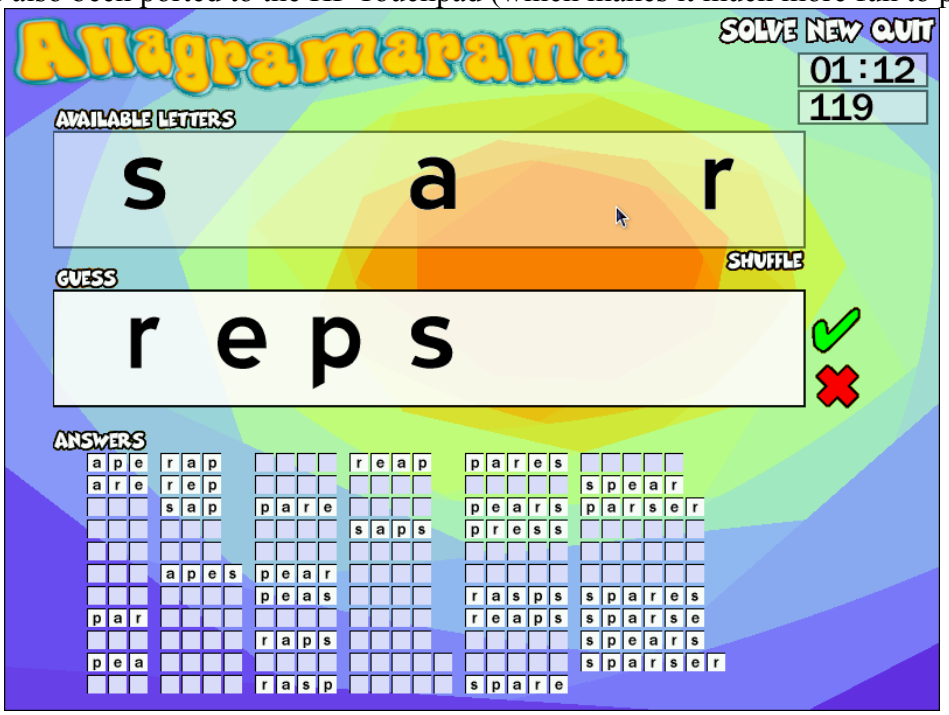
Generating Anagrams with Vertica UDTs

Introduction

My wife is currently addicted to a game called "Anagramarama". Initially developed for Linux desktops, it has also been ported to the HP Touchpad (which makes it much more fun to play).



The basic gameplay is:

1. The computer selects a 7-letter word from its dictionary.
2. All words that can be made from a permutation of the letters (or a subset thereof) are calculated by the computer, which makes empty boxes for them in the "answers" section of the game board.
3. The letters are randomly arranged in the "available letters" area.
4. The player has 5 minutes to arrange letters in the "guess" area. Each guess that is a valid word is recorded in the "answers" area. One point is scored for each letter box filled in in the "answers" area.

For those who want to play, the source code for the game can be found here:
http://code.google.com/p/anagramarama/
Also, we will use the (delightfully British) word list from here in our analysis of the game.

Analysis of the Game

The first idea that came to my mind is, "How do I get the computer to solve the puzzles for me?". (I believe the technical term for this is "cheating".) But I also had deeper questions, such as what puzzle is the easiest (has the least answers) or hardest (most answers).

At some point I realized that Vertica can actually help answer this, and is actually a decent tool for the job. All I needed to implement in a UDT is anagram logic, and then the results can be analyzed in standard SQL.

The UDT is quite simple. For each input string, it produces many output strings, one for each

permutation of every subset of letters. There are likely many duplicates, however we can let SQL filter these out. For 7-letter words, there are 13,699 output rows for each input row.

The example SQL for solving puzzles is straightforward:

```
SELECT anagram
FROM   (SELECT word,
               Gen_anagram(word) OVER (PARTITION BY word)
        FROM   (SELECT 'laabnoe' AS word) puzzle) sq
WHERE  anagram IN (SELECT word
                   FROM   words)
GROUP  BY anagram
ORDER  BY Length(anagram),
          anagram;
```

The output of this example is:

```
 anagram
---------
 alb
 ale
 baa
 ban
 boa
 eon
 lab
 lea
 lob
 nab
 one
 able
 aeon
 aloe
 anal
 bale
 bane
 bean
 bola
 bole
 bone
 lane
 lean
 loan
 lobe
 lone
 noel
 alone
 banal
 noble
 abalone
(31 rows)
```

This SQL is the kernel around which more sophisticated analysis can be performed. The details are in the example SQL file, but:

1. The puzzle with the most answers is “plaster” (also “stapler”, which is the same puzzle). It has 175 words, and requires 755 letters to be filled in in the answer area (this is over 2 and a half per second).
2. The puzzle with the least answers / lowest possible score is “vicious”, followed closely by

"gimmick".

3. There are 1,493 6-letter words in the word list that can't be made by generating anagrams of 7-letter words in the list.
4. The word "set" appears as an answer in the most puzzles, 1,364.
5. There are 2,223 puzzles that don't have any 6-letter solutions.
6. While there are 9,829 7-letter words on the list, there are only 8,896 unique puzzles.

Conclusion

This seems like a rather lame exercise in retrospect, but at least I didn't spend much time on it. There are, including comments and whitespace, less than 300 lines of SQL and C++ code here. Counting time spent playing the game (when I was supposed to be working), it was pretty much an afternoon and an evening.