

COL362/632 Introduction to Database Management Systems

Database Systems – Indexing

Kaustubh Beedkar

Department of Computer Science and Engineering
Indian Institute of Technology Delhi



Motivation

Contents

PART FIVE ■ STORAGE MANAGEMENT AND INDEXING

Chapter 12 Physical Storage Systems

12.1 Overview of Physical Storage Media	559	12.6 Disk/Block Access	577
12.2 Storage Interfaces	562	12.7 Summary	580
12.3 Magnetic Disks	563	Exercises	582
12.4 Flash Memory	567	Further Reading	584
12.5 RAID	570		

Chapter 13 Data Storage Structures

13.1 Database Storage Architecture	587	13.7 Storage Organization in Main-Memory	
13.2 File Organization	588	Databases	615
13.3 Organization of Records in Files	595	13.8 Summary	617
13.4 Data-Dictionary Storage	602	Exercises	619
13.5 Database Buffer	604	Further Reading	621
13.6 Column-Oriented Storage	611		

Chapter 14 Indexing

14.1 Basic Concepts	623	14.8 Write-Optimized Index Structures	665
14.2 Ordered Indices	625	14.9 Bitmap Indices	670
14.3 B*-Tree Index Files	634	14.10 Indexing of Spatial and Temporal Data	672
14.4 B*-Tree Extensions	650	14.11 Summary	677
14.5 Hash Indices	658	Exercises	679
14.6 Multiple-Key Access	661	Further Reading	683
14.7 Creation of Indices	664		

PART SIX ■ QUERY PROCESSING AND OPTIMIZATION

Chapter 15 Query Processing

15.1 Overview	689	15.7 Evaluation of Expressions	724
15.2 Measures of Query Cost	692	15.8 Query Processing in Memory	731
15.3 Selection Operation	695	15.9 Summary	734
15.4 Sorting	701	Exercises	736
15.5 Join Operation	704	Further Reading	740
15.6 Other Operations	719		

Contents

PART FIVE ■ STORAGE MANAGEMENT AND INDEXING

Chapter 12 Physical Storage Systems

- 12.1 Overview of Physical Storage Media 559
- 12.2 Storage Interfaces 562
- 12.3 Magnetic Disks 563
 - 12.3.1 Summary 580
 - 12.3.2 Exercises 582
- 12.4 Flash Memory 567
 - 12.4.1 Further Reading 584
- 12.5 RAID 570

Chapter 13 Data Storage Structures

- 13.1 Database Storage Architecture 587
- 13.2 File Organization 588
- 13.3 Organization of Records in Files 595
- 13.4 Data-Dictionary Storage 602
- 13.5 Database Buffer 604
- 13.6 Column-Oriented Storage 611
- 13.7 Storage Organization in Main-Memory
 - 13.7.1 Databases 615
 - 13.7.2 Summary 617
 - 13.7.3 Exercises 619
 - 13.7.4 Further Reading 621

Chapter 14 Indexing

- 14.1 Basic Concepts 623
- 14.2 Ordered Indices 625
- 14.3 B*-Tree Index Files 634
- 14.4 B*-Tree Extensions 650
- 14.5 Hash Indices 658
- 14.6 Multiple-Key Access 661
- 14.7 Creation of Indices 664
- 14.8 Write-Optimized Index Structures 665
- 14.9 Bitmap Indices 670
- 14.10 Indexing of Spatial and Temporal Data 672
- 14.11 Summary 677
 - 14.11.1 Exercises 679
 - 14.11.2 Further Reading 683

PART SIX ■ QUERY PROCESSING AND OPTIMIZATION

Chapter 15 Query Processing

- 15.1 Overview 689
- 15.2 Measures of Query Cost 692
- 15.3 Selection Operation 695
- 15.4 Sorting 701
- 15.5 Join Operation 704
- 15.6 Other Operations 719
- 15.7 Evaluation of Expressions 724
- 15.8 Query Processing in Memory 731
- 15.9 Summary 734
 - 15.9.1 Exercises 736
 - 15.9.2 Further Reading 740

Index

- aborted transactions, 805–807, 819–820
 - abstraction, 2, 9–12, 15
 - acceptors, 1148, 1152
 - accessing data. *See also* security
 - from application programs, 16–17
 - concurrent-access anomalies, 7
 - difficulties in, 6
 - indices for, 19
 - recovery systems and, 910–912
 - types of access, 15
 - access paths, 695
 - access time
 - indices and, 624, 627–628
 - query processing and, 692
 - storage and, 561, 566, 567, 578
 - access types, 624
 - active nonces, 1271
 - ACID properties. *See* atomicity; consistency; durability; isolation
 - Active Server Page (ASP), 405
 - active transactions, 806
 - ActiveX Data Objects (ADO), 1239
 - adaptive lock granularity, 969–970
 - add constraint, 146
 - ADO (ActiveX Data Objects), 1239
 - ADO.NET, 184, 1239
 - Advanced Encryption Standard (AES), 448, 449
 - advanced SQL, 183–231
 - accessing from programming languages, 183–198
 - aggregate features, 219–231
 - embedded, 197–198
 - functions and procedures, 198–206
 - JDBC and, 184–193
 - ODBC and, 194–197
 - Python and, 193–194
 - triggers and, 206–213
 - advertisement data, 469
 - AES (Advanced Encryption Standard), 448, 449
 - after triggers, 210
 - aggregate functions, 91–96
 - basic, 91–92
 - with Boolean values, 96
 - defined, 91
 - with grouping, 92–95
 - having clause, 95–96
 - with null values, 96
 - aggregation
 - defined, 277
 - entity-relationship (E-R) model and, 276–277
 - intraoperation parallelism and, 1049
 - on multidimensional data, 527–532
 - partial, 1049
 - pivoting and, 226–227, 530
 - query optimization and, 764
 - query processing and, 723
 - ranking and, 219–223
 - representation of, 279
 - rollup and cube, 227–231
 - skew and, 1049–1050
 - of transactions, 1278
 - view maintenance and, 781–782
 - windowing and, 223–226
- aggregation operation, 57
 - aggregation switch, 977
 - airlines, database applications for, 3
 - Ajax, 423–426, 1015
 - algebraic operations. *See* relational algebra
 - aliases, 81, 336, 1242
 - all construct, 100
 - alter table, 71, 146
 - alter trigger, 210
 - alter type, 159
 - Amid's law, 974
 - American National Standards Institute (ANSI), 65, 1237
 - analysis pass, 944
 - analytics. *See* data analytics
 - and connective, 74
 - and operation, 89–90
 - anonymity, 1252, 1253, 1258, 1259
 - ANSI (American National Standards Institute), 65, 1237
 - anticipatory standards, 1237
 - anti-join operation, 108, 776



Motivation

Contents

PART FIVE ■ STORAGE MANAGEMENT AND INDEXING

Chapter 12 Physical Storage Systems

- 12.1 Overview of Physical Storage Media 559
- 12.2 Storage Interfaces 562
- 12.3 Magnetic Disks 563
 - Exercises 563
- 12.4 Flash Memory 567
- 12.5 RAID 570

Chapter 13 Data Storage Structures

- 13.1 Database Storage Architecture 587
- 13.2 File Organization 588
- 13.3 Organization of Records in Files 595
- 13.4 Data-Dictionary Storage 602
- 13.5 Database Buffer 604
- 13.6 Column-Oriented Storage 611

Chapter 14 Indexing

- 14.1 Basic Concepts 623
- 14.2 Ordered Indices 625
- 14.3 B*-Tree Index Files 634
- 14.4 B*-Tree Extensions 650
- 14.5 Hash Indices 658
- 14.6 Multiple-Key Access 661
- 14.7 Creation of Indices 664

PART SIX ■ QUERY PROCESSING AND OPTIMIZATION

Chapter 15 Query Processing

- 15.1 Overview 689
- 15.2 Measures of Query Cost 692
- 15.3 Selection Operation 695
- 15.4 Sorting 701
- 15.5 Join Operation 704
- 15.6 Other Operations 719

Index



1239
ADO.NET, 184, 1239

pivoting and, 226–227, 530
query optimization and, 764

anticipatory standards, 1237
anti-join operation, 108, 776

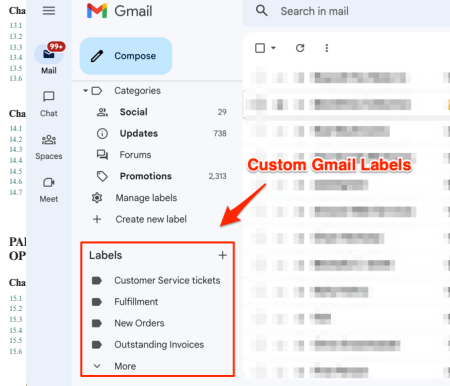
Motivation

Contents

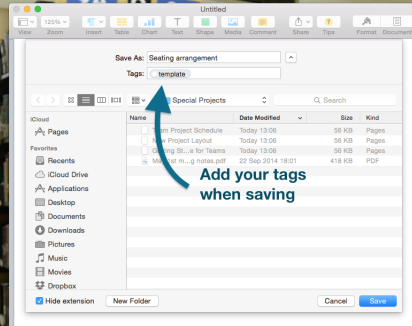
PART FIVE ■ STORAGE MANAGEMENT AND INDEXING

Chapter 12 Physical Storage Systems

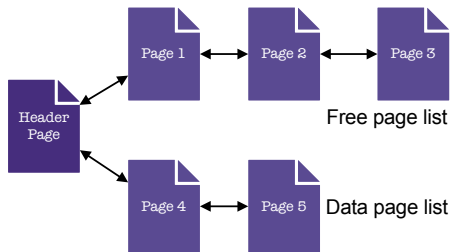
12.1 Overview of Physical Storage Media	559	12.6 DiskBlock A
12.2 Storage Interfaces	562	12.7 Summary
12.3 Magnetic Disks	563	Exercises
12.4 Flash Memory	567	Exercises
12.5		



Index

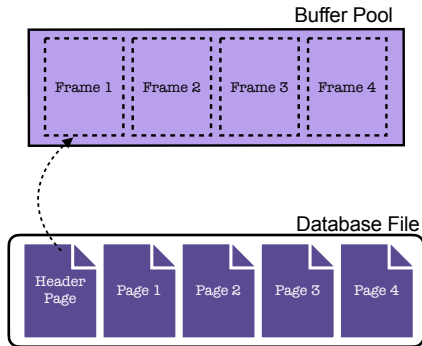


Accessing Records in Databases

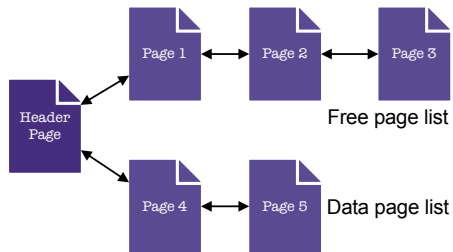


Recall: Heap File Organization

- ▶ Good for scanning all records

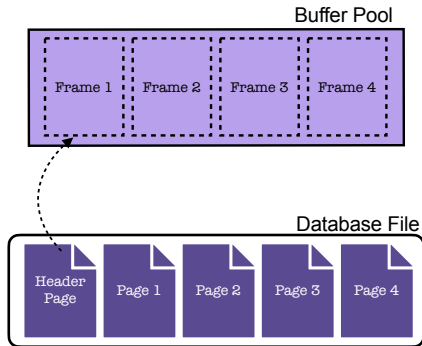


Accessing Records in Databases

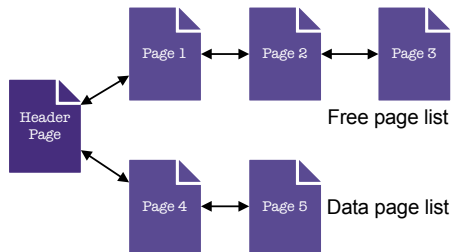


Recall: Heap File Organization

- ▶ Good for scanning all records
- ▶ But, what if we are looking for “certain” records
 - `select * from Order where return_status = 'I'`
 - `select name from Product where price < 100`

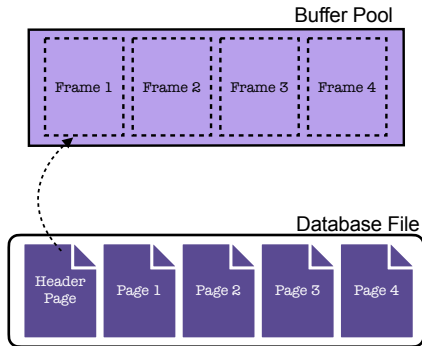


Accessing Records in Databases



Recall: Heap File Organization

- ▶ Good for scanning all records
- ▶ But, what if we are looking for “certain” records
 - `select * from Order where return_status = 'I'`
 - `select name from Product where price < 100`
- ▶ How to **optimally** retrieve records?



Accessing Records in Database



Accessing Records in Database



Accessing Records in Database

- ▶ Assume block is b (KB), each record is r (KB), and n records



Accessing Records in Database

- ▶ Assume block is b (KB), each record is r (KB), and n records
- ▶ $O(\frac{nr}{b})$ for sequential scan



Accessing Records in Database

- ▶ Assume block is b (KB), each record is r (KB), and n records
- ▶ $O(\frac{nr}{b})$ for sequential scan

- ▶ Improve access by
 - Keeping sorted files
 - $O(\log_2 \frac{nr}{b})$ for binary search



Accessing Records in Database

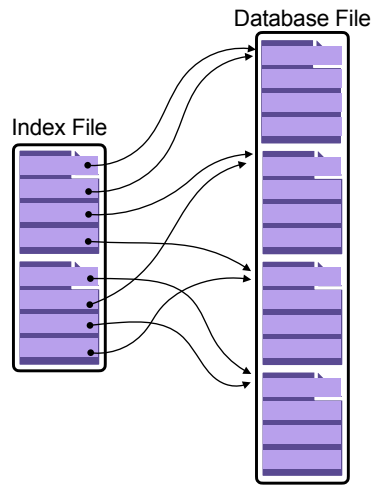
- ▶ Assume block is b (KB), each record is r (KB), and n records
- ▶ $O(\frac{nr}{b})$ for sequential scan
- ▶ Improve access by
 - Keeping sorted files
 - $O(\log_2 \frac{nr}{b})$ for binary search
 - **Using Indexes!**
 - B+ trees
 - Hash Index
 - Bit Maps
 - Bloom Filters
 - ...



Indexing - basics

Index

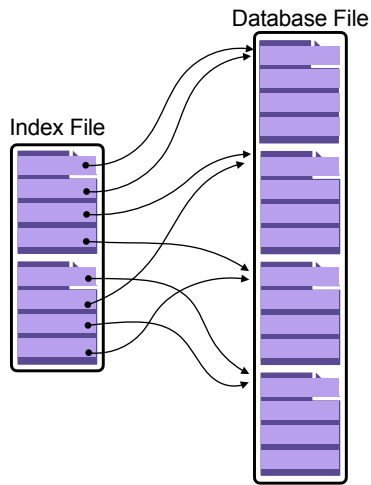
- ▶ Data structure that organizes records of a file
- ▶ Enables efficiently searching records using **search key**



Indexing - basics

Index

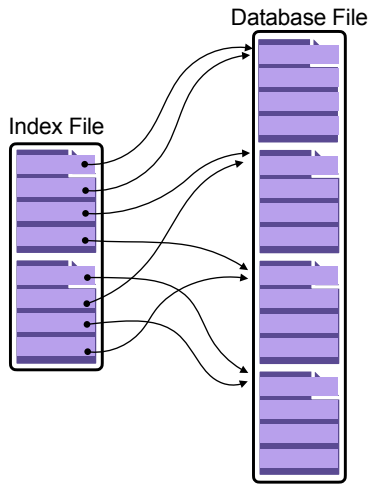
- ▶ Data structure that organizes records of a file
- ▶ Enables efficiently searching records using **search key**
- ▶ **Note:** Index is also a database file!



Indexing - basics

Index

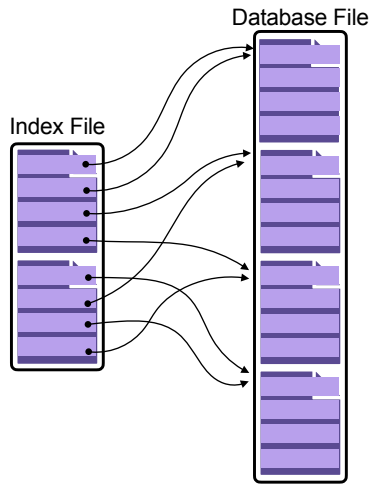
- ▶ Data structure that organizes records of a file
- ▶ Enables efficiently searching records using **search key**
- ▶ **Note:** Index is also a database file!
- ▶ Is a collection of **data entries**
 - An entry in index is different from record
 - But has information to locate records



Indexing - basics

Index

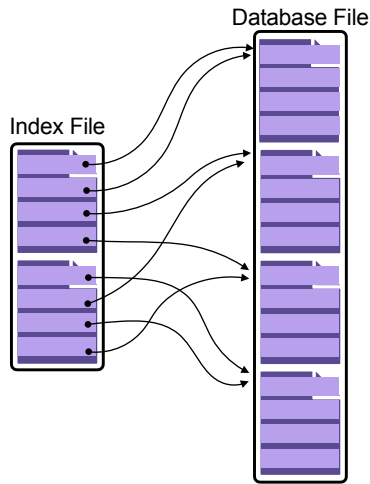
- ▶ Data structure that organizes records of a file
- ▶ Enables efficiently searching records using **search key**
- ▶ **Note:** Index is also a database file!
- ▶ Is a collection of **data entries**
 - An entry in index is different from record
 - But has information to locate records
- ▶ Two basic types
 1. Ordered Index – Search keys are stored in sorted order
 2. Hash Index – Search keys are distributed uniformly across *buckets* using a hash function



Indexing - basics

Index

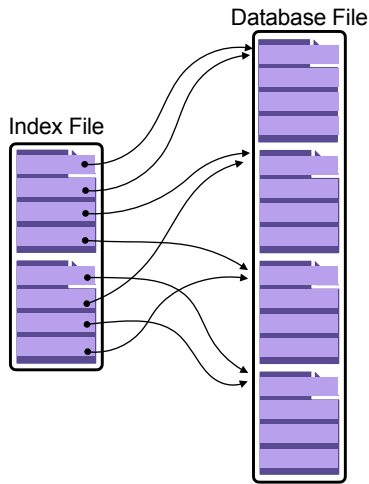
- ▶ Data structure that organizes records of a file
- ▶ Enables efficiently searching records using **search key**
- ▶ **Note:** Index is also a database file!
- ▶ Is a collection of **data entries**
 - An entry in index is different from record
 - But has information to locate records
- ▶ Two basic types
 1. Ordered Index – Search keys are stored in sorted order
 2. Hash Index – Search keys are distributed uniformly across *buckets* using a hash function
- ▶ A database file can have many indices



Indexing - basics

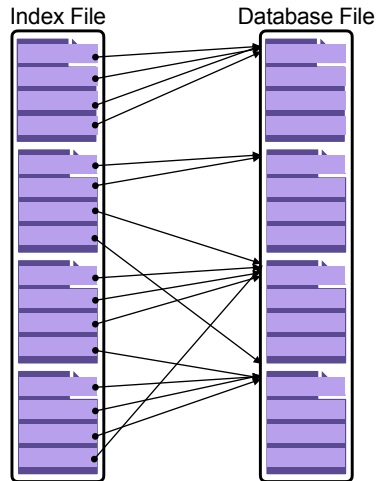
Index

- ▶ Data structure that organizes records of a file
- ▶ Enables efficiently searching records using **search key**
- ▶ **Note:** Index is also a database file!
- ▶ Is a collection of **data entries**
 - An entry in index is different from record
 - But has information to locate records
- ▶ Two basic types
 1. Ordered Index – Search keys are stored in sorted order
 2. Hash Index – Search keys are distributed uniformly across *buckets* using a hash function
- ▶ A database file can have many indices
- ▶ Updating file also requires updating the index



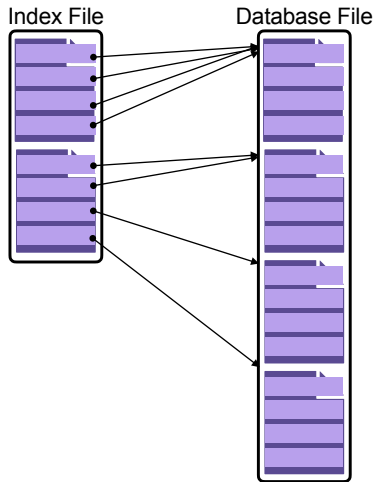
Dense Index

- ▶ Data entries based on sorted ordering of values
- ▶ There is an entry in the index **for every** search-key value in the file



Sparse Index

- ▶ Data entries based on sorted ordering of values
- ▶ There is an entry in the index **for only some** search-key value in the file
- ▶ **Note** Can only be used if the table is stored in sorted order of the search key



Primary Index and Secondary Index

Primary Index

- ▶ Search key contains the primary key
- ▶ No duplicate search keys
- ▶ A table can have only one primary index

Secondary Index

- ▶ Any index other than primary index

Note: Index on candidate key is called a **unique index**

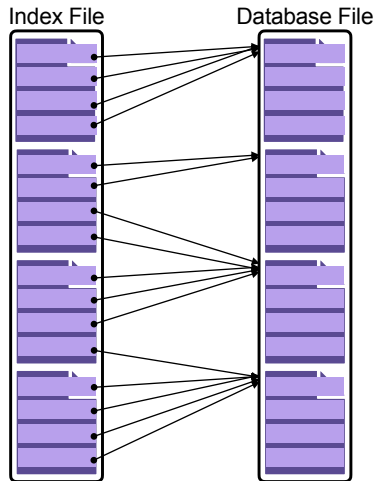
Clustered Index

- ▶ Search key also defines the sequential order of the file
- ▶ Search key often the primary key
- ▶ Can be both dense and sparse

Dense Clustered Index

- ▶ Data entry only points to block containing the first record

Note: Sparse Index \implies Clustered Index



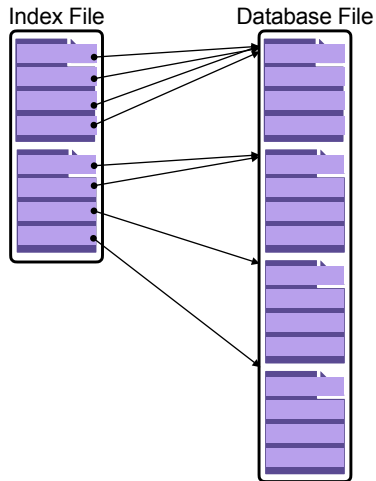
Clustered Index

- ▶ Search key also defines the sequential order of the file
- ▶ Search key often the primary key
- ▶ Can be both dense and sparse

Dense Clustered Index

- ▶ Data entry only points to block containing the first record

Note: Sparse Index \implies Clustered Index

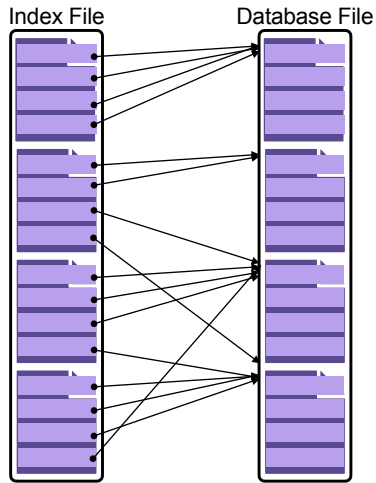


Non-clustered Index

- ▶ Search key specifies an order different from the sequential order of the file

Dense Non-clustered Index

- ▶ Data entry must store block pointers to all records with the same search key value

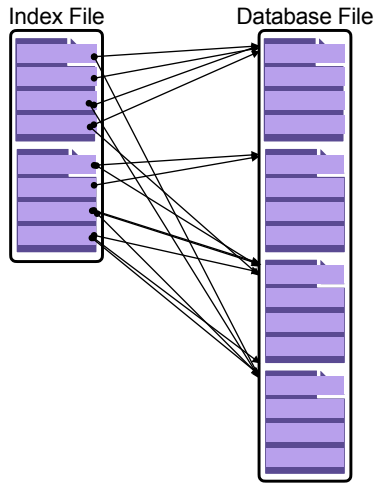


Non-clustered Index

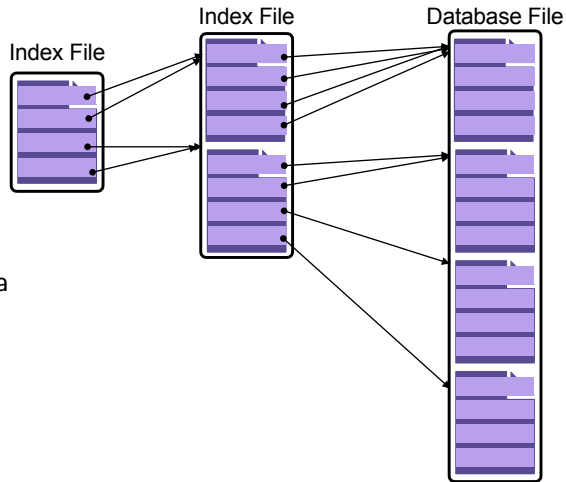
- ▶ Search key specifies an order different from the sequential order of the file

Dense Non-clustered Index

- ▶ Data entry must store block pointers to all records with the same search key value

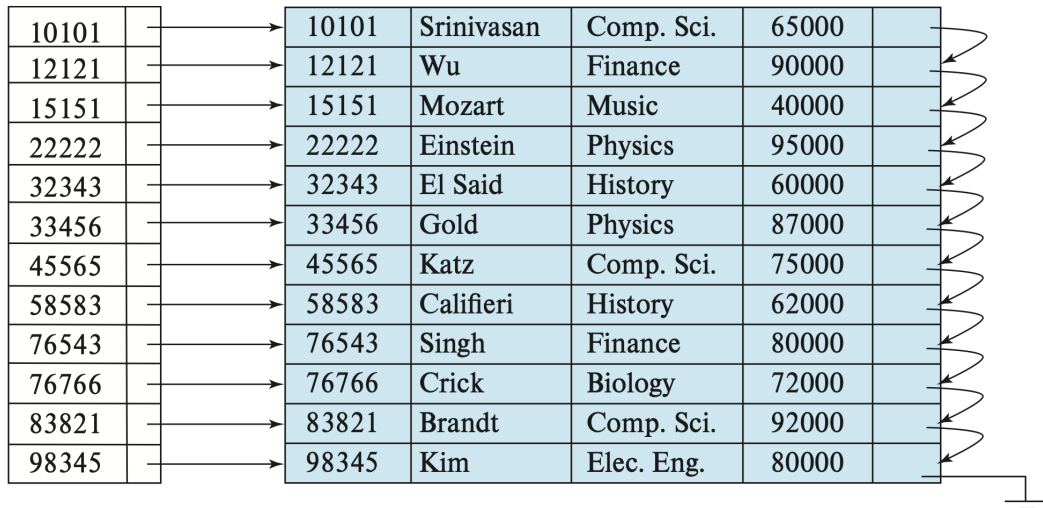


Multi-level Index

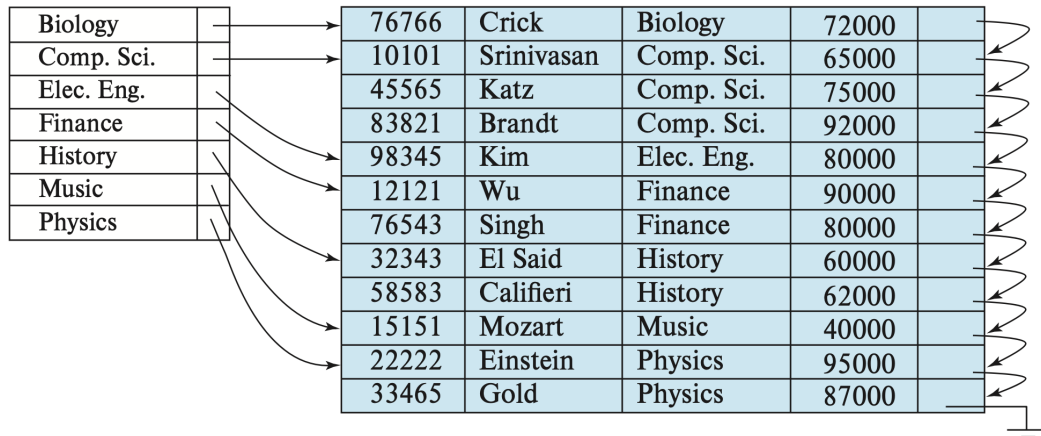


- ▶ Sometimes, index may not fit in memory
- ▶ Treat index as sequential file and create a **sparse outer** index
- ▶ Can be repeated

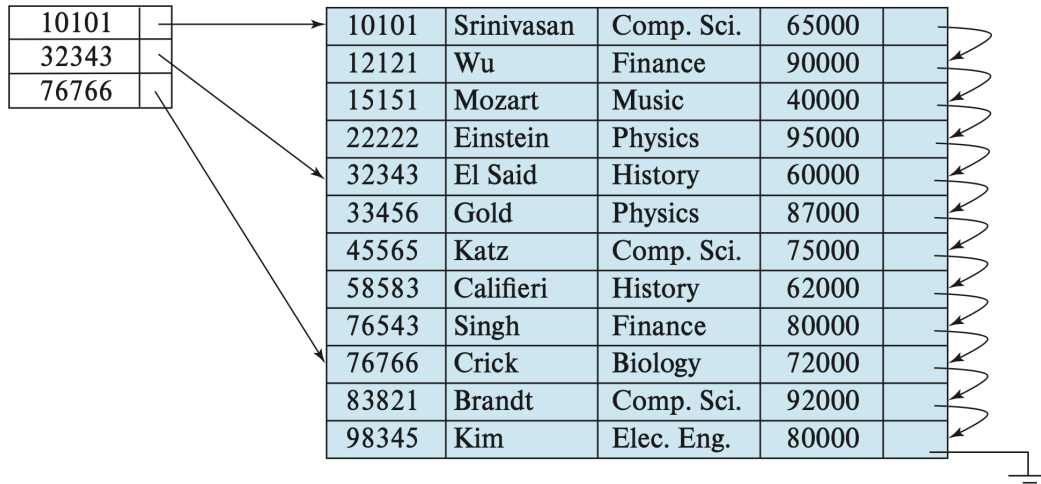
Examples (from Book B1)



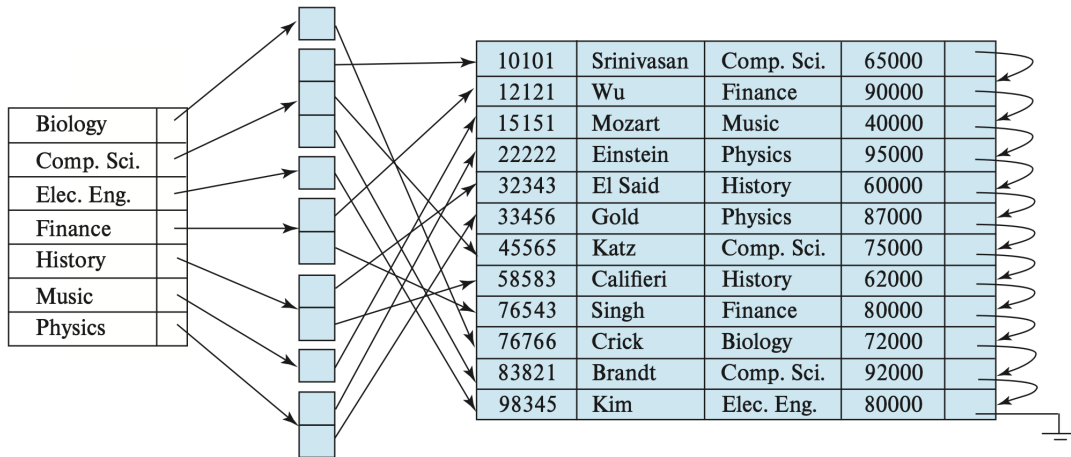
Examples (from Book B1)



Examples (from Book B1)



Examples (from Book B1)



Updating Index – Insertions

Dense Index

- ▶ If search key !in index
add a data entry at appropriate position
- ▶ Else If non-clustering index, add pointer to data entry
- ▶ Else place the **record** appropriately

Sparse Index

- ▶ If new block, add new data entry
- ▶ If new record has least search-key value, update the data entry!

Updating Index – Deletions

Dense Index

- ▶ If it was the only record, delete the index entry
- ▶ Else, for non-clustering index, delete the pointer
- ▶ Else, update the data entry by pointing the next record

Sparse Index

- ▶ If the index does not contain data entry with search key value, do nothing; Else,
- ▶ If it was the only record, update the index entry with next search key value
- ▶ If next search key value present in index, delete the index entry
- ▶ Else, if index entry points to recorded being deleted, update the index to point to the next record with the same search key value (record can be in another block)

- ▶ Find out which is the largest table
- ▶ Check if it has index
- ▶ create index if there is none for some attribute
- ▶ benchmark multiple queries with and without index

Creating/dropping an index in PostgreSQL

- ▶ `CREATE UNIQUE INDEX title_idx ON films (title)`
- ▶ `CREATE INDEX title_idx ON films (title) WITH (deduplicate_items = off)`
- ▶ More examples on
<https://www.postgresql.org/docs/current/sql-createindex.html>
- ▶ `drop index index_name`

Which one is clustered and which is non-clustered?

Contents

PART FIVE ■ STORAGE MANAGEMENT AND INDEXING

Chapter 12 Physical Storage Systems

- 12.1 Overview of Physical Storage Media 559
- 12.2 Storage Interfaces 562
- 12.3 Magnetic Disks 563
- 12.4 Flash Memory 567
- 12.5 RAID 570
- 12.6 Disk-Block Access 577
- 12.7 Summary 580
- Exercises 582
- Further Reading 584

Chapter 13 Data Storage Structures

- 13.1 Database Storage Architecture 587
- 13.2 File Organization 588
- 13.3 Organization of Records in Files 595
- 13.4 Data-Dictionary Storage 602
- 13.5 Database Buffer 604
- 13.6 Column-Oriented Storage 611
- 13.7 Storage Organization in Main-Memory Databases 615
- 13.8 Summary 617
- Exercises 619
- Further Reading 621

Chapter 14 Indexing

- 14.1 Basic Concepts 623
- 14.2 Ordered Indices 625
- 14.3 B*-Tree Index Files 634
- 14.4 B*-Tree Extensions 650
- 14.5 Hash Indices 658
- 14.6 Multiple-Key Access 661
- 14.7 Creation of Indices 664
- 14.8 Write-Optimized Index Structures 665
- 14.9 Bitmap Indices 670
- 14.10 Indexing of Spatial and Temporal Data 672
- 14.11 Summary 677
- Exercises 679
- Further Reading 683

PART SIX ■ QUERY PROCESSING AND OPTIMIZATION

Chapter 15 Query Processing

- 15.1 Overview 689
- 15.2 Measures of Query Cost 692
- 15.3 Selection Operation 695
- 15.4 Sorting 701
- 15.5 Join Operation 704
- 15.6 Other Operations 719
- 15.7 Evaluation of Expressions 724
- 15.8 Query Processing in Memory 731
- 15.9 Summary 734
- Exercises 736
- Further Reading 740

Index

- aborted transactions, 805–807, 819–820
 - abstraction, 2, 9–12, 15
 - acceptors, 1148, 1152
 - accessing data. *See also* security from application programs, 16–17 concurrent-access anomalies, 7 difficulties in, 6 indices for, 19 recovery systems and, 910–912 types of access, 15
 - access paths, 695
 - access time
 - indices and, 624, 627–628
 - query processing and, 692
 - storage and, 561, 566, 567, 578
 - access types, 624
 - account nonces, 1271
 - ACID properties. *See* atomicity; consistency; durability; isolation
 - Active Server Page (ASP), 405
 - active transactions, 806
 - ActiveX Data Objects (ADO), 1239
 - adaptive lock granularity, 969–970
 - add constraint, 146
 - ADO (ActiveX Data Objects), 1239
 - ADO.NET, 184, 1239
 - Advanced Encryption Standard (AES), 448, 449
 - advanced SQL, 183–231
 - accessing from programming languages, 183–198
 - aggregate features, 219–231
 - embedded, 197–198
 - functions and procedures, 198–206
 - JDBC and, 184–193
 - ODBC and, 194–197
 - Python and, 193–194
 - triggers and, 206–213
 - advertisement data, 469
 - AES (Advanced Encryption Standard), 448, 449
 - after triggers, 210
 - aggregate functions, 91–96
 - basic, 91–92
 - with Boolean values, 96
 - defined, 91
 - with grouping, 92–95
 - having clause, 95–96
 - with null values, 96
 - aggregation
 - defined, 277
 - entity-relationship (E-R) model and, 276–277
 - intraoperation parallelism and, 1049
 - on multidimensional data, 527–532
 - partial, 1049
 - pivoting and, 226–227, 530
 - query optimization and, 764
 - query processing and, 723
 - ranking and, 219–223
 - representation of, 279
 - rollup and cube, 227–231
 - skew and, 1049–1050
 - of transactions, 1278
 - view maintenance and, 781–782
 - windowing and, 223–226
- aggregation operation, 57
 - aggregation switch, 977
 - airlines, database applications for, 3
 - Ajax, 423–426, 1015
 - algebraic operations. *See* relational algebra
 - aliases, 81, 336, 1242
 - all construct, 100
 - alter table, 71, 146
 - alter trigger, 210
 - alter type, 159
 - Amdahl's law, 974
 - American National Standards Institute (ANSI), 65, 1237
 - analysis pass, 944
 - analytics. *See* data analytics
 - and connective, 74
 - and operation, 89–90
 - anonymity, 1252, 1253, 1258, 1259
 - ANSI (American National Standards Institute), 65, 1237
 - anticipatory standards, 1237
 - anti-join operation, 108, 776
