class 4

data layouts & column stores

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
from last time

database kernel

parser
optimizer
execution
storage

in/out
thread pool
transactions
buffer pool

cpu
memory
disk

applications

sql

applications

sql

from last time

database kernel

parser
optimizer
execution
storage

in/out
thread pool
transactions
buffer pool

cpu
memory
disk

applications

sql

from last time

database kernel

parser
optimizer
execution
storage

in/out
thread pool
transactions
buffer pool

cpu
memory
disk

applications

sql
can DBAs make wrong decisions?
can optimizers make wrong decisions?
```
select student.name
from students, enrolled, courses
where courses.name="cs165"
and enrolled.courseId=course.id
and student.id=enrolled.studentId
```
SQL

logical plan

optimizer

rules/cost model/statistics

physical plan

execution
suggestion:

install MonetDB & PostgreSQL and play with SQL

explain + SQL query to see query plans in MonetDB
(use read-only mode)

repeat throughout the semester
compare with your system
how to read research papers

1) abstract-intro-related work-conclusions
what is the problem
why is it important
why past solutions do not work
what is the core idea

2) core part-analysis
basic idea
what matters
any gaps?

3) follow a few citations and repeat

goal: by the end of the semester understand these papers fully
memory hierarchy

data layouts

column-stores basics
system where db runs

applications

sql

cpu - cpu - cpu - cpu

cpu registers

caches

memory

disk - disk - disk - disk

smaller/faster

+ flash

+ non volatile memory

memory hierarchy
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Innovations award

- 100Kx disk
- 100x memory
- 10x on board cache
- 2x on chip cache
- registers

Pluto
- 2 years

New York
- 1.5 hours

this building
- 10 min

this room
- 1 min

my head
- ~0
Registers
On-chip cache
On-board cache
Memory
Disk

CPU

Cache miss: looking for something which is not in the cache

Memory wall

Memory miss: looking for something which is not in memory

SRAM

DRAM

Approximate times:
- SRAM: \(~1\text{ns}\)
- DRAM: \(~10\text{ns}\)
- Memory: \(~100\text{ns}\)

Speed vs. time
CPU vs. Memory

Faster
Cheaper
and touch/access only what you need

design of storage/access methods/algorithms should minimize:

- data misses
- instruction misses
random access & page-based access

data value x

need to only read x... but have to read all of page 1

page1  page2  page3  ...

data move

CPU
registers
on chip cache
on board cache
memory
disk
query \( x < 5 \)

(size=120 bytes)
memory level N

memory level N-1

\[ \begin{align*}
&5 1 0 6 4 12 \quad 2 8 9 7 6 \quad 7 1 1 3 9 6 \quad \ldots \\
\end{align*} \]

page size: 5x8 bytes
query $x < 5$

scan

(size=120 bytes)
memory level N

memory level N-1

page size: 5x8 bytes
query $x < 5$

(size=120 bytes)

memory level $N$

5 10 6 4 12

scan

4

memory level $N-1$

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

page size: 5x8 bytes
query \ x<5

(scan)

5 10 6 4 12

4

(size=120 bytes)
memory level N

memory level N-1

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

...
**query** \( x < 5 \)

(memory level N)

```
5 10 6 4 12
```

(scan)

```
2 8 9 7 6
```

40 bytes

(memory level N-1)

```
5 10 6 4 12
```

```
2 8 9 7 6
```

```
7 11 3 9 6
```

(page size: 5x8 bytes)
query $x < 5$

memory level $N$

(size=120 bytes)

5 10 6 4 12

scan

2 8 9 7 6

scan

4 2

memory level $N-1$

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

... page size: 5x8 bytes
query $x < 5$

(memory level $N$)

(size = 120 bytes)

scan

5 10 6 4 12

scan

2 8 9 7 6

4 2

(memory level $N-1$)

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

... page size: 5x8 bytes
query \( x < 5 \)

(memory level \( N \))

(size=120 bytes)

(memory level \( N-1 \))

page size: 5x8 bytes
query \( x < 5 \)

(scan)

(size=120 bytes)
memory level \( N \)

\[
\begin{array}{cccc}
7 & 11 & 3 & 9 \\
\end{array}
\quad
\begin{array}{cccc}
2 & 8 & 9 & 7 \\
6 & 4 & 12 & 2
\end{array}
\quad
\begin{array}{c}
4 & 2
\end{array}
\]

memory level \( N-1 \)

\[
\begin{array}{cccc}
5 & 10 & 6 & 4 \\
12 & 2 & 8 & 9 \\
7 & 11 & 3 & 9 \\
6 & 7 & 11 & 3 & 9 & 6 & \ldots
\end{array}
\]

page size: 5x8 bytes
query $x<5$

Scan

(memory level $N$)

(size=120 bytes)

Page size: 5x8 bytes

(memory level $N-1$)
query $x < 5$

(size=120 bytes)

memory level N

scan

7 11 3 9 6

2 8 9 7 6

4 2 3

memory level N-1

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

... page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(size=120 bytes)
memory level N

memory level N-1

page size: 5x8 bytes
an oracle gives us the positions

query \( x < 5 \)

memory level N

(memory level N)

(oracle)

(size=120 bytes)

memory level N-1

(page size: 5x8 bytes)
an oracle gives us the positions

query $x < 5$

(size=120 bytes)
memory level N

oracle

5 10 6 4 12

memory level N-1

5 10 6 4 12 2 8 9 7 6 7 11 3 9 6 ...

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(memory level N)

\[\begin{array}{ccccccc}
5 & 10 & 6 & 4 & 12 & \quad\text{oracle}\quad & 4 \\
\end{array}\]

(size=120 bytes)

(memory level N-1)

\[\begin{array}{ccccccc}
5 & 10 & 6 & 4 & 12 & 2 & 8 & 9 & 7 & 6 & 7 & 11 & 3 & 9 & 6 & \ldots \\
\end{array}\]

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(size = 120 bytes)

memory level N

 oracle

 5 10 6 4 12

 oracle

 2 8 9 7 6

 4

memory level N-1

 5 10 6 4 12

 2 8 9 7 6

 7 11 3 9 6

... page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(size=120 bytes)
memory level $N$

memory level $N-1$

page size: 5x8 bytes
an oracle gives us the positions

\[ \text{query } x < 5 \]

(size=120 bytes)

memory level N

\[
\begin{array}{c}
5 & 10 & 6 & 4 & 12 \\
\end{array}
\]

oracle

\[
\begin{array}{c}
2 & 8 & 9 & 7 & 6 \\
\end{array}
\]

oracle

\[
\begin{array}{c}
4 & 2 \\
\end{array}
\]

memory level N-1

\[
\begin{array}{c}
5 & 10 & 6 & 4 & 12 \\
\end{array}
\]

\[
\begin{array}{c}
2 & 8 & 9 & 7 & 6 \\
\end{array}
\]

\[
\begin{array}{c}
7 & 11 & 3 & 9 & 6 \\
\end{array}
\]

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

(memory level N)

(size = 120 bytes)

page size: 5x8 bytes

(memory level N-1)
an oracle gives us the positions

query $x<5$

oracle

(size=120 bytes)
memory level N

7 11 3 9 6 2 8 9 7 6 4 2

memory level N-1

5 10 6 4 12 2 8 9 7 6 7 11 3 9 6 ...

page size: 5x8 bytes
an oracle gives us the positions

query $x < 5$

oracles

(size=120 bytes)
memory level N

<table>
<thead>
<tr>
<th>4 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 11 3 9 6</td>
</tr>
<tr>
<td>2 8 9 7 6</td>
</tr>
</tbody>
</table>

memory level N-1

| 7 11 3 9 6 |
| 5 10 6 4 12 |
| 2 8 9 7 6 |
| ... |

page size: 5x8 bytes
an oracle gives us the positions

query \( x < 5 \)

(size=120 bytes)

memory level N

oracle

7 11 3 9 6
2 8 9 7 6
4 2 3

memory level N-1

5 10 6 4 12
2 8 9 7 6
7 11 3 9 6
...

page size: 5x8 bytes
when does it make sense to have an oracle
**sequential access:**
read one block; consume it completely; discard it; read next

in parallel/prefetching

what is next?

1 2 3 4

hardware can better predict/buffer sequential pages to be read
e.g., 2MB buffers in modern DRAM
amortize cost of moving disk arms
random access:
read one block; consume it partially; discard it; might have to read it again in future; read “random” next;
device block size

os block size

dbms block size

os and db will typically refer to **pages**
level N

buffer pool

level N-1

remember hot blocks

why use not use OS caching
employee
(id:int, name:varchar(50), office:char(5), telephone:char(10), city:varchar(30), salary:int)

(1, name1, office1, tel1, city1, salary1)
(2, name2, office2, tel2, city2, salary2)
(3, name3, office3, tel3, city3, salary3)
(4, name4, office4, tel4, city4, salary4)
(5, name5, office5, tel5, city5, salary5)
(6, name6, office6, tel6, city6, salary6)
(7, name7, office7, tel7, city7, salary7)
(8, name8, office8, tel8, city8, salary8)
(9, name9, office9, tel9, city9, salary9)
...

data storage
blocks < pages < files

remember: the way we store data defines the best possible way we can access it
employee
(id:int, name:varchar(50), office:char(5), telephone:char(10), city:varchar(30), salary:int)

(1, name1, office1, tel1, city1, salary1)
(2, name2, office2, tel2, city2, salary2)
(3, name3, office3, tel3, city3, salary3)
(4, name4, office4, tel4, city4, salary4)
(5, name5, office5, tel5, city5, salary5)
(6, name6, office6, tel6, city6, salary6)
(7, name7, office7, tel7, city7, salary7)
(8, name8, office8, tel8, city8, salary8)
(9, name9, office9, tel9, city9, salary9)

…
slotted page

free_offset, N, offset1-length1, offset2-length2,…

free space

scan
null
update
var length
…
some things to “worry” about
how much data we transfer through the memory hierarchy
how many computations we do
row-store

one page contains all fields of multiple attributes

stored continuously

select A, B, C, D

select A

file
row-store

A B C D

stored continuously

column-store

A B C D

one page contains fields of a single attribute

select A, B, C, D

select A
1960s

- Rows
- Rows
- Rows
- Rows

1970: Column storage ideas start appearing

1985: First rather complete column-store model

1970:

~2000: Open source complete system

~2000:

2005-now: More ideas and industry adoption of column-store designs

2005-now:

- Monetdb
- C-store, Vertica, Vectorwise
- And then IBM, Microsoft, Oracle, and more
R(A,B,C)

column-store with materialized IDs

ID A
ID B
ID C

good idea
virtual ids/ positional alignment

columns do not need to have the same width

tuple 1  →  a1  →  b1  →  c1
tuple 2  →  a2  →  b2  →  c2
tuple 3  →  a3  →  b3  →  c3
tuple 4  →  a4  →  b4  →  c4
tuple 5  →  a5  →  b5  →  c5
tuple 6  →  a6  →  b6  →  c6

fixed-width
+
dense

positional lookups/joins

\[ A(i) = A + i \times \text{width}(A) \]
ok so now we can selectively read columns but how do we process them?

disk

| A | B | C | D |

memory

| A |

option 1

row-store engine

early tuple reconstruction/materialization

option 2
registers on chip cache on board cache memory DRAM ~100ns memory wall SRAM ~10ns ~1ns
chip faster cheaper

it is not just memory and disk we want to move as few data items as possible all the way up to the CPU
select min(C) from R where A<10 & B<20

disk

A  B  C  D

write the query plan and the code/logic of each operator
do not forget about intermediate results
describe data layouts at each step

(memory1 of project)
late reconstruction/materialization

\textbf{select} \ \textbf{min(C)} \ \textbf{from} \ R \ \textbf{where} \ A<10 \ & \ B<20
late reconstruction/materialization

\textbf{select } \text{min}(C) \text{ from } R \text{ where } A<10 \text{ & } B<20
late reconstruction/materialization

$$\text{select min}(C) \text{ from } R \text{ where } A<10 \& B<20$$

```
1: int *input=A
2: for (i=0;i<tuples;i++,input++)
3:    if *input<10
4:       *output=i
5:       output++
```
late reconstruction/materialization

```
select min(C) from R where A<10 & B<20
```
late reconstruction/materialization

\textbf{select} \text{\textit{min}}(C) \textbf{from} R \textbf{where} A<10 \& B<20
late reconstruction/materialization

```sql
select min(C) from R where A<10 & B<20
```
late reconstruction/materialization

\[
\text{select } \min(C) \text{ from } R \text{ where } A<10 \& B<20
\]
late reconstruction/materialization

\[
\text{select } \min(C) \text{ from } R \text{ where } A<10 \land B<20
\]
late reconstruction/materialization

\[
\text{select } \min(C) \text{ from } R \text{ where } A < 10 \& B < 20
\]

always sequential access patterns
memory contains only what is needed at any point in time
### Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

### Query and Query Plan (MAL Algebra)

```sql
select sum(R.a) from R, S where R.c = S.b and 5 < R.a < 20 and 40 < R.b < 50 and 30 < S.a < 40
```

1. inter1 = \text{select}(Ra,5,20)
2. inter2 = \text{reconstruct}(Rb,inter1)
3. inter3 = \text{select}(inter2,40,50)
4. join_input_R = \text{reconstruct}(Rc,inter3)
5. inter4 = \text{select}(Sa,50,65)
6. inter5 = \text{reconstruct}(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,55,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
**Initial Status**

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

**Query and Query Plan (MAL Algebra)**

select sum(R.a) from R, S where R.c = S.b and 5 < R.a < 20 and 40 < R.b < 50 and 30 < S.a < 40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
**Initial Status**

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

**Query and Query Plan (MAL Algebra)**

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. \( \text{inter1} = \text{select}(\text{Ra},5,20) \)
2. \( \text{inter2} = \text{reconstruct}(\text{Rb},\text{inter1}) \)
3. \( \text{inter3} = \text{select}(\text{inter2},40,50) \)
4. \( \text{join_input_R} = \text{reconstruct}(\text{Rc},\text{inter3}) \)
5. \( \text{inter4} = \text{select}(\text{Sa},50,65) \)
6. \( \text{inter5} = \text{reconstruct}(\text{Sb},\text{inter4}) \)
7. \( \text{join_input_S} = \text{reverse}(\text{inter5}) \)
8. \( \text{join_res_R_S} = \text{join}(\text{join_input_R},\text{join_input_S}) \)
9. \( \text{inter6} = \text{voidTail}(\text{join_res_R_S}) \)
10. \( \text{inter7} = \text{reconstruct}(\text{Rb},\text{inter6}) \)
11. \( \text{result} = \text{sum}(\text{inter7}) \)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

```
select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
```
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and
5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Initial Status

<table>
<thead>
<tr>
<th>Relation R</th>
<th>Relation S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ra</td>
<td>Rb</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
</tr>
<tr>
<td>41</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>55</td>
</tr>
</tbody>
</table>

Query and Query Plan (MAL Algebra)

```
select sum(R.a) from R, S where R.c = S.b and 
5<R.a<20 and 40<R.b<50 and 30<S.a<40
```

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
Query and Query Plan (MAL Algebra)

select sum(R.a) from R, S where R.c = S.b and 5<R.a<20 and 40<R.b<50 and 30<S.a<40

1. inter1 = select(Ra,5,20)
2. inter2 = reconstruct(Rb,inter1)
3. inter3 = select(inter2,40,50)
4. join_input_R = reconstruct(Rc,inter3)
5. inter4 = select(Sa,50,65)
6. inter5 = reconstruct(Sb,inter4)
7. join_input_S = reverse(inter5)
8. join_res_R_S = join(join_input_R,join_input_S)
9. inter6 = voidTail(join_res_R_S)
10. inter7 = reconstruct(Ra,inter6)
11. result = sum(inter7)
other data models

rdf, jason, xml, arrays, sciences?
reading
(also for next class)

The Design and Implementation of Modern Column-store Database Systems (Sections: all -4.6 & 4.8)
by D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden

IEEE Data Engineering Bulletin, 35(1), March 2012
Special Issue on Column-stores (9 short overview papers)

next class more details on column-store design
data layouts
column-stores

DATA SYSTEMS

prof. Stratos Idreos