Rudolf Bayer
Technische Universität München

B-Trees and Databases, Past & Future
## Computing Technology in 1969 vs 2001

<table>
<thead>
<tr>
<th>Feature</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>$3*10^3$</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 (accessibility)</td>
<td>130 s</td>
<td>1300 s</td>
<td>-10</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

Diagram:
- The root node contains keys: 5, 11, 16, 21.
- The left child of the root contains keys: 1, 2, 3, 4.
- The left child of the left child contains keys: 6, 7, 8, 9, 10.
- The right child of the root contains keys: 12, 13, 15.
- The right child of the left child contains keys: 17, 18, 19, 20.
- The right child of the right child contains keys: 22, 24, 25.
Basics of B-Trees: Overflow
Basics of B-Trees: the Split

![Diagram of a B-tree split]
Basics of B-Trees: recursive Split
Basics of B-Trees: Growth at Root
Mental Hurdles

- storage utilization: 50% was sacrilege in 1969
- growth behavior: different from natural trees

Scientific American 1984
Optimal Page Size?

\[
\alpha/\beta = 2(((\sqrt[k]{k+1})/\nu)*\ln(\sqrt[k]{k+1})) - (2k+1) = f(k, \nu)
\]

- \(\alpha\) latency time of disk
- \(\beta\) transfer rate
- \(\nu\) page occupancy
- \(k\) branching factor of B-Tree
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: height nodes size
1 1 8 KB
2 400 3.2 MB
3 $16*10^4$ 1.3 GB
4 $64*10^6$ 512 GB

⇒ < 4 logical I/O per operation !!
Independence of DB Size

Index part $\leq 1\%$ of file
remember: since 1969
disk size
memory size

$\approx \text{const} \approx 100$

$\Rightarrow < 2$ physical I/O per operation !!
DB-Models in 1969

**IMS:** hierarchical, commercial success
**CODASYL:** network model, 1966-1968
  M. Senko, C. Bachmann, IDS

**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 = Relations + Operators

today: $\pi \times \sigma$ set operators $\uparrow$

Codd: $\pi \downarrow_{\text{lossless}}$ restriction by table $\uparrow$

$\Rightarrow$ 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)

→ 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?

<table>
<thead>
<tr>
<th>Language</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL/I</td>
<td>IMS</td>
</tr>
<tr>
<td>CODASYL</td>
<td>COBOL + pointer chasing and currency indicators, Chamberlin</td>
</tr>
<tr>
<td>Rel. Algebra</td>
<td>Codd</td>
</tr>
<tr>
<td>DSL/Alpha</td>
<td>Codd</td>
</tr>
<tr>
<td>SQUARE</td>
<td>Chamberlin, et al.</td>
</tr>
<tr>
<td>SEQUEL</td>
<td>Chamberlin, Boyce, Reisner</td>
</tr>
<tr>
<td>QBE</td>
<td>Moshe Zloof</td>
</tr>
<tr>
<td>Rendezvous</td>
<td>Codd</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Component</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>Chamberlin, Reisner</td>
</tr>
<tr>
<td>Schemata:</td>
<td>normalization, Codd, Boyce</td>
</tr>
<tr>
<td>Rel. Algebra:</td>
<td>Codd et al., ~ 80 operators</td>
</tr>
<tr>
<td>Optimization:</td>
<td>Blasgen, Selinger, Eswaran</td>
</tr>
<tr>
<td>Cost Models:</td>
<td></td>
</tr>
<tr>
<td>Transactions:</td>
<td>Gray, Traiger</td>
</tr>
<tr>
<td>B-Trees:</td>
<td>Bayer, Schkolnick, Blasgen</td>
</tr>
<tr>
<td>Recovery:</td>
<td>Lorie, Putzolu</td>
</tr>
</tbody>
</table>
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

Bayer, Schkolnick: Acta Informatica 9, 1-21 (1977)

- everybody reads root
- root almost never changes
- low probability of conflicts near leaves

- combination of synchronization protocols
- no chance of testing real general case
Time for B-Tree of Height 6

\[ 10^6 \text{ B/s} \times 10^5 \text{ s/day} = 10^{11} \text{ B/day} = 100 \text{ GB/day} \]

→ B-Tree of height 4 takes 5 days to bulkload
  height 5 takes 2000 days
  height 6 takes ...

→ *general case of protocol at height 6, correctness proof is a must!*
Products based on B-Trees

- IBM System R, 1976/77 (prototype)
- Oracle 1979
- IBM SQL/DS
- Ingres, later versions
- IBM DB2
- Informix
- Sybase
- Microsoft, SQL-Server
UB-Trees: Multidimensional Indexing

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

• XML
• mobile, location based applications
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

\[ \text{Response time for query is proportional to size of the answer!} \]
UB-Tree: Regions and Query-Box

Z-region
[0.1 : 1.1.1]
World as self balancing UB-Tree