Search ALL Genome Annotations

Ryan M. Layer
ryan.layer@gmail.com
@ryanlayer
University of Utah
quinlanlab.org
Hierarchical Binning [UCSC Kent 2002, TABIX Li 2011, BEDTOOLS Quinlan 2010]

Database

Sweep [BEDTOOLS]
chr1:52000-53000

chr1 52057 52058 rs62637813
chr1 52237 52238 rs2691277
chr1 52726 52727 rs2691278

UCSC

query size empty
100bp  0.73
1000bp  0.67
10000bp  0.58
B+ Tree index of positions and offsets
**Query**
- **# of intersections**
  - chr1:52000-53000
  - 12039

**Intersections per file**
- chr1:52000-53000
- snp144
- neandertalMethylation
- genomicSuperDups
  - 13
  - 5
  - 2

**Intersecting intervals**
- chr1:52000-53000
  - #snp144
  - chr1 52057 52058 rs62637
  - chr1 52095 52096 rs36775
  - #neandertalMethylation
  - chr1 52015 52016 100
  - chr1 52028 52029 100
THE TIME INDEX:
AN ACCESS STRUCTURE FOR TEMPORAL DATA

Ramez Elmasri1," Gene T. J. Wu2, and Yeong-Joon Kim3
1 Department of Computer Science, University of Houston, Houston, TX 77204
2Bell Communications Research, 444 Hoos Lane, Piscataway, NJ 08854

ABSTRACT

In this paper, we describe a new indexing technique, the time index, for improving the performance of certain classes of temporal queries. The time index can be used to retrieve versions of objects that are valid during a specific time period. It supports the processing of the temporal WHEN operator and temporal aggregate functions efficiently. The time indexing scheme is also extended to improve the performance of the temporal SELECT operator, which retrieves objects that satisfy a certain condition during a specific time period. We will describe the indexing technique, and its search and insertion algorithms. We also describe an algorithm for processing a commonly used temporal JOIN operation. Some results of a simulation for comparing the performance of the time index with other proposed temporal access structures are presented.

languages [SS87, EW90]. These temporal data models define powerful operations for specifying complex temporal queries. There has been relatively less research in the area of defining efficient storage structures and access paths for temporal data [Lum84, Ahn86, AS88, SG89, GSu, RS87, KS89, LS89]. These proposals do not discuss indexing schemes for supporting the high-level temporal operators defined in [GY88, EW90]. This paper describes indexing techniques for improving the efficiency of temporal operations, such as when, select, and join [GY88], temporal selection and temporal projection [EW90], and aggregation functions.

The storage techniques for temporal data proposed in [AS88, Lum84] index or link the versions of each individual object separately. In order to retrieve object versions that are valid during a certain time period, it is necessary to first locate the first (current) version of each object, and then search through the version in-
Binary Tree

```
    4
   / \
  2   11
 / \ /  \  \
1  3 7   14
   / \  /  /  \
  5 10 15
```
Binary Tree

```
4
 /   \
/     \
2      11
|      /\n1  3   7
 |    |  |
5    10  14
      |    |
      15
```
Binary Tree

B+ Tree

at most $N$ (4) keys per node
linked leaves
Binary Tree

B+ Tree

1 2 3 4 5 7 10 11 14 15
Binary Tree

B+ Tree
Binary Tree

B+ Tree
Binary Tree

B+ Tree

Disk layout:
Binary Tree

Disk layout:

```
| 4 | 2 | 11 | 1 | 3 | 7 | 14 | 5 | 10 | 15 |
```

B+ Tree

Disk layout:

```
| 5 | 11 |
```

4 disk reads:

```
| 4 | 2 | 11 | 1 | 3 | 7 | 14 | 5 | 10 | 15 |
```

SSD disk speed

100,000 IOPS

~2,000,000X slower than CPU

2 disk reads:

```
| 5 | 11 | 1 | 2 | 3 | 4 | 5 | 7 | 10 | 11 | 14 | 15 |
```
index(X, start, end)
insert +X at start
insert -X at end + 1
append X at spanned L's
index(A1, 1, 9)

MAX_KEYS = 4
index(A1, 1, 9)
index(A1, 1, 9)
index(A2, 2, 4)
index(A2, 2, 4)

L 1 2 10

+A1 +A2 -A1
index(A2, 2, 4)
\text{index}(A_3, 5, 10)
index(A3, 5, 10)

\[
\begin{array}{cccccc}
\text{L} & 1 & 2 & 5 & 10 \\
\end{array}
\]

\[
\begin{align*}
-A1 & & -A2 & & & \\
\end{align*}
\]
index(A3, 5, 10)
index(A3, 5, 10)

Too Many Keys

L 1 2 5 10 11

+A1 +A2 +A3 -A1 -A3

-A2
Split Leaf

index(A3, 5, 10)
Promote a New Root

index(A3, 5, 10)
index(A3, 3, 6)

Promote a New Root
Promote a New Root

index(A3, 3, 6)
search(start, end)

I = []
from = find(start)
to = find(end);
I += from.leaf.L
for node upto from:
    I += node.starts
    I -= node.ends
for node in [from.next, to]:
    I += node.starts
search(7, 11)
search(7, 11)
search(7, 11)

I = [ ]
search(7, 11)

I = [ ]
search(7, 11)

I = [ ]
$I = [A1 \ A2 \ B1 \ C1]$

```
+ A1  + A2  + B1  + C1
\_\_\_\_\_\_
A1  A2  B1  C1
  |  |  |  |
  +A3 +B2 +C2 -B1
```

```
1  2  3  4  5  6  7  8  9  10  11  12  13  14
```

search(7, 11)
\[ I = [A_1 \ A_2 \ B_1 \ C_1] + [A_3] - [A_2] = [A_1 \ A_3 \ B_1 \ C_1] \]
\[ I = [A_1 A_3 B_1 C_1] + [B_2 C_2] - [B_1] = [A_1 A_3 B_2 C_1 C_3] \]
\[ I = [A1 \ A3 \ B2 \ C1 \ C3] \]

search(7, 11)
$I = [A_1 \ A_3 \ B_2 \ C_1 \ C_3]$

search(7, 11)
- B+ tree
  - Efficient on-disk database
  - Traversal between leaves

- Many files in one index

- Count intersections within the index
Count intersections for 10 - 1M 100bp intervals

127 reference epigenomes
15 genomic states [CHROHMM Ernst 2012]
1905 tracks

2.6GHz Intel Xeon CPU, 20MB cache

SQLITE3 w/ UCSC binning
TABIX
BEDTOOLS sorted
GIGGLE

Count intersections for 10 - 1M 100bp intervals

28M intervals, 939 tracks

55M intervals, 1905 tracks

Speed up: 60.8X 5.4X 7.7X

67X 5.9X 8.2X
DNase-seq of K562
Homo sapiens, adult 53 year
Lab: John Stamatoyannopoulos, UW
Project: ENCODE

ChIP-seq of forebrain
Mus musculus, embryonic 10.5 day
Target: H3K9me3
Lab: Bing Ren, UCSD
Project: ENCODE

ChIP-seq of heart
Mus musculus, embryonic 10.5 day
Target: H3K27me3
Lab: Bing Ren, UCSD
Project: ENCODE

ChIP-seq of embryonic facial prominence
Mus musculus, embryonic 10.5 day
Target: H3K27me3
Lab: Bing Ren, UCSD
Project: ENCODE
Monte Carlo simulations

Query

Database

Observe 3 intersections
Monte Carlo simulations

Query

Database

Observe 3 intersections

Expect ~4
0.75 "enrichment"
Monte Carlo simulations

Query

Database

Observe 3 intersections

Contingency table

<table>
<thead>
<tr>
<th>In query</th>
<th>In target</th>
<th>Not in target</th>
</tr>
</thead>
<tbody>
<tr>
<td>In query</td>
<td></td>
<td>Q T</td>
</tr>
</tbody>
</table>
| Not In query | | T | - | Q T | Genome size \[
\frac{\mu(Q) + \mu(T)}{\mu(Q) + \mu(T)} - (|Q| + |T| - |Q T|) \]

Expect ~4
0.75 "enrichment"

Brent Pedersen
Odds ratio vs. Monte Carlo (N=10000)

Difference in ranking by odds ration and Monte Carlo

[Graph showing the comparison between Odds ratio and Monte Carlo methods, with axes labeled for Odds ratio and Monte Carlo observations/expectations, and a secondary axis for the difference in ranking between the two methods.]
Genotype Query Tools

github.com/ryanlayer/gqt

40 families
20M variants

Tissues and Cell Types

De Novos in:

Affected child

- `gqt query -i all.vcf.gz`
- `-p "phenotype == 2"
- `-g "count(HET) == 1"
- `-p "phenotype == 1"
- `-g "HOM_REF"

Unaffected child

- `-p "phenotype == 1 & parental_id != -9"
- `-g "count(HET) == 1"
- `-p "phenotype != 1 || parental_id == -9"
- `-g "HOM_REF"

Affected female

- `-p "phenotype == 2 & sex == 2"
- `-g "count(HET) == 1"
- `-p "phenotype == 1"
- `-g "HOM_REF"

Affected European

- `-p "phenotype == 2 & population == 'EUR'
- `-g "count(HET) == 1"
- `-p "phenotype == 1"
- `-g "HOM_REF"

< 6s

127 tissues
15 genomic states

14740 families
20M variants
**Command line**

- `giggle index -i "tracks/*gz" -o tracks_i`
- `giggle search -i tracks_i -r chr1:1-1000`
- `giggle search -9 tracks_i -q query.bed.gz`

**C API**

```c
struct giggle_index *gi =
    giggle_load(index_dir_name, uint32_t_ll_giggle_set_data_handler);

struct giggle_query_result *gqr =
    giggle_query(gi, chr, beg, end, gqr);

for(i = 0; i < gqr->num_files; i++) {
    struct giggle_query_iter *gqi =
        giggle_get_query_itr(gqr, i);

    while (giggle_query_next(gqi, &result) == 0)
        printf("%s\n", result);

    giggle_iter_destroy(&gqi);
}
```

**Python API COMING SOON**

Brent Pedersen
TODO:

Index remote files

Faster indexing

Index updating (append)

Integration with Genotype Query Tools (GQT)
Ryan M. Layer
ryan.layer@gmail.com
@ryanlayer

Aaron Quinlan, Brent Pedersen, Tonya Di Sera