UNIVERSITAS INDONESIA

FAKULTAS ILMU KOMPUTER

Slide 10

Intro to Indexing + SQL Indexing

CSF2600700 - BASIS DATA

SEMESTER GANJIL 2017/2018

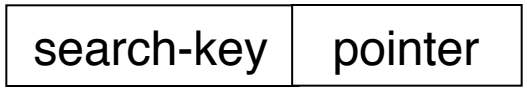
This slide is a modification to supplementary slide of "Database System", 6th edition, Elmasri/ Navathe, 2011: **Chapter 17 Database File Indexing Techiniques**

Index

- Indexing is a way of sorting a number of records on multiple fields
- Indexes are special lookup tables that the database search engine can use to speed up data retrieval, without reading the whole table
- Simply, an index is a pointer to data in a table.
- They are based upon one or more columns but stored as a separate entity
- An index in a database is very similar to an index in the back of a book.

Index

- An index file consists of records (called **index entries**) of the form



  - **Search Key**: attribute to set of attributes used to look up records in a file.
  - **Pointer**: pointer to the record or the block
- Index files are typically much smaller than the original file
- An index is an auxiliary file that makes it more efficient to search for a record in the data file .

Index

- An index helps speed up SELECT queries and WHERE clauses, but it slows down data input, with UPDATE and INSERT statements.
- Indexes can be created or dropped with no effect on the data.

Types of Index

- Primary Indexes vs. Secondary Indexes
  - Primary index**:** in a sorted organization file, the index whose search key is the same as the sequential order of the file.
  - Secondary index: an index whose search key is different from the sequential order of the file. Secondary indexes provide a mechanism for specifying an additional key for a base relation that can be used to retrieve data more efficiently

- Sparse vs. Dense Indexes
  - Sparse index: index on attribute for sorting the file, only one index entry per **block of data record**.
  - Dense index: one index entry per **data record**.

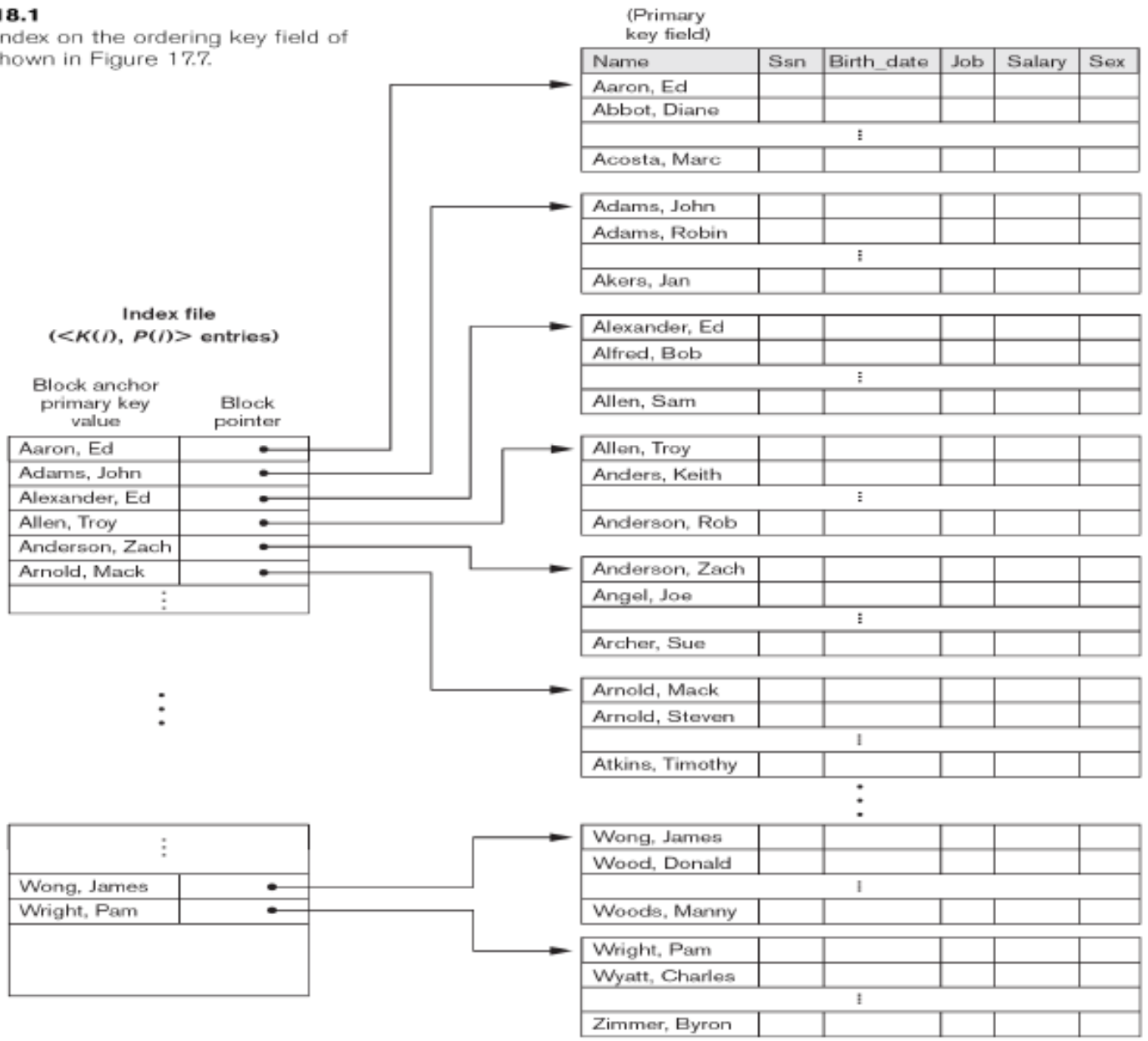
Types of Index

- Clustering Indexes
  - If the ordering field is not UNIQUE or non-key, such as DEPT_NUMBER in EMP table, we can create a clustering index.
  - In the index, one index entry for each distinct value

- Single-Level vs. Multi-Level Indexes
  - One index per file
  - Multiple index per file. Index of index.

Example of INDEX

Figure 18.1
Primary index on the ordering key field of the file shown in Figure 17.7.



Types of Index

Table 18.1 Types of Indexes Based on the Properties of the Indexing Field

	Index Field Used for Physical Ordering of the File	Index Field Not Used for Physical Ordering of the File
Indexing field is key	Primary index	Secondary index (Key)
Indexing field is nonkey	Clustering index	Secondary index (NonKey)

Choose indexes – Guidelines for choosing 'wish-list'

1. Do not index small relations.
2. Index PK of a relation if it is not a key of the file organization.
3. Add secondary index to a FK if it is frequently accessed.
4. Add secondary index to any attribute heavily used as a secondary key.
5. Add secondary index on attributes involved in: selection or join criteria; ORDER BY; GROUP BY; and other operations involving sorting (such as UNION or DISTINCT).

Choose indexes – Guidelines for choosing 'wish-list'

6. Add secondary index on attributes involved in built-in functions.
7. Avoid indexing an attribute or relation that is frequently updated.
8. Avoid indexing an attribute if the query will retrieve a significant proportion of the relation.
9. Avoid indexing attributes that consist of long character strings.

UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

SQL Indexing

INDEX

Creating an index involves the CREATE INDEX statement,

- Which allows you to name the index,
- To specify the table and which column(s) to index,
- And to indicate whether the index is in ascending or descending order

The basic syntax:

- **`CREATE INDEX index_name ON table_name (column_name);`**
- Ex: **`CREATE INDEX Emp_idx ON EMPLOYEE (Ssn)`**

INDEX

Multiple column index

```
CREATE INDEX index_name ON
table_name (column_name1,
column_name2);
```

Ex: `CREATE INDEX WORKS_ON_Idx ON WORKS_ON (pno, hours)`

Which column to choose is based on frequently queried column in WHERE clause

DROP INDEX

```
DROP INDEX index_name;
```

Care should be taken when dropping an index because performance may be slowed or improved

WHEN NOT TO USE INDEX

Although indexes are intended to enhance a database's performance, there are times when they should be avoided. The following guidelines indicate when the use of an index should be reconsidered:

- Indexes should not be used on small tables.
- Tables that have frequent, large batch update or insert operations.
- Indexes should not be used on columns that contain a high number of NULL values.
- Columns that are frequently manipulated should not be indexed.