Remnin Tuning Short Course, June 2007

Professor: Dennis Shasha   Teaching assistants: Wei Cao, Shaoyi Yin

Each question is worth 10 points. You may work with one partner and sign both of your names to your paper.

1. Day 1: Design a program in your favorite programming language that can generate tuples where field values can be unique, have a fixed set of distinct values that are allocated to tuples according to a uniform probability distribution, and a fixed set allocated to tuples according to a fractal probability distribution (70-30 rule).

   Populate table sales(id, itemid, quantity)

   such that there are 100,000 ids, 20,000 itemids distributed with uniform probability among the sale ids. The quantity field is distributed in a fractal manner.

   Then write one or more queries that find the ten items having the greatest total quantities and the ten items having the least total quantities.

2. Day 2: Choose one rule of thumb (e.g. the value of denormalization or scans vs. non-clustered indexes or the value of chopping) from the Shasha/Bonnet tuning book and find data distributions and loads where it is satisfied and ones where it either fails to be satisfied or does little good. Comment on whether the situations where the rule of thumb is satisfied are more likely than those when the rule of thumb is not satisfied. Also see how you would modify the rule of thumb to make it more precise.

3. Days 3 and 4: ReserveWithUs is an electronic portal for hotel rooms. It buys rooms at discounted rates and sells them for a bit more. Its clientele is international so the website is multi-lingual. From time to time, management needs the answers to certain summarization queries. Here are the most important tables in the original schema:

```
Hotels need to be named and located:

HOTEL ( id , name, street, city, zip_code, state , distance_to_center,
PRIMARY KEY (id) )

Rooms have features:

FEATURES ( id, feature, PRIMARY KEY (id) )

ROOM_FEATURES ( room_id, feature_id, PRIMARY KEY (room_id, feature_id),
FOREIGN KEY (room_id) REFERENCES ROOM (id), INDEX (feature_id), FOREIGN
KEY (feature_id) REFERENCES FEATURES (id) )


Hotel rooms have rates:

RATE_TYPE ( id, name, rate, PRIMARY KEY (id) )

RATE ( room_id, rate_type_id, day, PRIMARY KEY (room_id, day), FOREIGN
KEY (room_id) REFERENCES ROOM (id) , FOREIGN KEY (rate_type_id)
REFERENCES RATE_TYPE (id) )

Each hotel has a certain number of rooms of various types.
Thus, each ROOM.id is associated with a hotel and can fit
```

```
a certain number of guests.

ROOM ( id, number, max_guests, hotel_id, room_type, PRIMARY KEY (id),
INDEX (hotel_id), FOREIGN KEY (hotel_id) REFERENCES HOTEL (id) )

Guests have accounts and make reservations:

ACCOUNT ( id, username, password, first_name, last_name, home_street,
home_city, home_zip_code, home_state, business_street, business_city,
business_zip_code, business_state, home_phone, business_phone, email,
PRIMARY KEY (id) )

RESERVATION ( id, check_in, duration, room_id, account_id, PRIMARY KEY
(id), INDEX (room_id), FOREIGN KEY (room_id)
REFERENCES ROOM (id), INDEX (account_id), FOREIGN KEY (account_id)
REFERENCES ACCOUNT (id) )
```

The data is here. Here is a verbal description of the operations to perform. When I ask you to evaluate alternatives, please do so first without indexes and then after adding indexes.

An alternative representation of the data is to suppose that the rate depends only on the room type (as opposed to the specific room). So double rooms would cost more than single rooms, for example. This change would affect several tables, but might reduce the sizes of the tables and the times of the updates. We explore this possibility.

Here are some new or changed tables. 1. You should figure out the keys, foreign key relations and best indexes. You should populate the tables given the original data:

```
ROOMTYPE(hotel_id room_type_id, feature_id)

RATENEW ( hotel_id,
room_type_id, rate_type_id, day )

 -- The qty field is the number of rooms desired
 -- we treat each day separately
 -- for a reservation to succeed, there must be enough
 -- rooms of that type for every day.
RESERVATIONNEW ( hotel_id, room_type_id, date, qty)

ACCOUNTRES ( account_id, reservation_id)
```

2. You will be given 10,000 room reservations which we will generate, each of which tries to take a cettain number of rooms of some room type for some date range in some hotel for some account. The reservations should behave as follows: if there are not enough rooms of that type available on the given date, then the reservation is rejected. Otherwise the RESERVATIONNEW and ACCOUNTRES tables are appopriately updated. You should try two alternatives: (i) for each reserved date, insert into RESERVATIONNEW if there is no record for that date, hotelid, and room type or otherwise update that record (the "upsert" option); (ii) in a maintenance step, insert records for every possible hotel id, date, and room type with zero taken, so that this reservation operation will always be an update.

Do both (i) and (ii) without indexes and then add indexes. Time the alternatives.

3. Find hotels with above average sales in each city. Again use two techniques: (i) use a temporary table; (ii) use a subquery. State which indexes would speed up the query and use them then repeat (i) and (ii). Do not otherwise change the schema.

Be careful to specify the schemas you use for each experiment, the queries you write, and the measurements. It should be possible to reproduce your findings.