Databases 1

Daniel POP

Course Outline

1. ~~Introduction to database approach~~
2. ~~The database environment~~
3. ~~Introduction to The Relational Model~~
4. ~~Views~~
5. ~~Transactions~~
6. ~~SQL Constraints~~
7. ~~Relational Database Design. Theory and practice~~
8. An Introduction to Database Performance. Indexing
9. JSON Support in Relational Database Management Systems
10. NoSQL Databases

Week 12
Indexing. Basics

Agenda

1. What is an Index
2. Index implementation in relational DBMS
3. Demo

Indexes

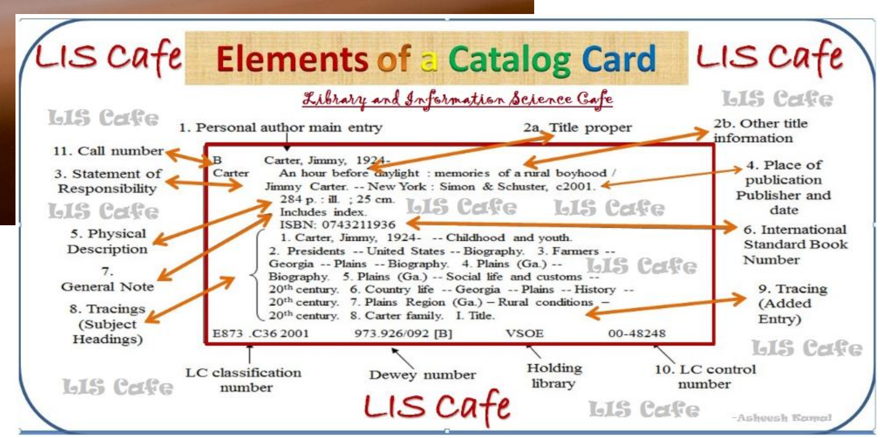
- Primary mechanism to improve the performance of database
- Persistent data structures stored in the database
- Difficult to implement in DBMS
- (Most of the times) Not directly used in queries

Indexes

- Analogy with
  - Index of terms in a book
  - Phonebook
  - Public libraries

Indexes

Sci/Tech. Ctr.
RA778 .P28
Pattammal, Madras Kanagasabapathy, 1910-
advice to daughter.
CD. DUPL.



LIS Cafe
Elements of a Catalog Card
LIS Cafe
Library and Information Science Cafe
1. Personal author main entry
2a. Title proper
2b. Other title information
11. Call number
3. Statement of Responsibility
4. Place of publication Publisher and date
5. Physical Description
6. International Standard Book Number
7. General Note
8. Tracings (Subject Headings)
9. Tracing (Added Entry)
B
Carter
Carter, Jimmy, 1924-
An hour before daylight : memories of a rural boyhood /
Jimmy Carter. -- New York : Simon & Schuster, c2001.
284 p. : ill. ; 25 cm.
Includes index.
ISBN: 0743211936
1. Carter, Jimmy, 1924- -- Childhood and youth.
2. Presidents -- United States -- Biography. 3. Farmers --
Georgia -- Plains -- Biography. 4. Plains (Ga.) --
Biography. 5. Plains (Ga.) -- Social life and customs --
20th century. 6. Country life -- Georgia -- Plains -- History --
20th century. 7. Plains Region (Ga.) -- Rural conditions --
20th century. 8. Carter family. I. Title.
E873 .C36 2001 973.926/092 [B] VSOE 00-48248
LC classification number
Dewey number
Holding library
10. LC control number
LIS Cafe
-Asheesh Kamal

Example

```
Course(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Student(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40))

Enrollment(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)
```

```
(Name,     TotalCredits, …)
------------------------
Popescu      50
Ionescu      70
Vasilescu    66
Popescu      45
Ionescu      60
```

SELECT * FROM Student
WHERE StudName='Popescu'

To speed up queries on columns Name (e.g. Student.StudName = 'Popescu') we will build an index on column Name that will quickly return the answer **without scanning the entire table** => Orders of magnitude performance improvements

Example

```
Course(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Student(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40))

Enrollment(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)
```

```
(Name,    TotalCredits, …)
----------------------
Popescu      50
Ionescu      70
Vasilescu    66
Popescu      45
Ionescu      60
```

SELECT StudID, StudName
FROM Student
WHERE StudName = ‘Popescu’ AND
TotalCredits > 60

Example

```
Course(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Student(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40))

Enrollment(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)
```

```
(Name,     TotalCredits, …)
------------------------
Popescu      50
Ionescu      70
Vasilescu    66
Popescu      45
Ionescu      60
```

SELECT StudID, StudName
FROM Student
WHERE StudName = 'Popescu' AND
TotalCredits > 60

SELECT StudName, CourseTitle
FROM Student JOIN Enrollment
ON Student.StudID = Enrollment.StudID

Index Implementation

- Balanced B-Tree
- Each node is represented by a 8k page
- Pages on same level -> double linked list
- Logarithmic running time
- Very good for:
  - Searching for a value "=“
  - Scanning a range of values "BETWEEN" (value1 < A < value2)
  - Sorted output (table joins)

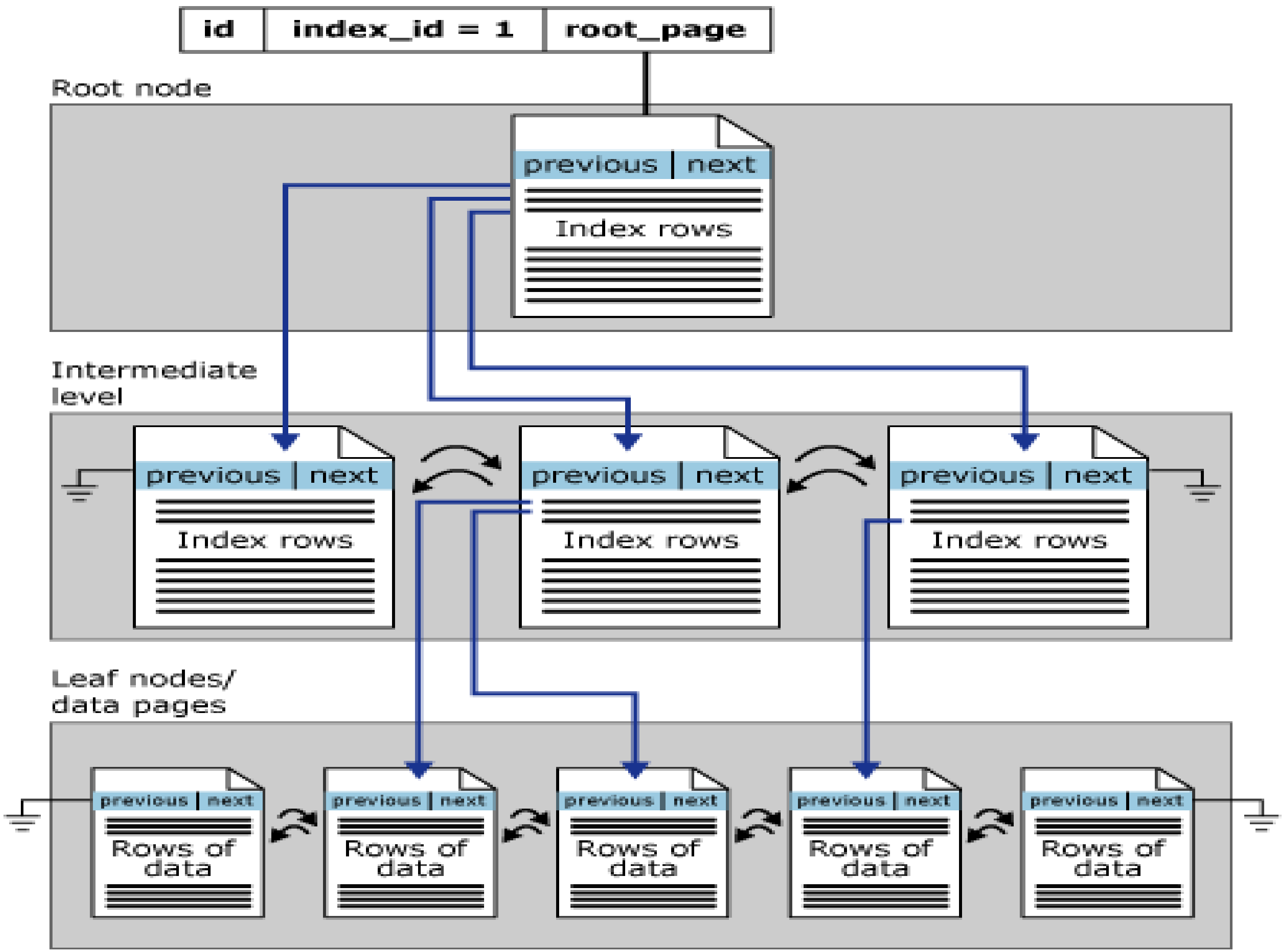
Reminder
A balanced B-Tree requires that each leaf is at the same distance from the root.

https://www.youtube.com/watch?v=HZRPa0kMOZE – Learn B-tree with cards

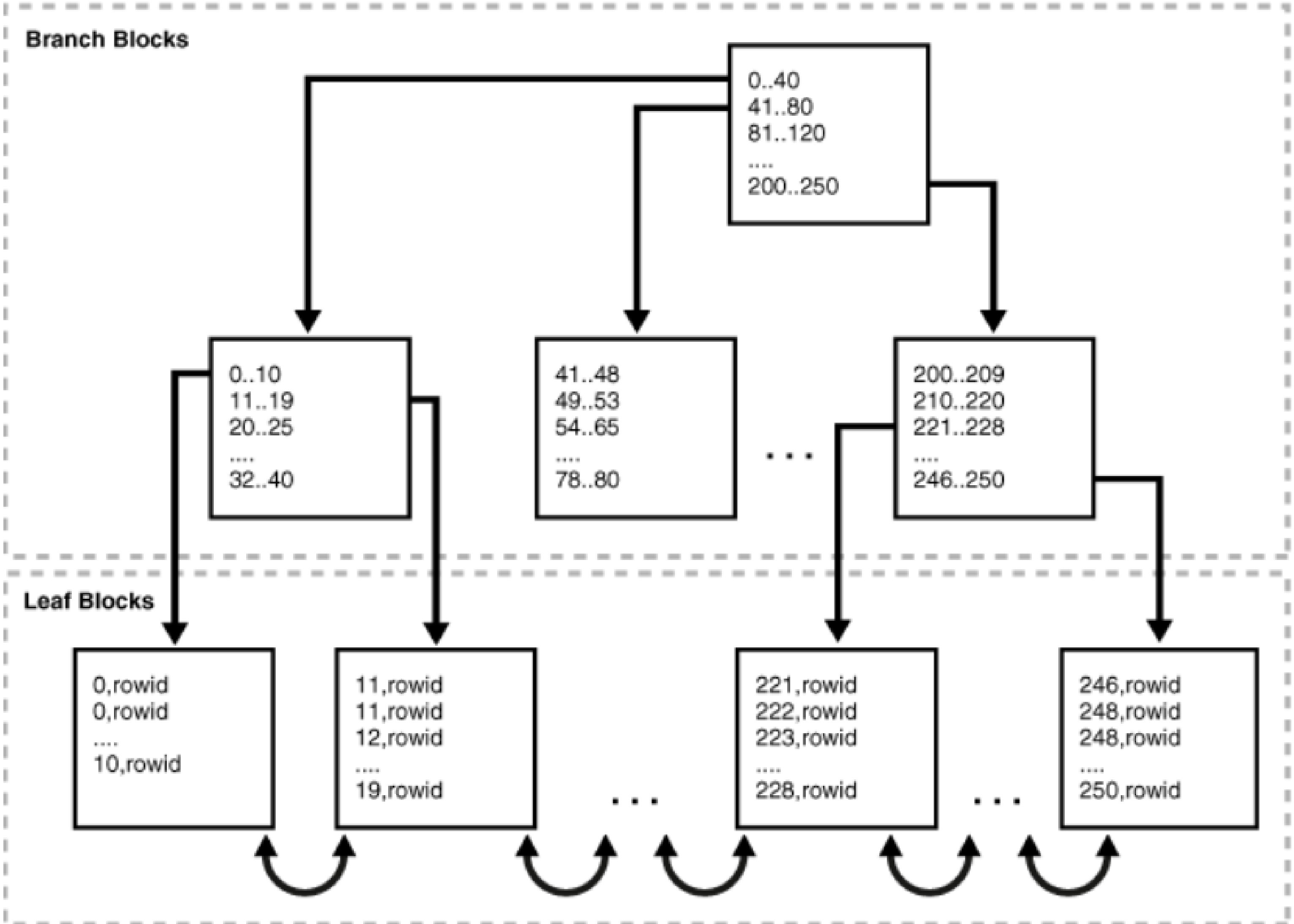
MS SQL Server Clustered vs. Non-clustered

- A clustered index
  - Type of index that reorders the way records in the table are physically stored
  - Therefore table can have only one clustered index.
  - The leaf nodes of a clustered index contain the data pages.
- A non-clustered index
  - Type of index in which the logical order of the index does not match the physical stored order of the rows on disk.
  - A table may have multiple non-clustered indexes
  - The leaf node of a non-clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

MS SQL Server Clustered Index



MS SQL Server Non-Clustered Index



Description of "Figure 3-1 Internal Structure of a B-tree Index"

SQL Server Clustered vs. Non-clustered

- A clustered index means you are telling the database to store close values actually close to one another on the disk. This has the benefit of rapid scan / retrieval of records falling into some range of clustered index values.

- Example: If you wish to quickly retrieve all courses on which a student is enrolled to, you may wish to create a clustered index on *StudID* column of the Enrollment table. This way the records with the same StudID will be physically stored close to each other on disk (clustered) which speeds up their retrieval.

- Remark: The index on StudID will obviously be not unique, so you either need to add a second field to "uniquify" the index or let the database handle that for you.

SQL Server Clustered vs. Non-clustered

- The key difference between clustered indexes and non clustered indexes is that the leaf level of the clustered index is the table. This has two implications
  - The rows on the clustered index leaf pages always contains something for each of the (non sparse) columns in the table (either the value, or a pointer to the actual value).
  - The clustered index is the primary copy of a table.
- Non clustered indexes can also store actual data by using the INCLUDE clause (Since SQL Server 2005) to explicitly include non key columns
- If a table has no clustered index it is called a heap. Non-clustered indexes can be created on both heap and clustered tables.
- When you create a PRIMARY KEY constraint, a unique clustered index on PK column(s) is automatically created if a clustered index on the table does not already exist.

https://stackoverflow.com/questions/1251636/what-do-clustered-and-non-clustered-index-actually-mean

Clustered vs. Non-clustered - Other DBMS

SQL SERVER

- SQL Server uses clustered indexes by default (index-organized tables), using the primary key as clustering key.
- To create a heap table you must use the NONCLUSTERED clause in the primary key definition

ORACLE

- Oracle database uses heap tables by default. Index-organized tables can be created using the ORGANIZATION INDEX clause, always using the primary key as the clustering key.

PostgreSQL

- PostgreSQL only uses heap tables
- You can, however, use the CLUSTER clause to align the contents of the heap table with an index.

MySQL

- The MyISAM engine only uses heap tables while the InnoDB engine always uses clustered indexes.

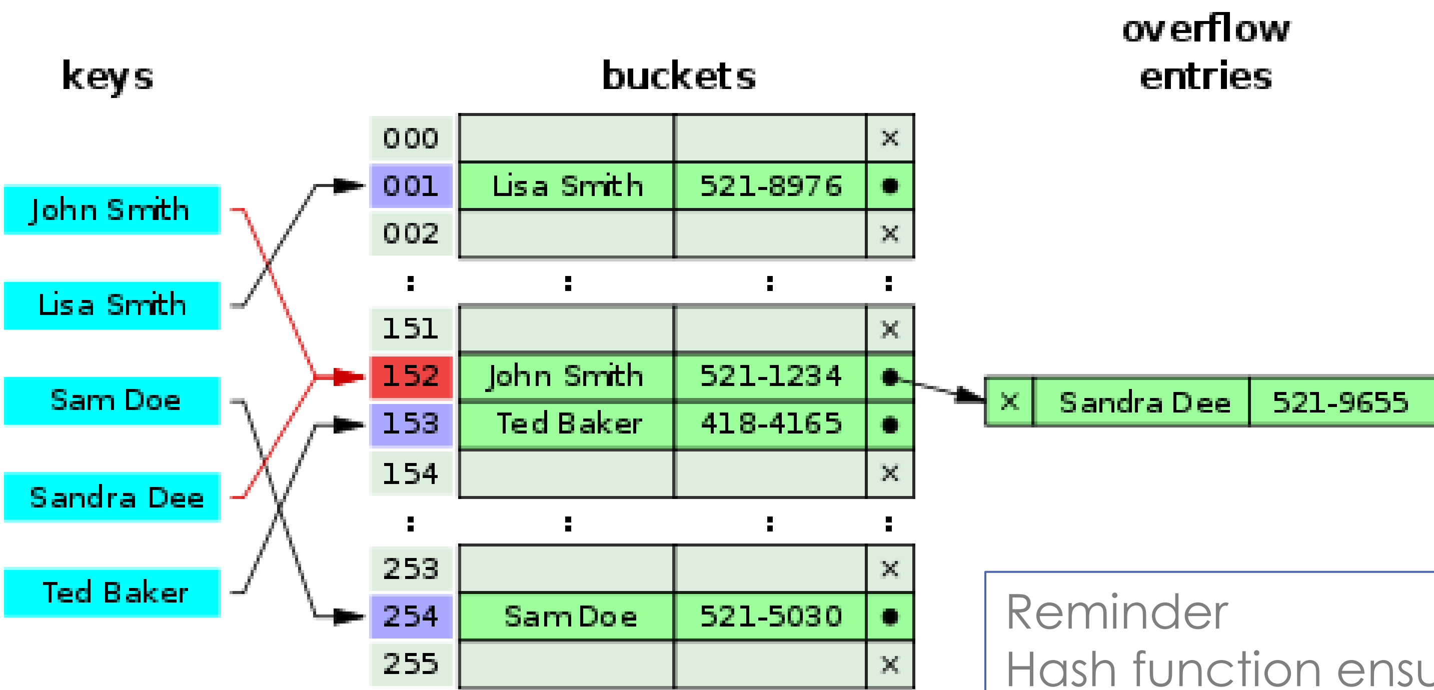
https://use-the-index-luke.com/sql/clustering/index-organized-clustered-index

MS SQL Server - Hash Index

- Only on Memory-Optimized tables
- Fixed size
- Deterministic
- Poisson or bell curve distribution
- Optimized for point lookups “=”
- Constant running time
- Not good for range scans or inequality clauses
- If on avg > 100 entries/value then use non-clustered index
- Number of buckets must be set when index is created

https://docs.microsoft.com/en-us/sql/database-engine/determining-the-correct-bucket-count-for-hash-indexes?view=sql-server-2014

Hash table

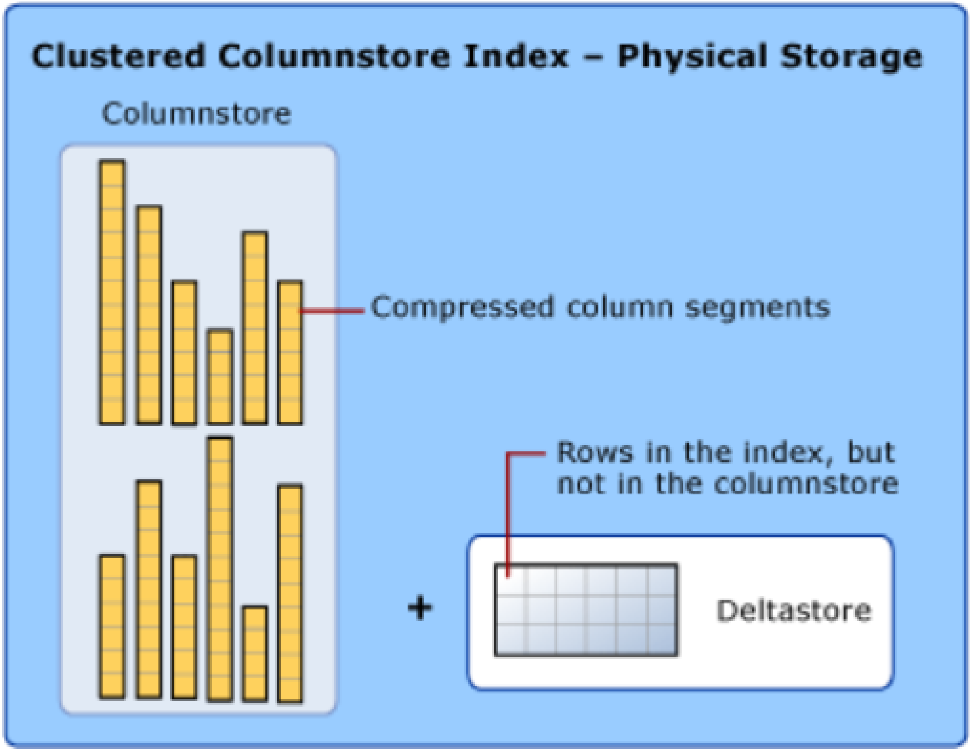


Reminder
Hash function ensures a uniform distribution of hash values.

Hash collision strategy: in this example, separate chaining with head records in the bucket array.

Source: http://en.wikipedia.org/wiki/Hash_table

MS SQL Server – Columnstore Index



- Can only be scanned
- High compression rates (up to 10x)
- No sort order
- Segment and column elimination
- Good for:
  - Scanning
  - Aggregation

https://www.red-gate.com/simple-talk/databases/sql-server/t-sql-programming-sql-server/what-are-columnstore-indexes

https://learn.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?view=sql-server-ver16

https://swarm64.com/post/postgresql-columnstore-index-intro/

Exercise

- Given the following query

  SELECT StudID, CourseTitle, Credits
  FROM Enrollment, Course
  WHERE Enrollment.CourseTitle = Course.CourseTitle AND
  Enrollment.Decision = TRUE and Course.Credits > 4

  which of the following indexes could NOT be useful in speeding-up the query?

  a) tree-based index on Enrollment.CourseTitle
  b) hash-based index on Enrollment.Decision
  c) hash-based index on Course.CourseTitle
  d) hash-based index on Course.Credits

Shortcomings of indexes

- Extra space on database
- Overhead of index creation (usually when the database is loaded)
- Maintenance of index: index has to be modified every time values in the table are changed so that it slows down the modifications (add/delete/update)
  - Defragment
  - Rebuild

SQL Implementation

- Indexes are created by default for PRIMARY KEY and UNIQUE constraints
- CREATE INDEX IdxName ON Student (Name)
- CREATE UNIQUE INDEX IdxCNP ON Student (CNP)
  - Check that all values for CNP are unique and will generate an error in case of duplicates
- ALTER INDEX IdxName ON TableName REBUILD
- DROP INDEX IdxName

MS SQL Server Implementation

```
CREATE CLUSTERED INDEX [CLI_SalesOrderDetailID]
ON [dbo].[BigTable] ([SalesOrderDetailID] ASC)

CREATE NONCLUSTERED INDEX [NonClusteredIndexDemo]
ON [dbo].[BigTable]
(
      [UnitPrice] ASC,
      [ModifiedDate] ASC
)
INCLUDE ([OrderQty])
```

Demo

- Adventure Works Sample Database from MS

- BigTable example
  - Structure. Number of records
  - Indexes

- Performance demo
  - CPU time vs. Elapsed time
  - Discuss the query plan
  - Look into physical structure of the index

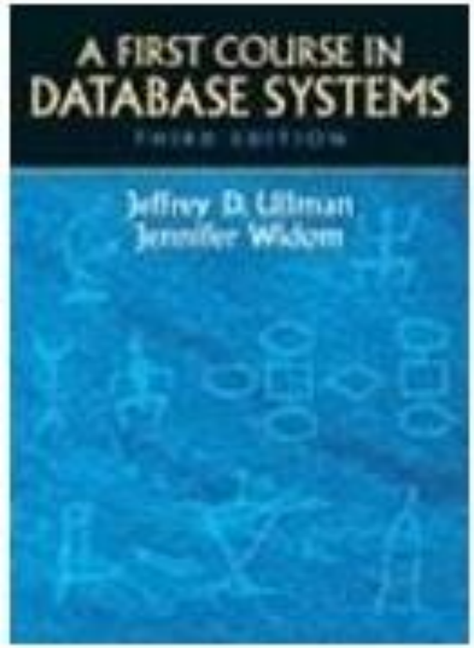
Demo

```
-- Enable the execution time tracking in the messages tab
SET STATISTICS TIME ON

-- Index Seek vs. Table Scan (clustered index) Performance
SELECT OrderQty FROM [dbo].[BigTable]
WITH (INDEX(CLI_SalesOrderDetailID))
WHERE UnitPrice = 5.70

SELECT OrderQty FROM [dbo].[BigTable]
WITH (INDEX(NonClusteredIndexDemo))
WHERE UnitPrice = 5.70


-- Aggregating a column using a column-store index vs. clustered index
SELECT AVG(UnitPrice) FROM [dbo].[BigTable]
WITH (INDEX(CLI_SalesOrderDetailID))

SELECT AVG(UnitPrice) FROM [dbo].[BigTable]
WITH (INDEX(NonClusteredColumnStoreIndexDemo))
```

Bibliography (recommended)

Teoria generala a bazelor de date, I. Despi, G. Petrov, R. Reisz, A. Stepan, Mirton, 2000

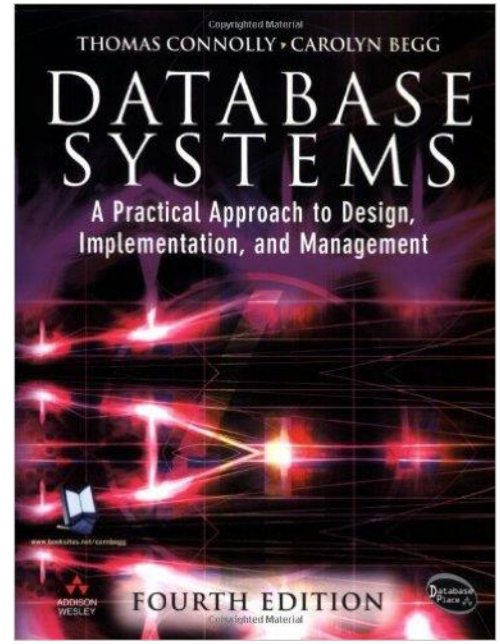
Cap 10.4 - 10.5



A First Course in Database Systems (3rd edition) by Jeffrey Ullman and Jennifer Widom, Prentice Hall, 2007

Chapter 7.1, 7.2, 7.3 – SQL Constraints

Chapter 8.3 – 8.4 - Indexes



Database Systems - A Practical Approach to Design, Implementation, and Management (4th edition) by Thomas Connolly and Carolyn Begg, Addison-Wesley, 2004

Chapter 17.3 & C5