Database Management Systems (CSN-351)
Recovery and File Structure

BTech 3rd Year (CS) + Minor

Instructor: **R**anita **B**iswas
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand - 247 667, India

Failure Classification

Transaction failure due to **Logical Error**

Failure Classification

Transaction failure due to **Logical Error**

Transaction failure due to **System Error**

Failure Classification

Transaction failure due to **Logical Error**

Transaction failure due to **System Error**

System Crash

Failure Classification

Transaction failure due to **Logical Error**

Transaction failure due to **System Error**

System Crash

Disk Failure

Transactions and System Log

T_0:	T_1:
read(A);	
$A := A - 50$;	
write(A);	
read(B);	read(C);
$B := B + 50$;	$C := C - 100$;
write(B).	write(C).

Transactions and System Log

T_0: read(A);		$<T_0$ start$>$
$A := A - 50$;		$<T_0, A, 1000, 950>$
write(A);		$<T_0, B, 2000, 2050>$
read(B);		$<T_0$ commit$>$
$B := B + 50$;	T_1: read(C);	$<T_1$ start$>$
write(B).	$C := C - 100$;	$<T_1, C, 700, 600>$
	write(C).	$<T_1$ commit$>$

Storage

Organization of memory devices: Primary, Secondary, Tertiary

Storage

Organization of memory devices: Primary, Secondary, Tertiary

Databases are stored in Disk because

Storage

Organization of memory devices: Primary, Secondary, Tertiary

Databases are stored in Disk because

- Databases are large

Storage

Organization of memory devices: Primary, Secondary, Tertiary

Databases are stored in Disk because

- Databases are large
- Loss of data from disk is less frequent

Storage

Organization of memory devices: Primary, Secondary, Tertiary

Databases are stored in Disk because

- Databases are large
- Loss of data from disk is less frequent
- Cost-effective

Storage

Organization of memory devices: Primary, Secondary, Tertiary

Databases are stored in Disk because

- Databases are large
- Loss of data from disk is less frequent
- Cost-effective
- Only a small portion of the database is required at a time

File Structure

Organization of database in a storage device:

- Blocks, Files, Records, Values

File Structure

Organization of database in a storage device:

- Blocks, Files, Records, Values
- **Blocking Factor:** Average number of records that can be stored in a disk block

File Structure

Organization of database in a storage device:

- Blocks, Files, Records, Values
- **Blocking Factor:** Average number of records that can be stored in a disk block
- **Record Type:** A collection of field names and their corresponding data types

File Structure

Organization of database in a storage device:

- Blocks, Files, Records, Values
- **Blocking Factor:** Average number of records that can be stored in a disk block
- **Record Type:** A collection of field names and their corresponding data types
- BLOB and CLOB

Records

Records → Fixed-length records and Variable-length records.

Records

Records → Fixed-length records and Variable-length records.

Variable-length records appear when using

Records

Records → Fixed-length records and Variable-length records.

Variable-length records appear when using

- Different record types

Records

Records → Fixed-length records and Variable-length records.

Variable-length records appear when using

- Different record types
- Variable-length fields (`varchar`) in same record type

Records

Records → Fixed-length records and Variable-length records.

Variable-length records appear when using

- Different record types
- Variable-length fields (`varchar`) in same record type
- Multi-valued attributes

Records

Records → Fixed-length records and Variable-length records.

Variable-length records appear when using

- Different record types
- Variable-length fields (`varchar`) in same record type
- Multi-valued attributes
- Optional fields

Blocking Factor

Number of records which can be stored in a block is called blocking factor (bfr).

$$B = \text{Block size}$$
$$R = \text{Record size}$$

Blocking Factor

Number of records which can be stored in a block is called blocking factor (bfr).

$$B = \text{Block size}$$
$$R = \text{Record size}$$
$$bfr = \left\lfloor \frac{B}{R} \right\rfloor$$

Storing Strategy

Span Strategy: Records can be stored in more than one block. One record can span in multiple blocks.

Storing Strategy

Span Strategy: Records can be stored in more than one block. One record can span in multiple blocks.

For a variable-length record using span organization, if

$$bfr = \text{Average blocking factor}$$
$$r = \text{Number of records}$$

Then, number of blocks (b) required

Storing Strategy

Span Strategy: Records can be stored in more than one block. One record can span in multiple blocks.

For a variable-length record using span organization, if

$$bfr = \text{Average blocking factor}$$
$$r = \text{Number of records}$$

Then, number of blocks (b) required

$$b = \left\lceil \frac{r}{bfr} \right\rceil$$

Storing Strategy

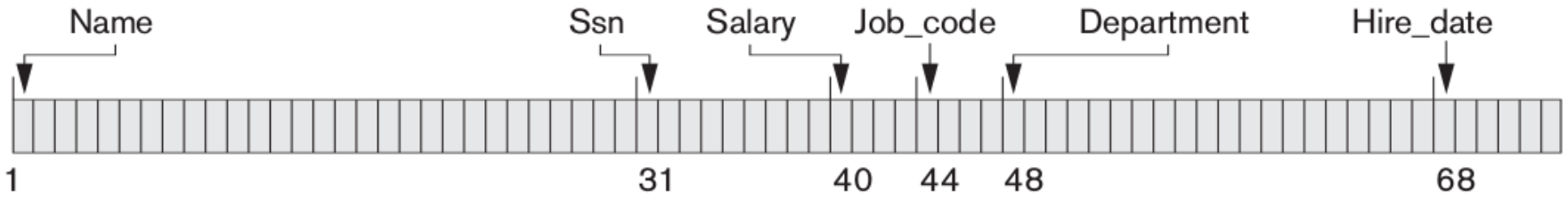
Unspan Strategy: Records are not allowed to cross block boundaries. This is used with fixed-length records having $B > R$ always.

Storing Strategy

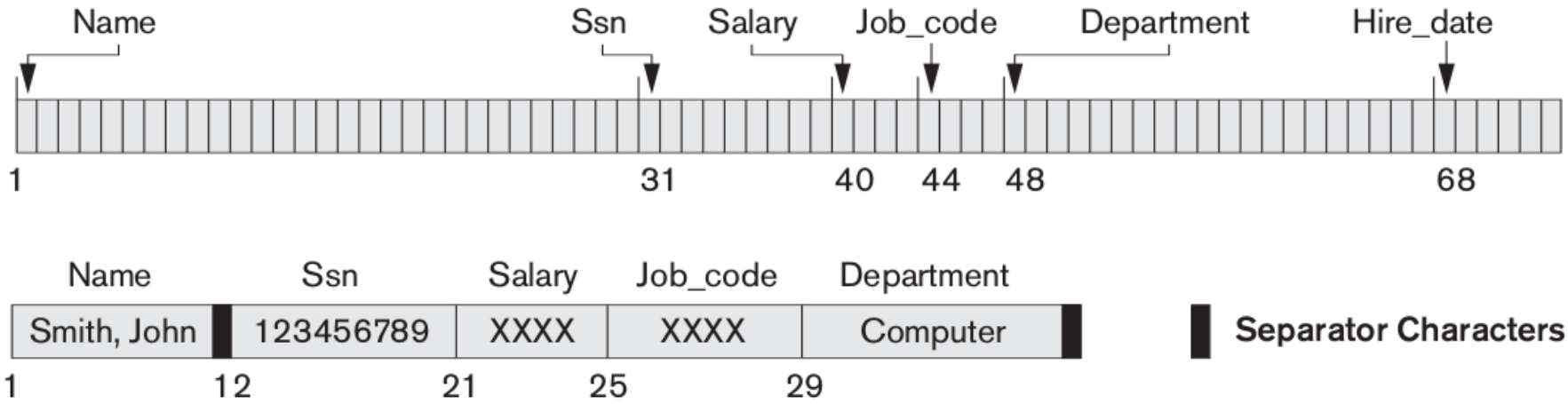
Unspan Strategy: Records are not allowed to cross block boundaries. This is used with fixed-length records having $B > R$ always.

Variable-length records are usually stored with **spanned strategy** which additionally requires to store separator characters and field types.

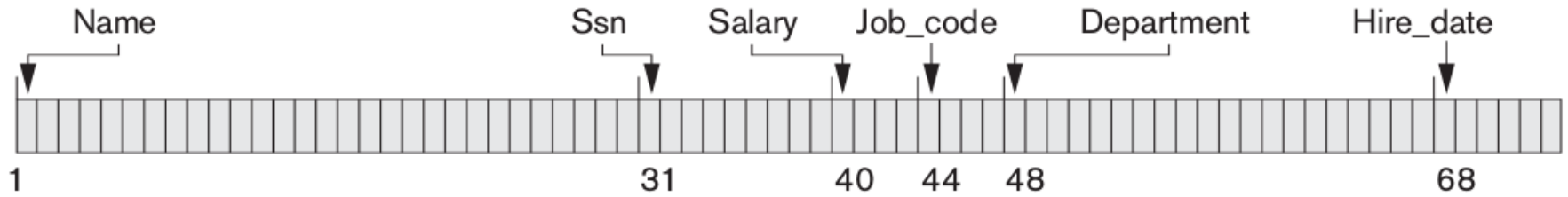
Example

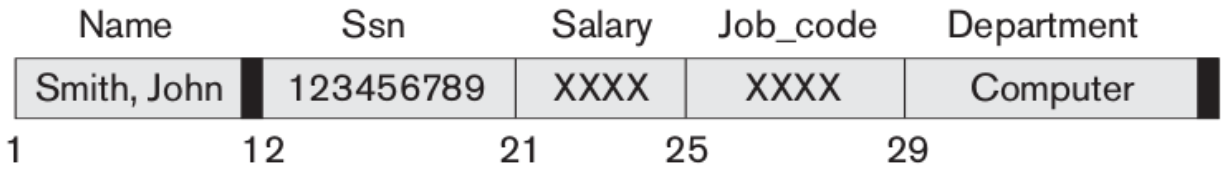


Example


Name
Ssn
Salary
Job_code
Department
Hire_date
1
31
40
44
48
68
Name
Ssn
Salary
Job_code
Department
Smith, John
123456789
XXXX
XXXX
Computer
1
12
21
25
29
Separator Characters

Example





Separator Characters

Separator Characters

= Separates field name from field value

Separates fields

Terminates record

Organization of Records in Files

Heap: a record can be placed anywhere in the file where there is space.

Organization of Records in Files

Heap: a record can be placed anywhere in the file where there is space.

Sequential: store records in sequential order, based on the value of the search key of each record.

Organization of Records in Files

Heap: a record can be placed anywhere in the file where there is space.

Sequential: store records in sequential order, based on the value of the search key of each record.

Hashing: a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed.

Organization of Records in Files

Heap: a record can be placed anywhere in the file where there is space.

Sequential: store records in sequential order, based on the value of the search key of each record.

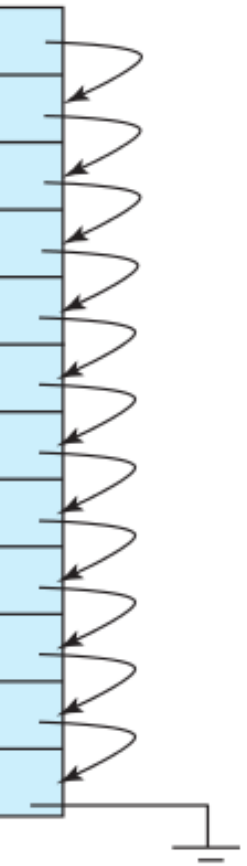
Hashing: a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed.

Records of each relation may be stored in a separate file. In a **multitable clustering file organization** records of several different relations can be stored in the same file. Related records are stored on the same block to minimize I/O.

Sequential File Organization

Suitable for applications that require **sequential processing** of the entire file.

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	



Sequential File Organization

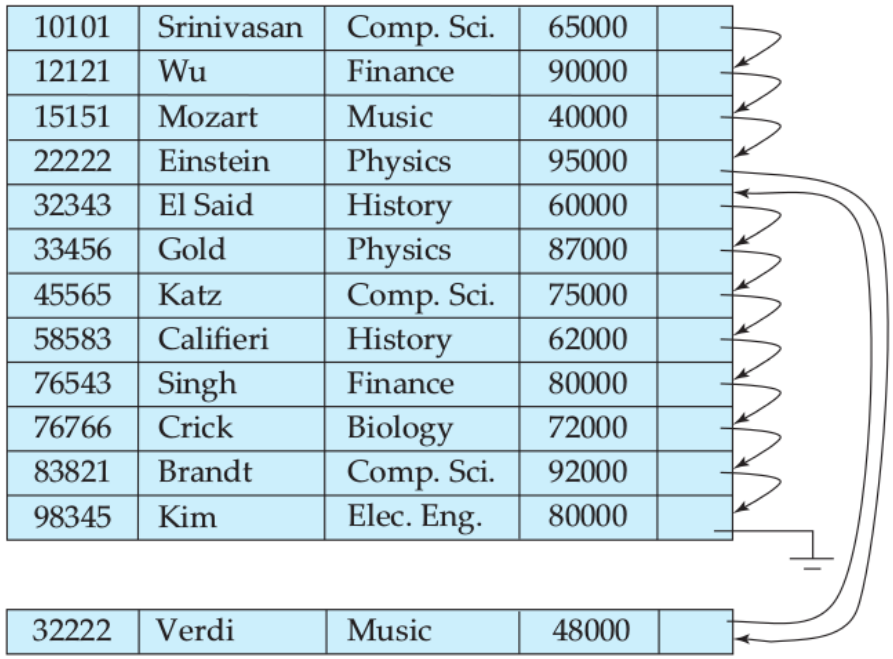
Deletion: use pointer chains

Sequential File Organization

Deletion: use pointer chains

Insertion: locate the position where the record is to be inserted

- if there is free space insert there
- if no free space, insert the record in an overflow block
- In either case, pointer chain must be updated.



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

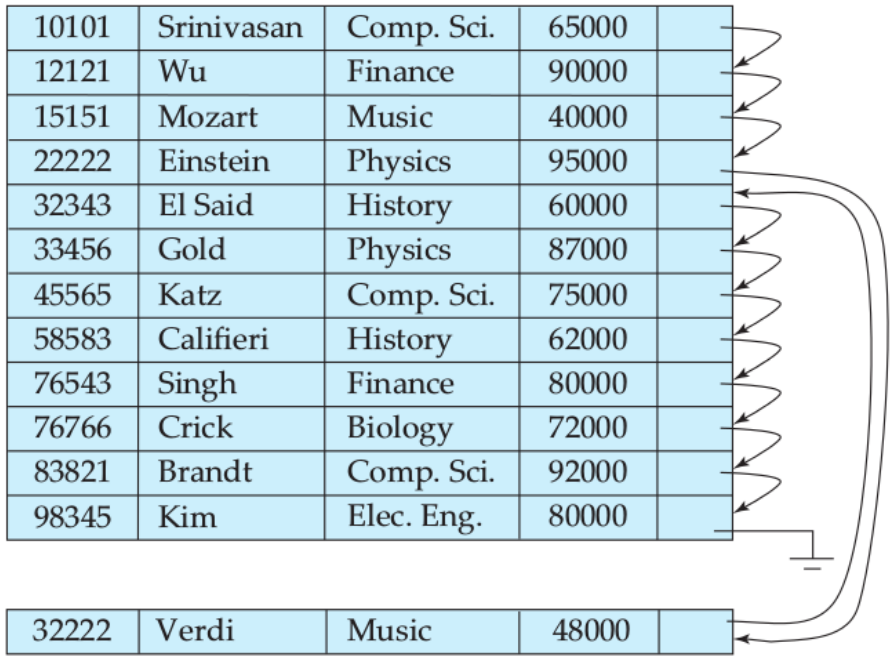
32222	Verdi	Music	48000	

Sequential File Organization

Deletion: use pointer chains

Insertion: locate the position where the record is to be inserted

- if there is free space insert there
- if no free space, insert the record in an overflow block
- In either case, pointer chain must be updated.



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

32222	Verdi	Music	48000	

Need to reorganize the file from time to time to restore sequential order

Multitable Clustering File Structure

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

select *dept_name, building, budget, ID, name, salary*
from *department* **natural join** *instructor;*

Multitable Clustering File Structure

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

select *dept_name, building, budget, ID, name, salary*
from *department* **natural join** *instructor;*

Comp. Sci.	Taylor	100000
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

Multitable Clustering File Structure

• Good for queries involving department `natural join` instructor, and for queries involving one single department and its instructors.

Multitable Clustering File Structure

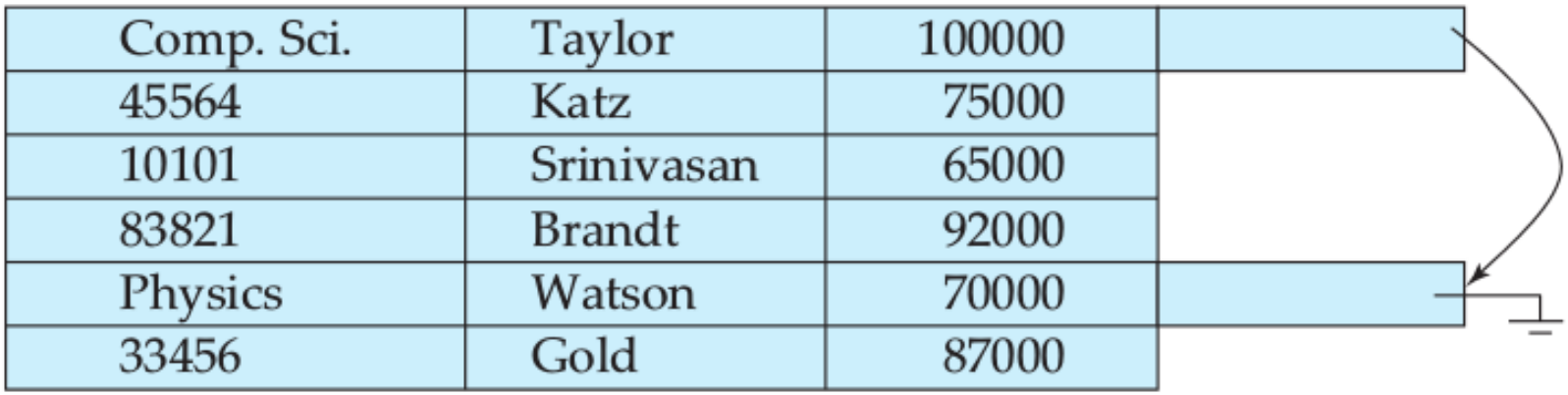
- Good for queries involving department `natural join` instructor, and for queries involving one single department and its instructors.
- Bad for queries involving only department.

Multitable Clustering File Structure

- Good for queries involving department `natural join` instructor, and for queries involving one single department and its instructors.
- Bad for queries involving only department.
- Results in variable size records.

Multitable Clustering File Structure

- Good for queries involving department `natural join` instructor, and for queries involving one single department and its instructors.
- Bad for queries involving only department.
- Results in variable size records.
- Can add pointer chains to link records of a particular relation.



Comp. Sci.	Taylor	100000	
45564	Katz	75000	
10101	Srinivasan	65000	
83821	Brandt	92000	
Physics	Watson	70000	
33456	Gold	87000	

Data Dictionary Storage

The **Data dictionary** (also called *system catalogue*) stores **metadata**; that is, data about data, such as

Information about relations:

- names of relations
- names, types and lengths of attributes of each relation
- names and definitions of views
- integrity constraints

Data Dictionary Storage

The **Data dictionary** (also called *system catalogue*) stores **metadata**; that is, data about data, such as

Information about relations:
- names of relations
- names, types and lengths of attributes of each relation
- names and definitions of views
- integrity constraints

User and accounting information, including passwords

Data Dictionary Storage

The **Data dictionary** (also called *system catalogue*) stores **metadata**; that is, data about data, such as

Information about relations:
- names of relations
- names, types and lengths of attributes of each relation
- names and definitions of views
- integrity constraints

User and accounting information, including passwords

Statistical and descriptive data
- number of tuples in each relation

Data Dictionary Storage

The **Data dictionary** (also called *system catalogue*) stores **metadata**; that is, data about data, such as

Information about relations:
- names of relations
- names, types and lengths of attributes of each relation
- names and definitions of views
- integrity constraints

User and accounting information, including passwords

Statistical and descriptive data
- number of tuples in each relation

Physical file organization information
- How relation is stored (sequential/hash/...)
- Physical location of relation

Data Dictionary Storage

The **Data dictionary** (also called *system catalogue*) stores **metadata**; that is, data about data, such as

Information about relations:
- names of relations
- names, types and lengths of attributes of each relation
- names and definitions of views
- integrity constraints

User and accounting information, including passwords

Statistical and descriptive data
- number of tuples in each relation

Physical file organization information
- How relation is stored (sequential/hash/...)
- Physical location of relation

Information about indices

Data Dictionary Storage