Computing Technology in 1969 vs 2001

<table>
<thead>
<tr>
<th></th>
<th>1969</th>
<th>2001</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>main memory</td>
<td>200 KB</td>
<td>200 MB</td>
<td>$10^3$</td>
</tr>
<tr>
<td>cache</td>
<td>20 KB</td>
<td>20 MB</td>
<td>$10^3$</td>
</tr>
<tr>
<td>cache pages</td>
<td>20</td>
<td>5000</td>
<td>&lt;$10^3$</td>
</tr>
<tr>
<td>disk size</td>
<td>7.5 MB</td>
<td>20 GB</td>
<td>$3 \times 10^3$</td>
</tr>
<tr>
<td>disk/memory size</td>
<td>40</td>
<td>100</td>
<td>-2.5</td>
</tr>
<tr>
<td>transfer rate</td>
<td>150 KB/s</td>
<td>15 MB/s</td>
<td>$10^2$</td>
</tr>
<tr>
<td>random access</td>
<td>50 ms</td>
<td>5 ms</td>
<td>10</td>
</tr>
<tr>
<td>scanning full disk</td>
<td>130 s</td>
<td>1300 s</td>
<td>-10</td>
</tr>
<tr>
<td>(accessibility)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Challenge of Applications in 1969

Space Industry
- Supersonic Transport SST
- C5A
- Boeing 747

Manufacturing
- parts explosion
- (spare) parts management

Commerce
- bank check management
- credit card management

Basics of B-Trees
Basics of B-Trees: Insertion

Basics of B-Trees: the Split
Basics of B-Trees: recursive Split

Basics of B-Trees: Growth at Root
Fundamental Properties of B-Trees

Time & I/O Complexity: $O(\log_k n)$; $k > 400$
for all elementary operations
  - find
  - insert & delete

Storage Utilization ~ 83%

Growth:

<table>
<thead>
<tr>
<th>Height</th>
<th>Nodes</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8 KB</td>
</tr>
<tr>
<td>2</td>
<td>400</td>
<td>3.2 MB</td>
</tr>
<tr>
<td>3</td>
<td>$16\times10^4$</td>
<td>1.3 GB</td>
</tr>
<tr>
<td>4</td>
<td>$64\times10^6$</td>
<td>512 GB</td>
</tr>
</tbody>
</table>

$\Rightarrow$ < 4 logical I/O per operation!!
Independence of DB Size

\[ \text{Index part} \leq 1\% \text{ of file} \]
\[ \text{remember: since 1969} \]
\[ \text{disk size} \]
\[ \text{memory size} \]
\[ \approx \text{const} \approx 100 \]

\[ \Rightarrow < 2 \text{ physical I/O per operation} !! \]

DB-Models in 1969

**IMS:** hierarchical, commercial success

**CODASYL:** network model,
M. Senko, C. Bachmann

**Relational:** E. F. Codd, theory only

Senko, Codd in same department:
Information Systems Department
IBM Research Lab, San José

Senko ↔ Codd
Efficiency ↔ Simplicity
Relational DB-Model, Ted Codd


Relational Algebra = Tables + Operators

today: \( \pi \times \sigma \) set operators \( \iota \)

Codd: \( \pi \iota_{\text{lossless}} \) restriction by table tie

\( \Rightarrow \) algebraic laws for query optimization

(Codd does not mention this aspect)

2 Languages

- imperative, procedural:
  algebraic expressions

- declarative, non-procedural:
  applied predicate calculus,
  DSL/Alpha (1971)

\( \Rightarrow \) no implementation of acceptable efficiency in sight!
Hard Questions from 1969-1974

- which model?
- which language?
- which implementation?

⇒ infighting, Codd to Systems Department
⇒ defer decisions: *rel. Storage System* RSS
to support all models and languages

1974 Cargese Workshop, Frank King

Which Language?

**DL/I:** IMS
**CODASYL:** COBOL + pointer chasing and
currency indicators, Chamberlin
**Rel. Algebra:** Codd
**DSL/Alpha:** Codd
**SQUARE:** Chamberlin, et al.
**SEQUEL:** Chamberlin, Boyce, Reisner
**QBE:** Moshe Zloof
**Rendezvous:** Codd

⇒ 3 survivors: DL/I, SQL, Rel.Algebra
Implementation: System R, IBM

SQL: Chamberlin, Reisner
Schemata: normalization, Codd, Boyce
Rel. Algebra: Codd et al.
Optimization: Blasgen, Selinger, Eswaran
Cost Models:
Transactions: Gray, Traiger
B-Trees: Bayer, Schkolnick, Blasgen
Recovery: Lorie, Putzolu

Factors for Product Success

- simple, formalized model
- simple user interface: SQL
- algebra + laws for optimization
- performance: B-trees
- multiuser: transactions (Gray)
- robustness: transactions + recovery, self-organization of B-trees
- scalability: B-trees with logarithmic growth, parallelism
**Prefix B-Trees**

- store shortest separators: **Simple Prefix B-trees**
- trim common prefixes: **Prefix B-trees**

**Concurrency and B-Trees**

- everybody reads root
- root almost never changes
- low probability of conflicts near leaves
- combination of synchronization protocols
- no chance of testing real general case
UB-Trees: Multidimensional Indexing

- geographic databases (GIS)
- Data-Warehousing: Star Schema
- all relational databases with n:m relationships

Basic Idea of UB-Tree

- linearize multidimensional space by space filling curve, e.g. Z-curve or Hilbert
- Use Z-address to store objects in B-Tree

→ **Response time for query is proportional to size of the answer!**
UB-Tree: Regions and Query-Box

Z-region
[0.1 : 1.1.1]

World as self balancing UB-Tree