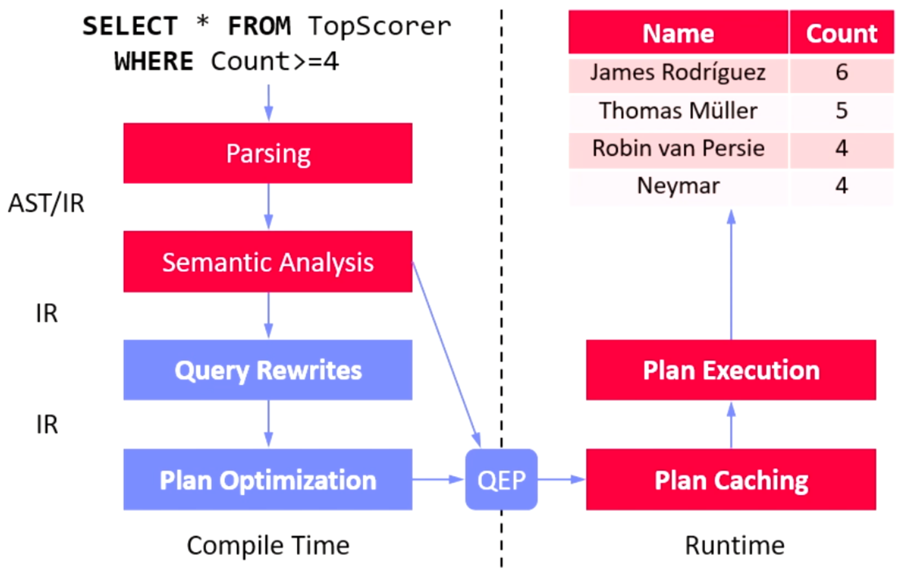
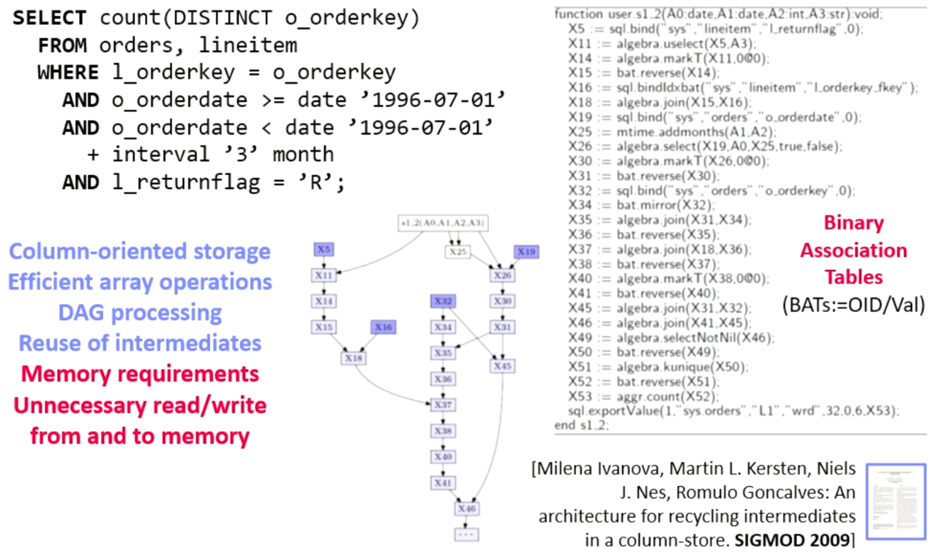
Overview

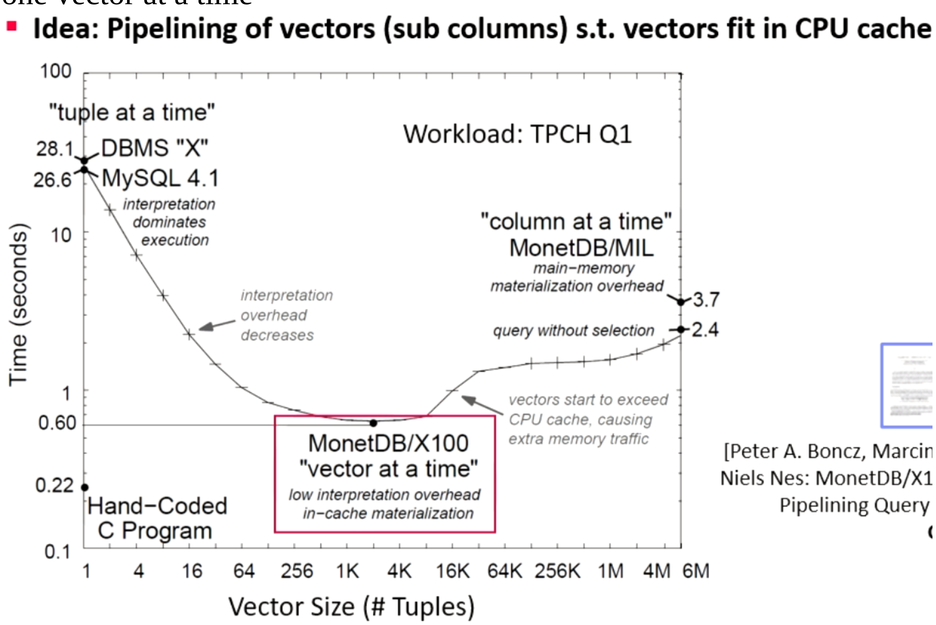
- query execution consists of four steps
  - parsing
  - semantic analysis
    - do all tables/tuples exist
    - checks user permissions
  - [[Query Rewriting]]
  - [[Plan Optimization]]
- query execution plan
  - semantic analysis creates QEP
  - plan optimization creates optimized QEP
  - can be executed by runtime
- runtime may store results in cache use again later
- 

Overview Execution Strategies

- different strategies with different pros and cons
- (Volcano) iterator model
  - see [[Physical Operators]]
- materialized intermediates
  - one column at a time
  - uses binary association tables (BATs)

- 

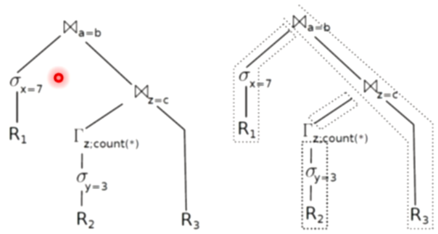
- vectorized (batched) execution
  - one vector at a time



- query compilation
  - no longer operator centric ==> data centric
  - blurred boundaries between operators

- **Idea: Data-centric, not op-centric processing + LLVM code generation**

Operator Trees
(w/o and w/ pipeline boundaries)



Compiled Query
(conceptual, not LLVM)

initialize memory of $\bowtie_{a=b}$, $\bowtie_{c=z}$, and Γ_z
for each tuple t in R_1
  if $t.x = 7$
    materialize t in hash table of $\bowtie_{a=b}$
for each tuple t in R_2
  if $t.y = 3$
    aggregate t in hash table of Γ_z
~~**for each** tuple t in Γ_z~~
  ~~materialize t in hash table of $\bowtie_{z=c}$~~
for each tuple t_3 in R_3
  for each match t_2 in $\bowtie_{z=c}[t_3.c]$
    for each match t_1 in $\bowtie_{a=b}[t_3.b]$
      output $t_1 \circ t_2 \circ t_3$

–  [Thomas Neumann: Efficiently Compiling Efficient Query Plans for Modern Hardware, **PVLDB 2011**]