Assignment 1:
PL/SQL Warmup

Wolfgang Richter ⟨wolfgang.richter@gmail.com⟩

January 27, 2015

Problem 1. Securely Storing Passwords

Design a table that can store user ID's, usernames, and passwords. User ID's should be a unique numeric identifier. No user should share the same user ID. Usernames should be unique strings identifying each user, assume that their maximum UTF-8 character length is 256 characters. Assume also that passwords are UTF-8 strings with a maximum length of 256 characters. However, unlike usernames, we should never store passwords in the clear. Passwords are sensitive pieces of information. If a hacker broke into the database, they would be able to steal people's passwords which are probably reused on other websites. Design some way of storing passwords which is secure and does not store plaintext.

Question 1.1. Why is it a good idea to use UTF-8 for string input from users?

Question 1.2. What are the options for securely storing passwords?

Question 1.3. How would you characterize the performance tradeoffs between these different options? For example, could you design an experiment to benchmark different strategies in PL/SQL?

Question 1.4. Using the benchmark you designed, what is the empirical (implement and run your benchmark) cost of adding complexity over simply storing passwords in plaintext?

Problem 2. Many Different Index Flavors

Tables in databases can easily grow to enormous size with hundreds of millions of rows. This causes tension with the database's primary mission: serve user queries as quickly as possible. As a DBA, the key tool generally applied to speed up user queries is an index. Generally, indexes create a summary with "pointers" into a larger set of data. For common user queries—those which are repeated frequently—creating an index can greatly decrease the latency users experience when running over very large tables. But, there are different types of indexes which can be applied.

Question 2.1. What are the types of indexes supported by Oracle DB today (don't read further until you know the answer to this question)?

Question 2.2. When would you use a B-tree style index? When would you use a bitmap index?

Question 2.3. What is the difference between a local index and a global index?

Question 2.4. Describe a scenario where a function-based index would be a good choice.

Question 2.5. Over what kind of data would you use a domain index?

Question 2.6. When would you use a reverse key index?

Question 2.7. What is index block contention? What type of query throughput does it hurt?