# PostgreSQL Performance Tuning

Bruce Momjian


PostgreSQL is an open-source, full-featured relational database. This presentation gives an overview of PostgreSQL performance tuning.

 *http://momjian.us/presentations*

*Last updated: January, 2017*

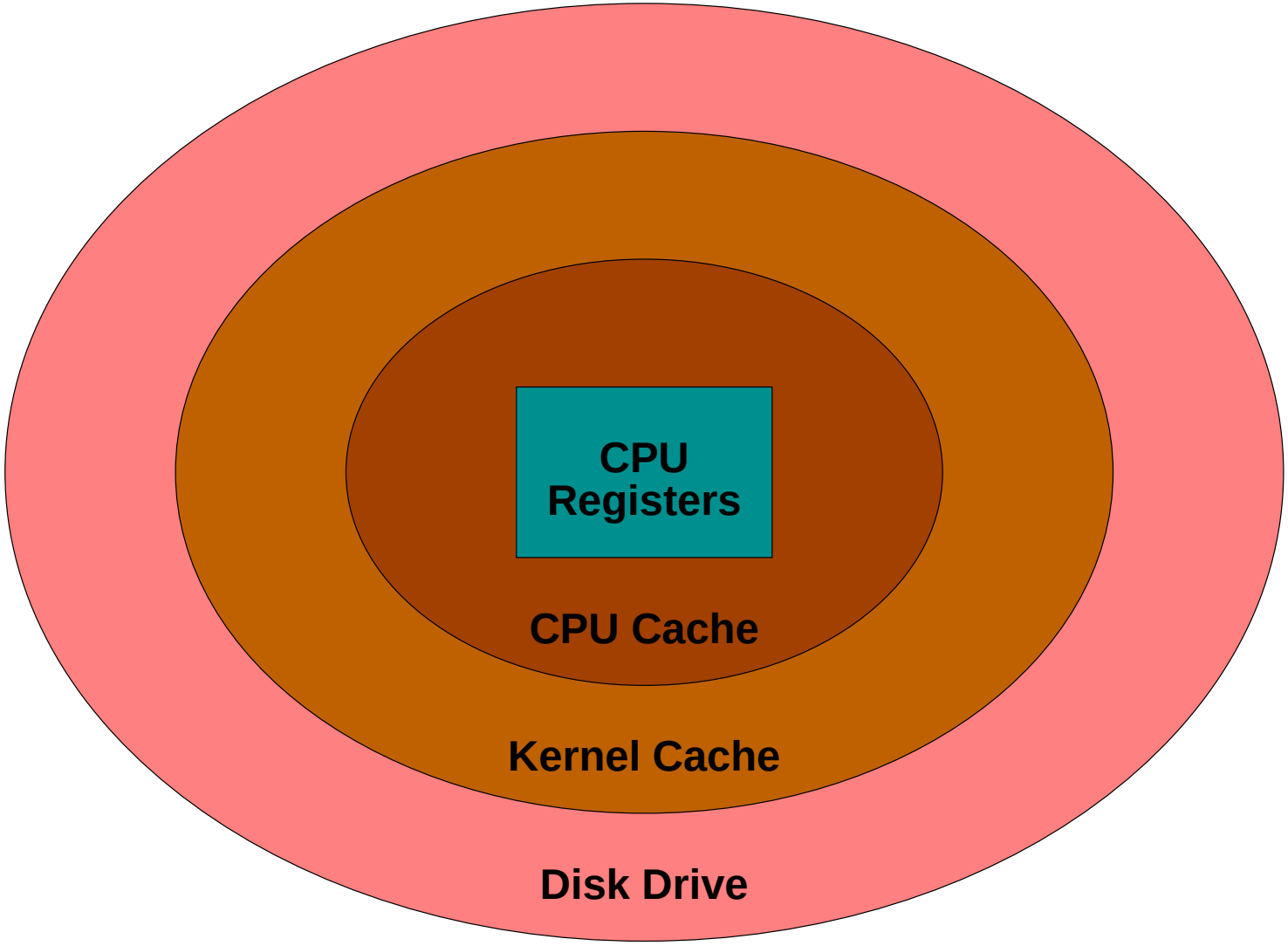
# Outline

1. Caching
2. Internals
3. Storage

# Caching

https://www.flickr.com/photos/storm-crypt/

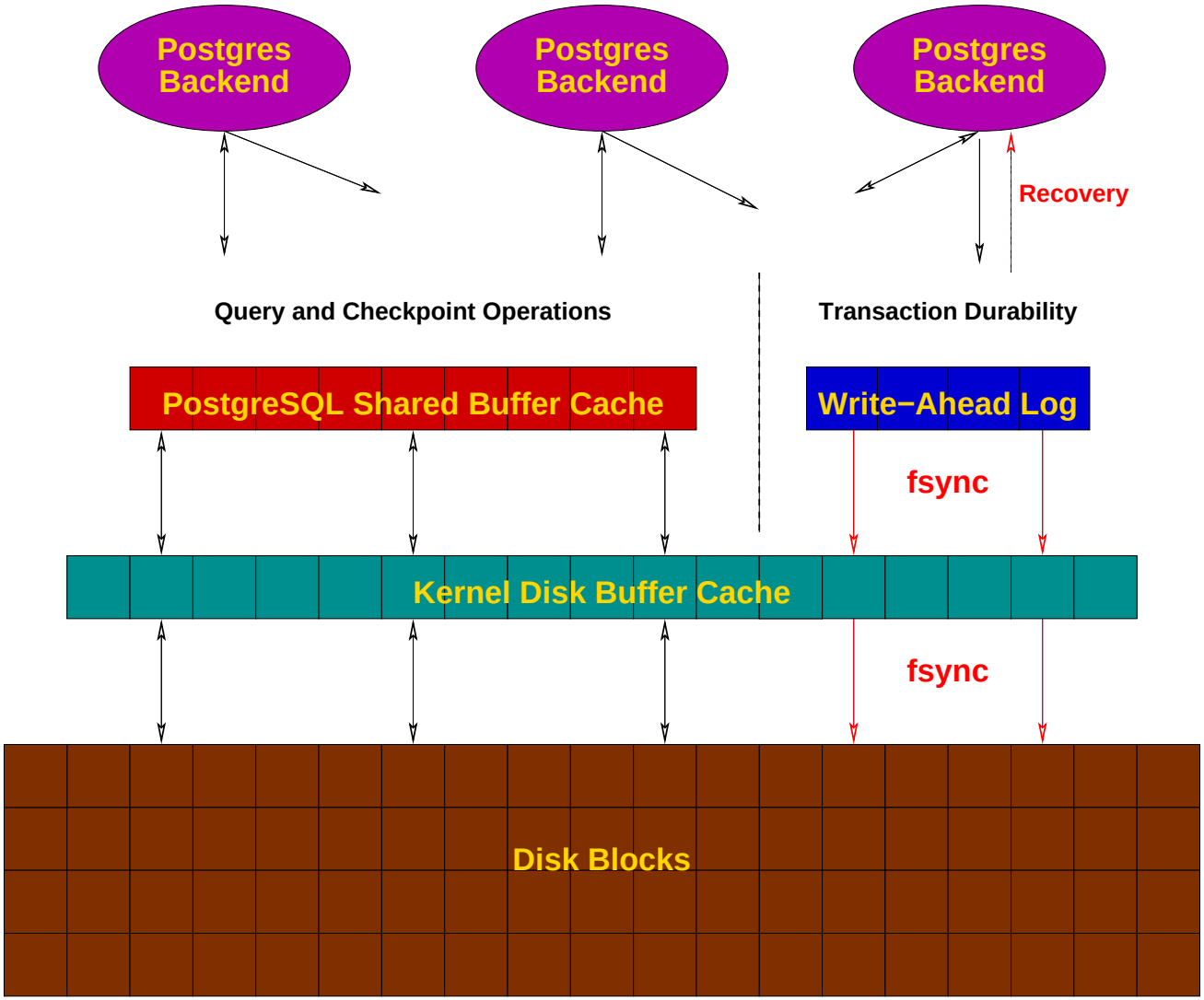
# Caches



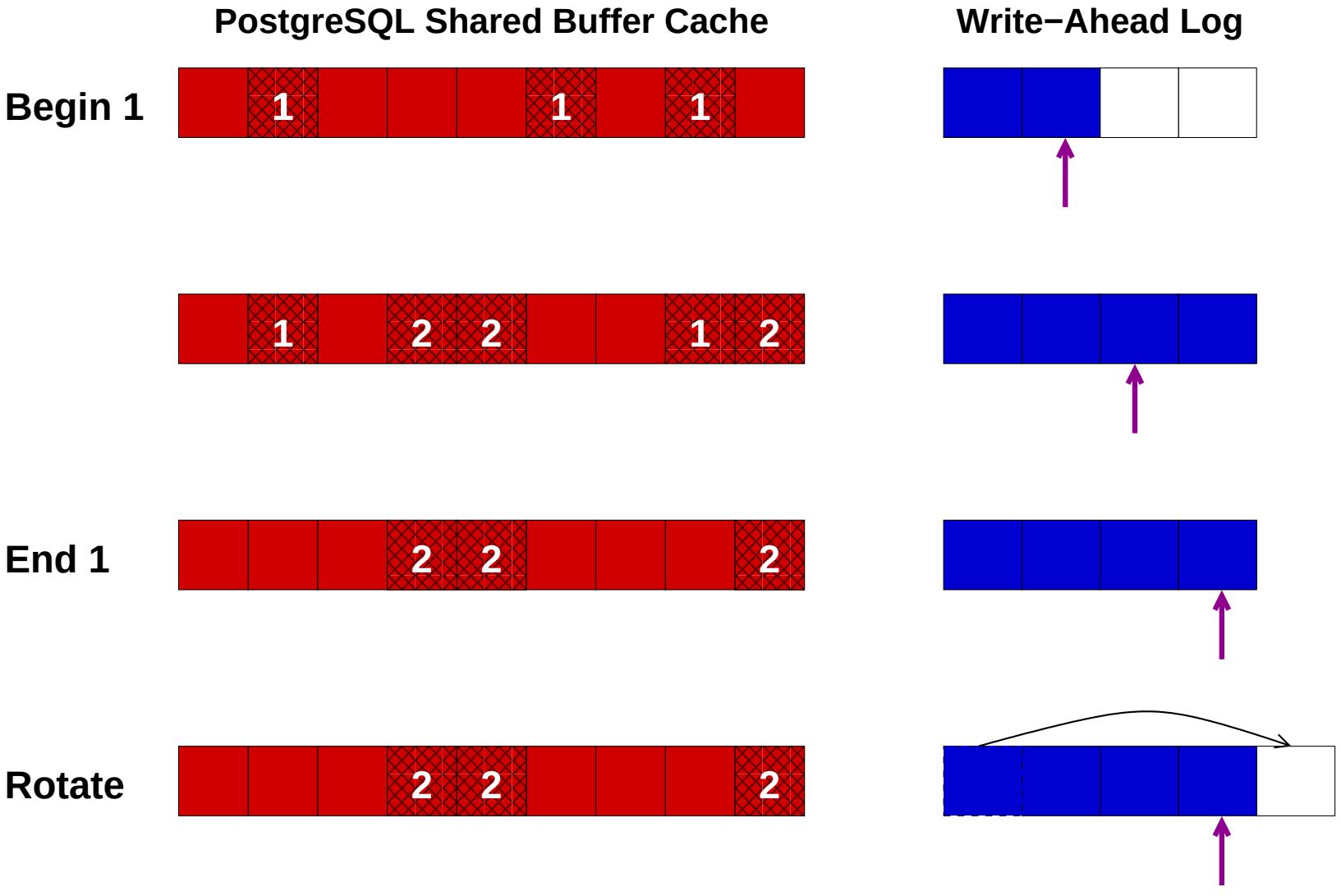
# Cache Sizes

| Storage Area | Measured in |
|---|---|
| CPU registers | bytes |
| CPU cache | megabytes |
| RAM | gigabytes |
| disk drives | terabytes |

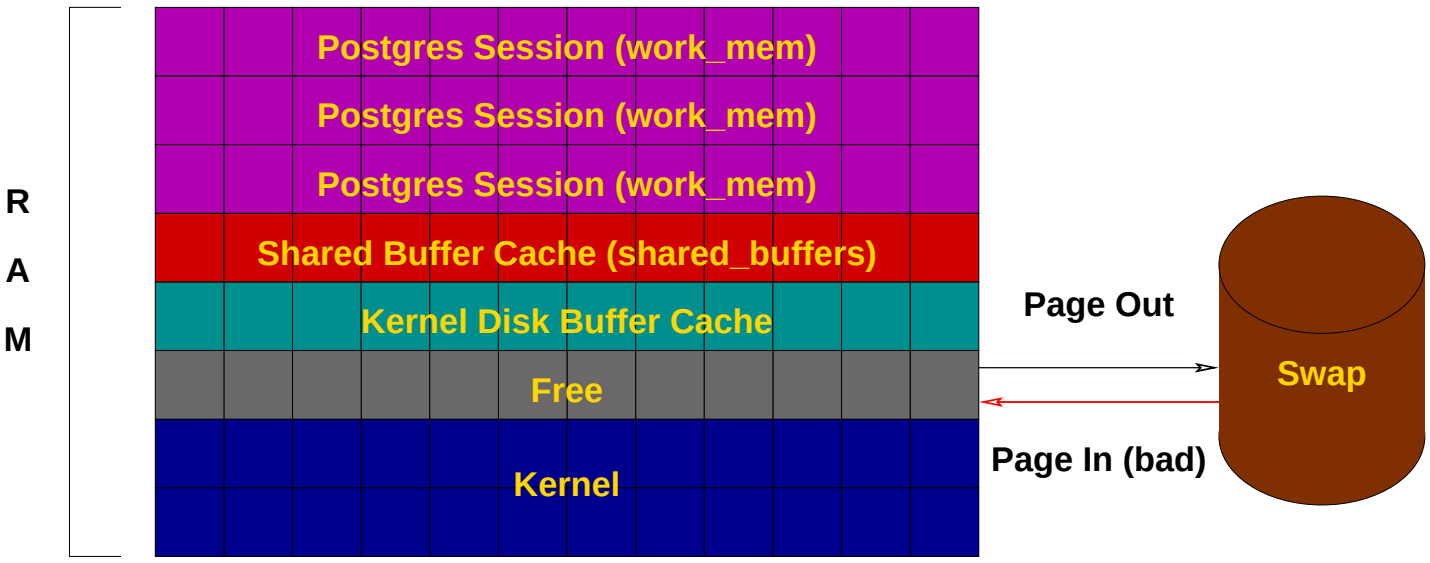
# Checkpoints and WAL Files



# Buffer / Disk Interaction



# Memory Usage



# Postgresql.conf Cache Parameters

```
shared_buffers = 32MB                   # min 128kB
                                        # (change requires restart)
#temp_buffers = 8MB                     # min 800kB

#work_mem = 1MB                         # min 64kB
#maintenance_work_mem = 16MB            # min 1MB

#effective_cache_size = 128MB
```

Kernel changes often required.

# Internals

The Anatomy Lesson of Dr. Nicolaes Tulp, Rembrandt van Rijn

# SQL Query

```
SELECT firstname
FROM friend
WHERE age = 33;
```

# Query in Psql

```
test=> SELECT firstname
test-> FROM friend
test-> WHERE age = 33;
    firstname
------------------
 Sandy
(1 row)
```

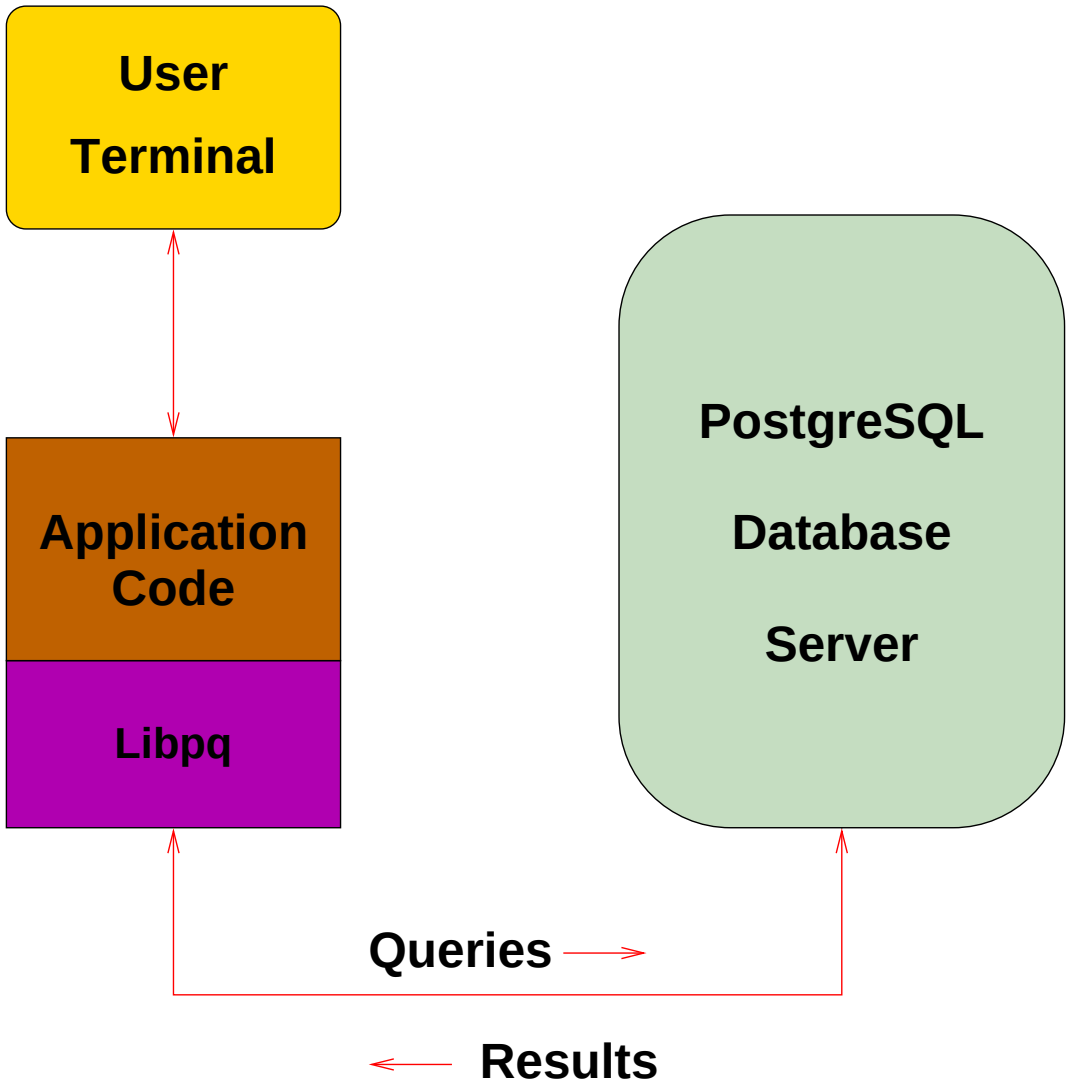
# Query Processing

```
test=> SELECT firstname
test-> FROM friend
test-> WHERE age = 33;

[ query is processed ]

    firstname
-----------------
 Sandy
(1 row)
```

# Query in Libpq

```
test=> SELECT firstname
test-> FROM friend
test-> WHERE age = 33;

Breakpoint 1, PQexec (conn=0x807a000,
    query=0x8081200 "SELECT firstname\nFROM friend\nWHERE age = 33
    at fe-exec.c:1195
```

# Libpq


User
Terminal
Application
Code
Libpq
PostgreSQL
Database
Server
Queries
Results

# TCP/IP Packet

```
17:05:22.715714 family.home.49165 > candle.navpoint.com.5432: P 354:400(46)
ack 61 win 8760 <nop,nop,timestamp 137847 7276138> (DF)

    0000: 00 d0 b7 b9 b6 c8 00 02   b3 04 09 dd 08 00 45 00  ________ ______E_
    0010: 00 62 45 31 40 00 40 06   b1 fe ac 14 00 02 a2 21  _bE1@_@_ ________!
    0020: f5 2e c0 0d 15 38 1c af   94 34 a8 1a 1e 39 80 18  _.___8__ _4___9__
    0030: 22 38 19 d5 00 00 01 01   08 0a 00 02 1a 77 00 6f  "8______ _____w_o
    0040: 06 6a 51 53 45 4c 45 43   54 20 66 69 72 73 74 6e  _jQSELEC T firstn
    0050: 61 6d 65 0a 46 52 4f 4d   20 66 72 69 65 6e 64 0a  ame_FROM  friend_
    0060: 57 48 45 52 45 20 61 67   65 20 3d 20 33 33 3b 00  WHERE ag e = 33;_
```

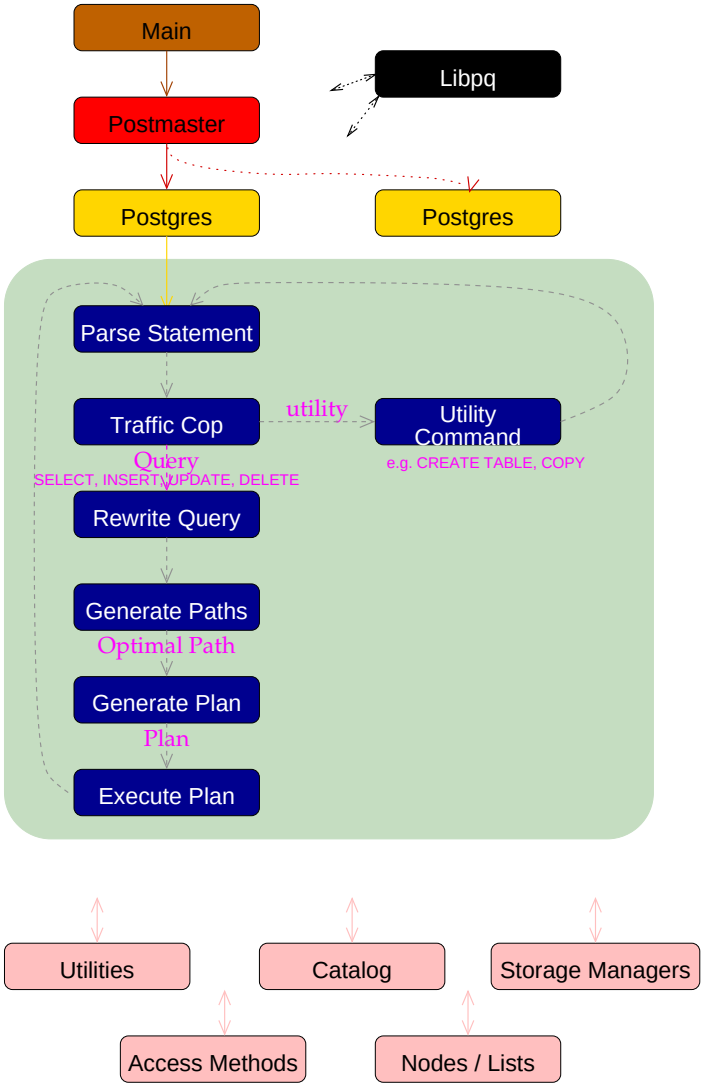
# Query Sent
# Result Received

```
FindExec: found "/var/local/postgres/./bin/postgres" using argv
DEBUG:  connection: host=[local] user=postgres database=test
DEBUG:  InitPostgres
DEBUG:  StartTransactionCommand
DEBUG:  query: SELECT firstname
               FROM friend
               WHERE age = 33;

[ query is processed ]

DEBUG:  ProcessQuery
DEBUG:  CommitTransactionCommand
DEBUG:  proc_exit(0)
DEBUG:  shmem_exit(0)
DEBUG:  exit(0)
```

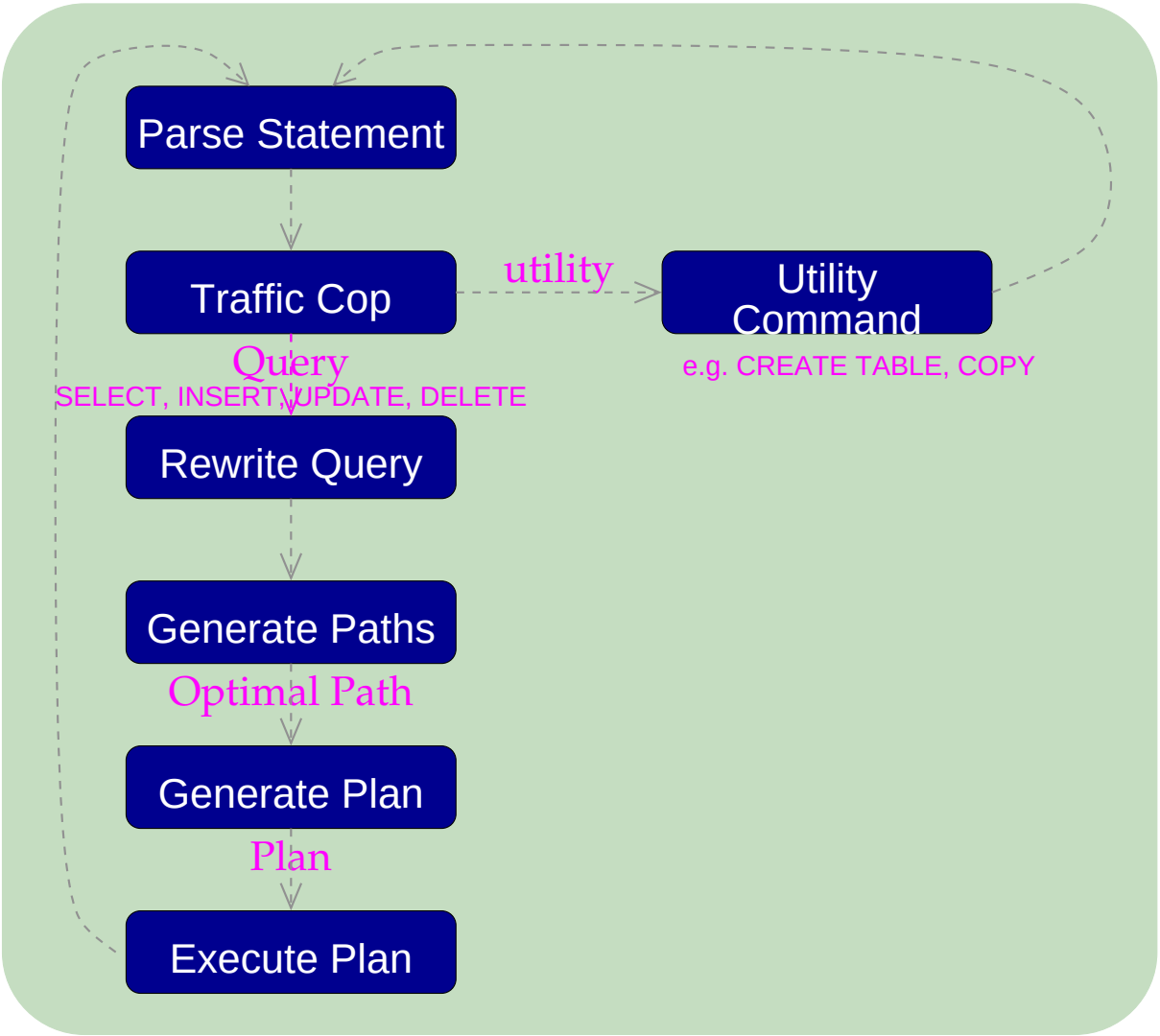
# Query Processing

```
FindExec: found "/var/local/postgres/./bin/postmaster" using argv[0]
./bin/postmaster: BackendStartup: pid 3320 user postgres db test socket 5
./bin/postmaster child[3320]: starting with (postgres -d99 -F -d99 -v131072 -p test )
FindExec: found "/var/local/postgres/./bin/postgres" using argv[0]
DEBUG:  connection: host=[local] user=postgres database=test
DEBUG:  InitPostgres
DEBUG:  StartTransactionCommand
DEBUG:  query: SELECT firstname
               FROM friend
               WHERE age = 33;
DEBUG:  parse tree: { QUERY :command 1  :utility <> :resultRelation 0 :into <> :isPortal false :isBinary false :isTemp false :h
s false :hasSubLinks false :rtable ({ RTE :relname friend :relid 26912  :subquery <> :alias <> :eref { ATTR :relname friend :at
 "firstname"  "lastname"  "city"  "state"  "age" )} :inh true :inFromCl true :checkForRead true :checkForWrite false :check
r 0}) :jointree { FROMEXPR :fromlist ({ RANGETBLREF 1 }) :quals { EXPR :typeOid 16  :opType op :oper { OPER :opno 96 :opid 0 :o
lttype 16 } :args ({ VAR :varno 1 :varattno 5 :vartype 23 :vartypmod -1  :varlevelsup 0 :varnoold 1 :varoattno 5} { CONST :cons
 23 :constlen 4 :constbyval true :constisnull false :constvalue  4 [ 33 0 0 0 ] })}} :rowMarks () :targetList ({ TARGETENTRY :r
 { RESDOM :resno 1 :restype 1042 :restypmod 19 :resname firstname :reskey 0 :reskeyop 0 :ressortgroupref 0 :resjunk false } :ex
VAR :varno 1 :varattno 1 :vartype 1042 :vartypmod 19  :varlevelsup 0 :varnoold 1 :varoattno 1}}) :groupClause <> :havingQual <>
tinctClause <> :sortClause <> :limitOffset <> :limitCount <> :setOperations <> :resultRelations ()}
DEBUG:  rewritten parse tree:
DEBUG:  { QUERY :command 1  :utility <> :resultRelation 0 :into <> :isPortal false :isBinary false :isTemp false :hasAggs false
SubLinks false :rtable ({ RTE :relname friend :relid 26912  :subquery <> :alias <> :eref { ATTR :relname friend :attrs ( "first
   "lastname"  "city"  "state"  "age" )} :inh true :inFromCl true :checkForRead true :checkForWrite false :checkAsUser 0}) :
ree { FROMEXPR :fromlist ({ RANGETBLREF 1 }) :quals { EXPR :typeOid 16  :opType op :oper { OPER :opno 96 :opid 0 :opresulttype
:args ({ VAR :varno 1 :varattno 5 :vartype 23 :vartypmod -1  :varlevelsup 0 :varnoold 1 :varoattno 5} { CONST :consttype 23 :co
n 4 :constbyval true :constisnull false :constvalue  4 [ 33 0 0 0 ] })}} :rowMarks () :targetList ({ TARGETENTRY :resdom { RESD
esno 1 :restype 1042 :restypmod 19 :resname firstname :reskey 0 :reskeyop 0 :ressortgroupref 0 :resjunk false } :expr { VAR :va
 :varattno 1 :vartype 1042 :vartypmod 19  :varlevelsup 0 :varnoold 1 :varoattno 1}}) :groupClause <> :havingQual <> :distinctCl
<> :sortClause <> :limitOffset <> :limitCount <> :setOperations <> :resultRelations ()}
DEBUG:  plan: { SEQSCAN :startup_cost 0.00 :total_cost 22.50 :rows 10 :width 12 :qptargetlist ({ TARGETENTRY :resdom { RESDOM :
 1 :restype 1042 :restypmod 19 :resname firstname :reskey 0 :reskeyop 0 :ressortgroupref 0 :resjunk false } :expr { VAR :varno
rattno 1 :vartype 1042 :vartypmod 19  :varlevelsup 0 :varnoold 1 :varoattno 1}}) :qpqual ({ EXPR :typeOid 16  :opType op :oper
R :opno 96 :opid 65 :opresulttype 16 } :args ({ VAR :varno 1 :varattno 5 :vartype 23 :vartypmod -1  :varlevelsup 0 :varnoold 1
attno 5} { CONST :consttype 23 :constlen 4 :constbyval true :constisnull false :constvalue  4 [ 33 0 0 0 ] })}) :lefttree <> :r
ree <> :extprm () :locprm () :initplan <> :nprm 0  :scanrelid 1 }
DEBUG:  ProcessQuery
DEBUG:  CommitTransactionCommand
DEBUG:  proc_exit(0)
DEBUG:  shmem_exit(0)
DEBUG:  exit(0)
./bin/postmaster: reaping dead processes...
./bin/postmaster: CleanupProc: pid 3320 exited with status 0
```

# Backend Flowchart



# Backend Flowchart - Magnified



# Statistics - Part 1

```
PARSER STATISTICS
  system usage stats:
        0.000002 elapsed 0.000000 user 0.000001 system sec
        [0.009992 user 0.049961 sys total]
        0/0 [0/1] filesystem blocks in/out
        0/0 [0/0] page faults/reclaims, 0 [0] swaps
        0 [0] signals rcvd, 0/0 [2/2] messages rcvd/sent
        0/0 [2/6] voluntary/involuntary context switches
  postgres usage stats:
        Shared blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Local  blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Direct blocks:          0 read,          0 written
PARSE ANALYSIS STATISTICS
  system usage stats:
        0.000002 elapsed 0.000001 user 0.000002 system sec
        [0.009993 user 0.049965 sys total]
        0/0 [0/1] filesystem blocks in/out
        0/0 [0/0] page faults/reclaims, 0 [0] swaps
        0 [0] signals rcvd, 0/0 [2/2] messages rcvd/sent
        0/0 [2/6] voluntary/involuntary context switches
  postgres usage stats:
        Shared blocks:          1 read,          0 written, buffer hit rate = 96.88%
        Local  blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Direct blocks:          0 read,          0 written
```

# Statistics - Part 2

```
REWRITER STATISTICS
  system usage stats:
        0.000002 elapsed 0.000000 user 0.000002 system sec
        [0.009993 user 0.049968 sys total]
        0/0 [0/1] filesystem blocks in/out
        0/0 [0/0] page faults/reclaims, 0 [0] swaps
        0 [0] signals rcvd, 0/0 [2/2] messages rcvd/sent
        0/0 [2/6] voluntary/involuntary context switches
  postgres usage stats:
        Shared blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Local  blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Direct blocks:          0 read,          0 written
PLANNER STATISTICS
  system usage stats:
        0.009974 elapsed 0.009988 user -1.999985 system sec
        [0.019982 user 0.049955 sys total]
        0/0 [0/1] filesystem blocks in/out
        0/0 [0/0] page faults/reclaims, 0 [0] swaps
        0 [0] signals rcvd, 0/0 [2/2] messages rcvd/sent
        0/0 [2/6] voluntary/involuntary context switches
  postgres usage stats:
        Shared blocks:          5 read,          0 written, buffer hit rate = 96.69%
        Local  blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Direct blocks:          0 read,          0 written
EXECUTOR STATISTICS
  system usage stats:
        0.040004 elapsed 0.039982 user 0.000013 system sec
        [0.059964 user 0.049970 sys total]
        0/0 [0/1] filesystem blocks in/out
        0/0 [0/0] page faults/reclaims, 0 [0] swaps
        0 [0] signals rcvd, 0/2 [2/4] messages rcvd/sent
        2/2 [4/8] voluntary/involuntary context switches
  postgres usage stats:
        Shared blocks:          2 read,          0 written, buffer hit rate = 83.33%
        Local  blocks:          0 read,          0 written, buffer hit rate = 0.00%
        Direct blocks:          0 read,          0 written
```

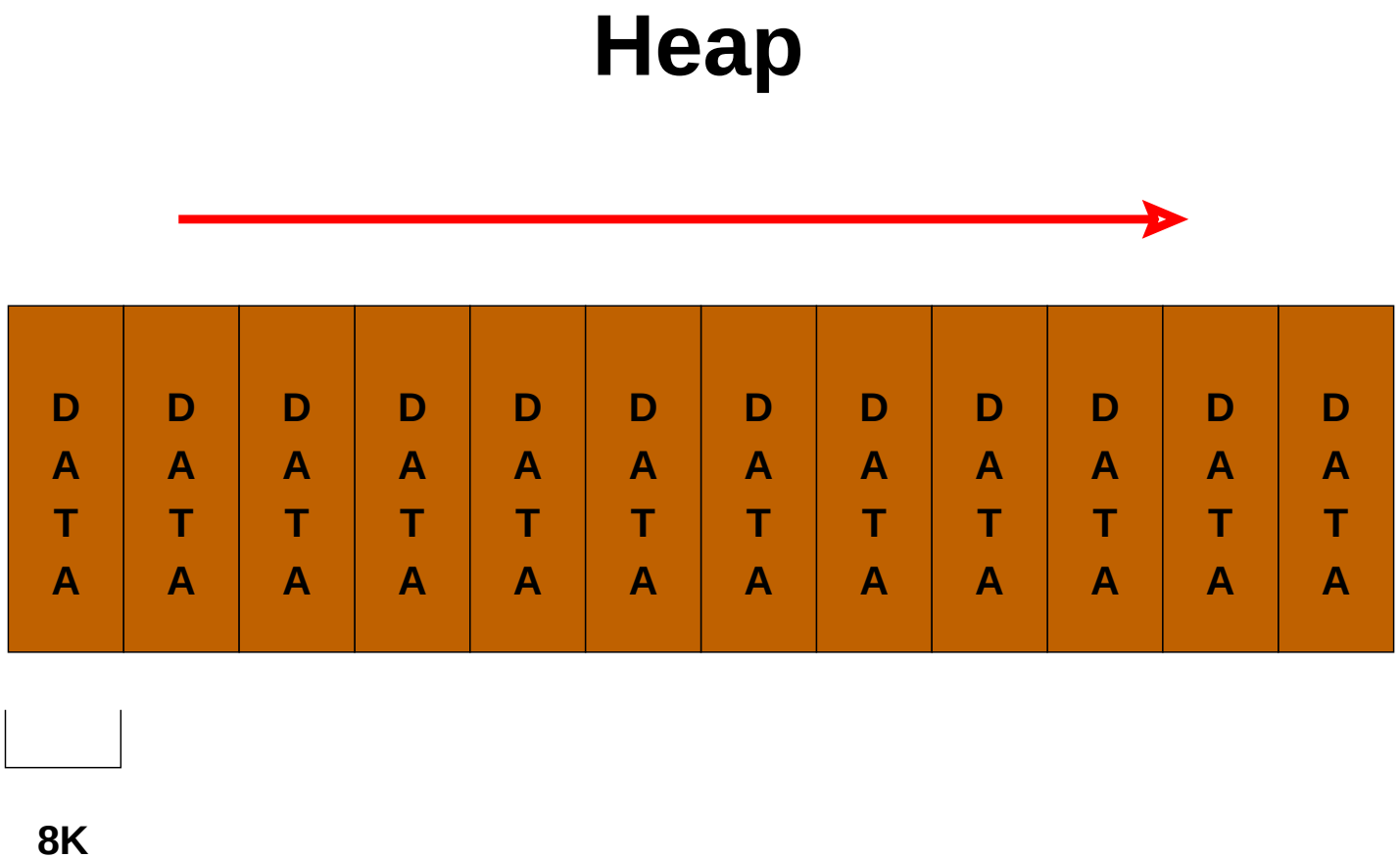
# Optimizer

- Scan Methods
- Join Methods
- Join Order

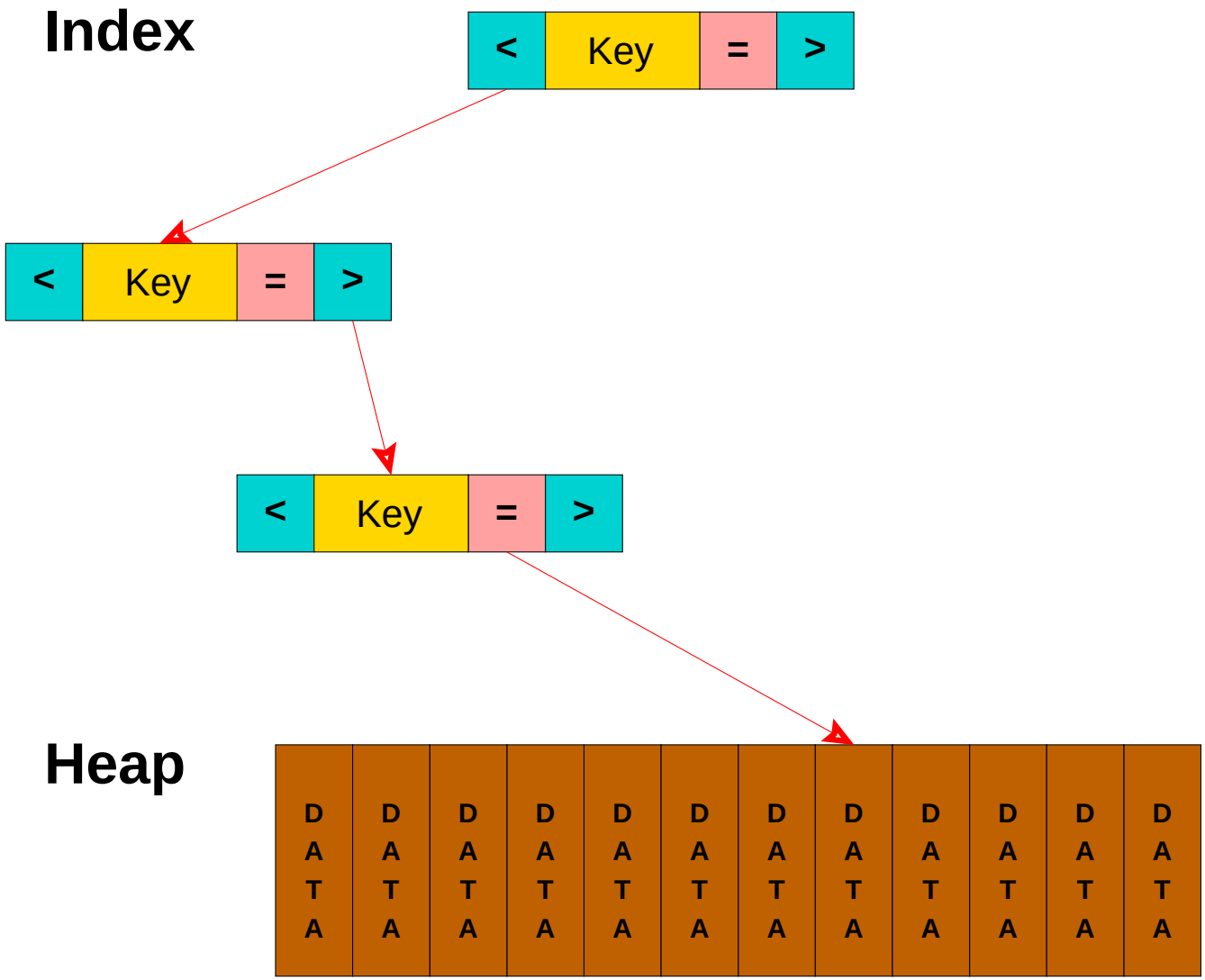
# Scan Methods

- Sequential Scan
- Index Scan
- Bitmap Scan

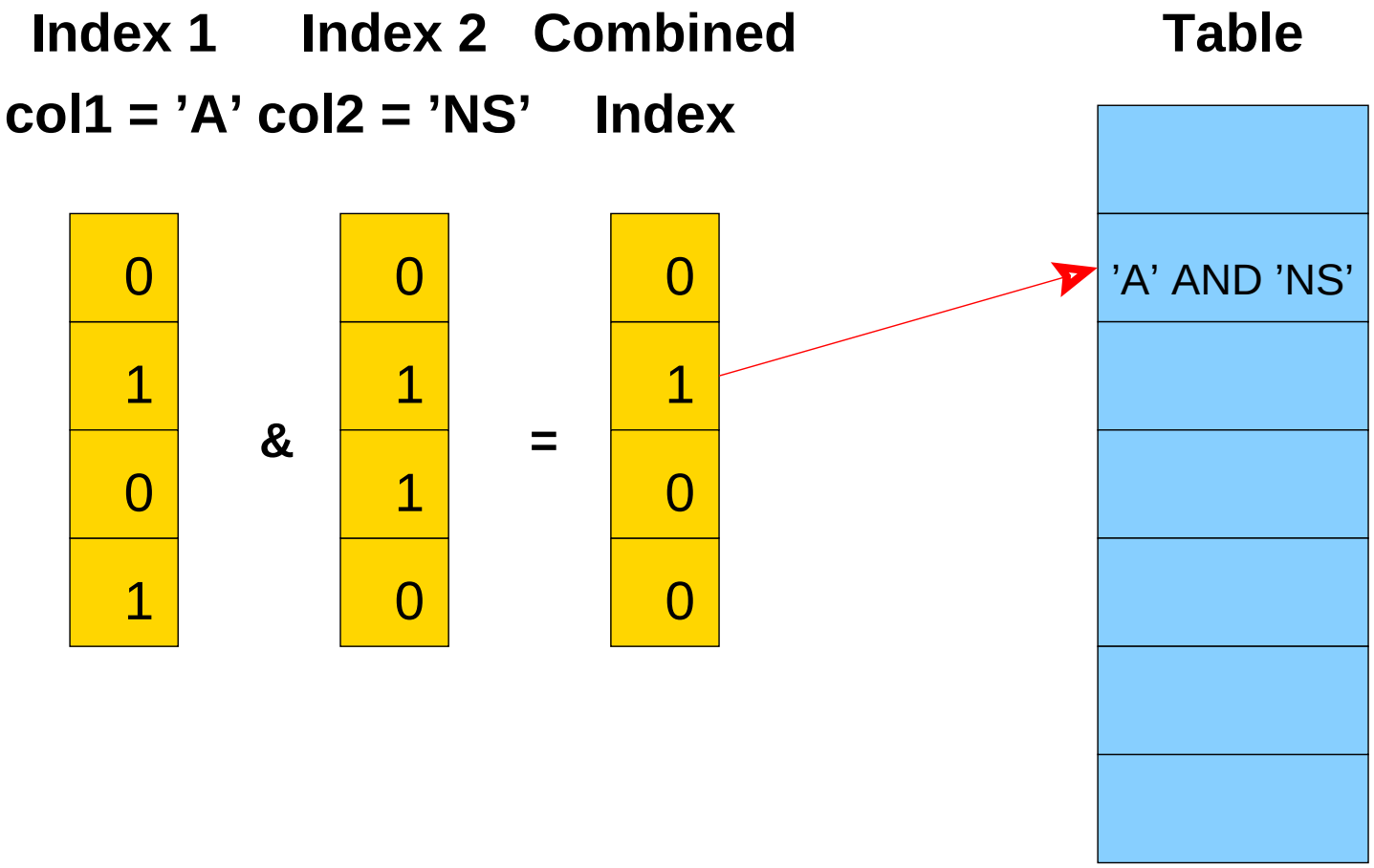
# Sequential Scan



# Btree Index Scan



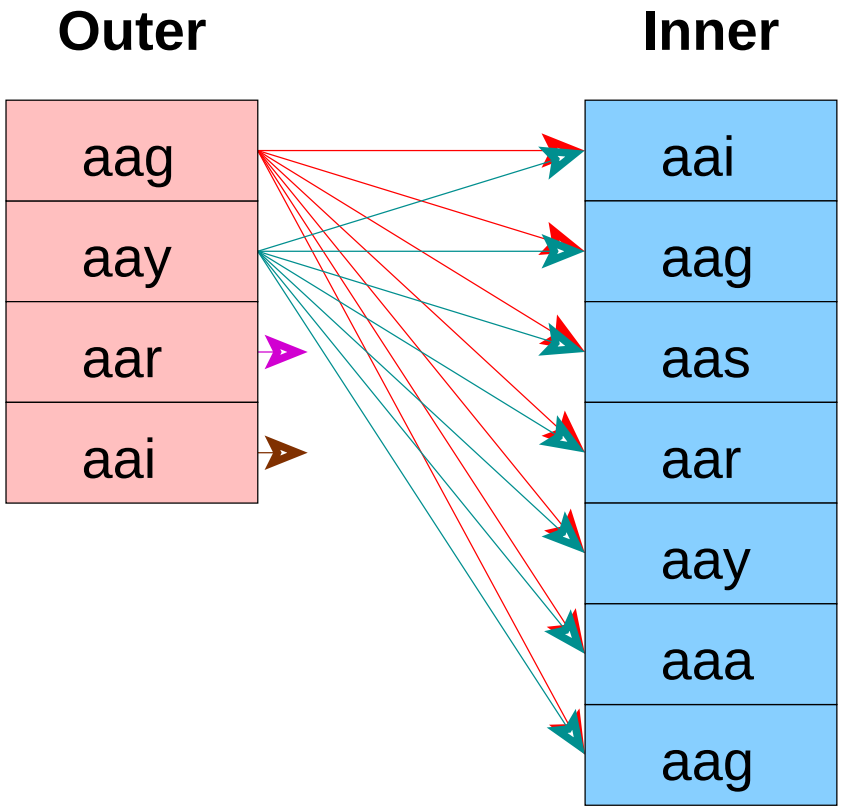
# Bitmap Scan


Index 1
col1 = 'A'
Index 2
col2 = 'NS'
Combined
Index
Table
0
1
0
1
&
0
1
1
0
=
0
1
0
0
'A' AND 'NS'

# Join Methods

- Nested Loop
  - With Inner Sequential Scan
  - With Inner Index Scan
- Hash Join
- Merge Join

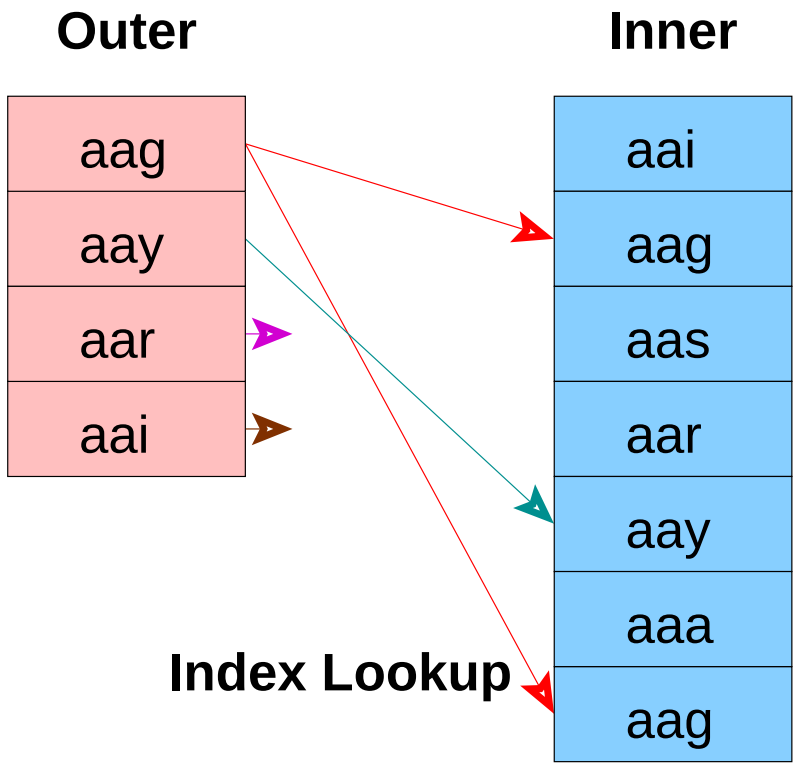
# Nested Loop Join with Inner Sequential Scan




No Setup Required

Used For Small Tables

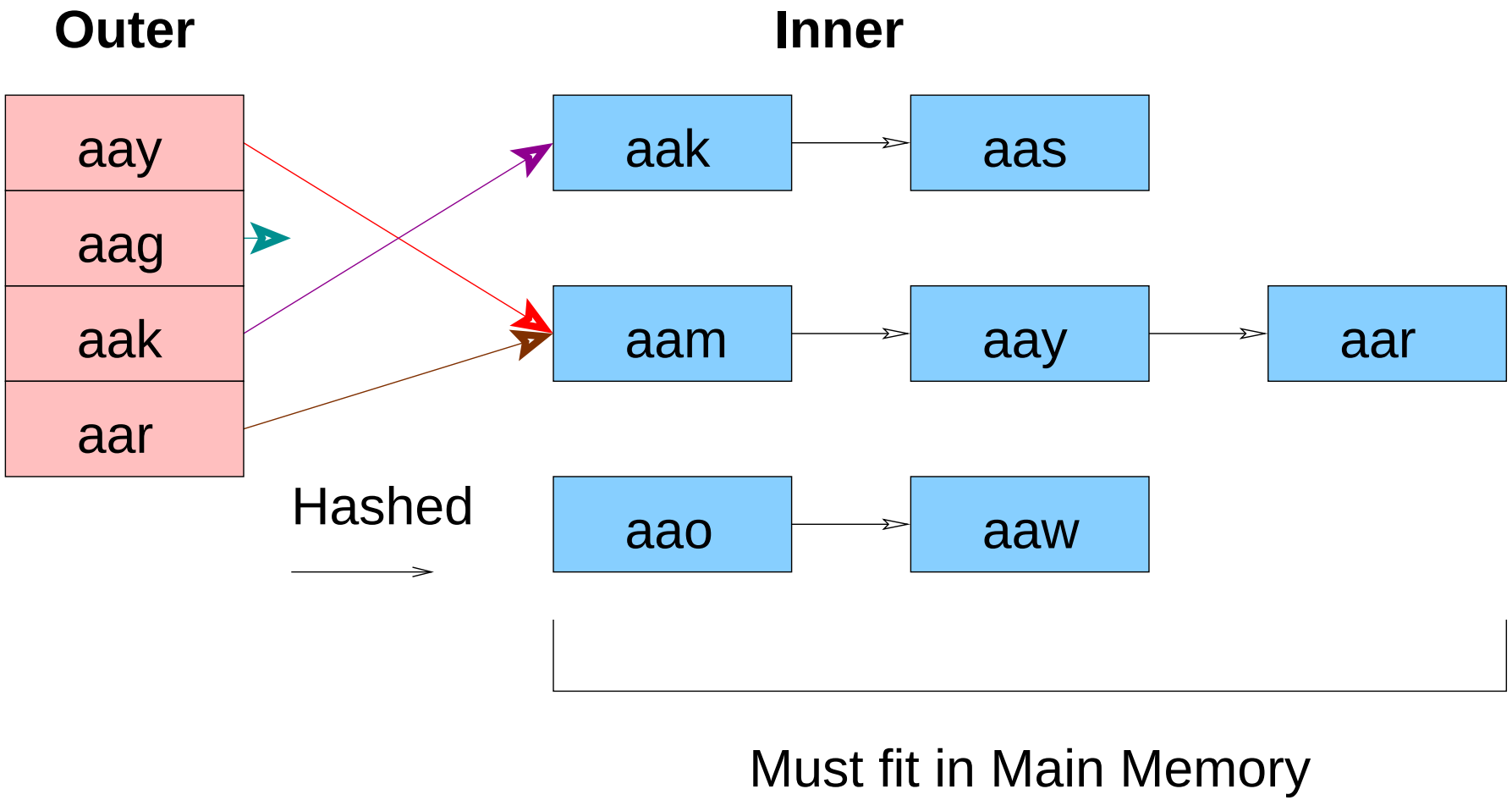
# Nested Loop Join with Inner Index Scan



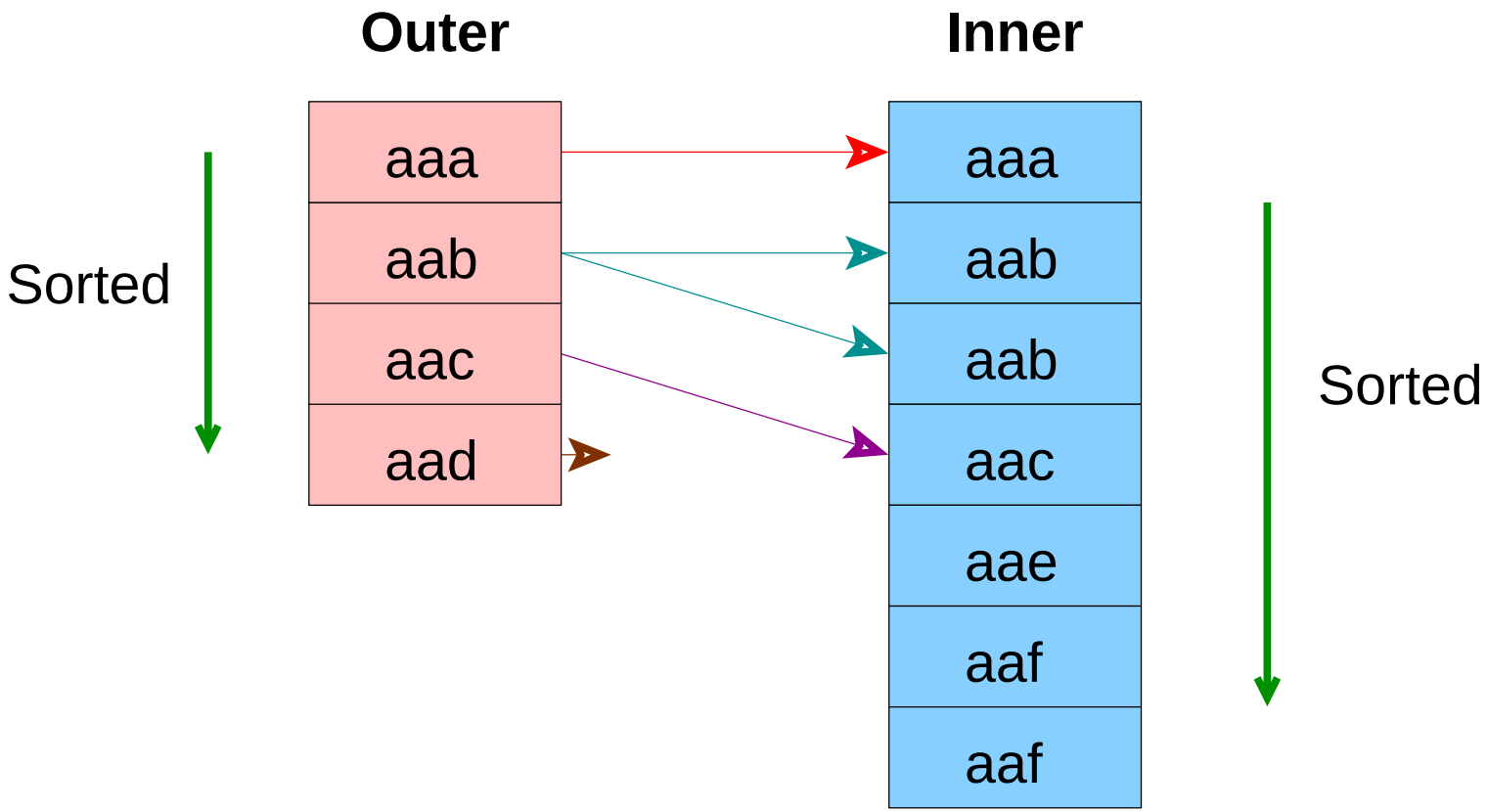

No Setup Required

Index Must Already Exist

# Hash Join


Outer
Inner
aay
aag
aak
aar
aak
aas
aam
aay
aar
aao
aaw
Hashed
Must fit in Main Memory

# Merge Join




Ideal for Large Tables

An Index Can Be Used to Eliminate the Sort

# Three-Table Join Query

```
SELECT  part.price
FROM    customer, salesorder, part
WHERE   customer.customer_id = salesorder.customer_id AND
        salesorder.part = part.part_id
```

# Three-Table Join, Pass 1, Part 1

```
(2 3 ): rows=575 width=76
        path list:
        HashJoin rows=575 cost=3.57..41.90
          clauses=(salesorder.part_id = part.part_id)
                SeqScan(2) rows=575 cost=0.00..13.75
                SeqScan(3) rows=126 cost=0.00..3.26
        Nestloop rows=575 cost=0.00..1178.70
                SeqScan(2) rows=575 cost=0.00..13.75
                IdxScan(3) rows=126 cost=0.00..2.01
        Nestloop rows=575 cost=0.00..1210.28
          pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(2) rows=575 cost=0.00..45.33
                  pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(3) rows=126 cost=0.00..2.01

        cheapest startup path:
        Nestloop rows=575 cost=0.00..1178.70
                SeqScan(2) rows=575 cost=0.00..13.75
                IdxScan(3) rows=126 cost=0.00..2.01

        cheapest total path:
        HashJoin rows=575 cost=3.57..41.90
          clauses=(salesorder.part_id = part.part_id)
                SeqScan(2) rows=575 cost=0.00..13.75
                SeqScan(3) rows=126 cost=0.00..3.26
```

# Three-Table Join, Pass 1, Part 2

```
(1 2 ): rows=575 width=76
        path list:
        HashJoin rows=575 cost=3.00..40.75
          clauses=(salesorder.customer_id = customer.customer_id)
                SeqScan(2) rows=575 cost=0.00..13.75
                SeqScan(1) rows=80 cost=0.00..2.80
        MergeJoin rows=575 cost=0.00..64.39
          clauses=(salesorder.customer_id = customer.customer_id)
                IdxScan(1) rows=80 cost=0.00..10.88
                  pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(2) rows=575 cost=0.00..45.33
                  pathkeys=((salesorder.customer_id, customer.customer_id) )

        cheapest startup path:
        MergeJoin rows=575 cost=0.00..64.39
          clauses=(salesorder.customer_id = customer.customer_id)
                IdxScan(1) rows=80 cost=0.00..10.88
                  pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(2) rows=575 cost=0.00..45.33
                  pathkeys=((salesorder.customer_id, customer.customer_id) )

        cheapest total path:
        HashJoin rows=575 cost=3.00..40.75
          clauses=(salesorder.customer_id = customer.customer_id)
                SeqScan(2) rows=575 cost=0.00..13.75
                SeqScan(1) rows=80 cost=0.00..2.80
```

# Three-Table Join, Pass 2, Part 1

```
(2 3 1 ): rows=575 width=112
        path list:
        HashJoin rows=575 cost=6.58..68.90
          clauses=(salesorder.customer_id = customer.customer_id)
                HashJoin rows=575 cost=3.57..41.90
                  clauses=(salesorder.part_id = part.part_id)
                        SeqScan(2) rows=575 cost=0.00..13.75
                        SeqScan(3) rows=126 cost=0.00..3.26
                SeqScan(1) rows=80 cost=0.00..2.80
        HashJoin rows=575 cost=3.57..92.54
          clauses=(salesorder.part_id = part.part_id)
                MergeJoin rows=575 cost=0.00..64.39
                  clauses=(salesorder.customer_id = customer.customer_id)
                        IdxScan(1) rows=80 cost=0.00..10.88
                          pathkeys=((salesorder.customer_id, customer.customer_id) )
                        IdxScan(2) rows=575 cost=0.00..45.33
                          pathkeys=((salesorder.customer_id, customer.customer_id) )
                SeqScan(3) rows=126 cost=0.00..3.26
        HashJoin rows=575 cost=3.00..1205.70
          clauses=(salesorder.customer_id = customer.customer_id)
                Nestloop rows=575 cost=0.00..1178.70
                        SeqScan(2) rows=575 cost=0.00..13.75
                        IdxScan(3) rows=126 cost=0.00..2.01
                SeqScan(1) rows=80 cost=0.00..2.80
```

# Three-Table Join, Pass 2, Part 2

```
MergeJoin rows=575 cost=0.00..1229.35
  clauses=(salesorder.customer_id = customer.customer_id)
        Nestloop rows=575 cost=0.00..1210.28
          pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(2) rows=575 cost=0.00..45.33
                  pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(3) rows=126 cost=0.00..2.01
        IdxScan(1) rows=80 cost=0.00..10.88
          pathkeys=((salesorder.customer_id, customer.customer_id) )

cheapest startup path:
MergeJoin rows=575 cost=0.00..1229.35
  clauses=(salesorder.customer_id = customer.customer_id)
        Nestloop rows=575 cost=0.00..1210.28
          pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(2) rows=575 cost=0.00..45.33
                  pathkeys=((salesorder.customer_id, customer.customer_id) )
                IdxScan(3) rows=126 cost=0.00..2.01
        IdxScan(1) rows=80 cost=0.00..10.88
          pathkeys=((salesorder.customer_id, customer.customer_id) )

cheapest total path:
HashJoin rows=575 cost=6.58..68.90
  clauses=(salesorder.customer_id = customer.customer_id)
        HashJoin rows=575 cost=3.57..41.90
          clauses=(salesorder.part_id = part.part_id)
                SeqScan(2) rows=575 cost=0.00..13.75
                SeqScan(3) rows=126 cost=0.00..3.26
        SeqScan(1) rows=80 cost=0.00..2.80
```

# Result Returned

```
test=> SELECT firstname
test-> FROM friend
test-> WHERE age = 33;

    1: firstname           (typeid = 1042, len = -1, typmod = 19, byval =
    ----
    1: firstname = "Sandy" (typeid = 1042, len = -1, typmod = 19, byval =
    ----

    firstname
-----------------
 Sandy
(1 row)
```

# Vacuum Analyze

```
test=> VACUUM ANALYZE VERBOSE customer;
INFO:  vacuuming "pg_catalog.pg_depend"
INFO:  index "pg_depend_depender_index" now contains 3616 row versions in 19 pages
DETAIL:  0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pg_depend_reference_index" now contains 3616 row versions in 23 pages
DETAIL:  0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pg_depend": found 0 removable, 3616 nonremovable row versions in 25 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/-1.99u sec elapsed 0.00 sec.
INFO:  analyzing "pg_catalog.pg_depend"
INFO:  "pg_depend": 25 pages, 3000 rows sampled, 3625 estimated total rows
VACUUM
```

```
starelid    | 16416
staattnum   | 4
stanullfrac | 0
stawidth    | 22
stadistinct | -0.4244
stakind1    | 1
stakind2    | 2
stakind3    | 3
stakind4    | 0
staop1      | 98
staop2      | 664
staop3      | 664
staop4      | 0
stanumbers1 | {0.146658,0.027904,0.0246593,0.0233615,0.0227125,0.0227125,0.0227125,0.0149254,0.01427
64,0.0123297}
stanumbers2 |
stanumbers3 | {-0.145569}
stanumbers4 |
stavalues1  | {I/O,equal,"not equal",less-than,greater-than,greater-than-or-equal,less-than-or-equal
,subtract,multiply,add}
stavalues2  | {"(Block, offset), physical location of tuple","absolute value","btree less-equal-grea
ter","convert int2 to float4","deparse an encoded expression","format int8 to text","is opclass visi
ble in search path?","matches LIKE expression","print type names of oidvector field",sine,"~18 digit
 integer, 8-byte storage"}
stavalues3  |
stavalues4  |
```

# EXPLAIN

```
test=> EXPLAIN SELECT name FROM customer;
NOTICE:  QUERY PLAN:

Seq Scan on customer (cost=0.00..225.88 rows=12288 width=34)
EXPLAIN
VACUUM
```

# Explain Analyze

```
test=> EXPLAIN ANALYZE SELECT name FROM customer;
NOTICE:  QUERY PLAN:

Seq Scan on customer (cost=0.00..225.88 rows=12288 width=34) (actual time=0.21..205.20 rows=12288 loops=1)
Total runtime: 249.10 msec
EXPLAIN
```

# Explain Using ansi Joins

```
test=> EXPLAIN INSERT INTO warehouse_tmp
test-> (uri, expression, n, relevance, spid_measure, size, title, sample)
test-> SELECT d.uri, dn.expression, n.n, dn.relevance, d.spid_measure,
test->        d.size, d.title, dn.sample
test-> FROM document as d
test->      INNER JOIN (document_n_gram AS dn
test(>         INNER JOIN n_gram AS n
test(>         ON (dn.expression = n.expression))
test->         ON (d.uri = dn.uri)
test-> ORDER BY dn.expression, n.n;
NOTICE:  QUERY PLAN:
Subquery Scan *SELECT*  (cost=3895109.07..3895109.07 rows=1009271 width=886)
  ->  Sort  (cost=3895109.07..3895109.07 rows=1009271 width=886)
        ->  Hash Join  (cost=1155071.81..2115045.12 rows=1009271 width=886)
              ->  Merge Join  (cost=1154294.92..1170599.85 rows=1009271 width=588)
                    ->  Sort  (cost=1001390.67..1001390.67 rows=1009271 width=439)
                          ->  Seq Scan on document_n_gram dn
                                      (cost=0.00..49251.71 rows=1009271 width=439)
                    ->  Sort  (cost=152904.25..152904.25 rows=466345 width=149)
                          ->  Seq Scan on n_gram n  (cost=0.00..12795.45 rows=466345 width=149)
              ->  Hash  (cost=767.71..767.71 rows=3671 width=298)
                    ->  Seq Scan on document d  (cost=0.00..767.71 rows=3671 width=298)
EXPLAIN
```

# Explain Using Subselect In FROM Clause

```
test=> EXPLAIN SELECT cs.entity_id as region, r.name, cs.status, count(*)
test-> FROM region r inner join
test->      (SELECT DISTINCT findregion(entity_id) AS entity_id, status
test(>       FROM current_status
test(>       ORDER BY 1
test(>      ) AS cs on r.region_id = cs.entity_id
test-> GROUP BY region, r.name, cs.status;
NOTICE:  QUERY PLAN:
Aggregate  (cost=13688.40..14338.40 rows=6500 width=24)
  ->  Group  (cost=13688.40..14175.90 rows=65000 width=24)
     ->  Sort  (cost=13688.40..13688.40 rows=65000 width=24)
          ->  Merge Join  (cost=7522.19..7674.94 rows=65000 width=24)
                ->  Index Scan using region_pkey on region r
                   (cost=0.00 59.00 rows=1000 width=16)
                ->  Sort  (cost=7522.19..7522.19 rows=6500 width=8)
                      ->  Subquery Scan cs  (cost=6785.54..7110.54
                                             rows=65 width=8)
                            ->  Unique  (cost=6785.54..7110.54 rows=6500
                                         with=8)
                                ->  Sort  (cost=6785.54..6785.54 rows=650
                                           width=8)
                                      ->  Seq Scan on current_status
                                          (st=0.00..1065.00 rows=65000 width=8)
EXPLAIN
```

# Postgresql.conf Optimizer Parameters

```
# - Planner Method Enabling -

#enable_hashagg = true
#enable_hashjoin = true
#enable_indexscan = true
#enable_mergejoin = true
#enable_nestloop = true
#enable_seqscan = true
#enable_sort = true
#enable_tidscan = true

# - Planner Cost Constants -

#effective_cache_size = 1000    # typically 8KB each
#random_page_cost = 4           # units are one sequential page fetch cost
#cpu_tuple_cost = 0.01          # (same)
#cpu_index_tuple_cost = 0.001   # (same)
#cpu_operator_cost = 0.0025     # (same)
```

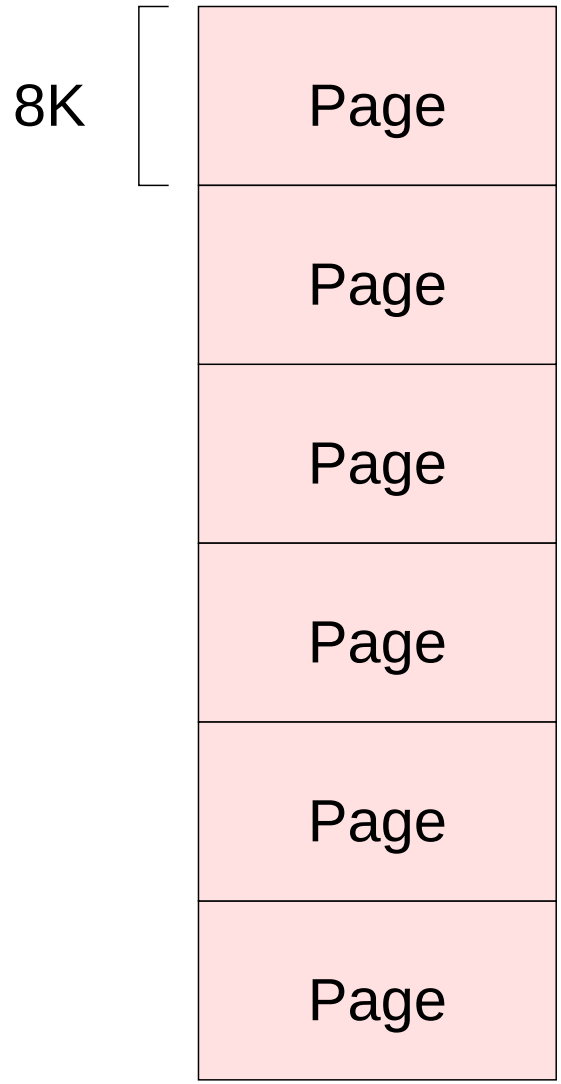
# More Postgresql.conf Optimizer Parameters

```
# - Genetic Query Optimizer -

#geqo = true
#geqo_threshold = 11
#geqo_effort = 1
#geqo_generations = 0
#geqo_pool_size = 0             # default based on tables in statement,
                                # range 128-1024
#geqo_selection_bias = 2.0      # range 1.5-2.0

# - Other Planner Options -

#default_statistics_target = 10 # range 1-1000
#from_collapse_limit = 8
#join_collapse_limit = 8        # 1 disables collapsing of explicit JOINs
```

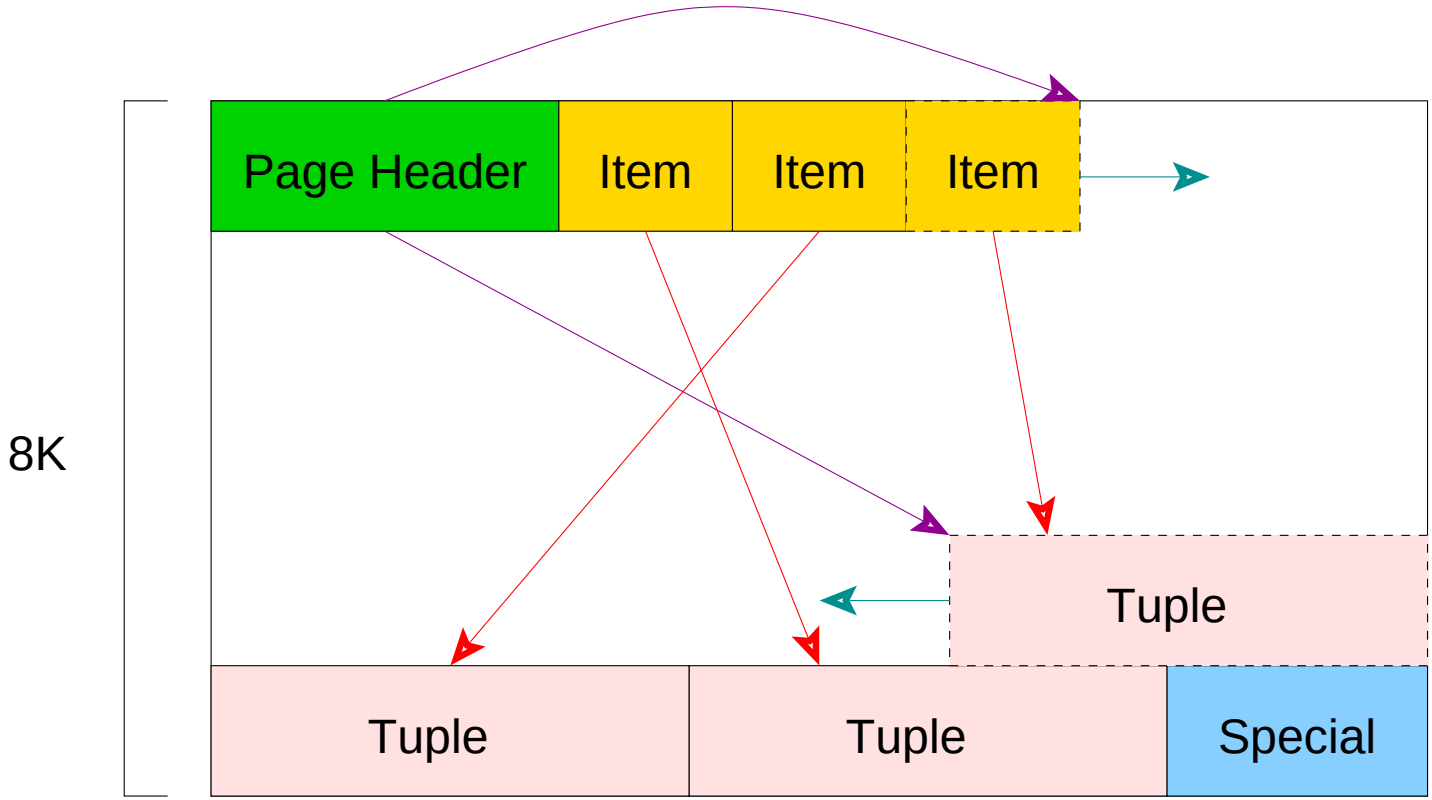
# Storage


https://www.flickr.com/photos/mirandala/

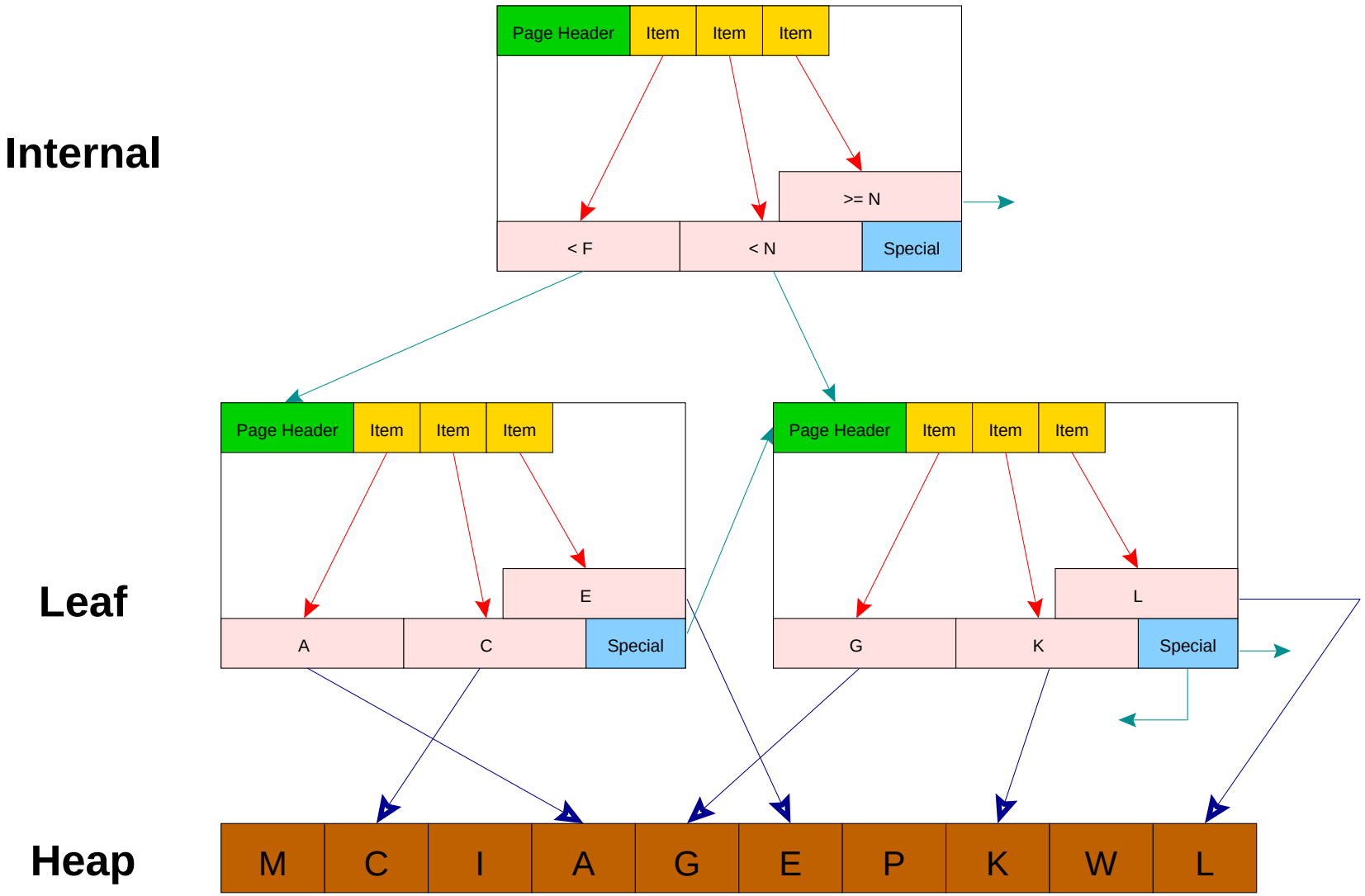
# File Structure



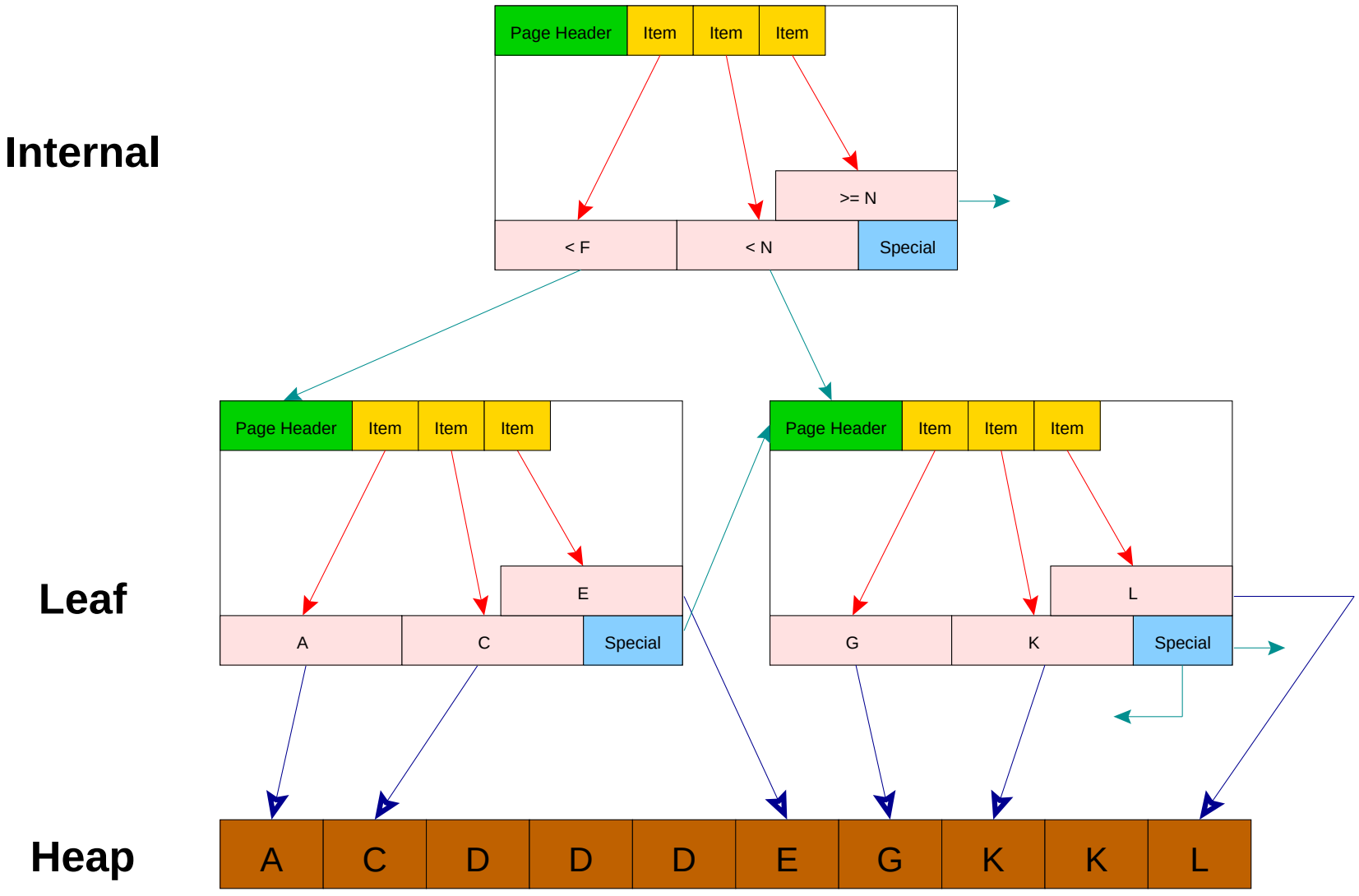
# Page Structure



# Index Page Structure



# Cluster


Internal
Leaf
Heap
Page Header
Item
Item
Item
>= N
< F
< N
Special
Page Header
Item
Item
Item
E
A
C
Special
Page Header
Item
Item
Item
L
G
K
Special
A
C
D
D
D
E
G
K
K
L

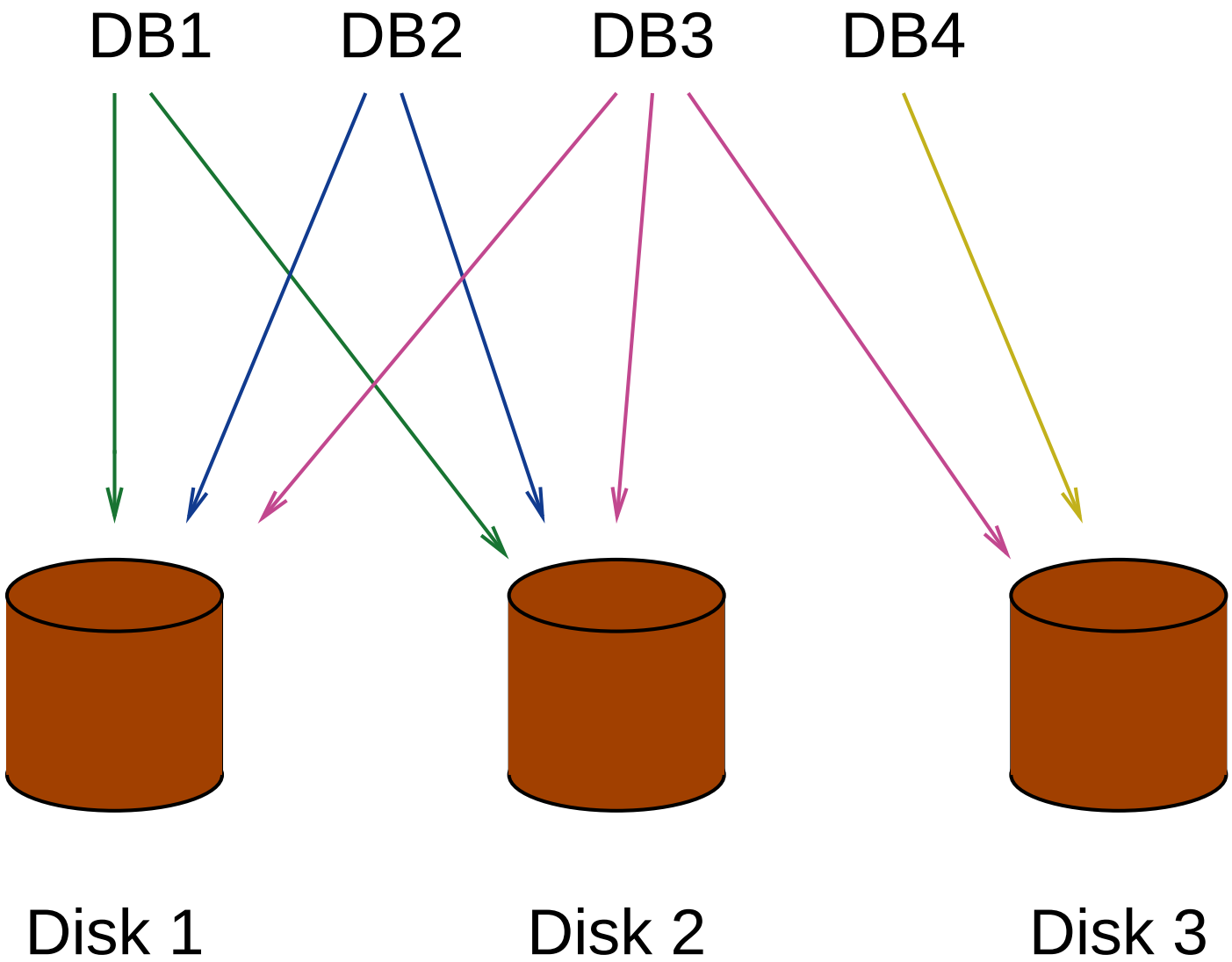
# Cluster

```
test=> CREATE TABLE customer (id SERIAL, name TEXT);
NOTICE: CREATE TABLE will create implicit sequence 'customer_id_seq' for SERIAL column 'customer.id'
test=> CREATE INDEX customer_id_index ON customer (id);
CREATE INDEX
test=> CLUSTER customer USING customer_id_index;
CLUSTER
```

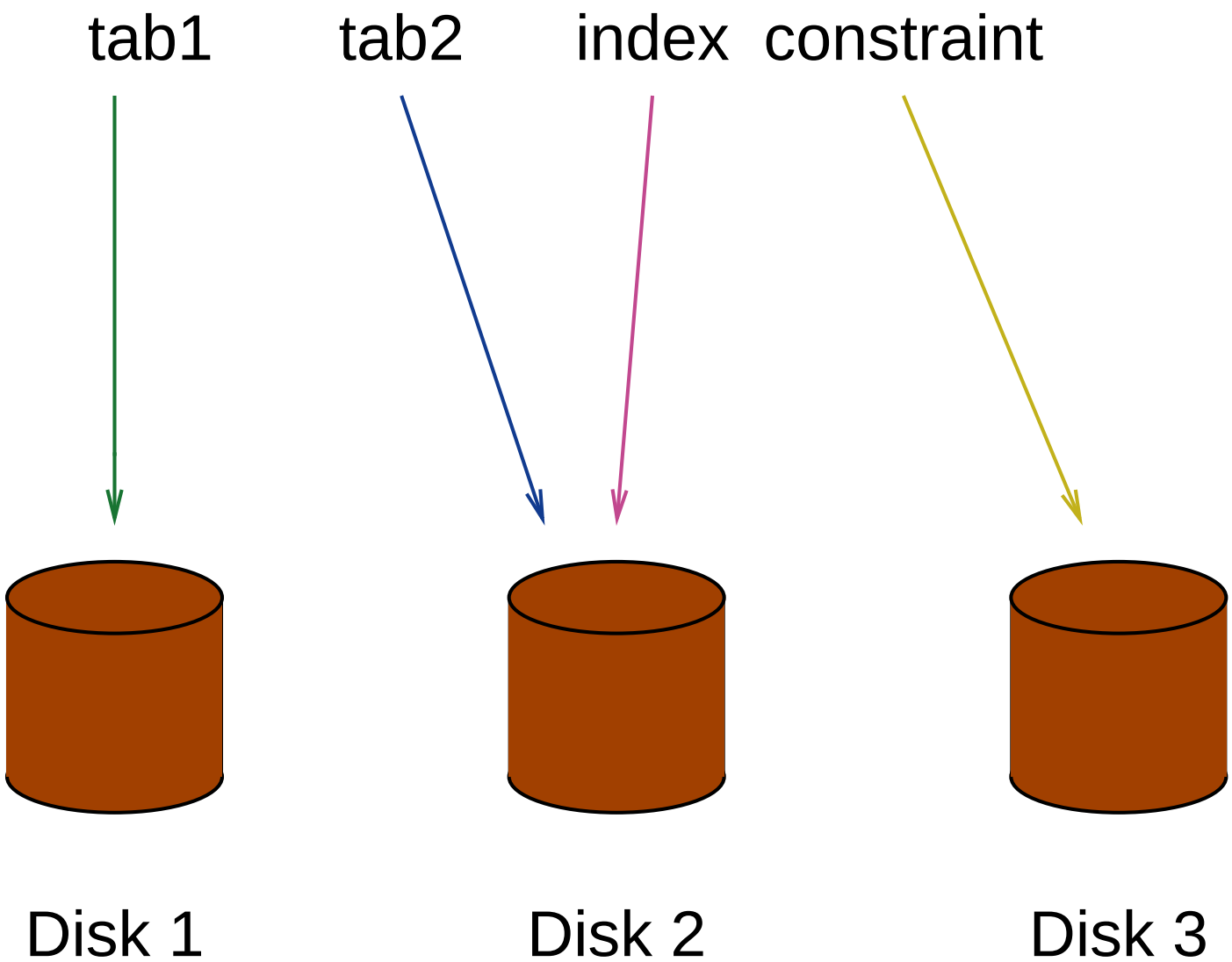
# Index Types (Access Methods)

- Btree
- Hash
- Rtree
- GiST
- GIN

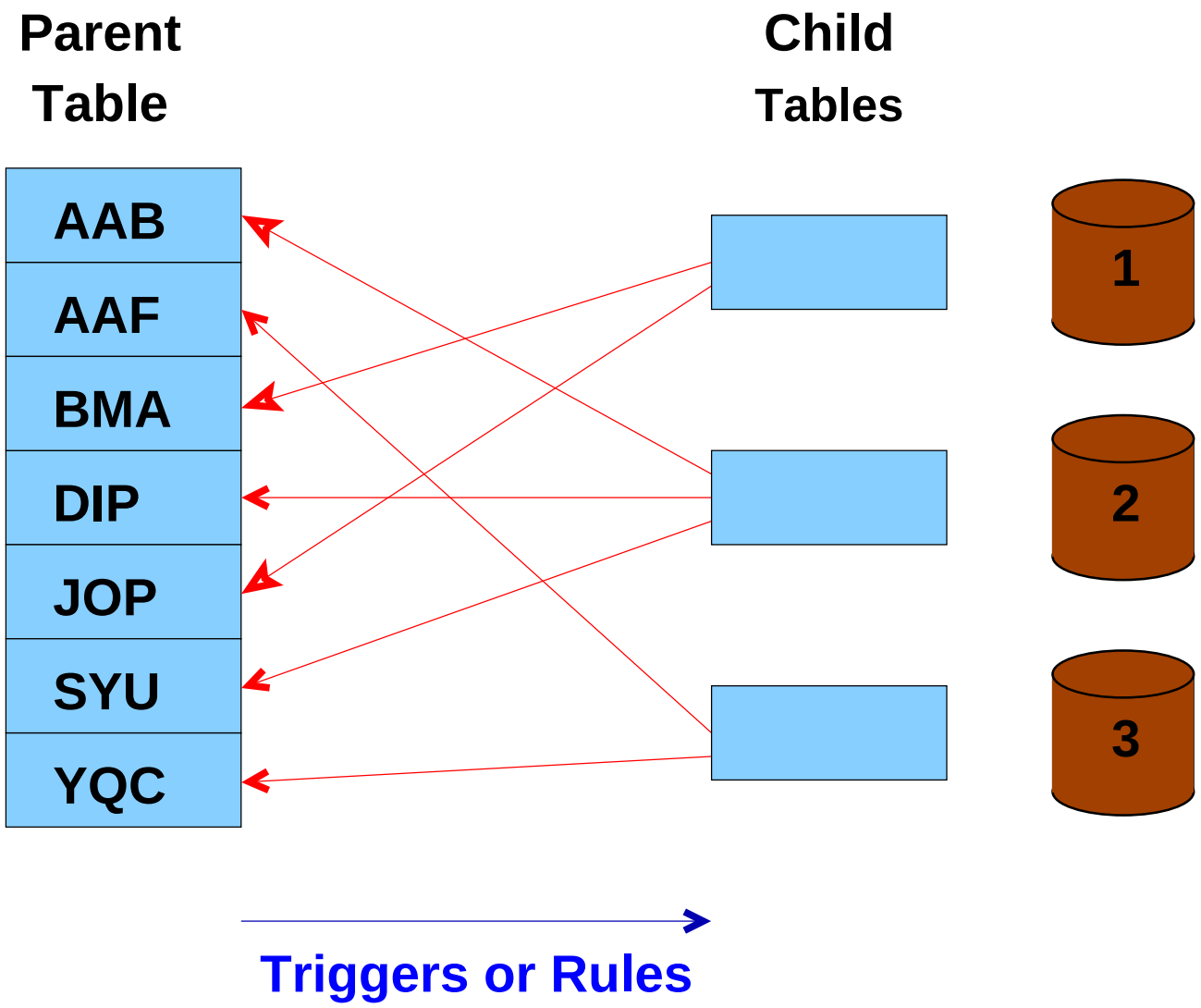
# Tablespaces For Database I/O Balancing



# Tablespaces For Table and Index I/O Balancing



# Table I/O Balancing Using constraint_exclusion




Range partitioning is also possible.

# Caches

- System Cache
- Relation Information Cache
- File Descriptor Cache

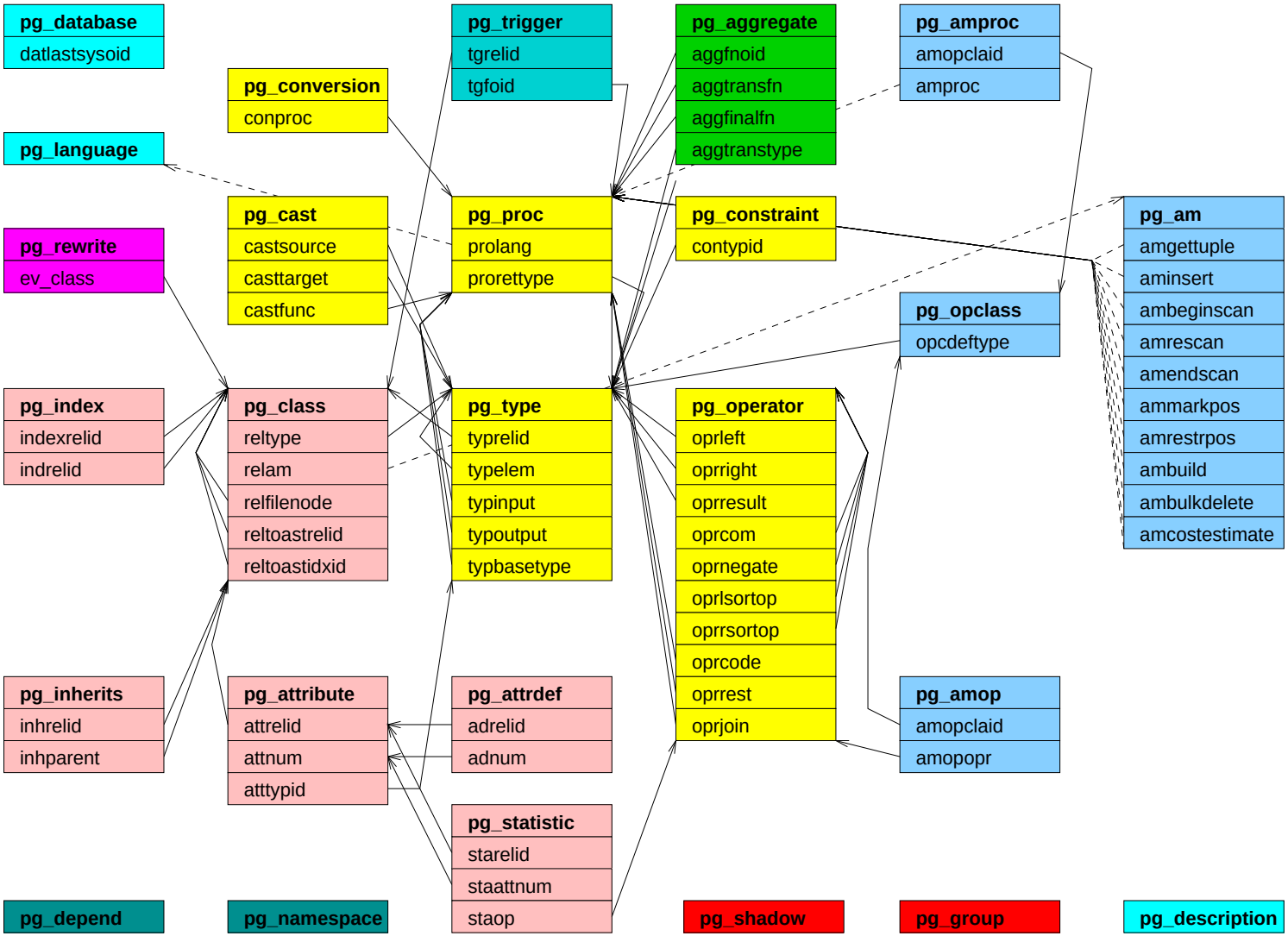
# Shared Memory

- Proc structure
- Lock structure
- Buffer structure
- Free space map

# Query Tips

- COPY vs. INSERT
- LIMIT vs. CURSOR
- TRUNCATE vs. DELETE
- Expression Indexes
- Partial Indexes
- Prepared Queries
- INTERSECT vs. AND (selfjoin)
- UNION vs. OR

# System Tables



# Conclusion

*http://momjian.us/presentations*

*https://www.flickr.com/photos/143948408@N03/*