MORE ABOUT DATABASES

Objectives

- After this unit you will be able to explain the concepts of:
  - Transaction, scheduling conflicts, ACID
  - How indexes speed up query procecessing

Transactions

- Queries by multiple users, can arrive simultaneously
- DBMS must handle concurrent execution of queries
  - high-throughput systems with 1,000s of TAs / sec
  - Disk access frequent & slow → keep CPU humming
- Transaction (TA) = sequence of queries forming a unit
  - Flight booking; Ebay buy; …
- OLTP = Online Transaction Processing

Transactions: The Challenge

- Every user can safely think of only its own TA, all others unknown
- Concurrency achieved by DBMS, interleaving reads/writes of active TAs
- TA must leave DB in consistent state
  - Ex: primary key & unique attributes; foreign keys…plus much more
- What can go wrong?
  - TAs conflict
  - TAs aborted
  - Server crashes

Transaction Syntax in SQL

- START TRANSACTION   start TA
- COMMIT   end TA successfully
- ROLLBACK   abort TA (undo any changes)
- If user omits/forgets: 1 query = 1 TA

ACID

- TA concept grounding on four basic properties:
- Atomic
  - all TA actions will be completed, or nothing
- Consistent
  - after commit/abort, data satisfy all integrity constraints
- Isolation
  - any changes are invisible to other TAs until commit
- Durable
  - nothing lost in future; failures occurring after commit cause no loss of data

Good Transaction Behavior

- Ex: Bank account services

```
T1:    BEGIN  A=A-100,   B=B+100   END
T2:    BEGIN  A=1.06*A,  B=1.06*B  END
```

  - T1 transfers $100 from B's account to A's account
  - T2 credits both accounts with a 6% interest payment

- no guarantee T1 will execute before T2 or vice-versa when submitted together; that's ok

- However, net effect must be equivalent to TAs running serially in some order

Anatomy of Conflicts

- Consider a possible interleaving (schedule):

```
T1:     A=A-100,                      B=B+100
T2:                   A=1.06*A,                       B=1.06*B
```

- This is OK. But what about:

```
T1:     A=A-100,                                      B=B+100
T2:                   A=1.06*A, B=1.06*B
```

- Problem: Reading uncommitted data (R/W conflicts, “dirty reads”)
- *…plus a few more such bad situations*

Lock-Based Concurrency Control

- DBMS schedules reads & writes in a way preserving serializability = consistent execution
  - Like traffic lights
- Support mechanism: lock table, for each tuple + activity (r,w)
  - TA must apply for S (shared) lock before reading, X (exclusive) lock before writing

```
  | S X
--+-----
S | + -
X | - -
```

- Locking protocols
  - two-phase locking (strict, non-strict, conservative, …)
  - Multi-version based
  - Optimistic concurrency control

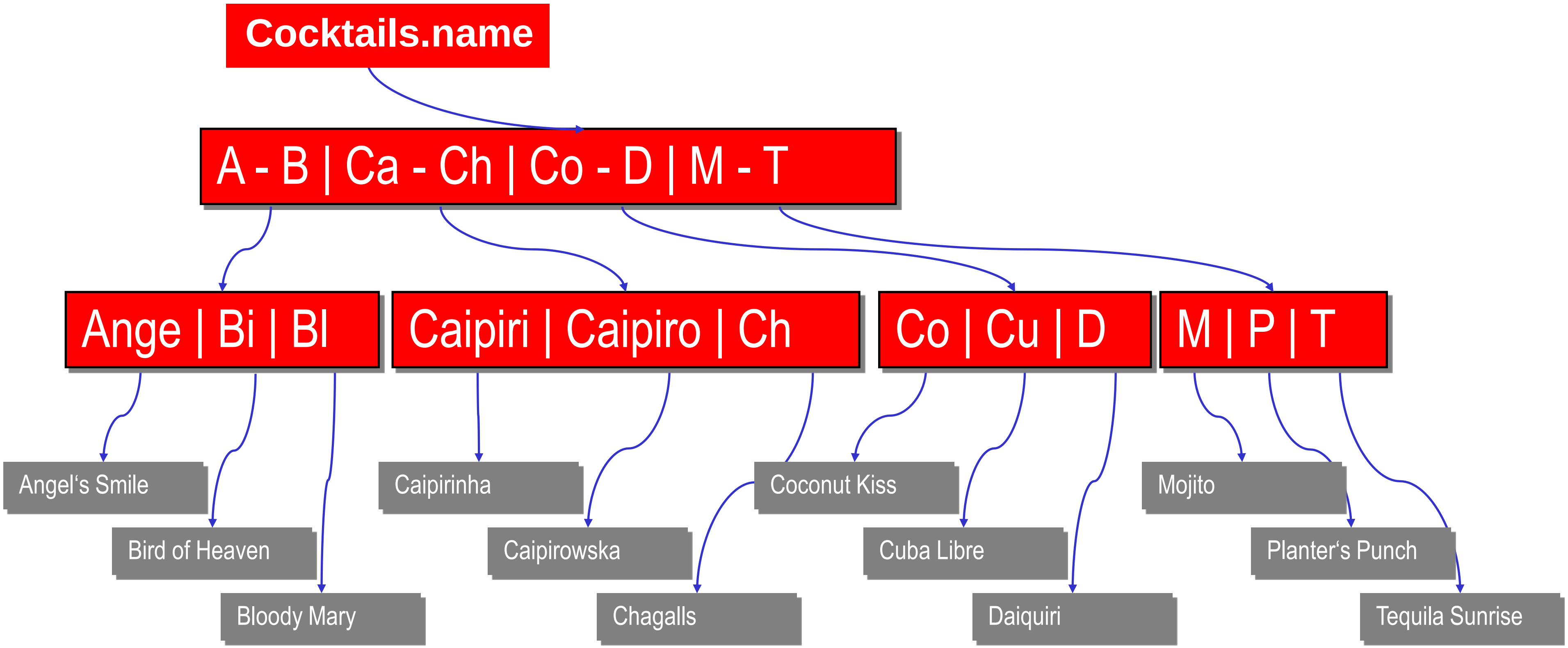
Indexing Data

- Problem: How to find specific tuples in a table?
- Alt 1: Brute force: table stored in file → scan file sequentially
  - 1 table with N tuples → N/2 search time on average
  - 2 tables with N, M tuples → N*M effort
- Alt 2: prepare small lookup table, called index
  - Not full tuple stored, only search criterion + path to data item
  - Extra magic for fast search
- Prominent: B-Tree index, Hash index

How Does a Tree Index Work?

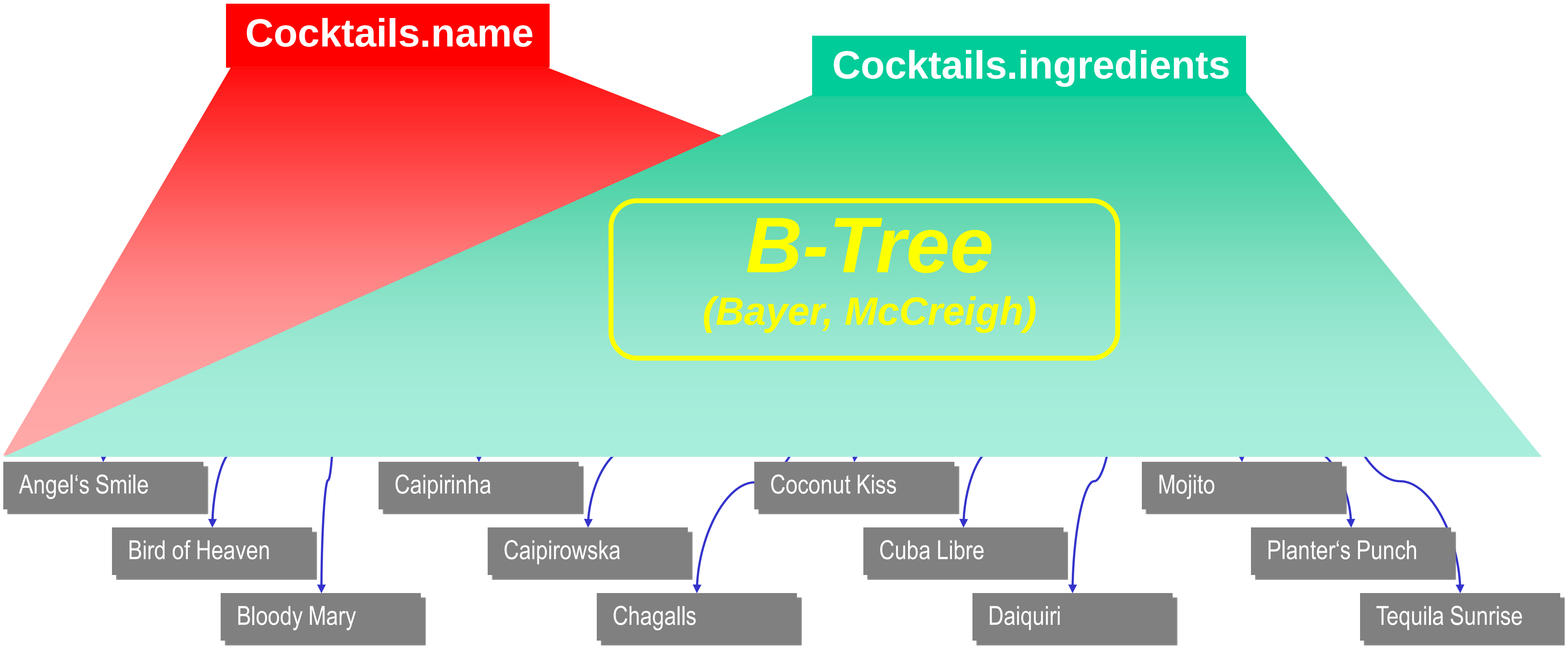
- Index on name attribute in Cocktails table, helpful for eg this query:

*SELECT * FROM Cocktails WHERE name like „Cai%“*



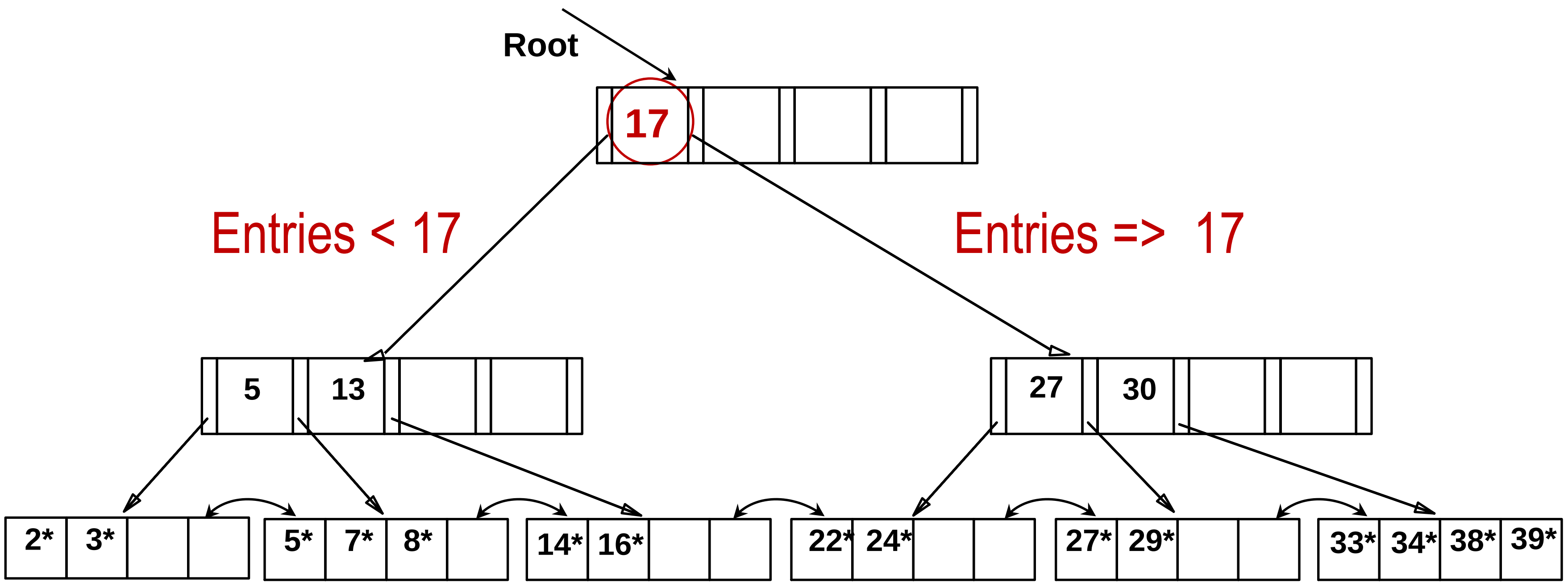
How Does a Tree Index Work?

- Need fast access path to more attributes? Create further indexes!
- But think of update frequency



Searching with B-Trees

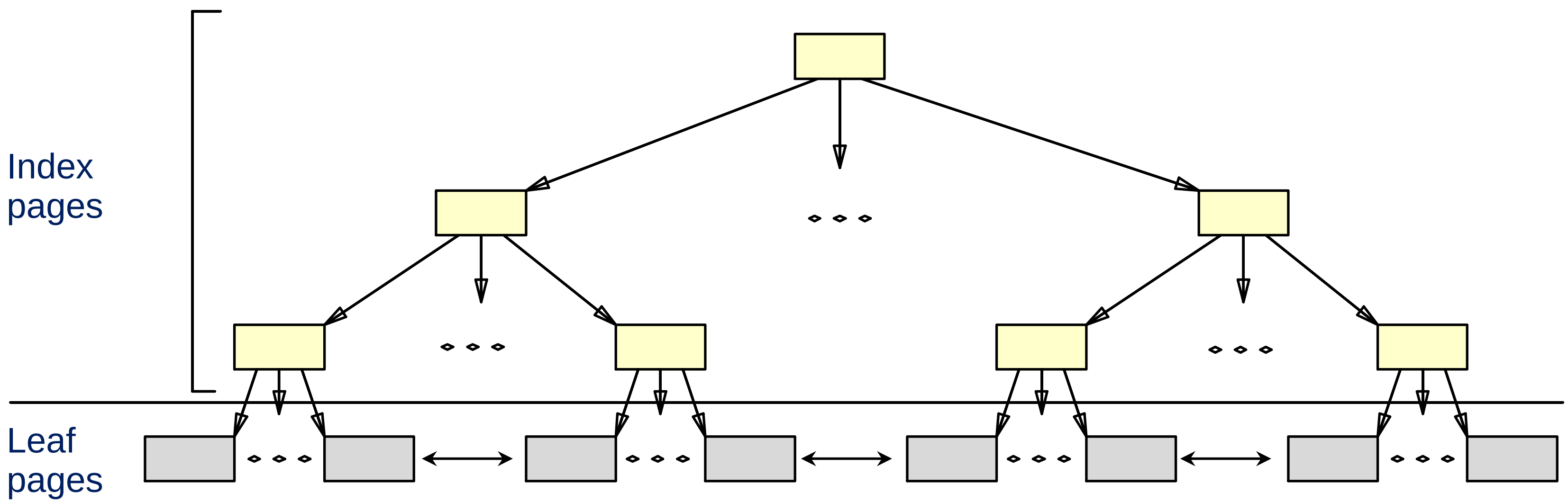
- Point query = retrieve by exact value (eg, grade)
- Range query = retrieve all within a range (eg, passing grades)



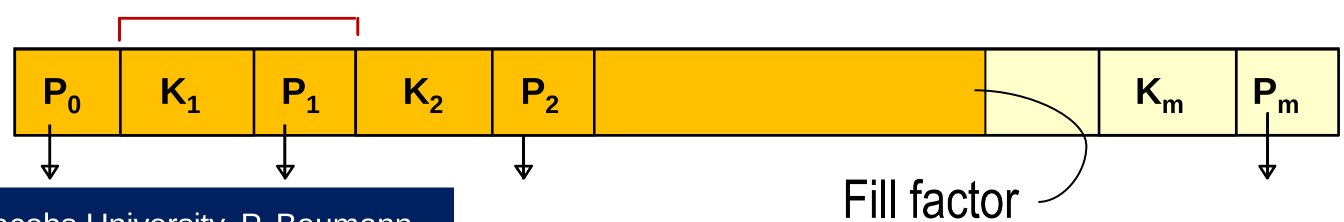
B-Tree

[Bayer & McCreight, 1972]

- B-Tree = n-ary tree optimized for black storage
  - Block = n discriminators + (n+1) pointers to subtrees or leafs
  - leaf chains for fast range queries



- B-tree block:



Why is This Fast?

- $O(\log_F N)$ where F = fan-out, N = # leaf pages
- Typical fan-out: 133
- Typical capacities:
  - Height 3: 133^3 = 2,352,637 records
  - Height 4: 133^4 = 312,900,700 records
- Can often hold top levels in buffer pool:
  - Level 1 = 1 page = 8 Kbytes
  - Level 2 = 133 pages = 1 Mbyte
  - Level 3 = 17,689 pages = 133 Mbytes

Summary

- Picked 2 distinct aspects for looking inside a DBMS
- Concurrency control
  - Transactions, scheduled via locks
  - ACID
- Indexing
  - Access paths to data, speeding up queries