class 4

basic db architectures & layouts

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
videos for sections 3 & 4 are online
check back every week (1-2 sections weekly)
there is a schedule but more will be added as we go
query plan

database kernel

algorithms/operators

data

data

data
\textbf{select} name \\
\textbf{from} student \\
\textbf{where} GPA \textgreater 3.0
```
select name
from student
where GPA > 3.0
```
```sql
select name
from student
where GPA > 3.0
```
\textbf{physical plans}

\begin{center}
\begin{tikzpicture}
  \node (s1) at (0,0) [circle, draw] {students};
  \node (s2) at (2,0) [circle, draw] {students};
  \node (p1) at (0,2) [circle, draw] {project name};
  \node (p2) at (2,2) [circle, draw] {project name};
  \node (r1) at (0,4) [circle, draw] {result};
  \node (r2) at (2,4) [circle, draw] {result};
  \node (sc1) at (0,6) [circle, draw] {scan GPA>3.0};
  \node (sc2) at (2,6) [circle, draw] {index scan GPA>3.0};
  \draw [->] (s1) -- (p1);
  \draw [->] (p1) -- (sc1);
  \draw [->] (sc1) -- (r1);
  \draw [->] (s2) -- (p2);
  \draw [->] (p2) -- (sc2);
  \draw [->] (sc2) -- (r2);
\end{tikzpicture}
\end{center}

\textbf{select name from student where GPA>3.0}
select avg(GPA) from student where class=2017
select avg(GPA) from student where class=2017
give me all students enrolled in cs165

```sql
select student.name
from student, enrolled, course
where course.name="cs165"
and enrolled.courseld=course.id
and student.id=enrolled.studentId
```
select student.name
from student, enrolled, course
where course.name = "cs165"
and enrolled.courseId = course.id
and student.id = enrolled.studentId

project student.name

join
student.id = enrolled.studentid

select course.name = "cs165"

join
enrolled.courseid = course.id

student

enrolled

course

good plan?
```
select student.name
from student, enrolled, course
where course.name = 'cs165'
and enrolled.courseId = course.id
and student.id = enrolled.studentId
```
\textbf{select} \textit{min(A)} \textbf{from} \textit{R where} \textit{B<10 and C<80}
\textbf{select} min(A) \textbf{from} R \textbf{where} B<10 \text{ and } C<80
select \( \min(A) \) from \( R \) where \( B<10 \) and \( C<80 \)
can DBAs make wrong decisions?

can optimizers make wrong decisions?

tuning

optimizer

execution

storage

db kernel
memory hierarchy

data layouts

column-stores basics
system where db runs

- cpu - cpu - cpu - cpu
- cpu registers
- caches
- memory
- disk - disk - disk - disk

memory hierarchy

+ flash
+ non volatile memory
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Innovations Award

registrers
my head
~0

2x
on chip cache
this room
1 min

10x
on board cache
this building
10 min

100x
memory
New York
1.5 hours

100Kx
disk
Pluto
2 years
Registers on chip cache

On board cache

Memory

Disk

CPU

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

~1ns

~10ns

~100ns

Memory wall

Memory miss: looking for something which is not in memory

Speed

Time

CPU

Mem

Faster

Cheaper
design of storage/access methods/algorithms should minimize:

- data misses
- instruction misses

Don’t miss!
touch/access only what you need
random access & page-based access

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

data value $x$

page1  page2  page3  ...
query \( x < 5 \)

(memory level N)

(page size: 5x8 bytes)

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

(memory level \( N \))

(scan)

(size=120 bytes)

(memory level \( N-1 \))

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

Scan

(memory level N)

(size = 120 bytes)

page size: 5x8 bytes

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

(memory level N)

(scan)

(size = 120 bytes)

(memory level N-1)

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

memory level $N$

(size=120 bytes)

memory level $N-1$

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

memory level $N$

(scan)

5 10 6 4 12

(scan)

2 8 9 7 6

4 2

(size = 120 bytes)

memory level $N-1$

5 10 6 4 12

2 8 9 7 6

7 11 3 9 6

... page size: 5x8 bytes

40 bytes
query \( x < 5 \)

memory level \( N \)

memory level \( N-1 \)

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

Scan

(size = 120 bytes)
memory level N

memory level N-1

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

(memory level N)

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

(scan)

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

(memory level N-1)

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

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

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

... page size: 5x8 bytes

80 bytes
query \ x < 5

scan

(size=120 bytes)
memory level N

memory level N-1

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

query $x < 5$

(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 ...

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

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

orcale

(size=120 bytes)

memory level N

memory level N-1

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

query \( x < 5 \)

<table>
<thead>
<tr>
<th>oracle</th>
<th>page size: 5x8 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(size=120 bytes)</td>
<td></td>
</tr>
<tr>
<td>memory level N</td>
<td>5 10 6 4 12</td>
</tr>
<tr>
<td></td>
<td>2 8 9 7 6</td>
</tr>
<tr>
<td></td>
<td>7 11 3 9 6</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>memory level N-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 10 6 4 12</td>
</tr>
<tr>
<td></td>
<td>2 8 9 7 6</td>
</tr>
<tr>
<td></td>
<td>7 11 3 9 6</td>
</tr>
</tbody>
</table>
an oracle gives us the positions

query $x < 5$

(size=120 bytes)
memory level N

oracle

5 10 6 4 12

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 $x < 5$

memory level $N$

(size=120 bytes)

memory level $N-1$

page size: 5x8 bytes

oracle

5 10 6 4 12

oracle

2 8 9 7 6

4
an oracle gives us the positions

query \( x < 5 \)

\[
\begin{align*}
\text{memory level } N & : 5, 10, 6, 4, 12 \\
\text{oracle} & \Rightarrow 5, 10, 6, 4, 12 \\
\end{align*}
\]

\[
\begin{align*}
\text{memory level } N-1 & : 2, 8, 9, 7, 6 \\
\text{oracle} & \Rightarrow 2, 8, 9, 7, 6 \\
\end{align*}
\]

\[
\begin{align*}
\text{oracle} & \Rightarrow 4, 2 \\
\end{align*}
\]

(size=120 bytes)

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

query $x < 5$

(size=120 bytes)
memory level N

<table>
<thead>
<tr>
<th>oracle</th>
<th>oracle</th>
<th>oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 10 6 4 12</td>
<td>2 8 9 7 6</td>
<td>4 2</td>
</tr>
</tbody>
</table>

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 $x < 5$

oracle

(size=120 bytes)

memory level N

\[\begin{array}{cccccc}
7 & 11 & 3 & 9 & 6 \\
2 & 8 & 9 & 7 & 6 \\
4 & 2 & 3 \\
\end{array}\]

memory level N-1

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

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

\[
x < 5
\]

memory level N

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

(size = 120 bytes)

memory level N-1

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

page size: 5x8 bytes
scan=120bytes vs oracle=120bytes
(and there is no such thing as an Oracle so Oracle is not for free…)

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?

hardware/software can better predict/buffer sequential pages to be read
random access:
read one block; consume it partially; discard it;
might have to read it again in future; read “random” next;
level N

buffer pool

level N-1

remember **hot** blocks

why not use OS caching
device block size

dbms block size

os block size

os and db will typically refer to **pages**
**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)
...

---

**Diagram:**
- **Page**
  - **Header**
  - **Row 1**
  - **Row 2**
  - **Row 3**
  - "..."
slotted page

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

free space

scan
null
update
var length
...

CS165, Fall 2016
Stratos Idreos
some things to “worry” about
how much data we transfer through the memory hierarchy
how many computations we do
row-store

file

A B C D

one page contains all fields of multiple attributes

stored continuously

select A, B, C, D

select A
select A,B,C,D
select A

row-store

column-store

A B C D

A B C D

stored continuously

one page contains fields of a single attribute
~1960s

1970: column storage ideas start appearing

~2000: open source complete system

1985: first rather complete column-store model

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

c-store, vertica, vectorwise and then ibm, microsoft, oracle, and more
R(A,B,C)

header

row1

row2

row3

column-store with materialized IDs

ID A ID B ID C
good idea
virtual ids/ positional alignment

### 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

column-store engine

option 2

A B C

early tuple reconstruction/materialization

row-store engine
it is not just memory and disk

we want to move as few data items as possible all the way up to the CPU
\textbf{select} min(C) \textbf{from} R \textbf{where} A<10 \& B<20

\begin{itemize}
  \item disk
  \begin{itemize}
    \item A
    \item B
    \item C
    \item D
  \end{itemize}
  \item memory
  \item write the query plan and the code/logic of each operator
  \item do not forget about intermediate results
  \item describe data layouts at each step
  \item (milestone 1 of project)
\end{itemize}
select min(C) from R where A<10 & B<20

write the query plan and the code/logic of each operator

do not forget about intermediate results
describe data layouts at each step

(milestone 1 of project)

no precise final answer is OK
understanding what matters is key
concepts & designs will be repeated >>1
late reconstruction/materialization

```
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

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

\begin{center}
\begin{tabular}{cccc}
\textbf{A} & \textbf{B} & \textbf{C} & \textbf{D} \\
\begin{tabular}{c}
\includegraphics[width=0.1\textwidth]{disk}
\end{tabular} & \begin{tabular}{c}
\includegraphics[width=0.1\textwidth]{memory}
\end{tabular} & \begin{tabular}{c}
\includegraphics[width=0.1\textwidth]{memory}
\end{tabular} & \begin{tabular}{c}
\includegraphics[width=0.1\textwidth]{memory}
\end{tabular} \\
\end{tabular}
\end{center}

\begin{enumerate}
\item \texttt{int *input=A} \\
\item \texttt{for (i=0;i<tuples;i++,input++)} \\
\item \texttt{if *input<10} \\
\item \texttt{*output=i} \\
\item \texttt{output++}
\end{enumerate}
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

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

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

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

```
select min(C) from R where A<10 & B<20
```

always sequential access patterns
memory contains only what is needed at any point in time
Notes to remember

column-stores vs row-stores
it all starts with how we store the data
still basic concepts are the same
moving data is a major cost component
it is not just about disk…
the whole memory hierarchy matters
please keep up with reading!

Architecture of a Database System  (Sections 1,2,3,4)
by J. Hellerstein, M. Stonebraker and J. Hamilton

The Design and Implementation of Modern Column-store Database Systems
by D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden
basic db architectures & layouts

DATA SYSTEMS

prof. Stratos Idreos