BENCHBASE FOR POSTGRES TEMPORAL QUERIES

Paul A. Jungwirth

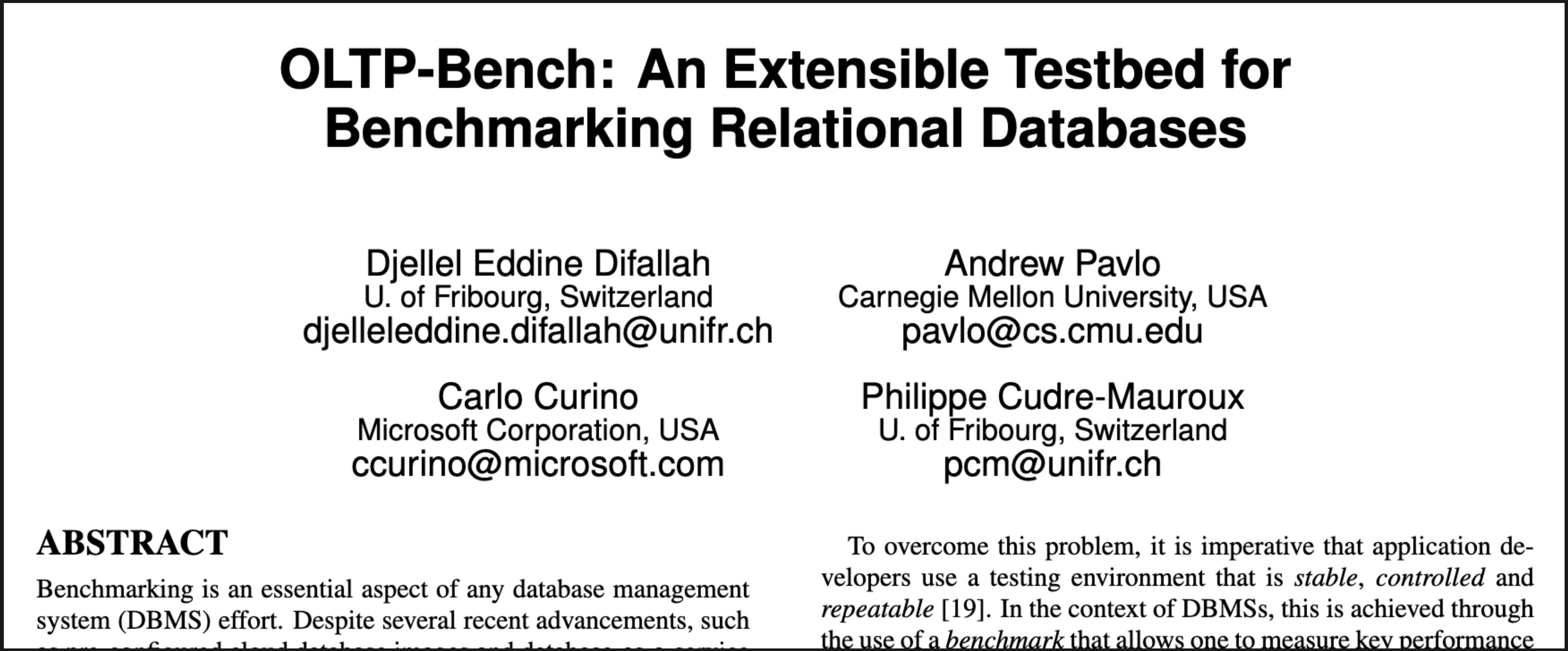
22 August 2024

pdxpug

OUTLINE

- How to use Benchbase
- Comparing temporal foreign key implementations
- ~~More temporal procedures~~
- Benchmarking mistakes and lessons

HISTORY



OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases

Djellel Eddine Difallah
U. of Fribourg, Switzerland
djelleleddine.difallah@unifr.ch

Andrew Pavlo
Carnegie Mellon University, USA
pavlo@cs.cmu.edu

Carlo Curino
Microsoft Corporation, USA
ccurino@microsoft.com

Philippe Cudre-Mauroux
U. of Fribourg, Switzerland
pcm@unifr.ch

ABSTRACT

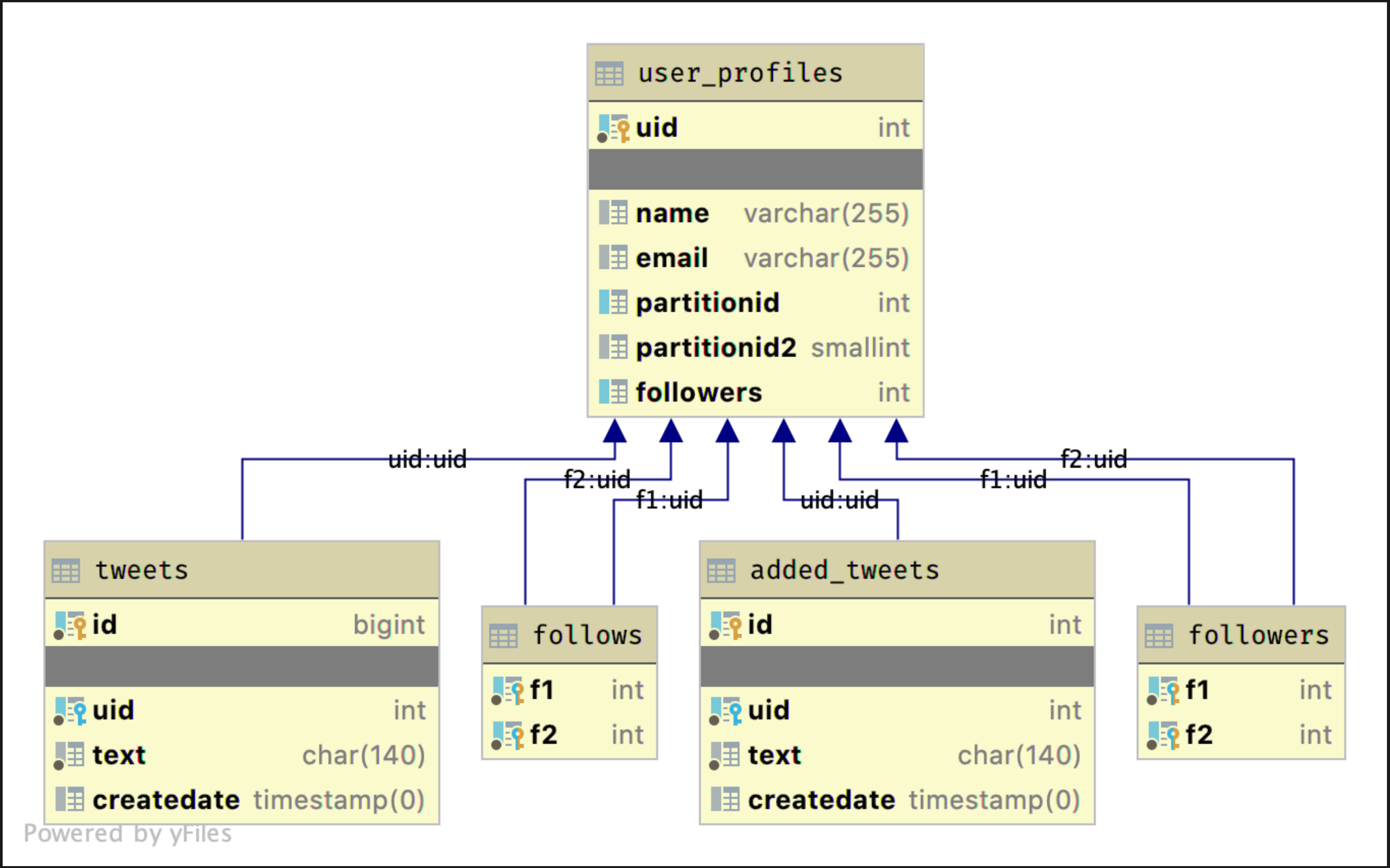
Benchmarking is an essential aspect of any database management system (DBMS) effort. Despite several recent advancements, such

To overcome this problem, it is imperative that application developers use a testing environment that is *stable*, *controlled* and *repeatable* [19]. In the context of DBMSs, this is achieved through the use of a *benchmark* that allows one to measure key performance

BENCHMARKS

```
paul@tal:~/src/benchbase$ ls \
> src/main/java/com/oltpbenchmark/benchmarks/
auctionmark  README.md         templated  voter
chbenchmark  resourcestresser  temporal   wikipedia
epinions     seats             tpcc       ycsb
hyadapt      sibench           tpcds
noop         smallbank         tpch
otmetrics    tatp              twitter
```

BENCHMARKS


user_profiles
uid int
name varchar(255)
email varchar(255)
partitionid int
partitionid2 smallint
followers int
uid:uid
f2:uid
f1:uid
uid:uid
f1:uid
f2:uid
tweets
id bigint
uid int
text char(140)
createdate timestamp(0)
follows
f1 int
f2 int
added_tweets
id int
uid int
text char(140)
createdate timestamp(0)
followers
f1 int
f2 int
Powered by yFiles

DBMSES

```
paul@tal:~/src/benchbase$ find \
> src/main/resources/benchmarks/ \
> -name 'ddl-*' |
> xargs -L1 basename |
> sort -u |
> pr -t -2
ddl-cassandra.sql           ddl-nuodb.sql
ddl-cockroachdb.sql         ddl-oracle.sql
ddl-db2.sql                 ddl-phoenix.sql
ddl-generic.sql             ddl-postgres.sql
ddl-hsqldb.sql              ddl-singlestore.sql
ddl-monetdb.sql             ddl-spanner.sql
ddl-myrocks.sql             ddl-sqlite.sql
ddl-mysql.sql               ddl-sqlserver.sql
ddl-noisepage.sql           ddl-timesten.sql
```

QUICKSTART

```
git clone https://github.com/cmu-db/benchbase.git
cd benchbase
./mvnw clean package -P postgres

cd target
tar xvzf benchbase-postgres.tgz
cd benchbase-postgres

java -jar benchbase.jar -b tpcc \
  -c config/postgres/sample_tpcc_config.xml \
  --create=true --load=true --execute=true
```

ONELINER

```
mvn clean compile exec:java -P postgres \
  -Dexec.args="-b tpcc \
  -c config/postgres/sample_tpcc_config.xml \
  --create=true --load=true --execute=true"
```

https://github.com/cmu-db/benchbase/pull/548

ONELINER

```
mvn clean compile exec:java -P postgres \
  -Dexec.args="-b tpcc \
  -c config/postgres/sample_tpcc_config.xml \
  --create=true --load=true --execute=true"
```

https://github.com/cmu-db/benchbase/pull/548

ONELINER

```
mvn clean compile exec:java -P postgres \
  -Dexec.args="-b tpcc \
  -c config/postgres/sample_tpcc_config.xml \
  --create=true --load=true --execute=true"
```

https://github.com/cmu-db/benchbase/pull/548

ONELINER

```
mvn clean compile exec:java -P postgres \
  -Dexec.args="-b tpcc \
  -c config/postgres/sample_tpcc_config.xml \
  --create=true --load=true --execute=true"
```

https://github.com/cmu-db/benchbase/pull/548

CONFIG FILE

```xml
<type>POSTGRES</type>
<driver>org.postgresql.Driver</driver>
<url>jdbc:postgresql://localhost:5460/benchbase?reWriteBatched
<username>paul</username>
<password></password>
<reconnectOnConnectionFailure>true</reconnectOnConnectionFailu
<isolation>TRANSACTION_READ_COMMITTED</isolation>
<newConnectionPerTxn>false</newConnectionPerTxn>
```

CONFIG FILE

```
<scalefactor>10000</scalefactor>
<batchsize>128</batchsize>
```

CONFIG FILE

```xml
<transactiontypes>
    <transactiontype>
        <name>InsertPosition</name>
    </transactiontype>
    <transactiontype>
        <name>UpdatePosition</name>
    </transactiontype>
    <transactiontype>
        <name>UpdateEmployee</name>
    </transactiontype>
    <transactiontype>
        <name>DeleteEmployee</name>
    </transactiontype>
</transactiontypes>
```

CONFIG FILE

```
<terminals>10</terminals>
<works>
    <work>
        <time>600</time>
        <rate>10000</rate>
        <weights>28,28,28,16</weights>
    </work>
</works>
```

CONFIG FILE

```
<terminals>10</terminals>
<works>
    <work arrival="poisson">
        <time>600</time>
        <rate>10000</rate>
        <weights>28,28,28,16</weights>
        <active_terinals>5</active_terminals>
    </work>
</works>
```

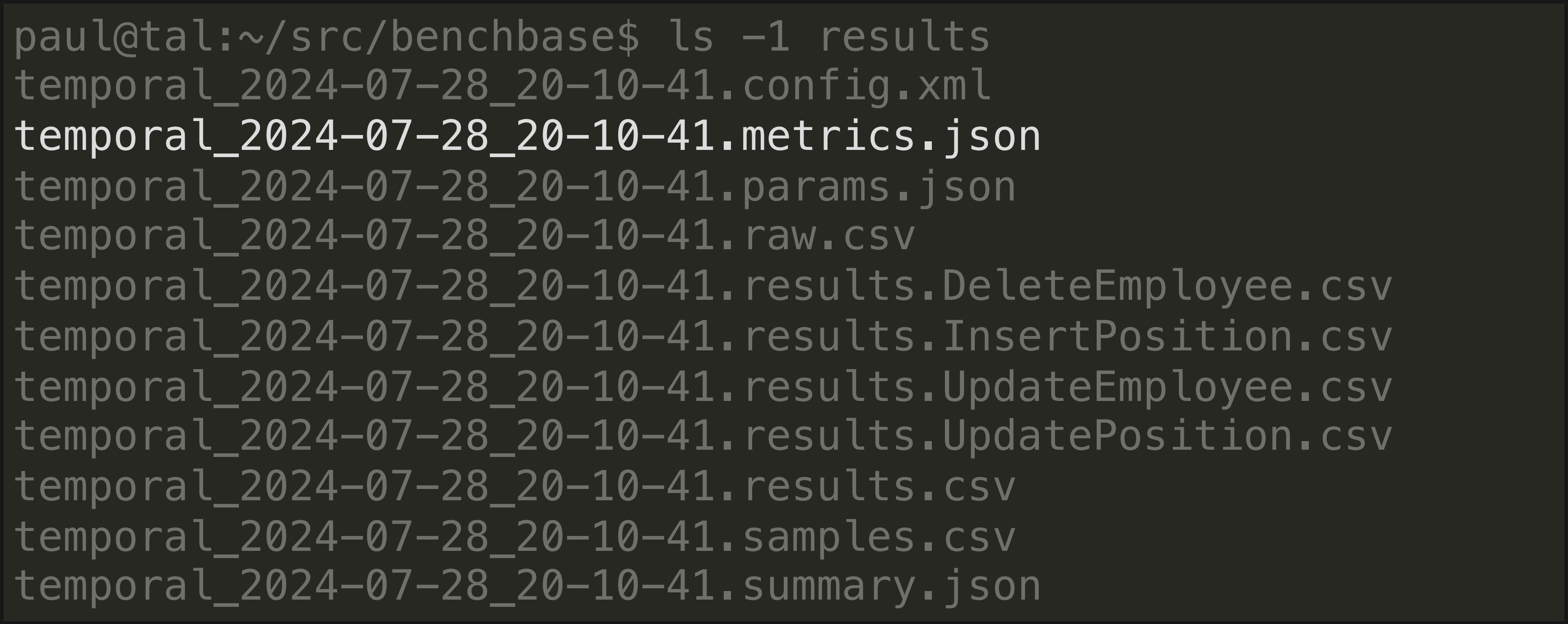
RESULTS

```
paul@tal:~/src/benchbase$ ls -1 results
temporal_2024-07-28_20-10-41.config.xml
temporal_2024-07-28_20-10-41.metrics.json
temporal_2024-07-28_20-10-41.params.json
temporal_2024-07-28_20-10-41.raw.csv
temporal_2024-07-28_20-10-41.results.DeleteEmployee.csv
temporal_2024-07-28_20-10-41.results.InsertPosition.csv
temporal_2024-07-28_20-10-41.results.UpdateEmployee.csv
temporal_2024-07-28_20-10-41.results.UpdatePosition.csv
temporal_2024-07-28_20-10-41.results.csv
temporal_2024-07-28_20-10-41.samples.csv
temporal_2024-07-28_20-10-41.summary.json
```

RESULTS

```
paul@tal:~/src/benchbase$ ls -1 results
temporal_2024-07-28_20-10-41.config.xml
temporal_2024-07-28_20-10-41.metrics.json
temporal_2024-07-28_20-10-41.params.json
temporal_2024-07-28_20-10-41.raw.csv
temporal_2024-07-28_20-10-41.results.DeleteEmployee.csv
temporal_2024-07-28_20-10-41.results.InsertPosition.csv
temporal_2024-07-28_20-10-41.results.UpdateEmployee.csv
temporal_2024-07-28_20-10-41.results.UpdatePosition.csv
temporal_2024-07-28_20-10-41.results.csv
temporal_2024-07-28_20-10-41.samples.csv
temporal_2024-07-28_20-10-41.summary.json
```

RESULTS



```
paul@tal:~/src/benchbase$ ls -1 results
temporal_2024-07-28_20-10-41.config.xml
temporal_2024-07-28_20-10-41.metrics.json
temporal_2024-07-28_20-10-41.params.json
temporal_2024-07-28_20-10-41.raw.csv
temporal_2024-07-28_20-10-41.results.DeleteEmployee.csv
temporal_2024-07-28_20-10-41.results.InsertPosition.csv
temporal_2024-07-28_20-10-41.results.UpdateEmployee.csv
temporal_2024-07-28_20-10-41.results.UpdatePosition.csv
temporal_2024-07-28_20-10-41.results.csv
temporal_2024-07-28_20-10-41.samples.csv
temporal_2024-07-28_20-10-41.summary.json
```

RESULTS

```
paul@tal:~/src/benchbase$ ls -1 results
temporal_2024-07-28_20-10-41.config.xml
temporal_2024-07-28_20-10-41.metrics.json
temporal_2024-07-28_20-10-41.params.json
temporal_2024-07-28_20-10-41.raw.csv
temporal_2024-07-28_20-10-41.results.DeleteEmployee.csv
temporal_2024-07-28_20-10-41.results.InsertPosition.csv
temporal_2024-07-28_20-10-41.results.UpdateEmployee.csv
temporal_2024-07-28_20-10-41.results.UpdatePosition.csv
temporal_2024-07-28_20-10-41.results.csv
temporal_2024-07-28_20-10-41.samples.csv
temporal_2024-07-28_20-10-41.summary.json
```

*.raw.csv

```
Transaction Type Index,Transaction Name,Start Time (microsecon
4,DeleteEmployee,1722222636.414196,6786,0,1
3,UpdateEmployee,1722222636.421013,1045,0,1
1,InsertPosition,1722222636.422064,1402,0,1
4,DeleteEmployee,1722222636.423471,1555,0,1
1,InsertPosition,1722222636.425031,722,0,1
1,InsertPosition,1722222636.425761,682,0,1
```

*.results.csv

```
Time (seconds),Throughput (requests/second),Average Latency (m
0,509.600,6.567,4.936,6.317,6.503,6.645,6.822,6.964,9.793,12.6
5,540.200,6.213,5.167,6.034,6.120,6.245,6.358,6.629,9.088,13.3
10,533.200,6.241,5.294,6.032,6.164,6.277,6.459,6.578,9.078,13.
15,539.800,6.214,2.761,5.934,6.178,6.529,6.755,7.813,12.855,30
20,739.600,4.531,2.659,3.466,3.546,5.144,6.877,7.059,11.450,58
```

*.samples.csv

```
Time (seconds),Requests,Throughput (requests/second),Minimum L
0,1780,1780.000,153,6225,6402,5613,6576,6756,7137,9928,12692
1,1794,1794.000,132,6360,6592,5582,6690,6835,6870,8501,9909
2,1795,1795.000,136,6351,6570,5568,6693,6833,6951,9773,10577
3,1803,1803.000,122,6241,6468,5531,6591,6750,6893,9495,9803
4,1901,1901.000,95,6035,6216,5249,6318,6438,6497,7804,10805
```

*.summary.json

```
{
 "Start timestamp (milliseconds)": 1723441031104,
 "Current Timestamp (milliseconds)": 1723441634718,
 "Elapsed Time (nanoseconds)": 600000068387,
 ...
 "Measured Requests": 1147862,
 "Latency Distribution": {
  "95th Percentile Latency (microseconds)": 7405,
  "Maximum Latency (microseconds)": 58595,
  "Median Latency (microseconds)": 6198,
  "Minimum Latency (microseconds)": 80,
  "25th Percentile Latency (microseconds)": 5135,
  "90th Percentile Latency (microseconds)": 6636,
  "99th Percentile Latency (microseconds)": 11254,
  "75th Percentile Latency (microseconds)": 6411
```

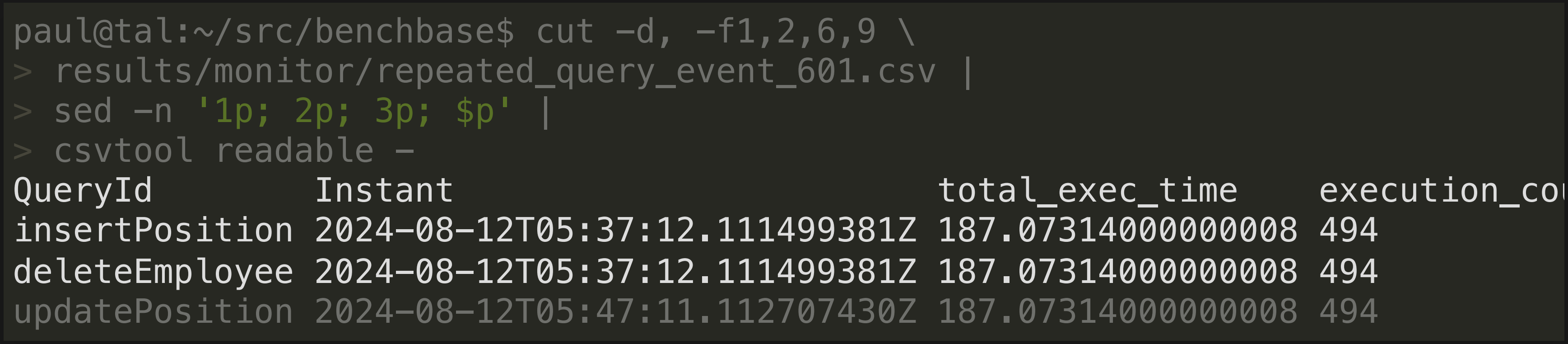
ADVANCED MONITORING

```
--monitor-type=advanced
--interval-monitor=1000
```

ADVANCED MONITORING

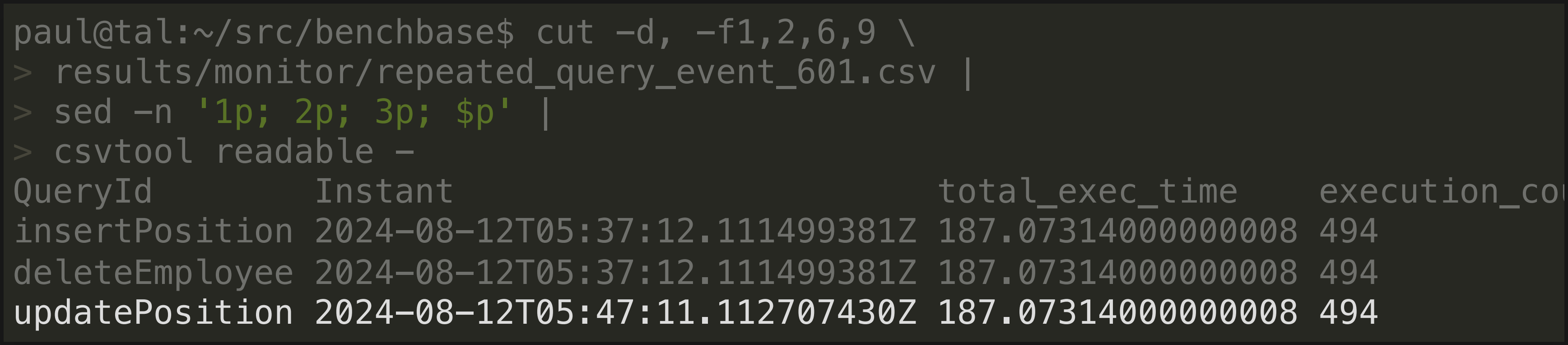
```
paul@tal:~/src/benchbase$ cut -d, -f1,2,6,9 \
> results/monitor/repeated_query_event_601.csv |
> sed -n '1p; 2p; 3p; $p' |
> csvtool readable -
QueryId        Instant                        total_exec_time    execution_co
insertPosition 2024-08-12T05:37:12.111499381Z 187.07314000000008 494
deleteEmployee 2024-08-12T05:37:12.111499381Z 187.07314000000008 494
updatePosition 2024-08-12T05:47:11.112707430Z 187.07314000000008 494
```

ADVANCED MONITORING



```
paul@tal:~/src/benchbase$ cut -d, -f1,2,6,9 \
> results/monitor/repeated_query_event_601.csv |
> sed -n '1p; 2p; 3p; $p' |
> csvtool readable -
QueryId        Instant                        total_exec_time    execution_co
insertPosition 2024-08-12T05:37:12.111499381Z 187.07314000000008 494
deleteEmployee 2024-08-12T05:37:12.111499381Z 187.07314000000008 494
updatePosition 2024-08-12T05:47:11.112707430Z 187.07314000000008 494
```

ADVANCED MONITORING



```
paul@tal:~/src/benchbase$ cut -d, -f1,2,6,9 \
> results/monitor/repeated_query_event_601.csv |
> sed -n '1p; 2p; 3p; $p' |
> csvtool readable -
QueryId        Instant                        total_exec_time    execution_co
insertPosition 2024-08-12T05:37:12.111499381Z 187.07314000000008 494
deleteEmployee 2024-08-12T05:37:12.111499381Z 187.07314000000008 494
updatePosition 2024-08-12T05:47:11.112707430Z 187.07314000000008 494
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

BENCHMARK ARCHITECTURE

```
paul@tal:~/src/benchbase$ ls -1 \
> src/main/java/com/oltpbenchmark/benchmarks/temporal/
DateRange.java
Employee.java
Position.java
TemporalBenchmark.java
TemporalConfiguration.java
TemporalConstants.java
TemporalLoader.java
TemporalModel.java
TemporalWorker.java
procedures
```

TemporalWorker

```
@Override
protected TransactionStatus executeWork(
  Connection conn, TransactionType nextTrans)
  throws UserAbortException, SQLException {
try {
  if (nextTrans.getProcedureClass().equals(InsertPosition.clas
    RandomEmployee emp = makeRandomEmployee(
            TemporalConstants.CHECK_FK_GAUSSIAN_RANGE,
            config.getMaxYearsInsertPositionRange());
    String duty = TemporalConstants.POSITION_NAMES[
            rng().nextInt(TemporalConstants.POSITION_NAMES.len
    int rank = 1;

    getProcedure(InsertPosition.class).run(conn, emp.id, duty,
```

TemporalWorker

```
@Override
protected TransactionStatus executeWork(
  Connection conn, TransactionType nextTrans)
  throws UserAbortException, SQLException {
try {
  if (nextTrans.getProcedureClass().equals(InsertPosition.clas
    RandomEmployee emp = makeRandomEmployee(
            TemporalConstants.CHECK_FK_GAUSSIAN_RANGE,
            config.getMaxYearsInsertPositionRange());
    String duty = TemporalConstants.POSITION_NAMES[
            rng().nextInt(TemporalConstants.POSITION_NAMES.len
    int rank = 1;

    getProcedure(InsertPosition.class).run(conn, emp.id, duty,
```

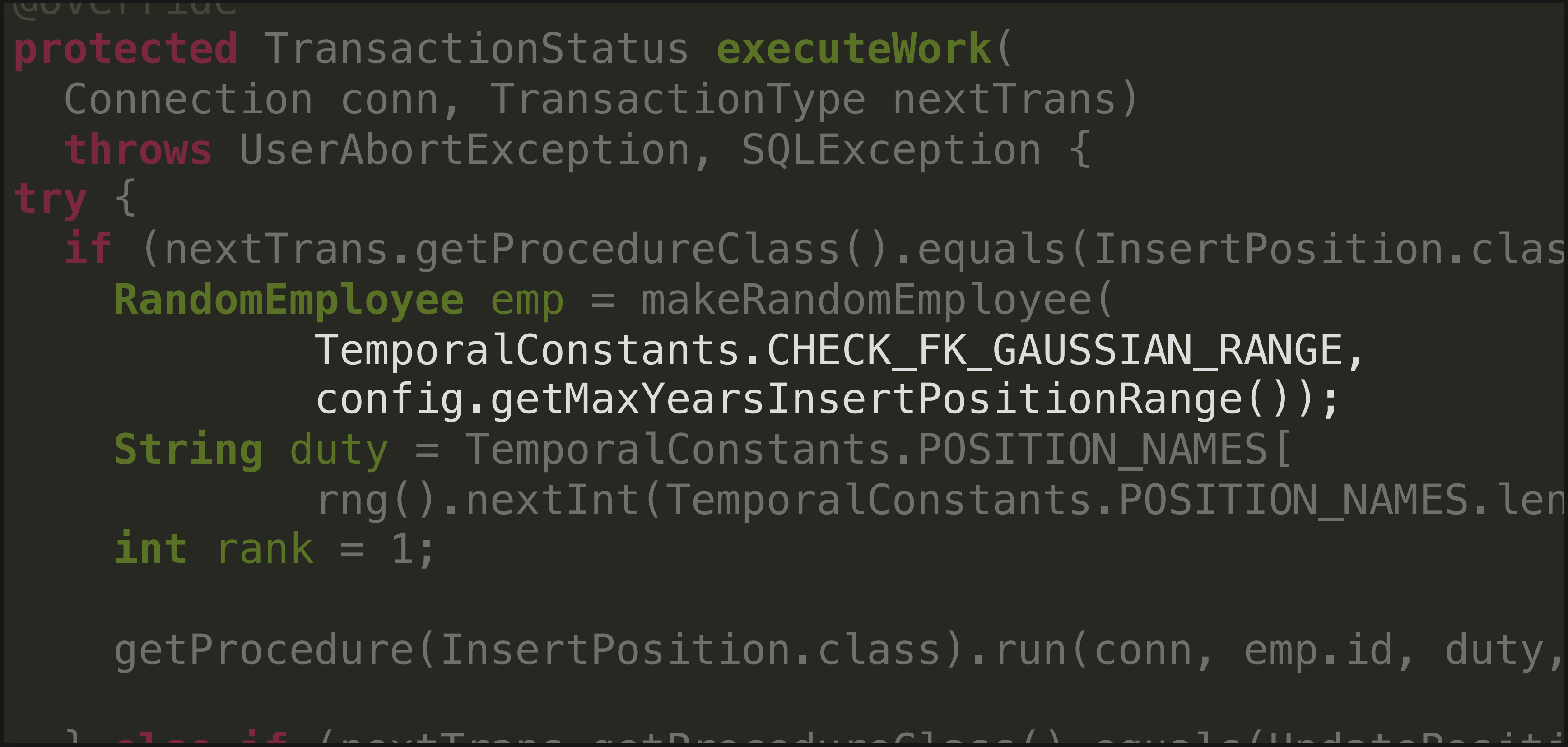
TemporalWorker

```java
@Override
protected TransactionStatus executeWork(
  Connection conn, TransactionType nextTrans)
  throws UserAbortException, SQLException {
try {
  if (nextTrans.getProcedureClass().equals(InsertPosition.clas
    RandomEmployee emp = makeRandomEmployee(
            TemporalConstants.CHECK_FK_GAUSSIAN_RANGE,
            config.getMaxYearsInsertPositionRange());
    String duty = TemporalConstants.POSITION_NAMES[
            rng().nextInt(TemporalConstants.POSITION_NAMES.len
    int rank = 1;

    getProcedure(InsertPosition.class).run(conn, emp.id, duty,
```

TemporalWorker

```
  Connection conn, TransactionType nextTrans)
  throws UserAbortException, SQLException {
try {
  if (nextTrans.getProcedureClass().equals(InsertPosition.clas
    RandomEmployee emp = makeRandomEmployee(
            TemporalConstants.CHECK_FK_GAUSSIAN_RANGE,
            config.getMaxYearsInsertPositionRange());
    String duty = TemporalConstants.POSITION_NAMES[
            rng().nextInt(TemporalConstants.POSITION_NAMES.len
    int rank = 1;

    getProcedure(InsertPosition.class).run(conn, emp.id, duty,

  } else if (nextTrans.getProcedureClass().equals(UpdatePositi
    RandomPosition p = makeRandomPosition(config.getMaxYearsUp
```

TemporalWorker



```
@Override
protected TransactionStatus executeWork(
  Connection conn, TransactionType nextTrans)
  throws UserAbortException, SQLException {
try {
  if (nextTrans.getProcedureClass().equals(InsertPosition.clas
    RandomEmployee emp = makeRandomEmployee(
            TemporalConstants.CHECK_FK_GAUSSIAN_RANGE,
            config.getMaxYearsInsertPositionRange());
    String duty = TemporalConstants.POSITION_NAMES[
            rng().nextInt(TemporalConstants.POSITION_NAMES.len
    int rank = 1;

    getProcedure(InsertPosition.class).run(conn, emp.id, duty,

  } else if (nextTrans.getProcedureClass().equals(UpdatePositi
```

PROCEDURES

```
public class InsertPosition extends Procedure {
  public final SQLStmt insertPosition =
    new SQLStmt(
      "INSERT INTO positions (employee_id, valid_at, name) " +
      "VALUES (?, daterange(?, ?), " +
      "concat(?, ' ', to_char(?, 'RN'))) RETURNING id");

  public int run(
    Connection conn, int employeeId, String duty,
    LocalDate s, LocalDate e, int rank)
    throws SQLException {
    try (PreparedStatement stmt = this.getPreparedStatement(co
      stmt.setInt(1, employeeId);
      stmt.setDate(2, s == null ? null : Date.valueOf(s));
      stmt.setDate(3, e == null ? null : Date.valueOf(e));
```

PROCEDURES

```java
public class InsertPosition extends Procedure {
  public final SQLStmt insertPosition =
    new SQLStmt(
      "INSERT INTO positions (employee_id, valid_at, name) " +
      "VALUES (?, daterange(?, ?), " +
      "concat(?, ' ', to_char(?, 'RN'))) RETURNING id");

  public int run(
    Connection conn, int employeeId, String duty,
    LocalDate s, LocalDate e, int rank)
    throws SQLException {
    try (PreparedStatement stmt = this.getPreparedStatement(co
      stmt.setInt(1, employeeId);
      stmt.setDate(2, s == null ? null : Date.valueOf(s));
      stmt.setDate(3, e == null ? null : Date.valueOf(e));
```

PROCEDURES

```
public class InsertPosition extends Procedure {
  public final SQLStmt insertPosition =
    new SQLStmt(
      "INSERT INTO positions (employee_id, valid_at, name) " +
      "VALUES (?, daterange(?, ?), " +
      "concat(?, ' ', to_char(?, 'RN'))) RETURNING id");

  public int run(
    Connection conn, int employeeId, String duty,
    LocalDate s, LocalDate e, int rank)
    throws SQLException {
    try (PreparedStatement stmt = this.getPreparedStatement(co
      stmt.setInt(1, employeeId);
      stmt.setDate(2, s == null ? null : Date.valueOf(s));
      stmt.setDate(3, e == null ? null : Date.valueOf(e));
```

TESTS

```
paul@tal:~/src/benchbase$ ls \
> src/test/java/com/oltpbenchmark/benchmarks/
auctionmark  resourcestresser  temporal  wikipedia
chbenchmark  seats             tpcc      ycsb
epinions     smallbank         tpch
noop         tatp              twitter
otmetrics    templated         voter
```

TEMPORAL DDL

```sql
DROP EXTENSION IF EXISTS btree_gist CASCADE;
DROP TABLE IF EXISTS employees CASCADE;
DROP TABLE IF EXISTS positions CASCADE;

CREATE EXTENSION btree_gist;

CREATE TABLE employees (
  id          int GENERATED BY DEFAULT AS IDENTITY NOT NULL,
  valid_at    daterange NOT NULL,
  name        text NOT NULL,
  salary      int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS)
);

CREATE TABLE positions (
```

TEMPORAL DDL

```
DROP TABLE IF EXISTS positions CASCADE;

CREATE EXTENSION btree_gist;

CREATE TABLE employees (
  id            int GENERATED BY DEFAULT AS IDENTITY NOT NULL,
  valid_at      daterange NOT NULL,
  name          text NOT NULL,
  salary        int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS)
);

CREATE TABLE positions (
  id            int GENERATED BY DEFAULT AS IDENTITY NOT NULL,
  valid_at      daterange NOT NULL,
```

TEMPORAL DDL

```sql
  salary      int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS)
);

CREATE TABLE positions (
  id          int GENERATED BY DEFAULT AS IDENTITY NOT NULL,
  valid_at    daterange NOT NULL,
  name        text NOT NULL,
  employee_id int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS),
  FOREIGN KEY (employee_id, PERIOD valid_at)
    REFERENCES employees (id, PERIOD valid_at)
);
CREATE INDEX idx_positions_employee_id ON positions
  USING gist (employee_id, valid_at);
```

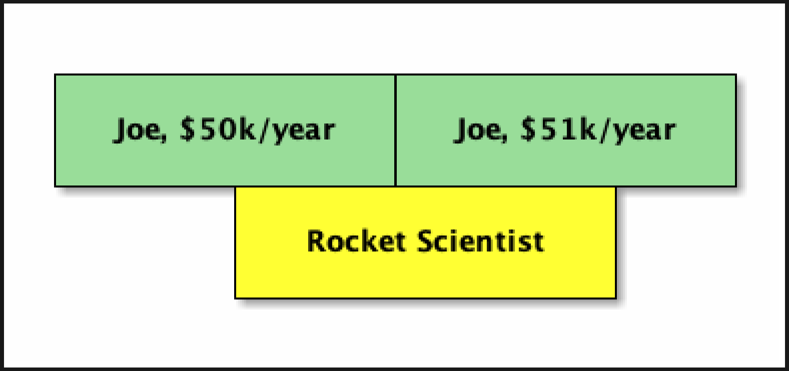
TEMPORAL DDL

```sql
  salary      int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS)
);

CREATE TABLE positions (
  id          int GENERATED BY DEFAULT AS IDENTITY NOT NULL,
  valid_at    daterange NOT NULL,
  name        text NOT NULL,
  employee_id int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS),
  FOREIGN KEY (employee_id, PERIOD valid_at)
    REFERENCES employees (id, PERIOD valid_at)
);
CREATE INDEX idx_positions_employee_id ON positions
  USING gist (employee_id, valid_at);
```

TEMPORAL DDL

```
  salary      int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS)
);

CREATE TABLE positions (
  id          int GENERATED BY DEFAULT AS IDENTITY NOT NULL,
  valid_at    daterange NOT NULL,
  name        text NOT NULL,
  employee_id int NOT NULL,
  PRIMARY KEY (id, valid_at WITHOUT OVERLAPS),
  FOREIGN KEY (employee_id, PERIOD valid_at)
    REFERENCES employees (id, PERIOD valid_at)
);
CREATE INDEX idx_positions_employee_id ON positions
  USING gist (employee_id, valid_at);
```

TEMPORAL FOREIGN KEYS



range_agg IMPLEMENTATION

```
SELECT 1
FROM    (
  SELECT pkperiodatt AS r
  FROM   [ONLY] pktable x
  WHERE  pkatt1 = $1 [AND ...]
  AND    pkperiodatt && $n
FOR KEY SHARE OF x
) x1
HAVING $n <@ range_agg(x1.r)
```

EXISTS IMPLEMENTATION

```
SELECT 1
-- There was a PK when the FK started:
WHERE EXISTS
  SELECT  1
  FROM    [ONLY] <pktable>
  WHERE   pkatt1 = $1 [AND ...]
  AND     COALESCE(lower(pkperiodatt), '-Infinity')
       <= COALESCE(lower($n), '-Infinity')
  AND     COALESCE(lower($n), '-Infinity')
       <  COALESCE(upper(pkperiodatt), 'Infinity')
)
...
```

EXISTS IMPLEMENTATION

```
-- There was a PK when the FK ended:
AND EXISTS (
  SELECT  1
  FROM    [ONLY] <pktable>
  WHERE   pkatt1 = $1 [AND ...]
  AND     COALESCE(lower(pkperiodatt), '-Infinity')
       <  COALESCE(upper($n), 'Infinity')
  AND     COALESCE(upper($n), 'Infinity')
       <= COALESCE(upper(pkperiodatt), 'Infinity')
)
...
```

EXISTS IMPLEMENTATION

```
-- There are no gaps in the PK:
-- (i.e. there is no PK that ends early,
-- unless a matching PK record starts right away)
AND NOT EXISTS (
  SELECT  1
  FROM    [ONLY] <pktable> AS pk1
  WHERE   pkatt1 = $1 [AND ...]
  AND     COALESCE(lower($n), '-Infinity')
       <  COALESCE(upper(pkperiodatt), 'Infinity')
  AND     COALESCE(upper(pkperiodatt), 'Infinity')
       <  COALESCE(upper($n), 'Infinity')
  ...
```

EXISTS IMPLEMENTATION

```
  AND     NOT EXISTS (
    SELECT  1
    FROM    [ONLY] <pktable> AS pk2
    WHERE   pk1.pkatt1 = pk2.pkatt1 [AND ...]
            -- but skip pk1.pkperiodatt && pk2.pkperiodatt
    AND     COALESCE(lower(pk2.pkperiodatt), '-Infinity')
         <= COALESCE(upper(pk1.pkperiodatt), 'Infinity')
            COALESCE(upper(pk1.pkperiodatt), 'Infinity')
         <  COALESCE(upper(pk2.pkperiodatt), 'Infinity')
  )
)
```

lag IMPLEMENTATION

```
SELECT  1
FROM    (
  SELECT  uk.uk_start_value,
          uk.uk_end_value,
          NULLIF(LAG(uk.uk_end_value) OVER (ORDER BY uk.uk_sta
  FROM   (
    SELECT  coalesce(lower(x.pkperiodatt), '-Infinity') AS uk_
            coalesce(upper(x.pkperiodatt), 'Infinity') AS uk_e
    FROM    pktable AS x
    WHERE   pkatt1 = $1 [AND ...]
    AND     uk.pkperiodatt && $n
    FOR KEY SHARE OF x
  ) AS uk
) AS uk
WHERE   uk.uk_start_value < upper($n)
```

COMPILER FLAGS

```
#if defined(RI_TEMPORAL_IMPL_LAG)
    quoteOneName(attname,

    appendStringInfo(&querybuf, "SELECT 1 FROM ( ");
    appendStringInfo(&querybuf, "  SELECT  uk.uk_start_value,
    appendStringInfo(&querybuf, "          NULLIF(LAG(uk.uk_en
    appendStringInfo(&querybuf, "  FROM    ( ");
    appendStringInfo(&querybuf, "    SELECT  COALESCE(LOWER(x.
    appendStringInfo(&querybuf, "            COALESCE(UPPER(x.
    appendStringInfo(&querybuf, "    FROM    %s%s AS x", pk_on
#elif defined(RI_TEMPORAL_IMPL_EXISTS)
    appendStringInfo(&querybuf,
                     "SELECT 1 ");
#else
    quoteOneName(attname,
```

EXPLAIN range_agg

```
Aggregate
  Filter: ('[2020-10-10,2020-12-12)'::daterange <@ range_agg(x
  ->  Subquery Scan on x1
    ->  LockRows
      ->  Index Scan using employees_pkey on employees x
        Index Cond: ((id = 500) AND (valid_at && '[2020-10-10,
```

EXPLAIN lag

```
Aggregate
  Filter: ((array_agg(uk.x) FILTER (WHERE (uk.x IS NOT NULL))
  ->  Subquery Scan on uk
    Filter: ((uk.uk_start_value < '2020-12-12'::date) AND (uk.
    ->  WindowAgg
      ->  Sort
        Sort Key: uk_1.uk_start_value
        ->  Subquery Scan on uk_1
          ->  LockRows
            ->  Index Scan using employees_pkey on employees x
              Index Cond: ((id = 500) AND (valid_at && '[2020-
```

EXPLAIN EXISTS

```
Result
  One-Time Filter: ((InitPlan 1).col1 AND (InitPlan 2).col1 AN
  InitPlan 1
  ->  LockRows
    ->  Index Scan using employees_pkey on employees x
      Index Cond: ((id = 500) AND (valid_at && '[2020-10-10,20
      Filter: ((COALESCE(lower(valid_at), '-infinity'::date) <
  InitPlan 2
  ->  LockRows
    ->  Index Scan using employees_pkey on employees x_1
      Index Cond: ((id = 500) AND (valid_at && '[2020-10-10,20
      Filter: ((COALESCE(lower(valid_at), '-infinity'::date) <
  InitPlan 4
  ->  LockRows
    ->  Index Scan using employees_pkey on employees pk1
```

ASSUMPTIONS

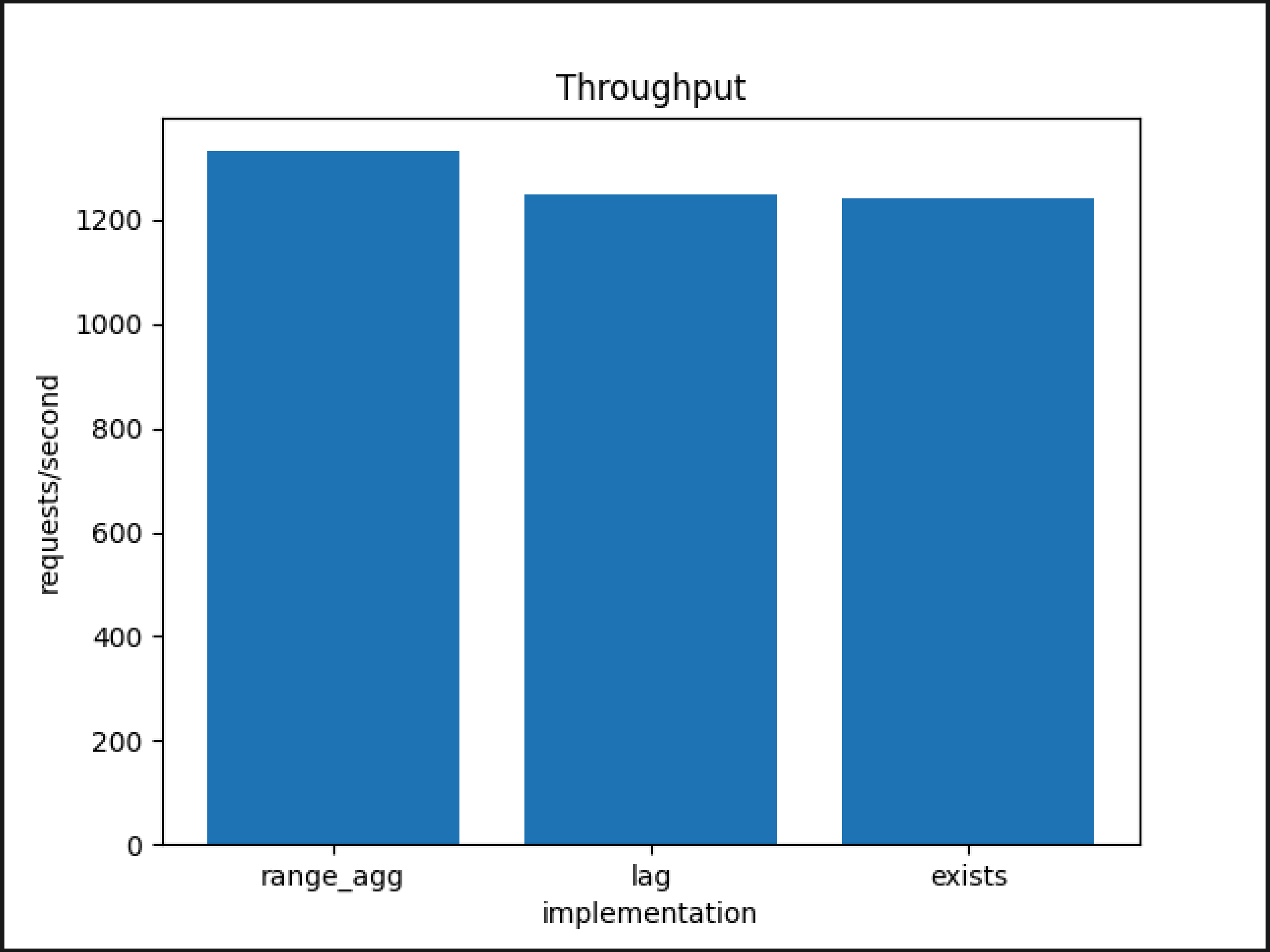
Relevant:

- CPU
- Tuples examined
- Number of index scans

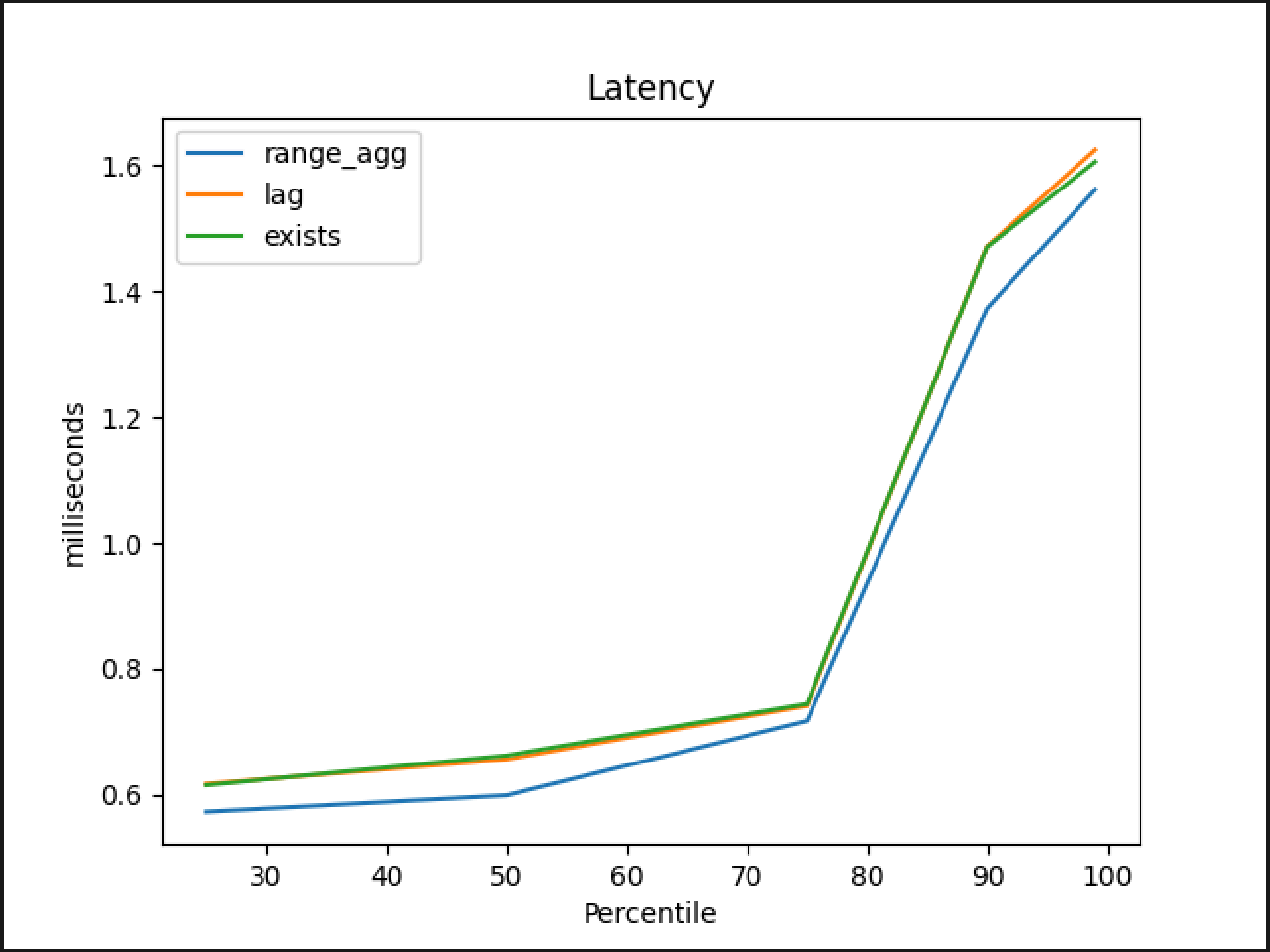
Not relevant:

- Shared Buffers
- I/O

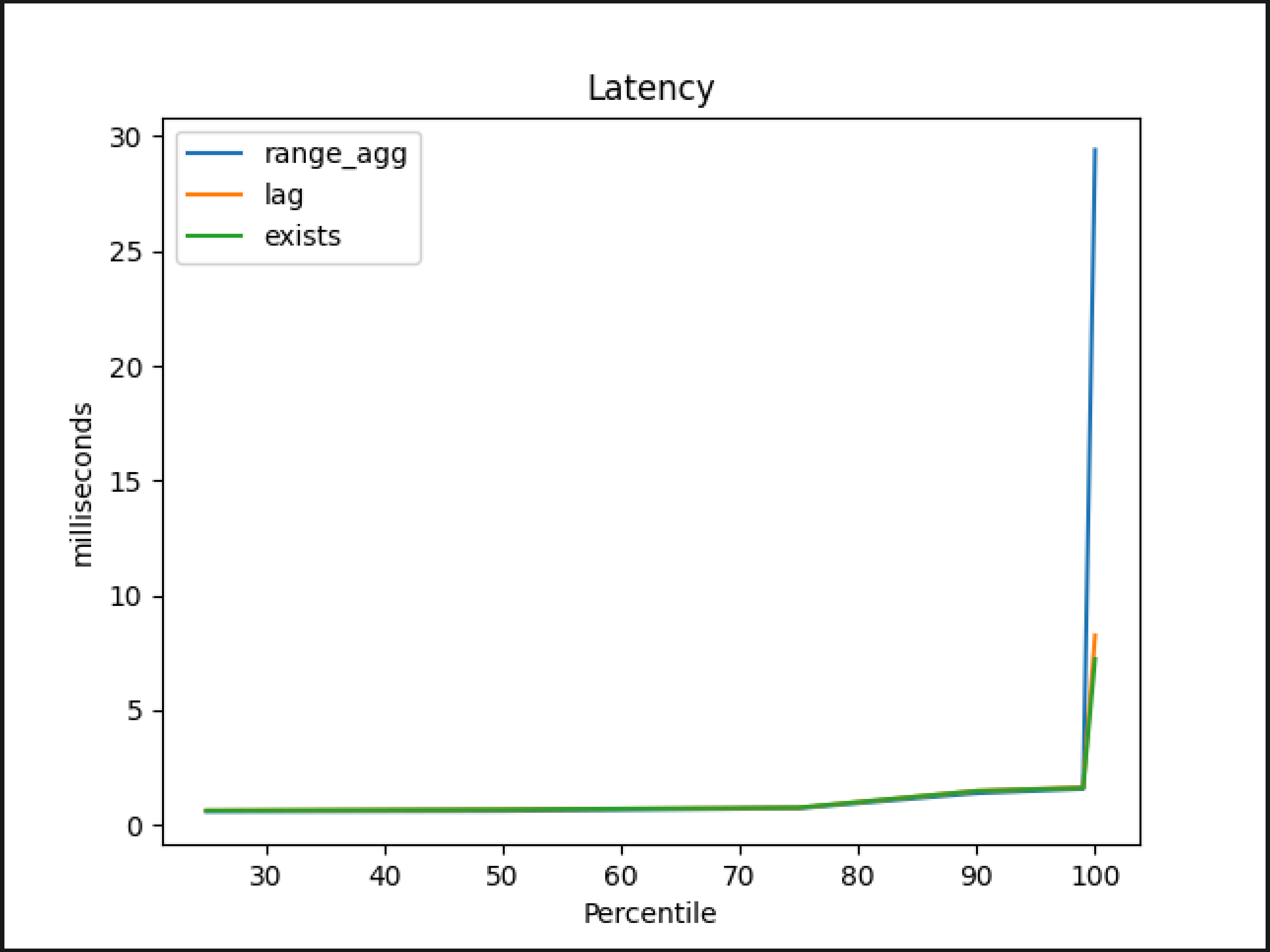
EARLY RESULTS



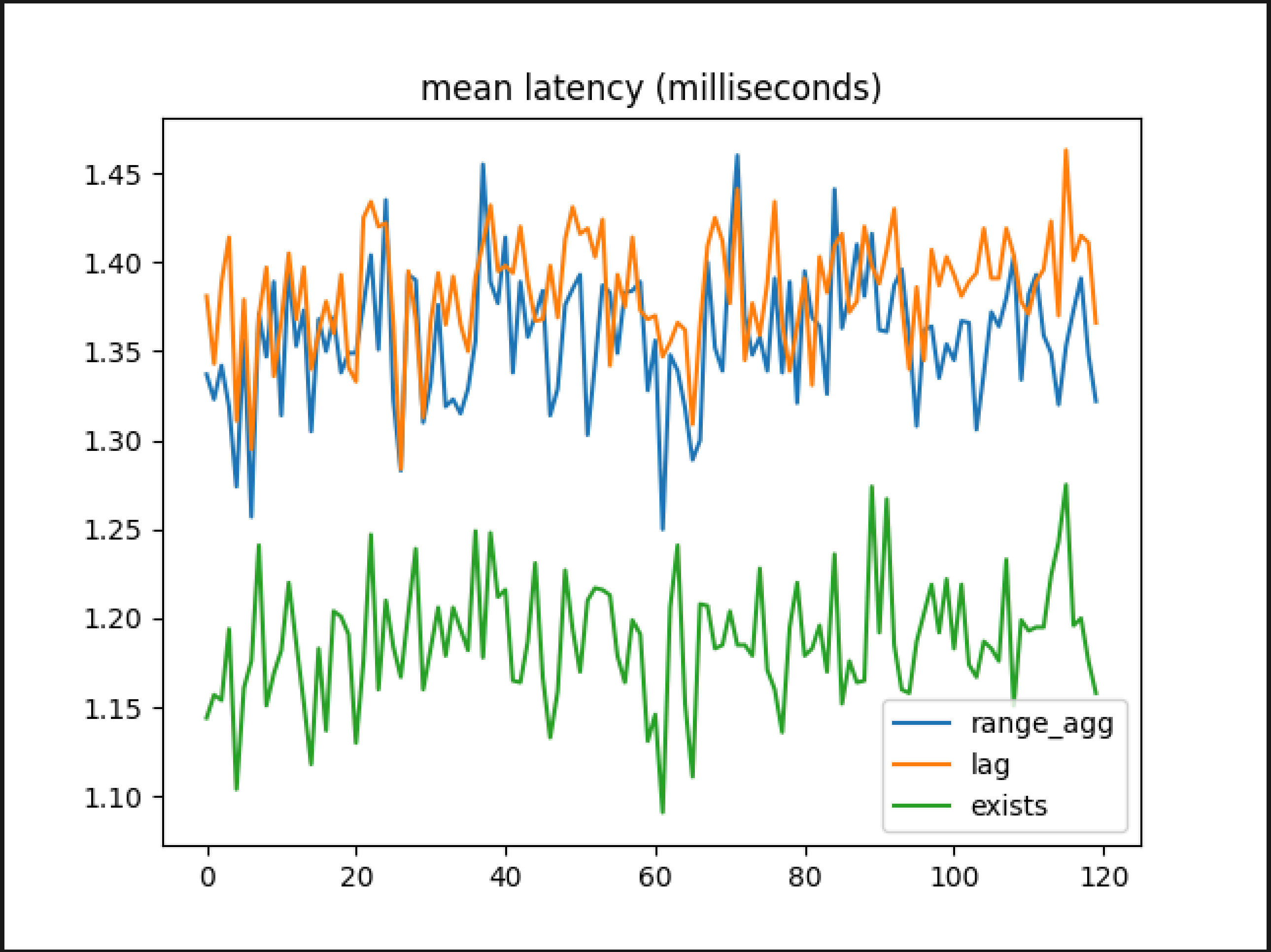
EARLY RESULTS



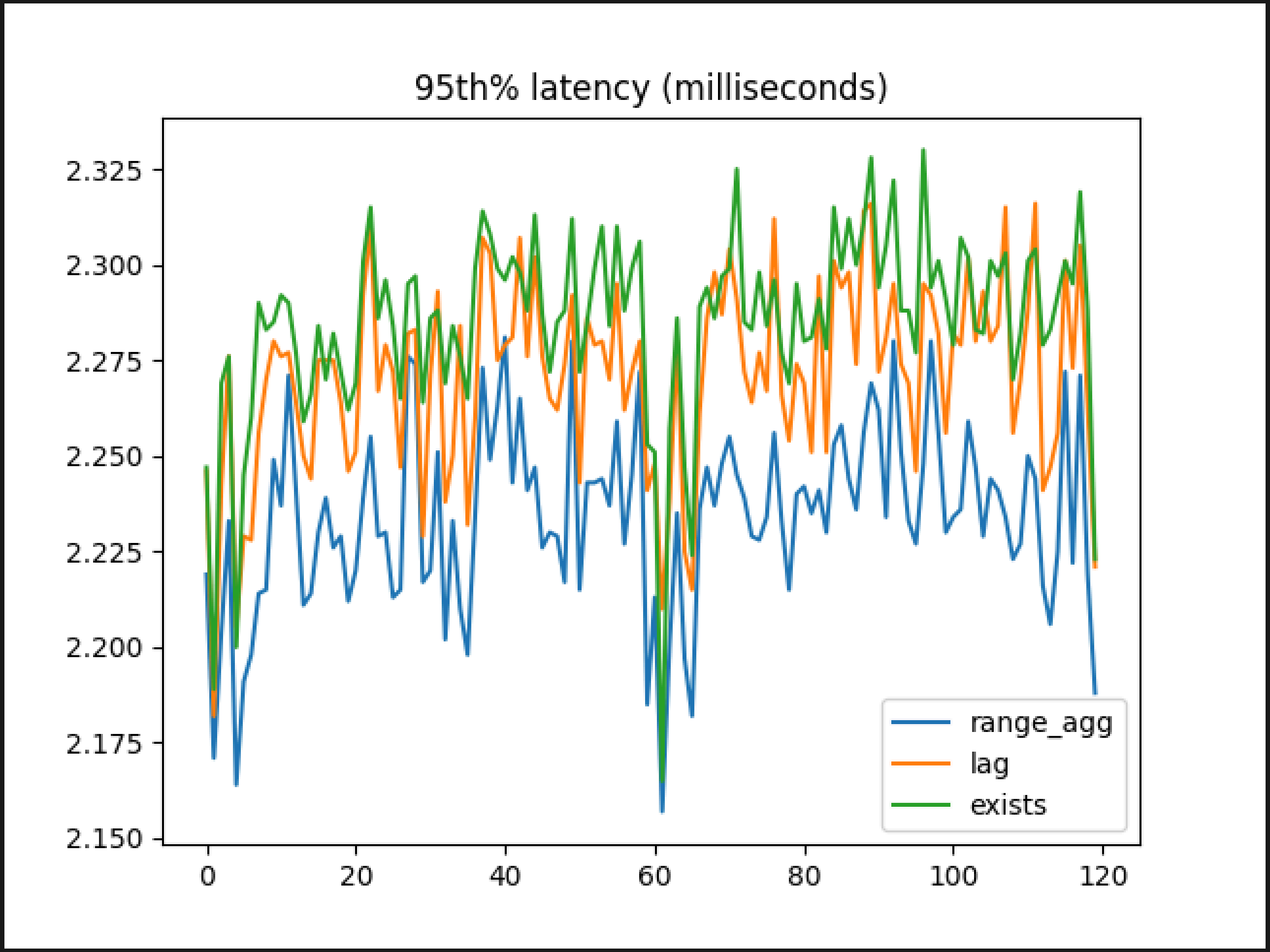
EARLY RESULTS



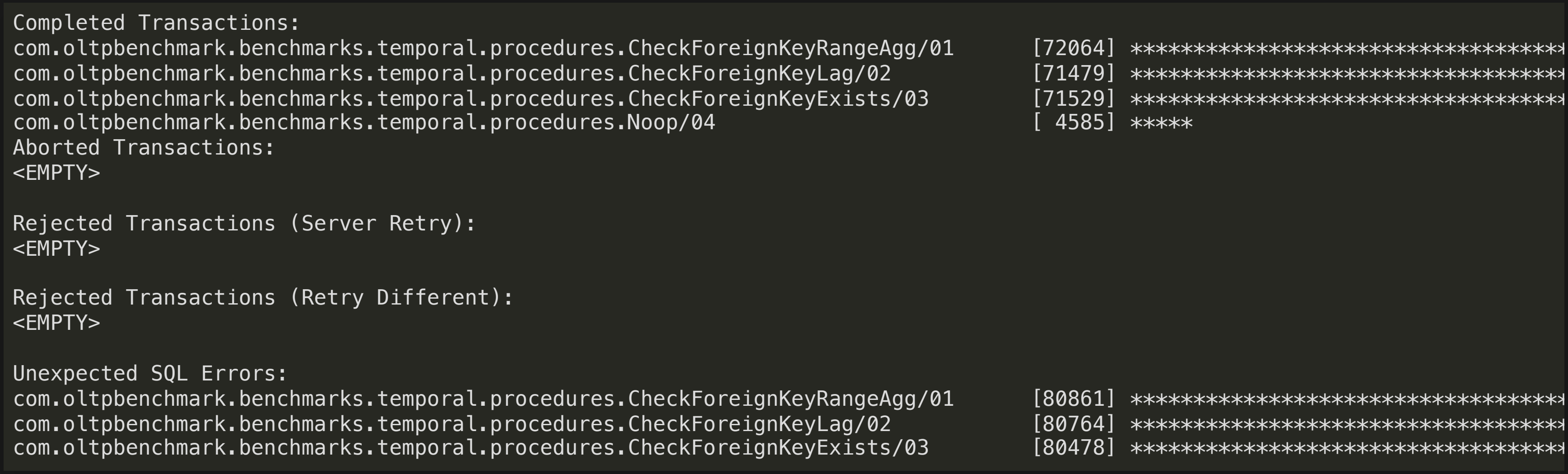
SURPRISE


mean latency (milliseconds)
1.45
1.40
1.35
1.30
1.25
1.20
1.15
1.10
0
20
40
60
80
100
120
range_agg
lag
exists

SURPRISE


95th% latency (milliseconds)
2.325
2.300
2.275
2.250
2.225
2.200
2.175
2.150
0
20
40
60
80
100
120
range_agg
lag
exists

50% ERRORS



```
Completed Transactions:
com.oltpbenchmark.benchmarks.temporal.procedures.CheckForeignKeyRangeAgg/01      [72064] ***********************************
com.oltpbenchmark.benchmarks.temporal.procedures.CheckForeignKeyLag/02           [71479] ***********************************
com.oltpbenchmark.benchmarks.temporal.procedures.CheckForeignKeyExists/03        [71529] ***********************************
com.oltpbenchmark.benchmarks.temporal.procedures.Noop/04                         [ 4585] *****
Aborted Transactions:
<EMPTY>

Rejected Transactions (Server Retry):
<EMPTY>

Rejected Transactions (Retry Different):
<EMPTY>

Unexpected SQL Errors:
com.oltpbenchmark.benchmarks.temporal.procedures.CheckForeignKeyRangeAgg/01      [80861] ***********************************
com.oltpbenchmark.benchmarks.temporal.procedures.CheckForeignKeyLag/02           [80764] ***********************************
com.oltpbenchmark.benchmarks.temporal.procedures.CheckForeignKeyExists/03        [80478] ***********************************
```

EXPLAIN ANALYZE EXISTS

```
Result (actual time=0.034..0.035 rows=0 loops=1)
  One-Time Filter: ((InitPlan 1).col1 AND (InitPlan 2).col1 AN
  InitPlan 1
  ->  LockRows (actual time=0.033..0.033 rows=0 loops=1)
    ->  Index Scan using employees_pkey on employees x (actual
      Index Cond: ((id = 5999) AND (valid_at && '[2020-10-10,2
      Filter: ((COALESCE(lower(valid_at), '-infinity'::date) <
  InitPlan 2
  ->  LockRows (never executed)
    ->  Index Scan using employees_pkey on employees x_1 (neve
      Index Cond: ((id = 5999) AND (valid_at && '[2020-10-10,2
      Filter: ((COALESCE(lower(valid_at), '-infinity'::date) <
  InitPlan 4
  ->  LockRows (never executed)
    ->  Index Scan using employees_pkey on employees pk1 (neve
```

EXPLAIN ANALYZE EXISTS

```
    Filter: ((COALESCE(lower(valid_at), '-infinity'::date) <
InitPlan 2
->  LockRows (never executed)
  ->  Index Scan using employees_pkey on employees x_1 (neve
    Index Cond: ((id = 5999) AND (valid_at && '[2020-10-10,2
    Filter: ((COALESCE(lower(valid_at), '-infinity'::date) <
InitPlan 4
->  LockRows (never executed)
  ->  Index Scan using employees_pkey on employees pk1 (neve
    Index Cond: ((id = 5999) AND (valid_at && '[2020-10-10,2
    Filter: (('2020-10-10'::date < COALESCE(upper(valid_at),
    SubPlan 3
    ->  LockRows (never executed)
      ->  Index Scan using employees_pkey on employees pk2 (
        Index Cond: (id = pk1.id)
```

bpftrace ExecProcNode

```
// Count how many exec nodes per query were required,
// and print a histogram of how often each count happens.
// Run this for each FK implementation separately.
// My hypothesis is that the EXISTS implementation calls ExecP
// but only if the FK is invalid.

u:/home/paul/local/bench-*/bin/postgres:standard_ExecutorStart
  @nodes[tid] = 0
}
u:/home/paul/local/bench-*/bin/postgres:ExecProcNode {
  @nodes[tid] += 1
}
u:/home/paul/local/bench-*/bin/postgres:standard_ExecutorEnd {
  @calls = hist(@nodes[tid]);
  delete(@nodes[tid]);
```

bpftrace ExecProcNode

all valid:

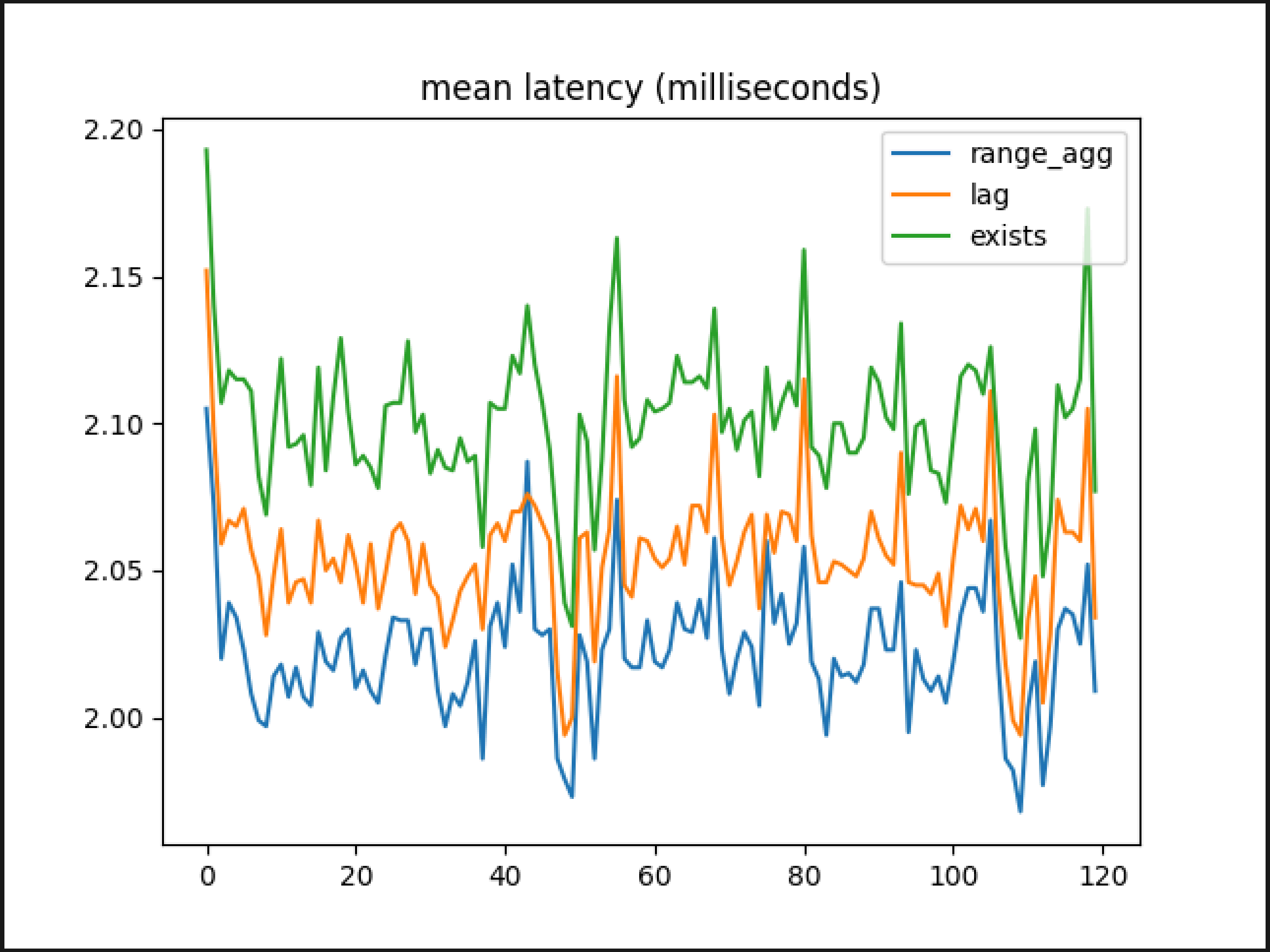
```
@calls:
[0]                    6 |
[1]                    0 |
[2, 4)                 0 |
[4, 8)            228851 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[8, 16)                1 |
[16, 32)               1 |
[32, 64)               2 |
[64, 128)              2 |
[128, 256)             2 |
[256, 512)             5 |
```

bpftrace ExecProcNode

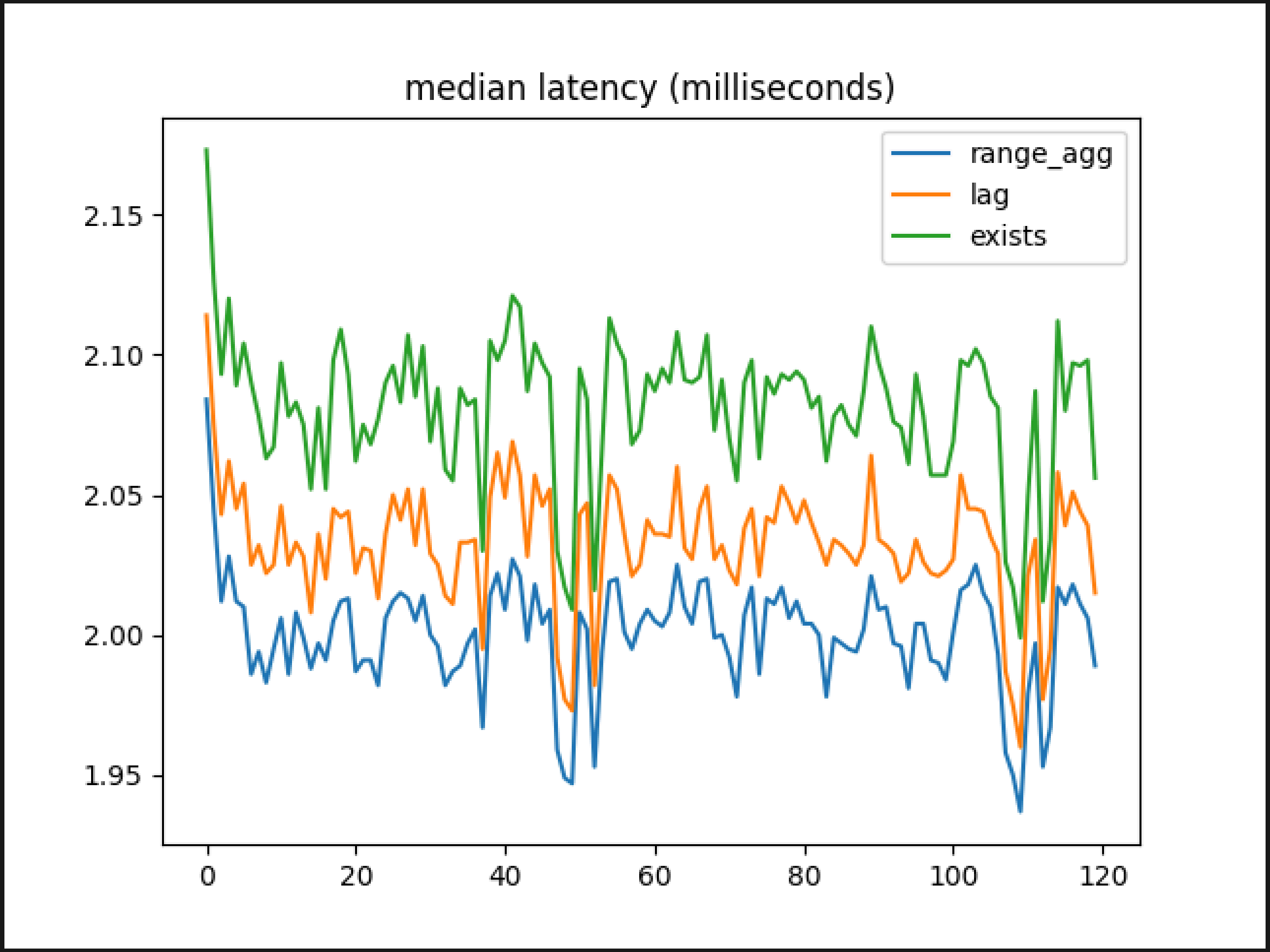
50+% invalid:

```
@calls:
[0]                    6 |
[1]                    0 |
[2, 4)            218294 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[4, 8)            183438 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[8, 16)              231 |
[16, 32)               1 |
[32, 64)               2 |
[64, 128)              2 |
[128, 256)             2 |
[256, 512)             5 |
```

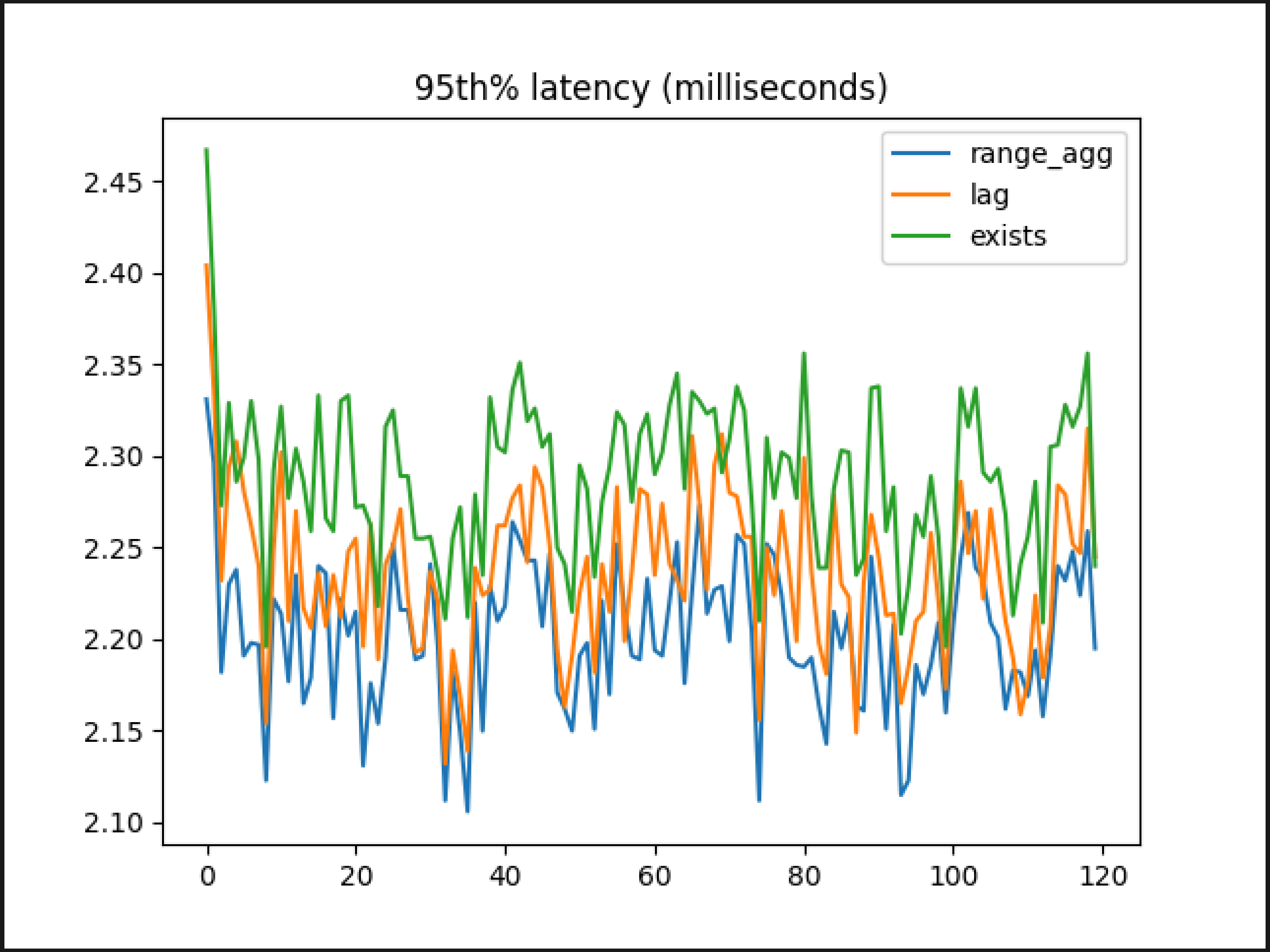
MOSTLY VALID



MOSTLY VALID



MOSTLY VALID



METHODOLOGY

- Short iterations
- Automate
- Keep notes
- Automating keeping notes?

NEXT

- More questions
- More procedures
- How to distribute the benchmark?

THANKS!

REFERENCES

THIS TALK

- This talk: https://github.com/pjungwir/pdxpug2024-benchbase-and-temporal-foreign-keys
- Temporal benchmark on my Basebase branch: https://github.com/pjungwir/benchbase/tree/temporal
- Benchmark notes and tools: https://github.com/pjungwir/benchmarking-temporal-tables

BENCHBASE

- Djellel Eddine Difallah and Andrew Pavlo and Carlo Curino and Philippe Cudré-Mauroux, "OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases," *PVLDB* 7.4, 2013, http://www.vldb.org/pvldb/vol7/p277-difallah.pdf
- Original OLTP-Bench repo wiki: https://github.com/oltpbenchmark/oltpbench/wiki
- Benchbase repo: https://github.com/cmu-db/benchbase
- PR to fix the exec:java target: https://github.com/cmu-db/benchbase/pull/548

TEMPORAL

- Richard Snodgrass, *Developing Time-Oriented Applications in SQL*, https://www2.cs.arizona.edu/~rts/tdbbook.pdf
- periods extension: https://github.com/xocolatl/periods
- My temporal patches: https://commitfest.postgresql.org/49/4308/
- My temporal branch: https://github.com/pjungwir/postgresql/tree/valid-time
- My temporal FK comparison branch: https://github.com/pjungwir/postgresql/tree/temporal-fk-comparison

BENCHMARKING

- Andres Freund, "Analyzing Postgres performance problems using perf and eBPF," https://www.youtube.com/watch?v=HghP4D72Noc
- Claire Giordano, "How I got started as a developer (& in Postgres), with Melanie Plageman & Thomas Munro," PathToCitusCon episode 4, https://www.youtube.com/watch?v=72OdrpZXjEg
- Mark Callaghan, Small Datum, https://smalldatum.blogspot.com
- Melanie Plageman, "Visualizing Postgres I/O Performance," PGCon 2023, https://www.youtube.com/watch?v=CxyPZHG5beI
- Melanie Plageman, "Postgres Performance Observability Sources and Analysis Techniques," https://www.youtube.com/watch?v=laxZdbE1Nuw
- Michael Christofides and Nikolay Samokhvalov, "Getting started with benchmarking," Postgres.FM episode 110, https://www.youtube.com/watch?v=xR-VJjR9DPQ