1. PostgreSQL: Up and Running - A Practical Guide to the Advanced Open Source Database , By Regina O. Obe, Leo S. Hsu .

2. PostgreSQL 9.3 Using EXPLAIN
http://www.postgresql.org/docs/9.3/static/using-explain.html#USING-EXPLAIN-BASICS

3. PostgreSQL Query Plan Analysis
http://revenant.ca/www/postgis/workshop/analysis.html

4. Quick PostgreSQL Optimization
http://www.mohawksoft.org/?q=node/56

# Ερωτήσεις