SAP HANA/HSQLDB

Mauricio Coello, A01328258, Eric Parton A01023503

2019-02-25

1 HSQLDB

1.1 £Por qué HSQLDB y no InfinityDB?

Debido a que había un error con la versión actual de prueba de InfinityDB, usé HSQLDB para las pruebas ya que también está escrito en Java. La diferencia más fuerte entre InfinityDB y HSQLDB es que el primero es un DBMS no-SQL y el segundo, claramente, es un DBMS SQL.

1.2 Introducción a las pruebas

Para probar la eficiencia del DBMS, la idea fue probar diferentes queries con la base de datos cargada con 10,000,000 de entradas. El problema con esto es que HSQLDB es un DBMS en memoria que no tiene la opción de guardar los datos en disco. El resultado es que en mi computadora de 8GB, sólo pude almacenar 1,750,000 datos para las pruebas.

Debido a esta limitación, el método de análisis fue medir los tiempos de los queries con intervalos de 250,000 datos y extrapolar esa información para predecir su comportamiento con 10,000,000.

La tabla usada para las pruebas fue la siguiente:

Nombre	id	email	continent	discovery
Tipo de dato	INT	VARCHAR(20)	VARCHAR(20)	INT
Ejemplo	3	cow57@gmail.com	North America	1473

El ID se generó incrementalmente. El email se contruyó a partir de un animal aleatorio de una selección (rabbit, horse, duck, camel, etc), el ID - 1, y una terminación aleatoria de una selección (gmail.com, hotmail.com, yahoo.com). El continente es un continente aleatorio. El discovery es un número aleatorio entre 1000 y 2000.

1.3 SELECT

El query que se ejecutó fue SELECT * FROM AnimalsOnTheInternet en todos los casos.

Número de datos	Tiempo de ejecución (ms)
250,000	106.4
500,000	67.6
750,000	98.5
1,000,000	129.2
1,250,000	166.8
1,500,000	194.1
1,750,000	238.8

Quitando la medición con 250,000 datos, el mejor polinomio de regresión para estos datos es lineal:

$$y = -2.599 + 0.000135x \tag{1}$$

Esta ecuación se ajusta muy bien a los datos con una r^2 de 0.996.

Dada esta ecuación, la predicción de tiempo de ejecución con 10,000,000 de entradas sería aproximadamente 1346ms.

1.4 AVG

El query que se ejecutó fue SELECT AVG (Discovery) AS DiscoveryDate FROM AnimalsOnTheInternet en todos los casos.

La tendencia de los datos con este query es más difícil ver. Otra vez quité el registro de 250,000 para la regresión ya que parece ser una medición no representativa. El mejor polinomio de regresión para estos datos es parabólico:

$$y = 1.110 - 0.000145x + 0.000000000162x^2 \tag{2}$$

Número de datos	Tiempo de ejecución (ms)
250,000	161.4
500,000	61.6
750,000	89.4
1,000,000	196.6
1,250,000	156.7
1,500,000	191.3
1,750,000	390.7

Esta ecuación se ajusta dudosamente a los datos con una r^2 de 0.829.

Dada esta ecuación, la predicción de tiempo de ejecución con 10,000,000 de entradas sería aproximadamente 17682ms.

Este número se debe tomar con un nivel de escepticismo por la r relativamente baja de la regresión.

1.5 GROUP BY

HSQLDB tiene un implemtación de GROUP BY diferente que requiere de una tabla especial de tipo GROUP BY para ejecutar ese query. Para evadir esa condición y seguir usando las mismas tablas que las otras pruebas, usé DISTINCT. Usé DISTINCT ya que puede representar las mismas queries que se pueden implementar con GROUP BY.

El query que se ejecutó fue SELECT DISTINCT Continent FROM AnimalsOnTheInternet en todos los casos.

Número de datos	Tiempo de ejecución (ms)
250,000	660.8
500,000	1213.2
750,000	1704.5
1,000,000	2277.5
1,250,000	2820.4
1,500,000	3504.2
1,750,000	4132.5

Quitando la medición con 250,000 datos, el mejor polinomio de regresión

para estos datos es lineal:

$$y = 28.586 + 0.00230x \tag{3}$$

Esta ecuación se ajusta muy bien a los datos con una r^2 de 0.997.

Dada esta ecuación, la predicción de tiempo de ejecución con 10,000,000 de entradas sería aproximadamente 23028ms.

1.6 WHERE

El query que se ejecutó fue SELECT Email, Continent FROM AnimalsOnTheInternet WHERE Continent = 'Africa' en todos los casos.

Número de datos	Tiempo de ejecución (ms)
250,000	99.0
500,000	118.5
750,000	174.2
1,000,000	230.5
1,250,000	256.4
1,500,000	304.4
1,750,000	358.9

Quitando la medición con 250,000 datos, el mejor polinomio de regresión para estos datos es lineal:

$$y = 44.029 + 0.000176x \tag{4}$$

Esta ecuación se ajusta bien a los datos con una r^2 de 0.990.

Dada esta ecuación, la predicción de tiempo de ejecución con 10,000,000 de entradas sería aproximadamente 1804ms.

1.7 Análisis de resultados

De los queries de SELECT, DISTINCT, y WHERE, se puede decir con gran certeza que se ejectutan en un tiempo lineal. Los resultados de AVG son más difíciles de interpretar dado el número de pruebas. La función parabólica da una r^2 más cercana a 1 pero no se ajusta muy bien a los datos. Mientras

que polinomios más grandes se ajustan mejor, no tiene sentido lógico que el algoritmo usado sea de tan poca eficiencia.

Los resultados de las extrapolaciónes de 10 millones de entradas quedan así:

Query	Tiempo con 10M (ms)	Mejor representación polinomial
SELECT	1346	Lineal
AVG	17682	Parabólica
DISTINCT	23028	Lineal
WHERE	1804	Lineal

SELECT y WHERE tienen tiempos comparables, tiene sentido considerando que es una base de datos ordenada por filas. DISTINCT es una función más costosa para la base de datos y es relevante notar que con 10 millones de registros, toma alrededor de 23 segundos en ejecutarse, probablemente demasiado tiempo para poder ser usado en muchos tipos de sistemas.

El tiempo de ejecución de AVG es una sorpresa. Es posible y probable que el ajuste para los datos es lineal y que, por hacer un ajuste parabólico se generó un estimado de ejecución tan alto (18 segundos). Por lógica, se pensaría que AVG debería seguir el mismo tiempo lineal que SELECT.

2 HANA

2.1 Introducción a las pruebas

Para el caso de HANA, tratamos de instalarlo en 2 computadoras, una con LINUX y otra con Windows, sin embargo, la instalación siempre fallaba al tratar de inicializar el motor de la base, probablemente debido a que ambas computadoras contaban con 8GB de RAM. Es por ello que optamos por montar nuestra instancia de HANA en Google Cloud.

Al igual que para las pruebas de HSQLDB, las pruebas que se hicieron sobre la base de datos fueron con una tabla con 10 millones de registros. La estructura de la tabla, cuyo nombre es $\text{BIKE}_{STATIONSfuelasiguiente}$: $IDstationnamestatusvalueaddressavailablebikestotaldocksavailabledocks$

2.2 SELECT

El query que se ejecutó fue SELECT * FROM bikestations.

La herramienta de bases de datos que pude utilizar con HANA, dado a que la tuve que ejecutar en Google Cloud fue DBeaver. Esta herramienta permite limitar el número de filas obtenidas del query, lo cual fue muy útil porque inclusive para ese tipo de Queries, mi computadora se quedaba sin memoria, por lo que particioné la prueba en 1,000,000 de filas, lo cual tardó 26 segundos. por lo que para 10,000,000 tardaría aproximadamente 260 segundos aproximadamente.

2.3 SUM

El query que se ejecutó fue SELECT SUM(availablebikes) from SYSTEM.bikestations;

Este query fue más fácil de realizar dado que solo regresa una fila con el resultado de la suma, la cual la hice sobre la columna de bicicletas disponibles, la cual realicé 10 veces para obtener un resultado más exacto, el cual tardó lo siguiente:

Número de ejecución	Tiempo de ejecución (ms)
1	510
2	484
3	480
4	558
5	466
6	463
7	484
8	478
9	488
10	503

Esto, en promedio se tradujo en 491.4 ms en promedio.

2.4 AVG

El query que se ejecutó fue SELECT AVG(availablebikes) from SYSTEM.bikestations;

Este query lo hice igualmente sobre la columna de bicicletas disponibles, la cual realicé 10 veces para obtener un resultado más exacto, el cual tardó lo siguiente:

Número de ejecución	Tiempo de ejecución (ms)
1	670
2	496
3	487
4	480
5	499
6	485
7	493
8	521
9	619
10	488

Esto, en promedio se tradujo en 523.8 ms en promedio.

2.5 GROUP BY

El query que se ejecutó fue SELECT statusvalue from SYSTEM.bikestations GROUP BY(statusvalue);

Este query lo hice igualmente sobre la columna de estatus de la estación, esto para hacerlo sobre una columna un poco más "pesada" de procesar, ya que esta indica "In service" o "Not in service", la cual, al igual que las anteriores, la realicé 10 veces para obtener un resultado más exacto, el cual tardó lo siguiente:

Número de ejecución	Tiempo de ejecución (ms)
1	2333
2	2585
3	2333
4	2329
5	2325
6	2326
7	2601
8	2319
9	2498
10	2796

Esto, en promedio se tradujo en 2444.6 ms en promedio.

2.6 WHERE

El query que se ejecutó fue SELECT * from SYSTEM.bikestations WHERE ID >= 100 AND ID <= 300;

Este query lo hice igualmente sobre la columna de bicicletas disponibles, la cual realicé 10 veces para obtener un resultado más exacto, el cual tardó lo siguiente:

Número de ejecución	Tiempo de ejecución (ms)
1	88
2	54
3	49
4	50
5	71
6	51
7	50
8	48
9	49
10	49

Esto, en promedio se tradujo en 55.9 ms en promedio.

2.7 Análisis de resultados

Como podemos observar de las pruebas realizadas, las pruebas en general tardaron alrededor de 500ms para las pruebas de agregación. En el caso del SELECT, el tiempo fue significativamente más altos por la cantidad de tuplas que el DBMS tuvo que reconstruir, además de que un factor muy alto fue la latencia por el hecho de que el DBMS se ejecutaba en Google Cloud y la falta de memoria de mi máquina.

En general, creo que el perfomance de la base fue muy bueno, dado que en general los tiempos de ejecución fueron muy bajos para la cantidad de datos que se procesaron.

2.8 Conclusión

En conclusión, podemos ver que en general, SAP HANA muestra mejores resultados en general, comparado con HSQLDB, esto inclusive considerando

el tiempo que tomaban los Queries en llegar a la máquina donde se probó, dado a que como mencionamos anteriormente, se desplegó en Google Cloud.

Sin duda fue una práctica interesante y retadora, dado a que la naturaleza de la misma, requería equipos con hardware fuera de la media común de equipos, lo que conllevó a buscar diferentes alternativas para poder desplegar las bases correspondientes y hacer las pruebas requeridas.